tek.com/keithley

Model 4200A-SCS Parameter Analyzer

Reference Manual

4200A-901-01 Rev. H March 2020

4200A-901-01H

KEITHLEY
A Tektronix Company

4200A-SCS

Parameter Analyzer

Reference Manual

Document number: 4200A-SCS-901-01 Rev. H / April 2020

KEITHLEY
A Tektronix Company

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 nonhazardous voltages, there are situations where hazardous conditions may be present.

This product is intended for use by 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 products are designed for use with electrical signals that are measurement, control, and data I/O connections, with low transient overvoltages, and must not be directly connected to mains voltage or to voltage sources with high transient overvoltages. Measurement Category II (as referenced in IEC 60664) connections require protection for high transient overvoltages often associated with local AC mains connections. Certain Keithley measuring instruments may be connected to mains. These instruments will be marked as category II or higher.

Unless explicitly allowed in the specifications, operating manual, and instrument labels, do not connect any instrument to mains.

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 30 V RMS, 42.4 V peak, or 60 VDC 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 1000 V, 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.

For safety, instruments and accessories must be used in accordance with the operating instructions. If the instruments or accessories are used in a manner not specified in the operating instructions, the protection provided by the equipment may be impaired.

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

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 protective earth (safety 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 protective earth (safety ground) using the wire recommended in the user documentation.

The  symbol on an instrument means caution, risk of hazard. The user must 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 warning, risk of electric shock. 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 hazards 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.

The **CAUTION** heading with the  symbol in the user documentation explains hazards that could result in moderate or minor injury or damage the instrument. Always read the associated information very carefully before performing the indicated procedure. Damage to the instrument 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. Standard fuses with applicable national safety approvals may be used if the rating and type are the same. The detachable mains power cord provided with the instrument may only be replaced with a similarly rated power cord. 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 to maintain accuracy and functionality of the product). If you are unsure about the applicability of a replacement component, call a Keithley office for information.

Unless otherwise noted in product-specific literature, Keithley instruments are designed to operate indoors only, in the following environment: Altitude at or below 2,000 m (6,562 ft); temperature 0 °C to 50 °C (32 °F to 122 °F); and pollution degree 1 or 2.

To clean an instrument, use a cloth dampened with deionized water 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.

Safety precaution revision as of June 2017.

Table of contents

Section 1

Introduction

In this section:

Introduction

The Keithley Instruments Model 4200A-SCS Parameter Analyzer is a customizable and fully-integrated parameter analyzer that provides synchronized insight into current-voltage (I-V), capacitance-voltage (C-V), and ultra-fast pulsed I-V characterization. The highest performance parameter analyzer, the 4200A-SCS accelerates semiconductor, materials, and process development.

The 4200A-SCS Clarius™ GUI-based software provides clear, uncompromised measurement and analysis capability. Furnished with embedded measurement expertise and hundreds of ready-to-use application tests, Clarius enables you to dig deeper into your research with speed and confidence.

The following topics introduce you to the 4200A-SCS, as follows:

- 4200A-SCS system overview (on page 1-2):
  - An overall block diagram and summary of the system.
  - Basic configurations and capabilities of the source-measure units (SMUs), preamplifiers, pulse generator and measure units, and capacitance-voltage units (CVUs) that source and measure the electrical signals that are connected to your devices under test (DUTs).
- Embedded PC policy (on page 1-5): Explains Keithley Instruments' policy concerning the use of third-party software and its no-tamper policy for the Microsoft® Windows® operating system.

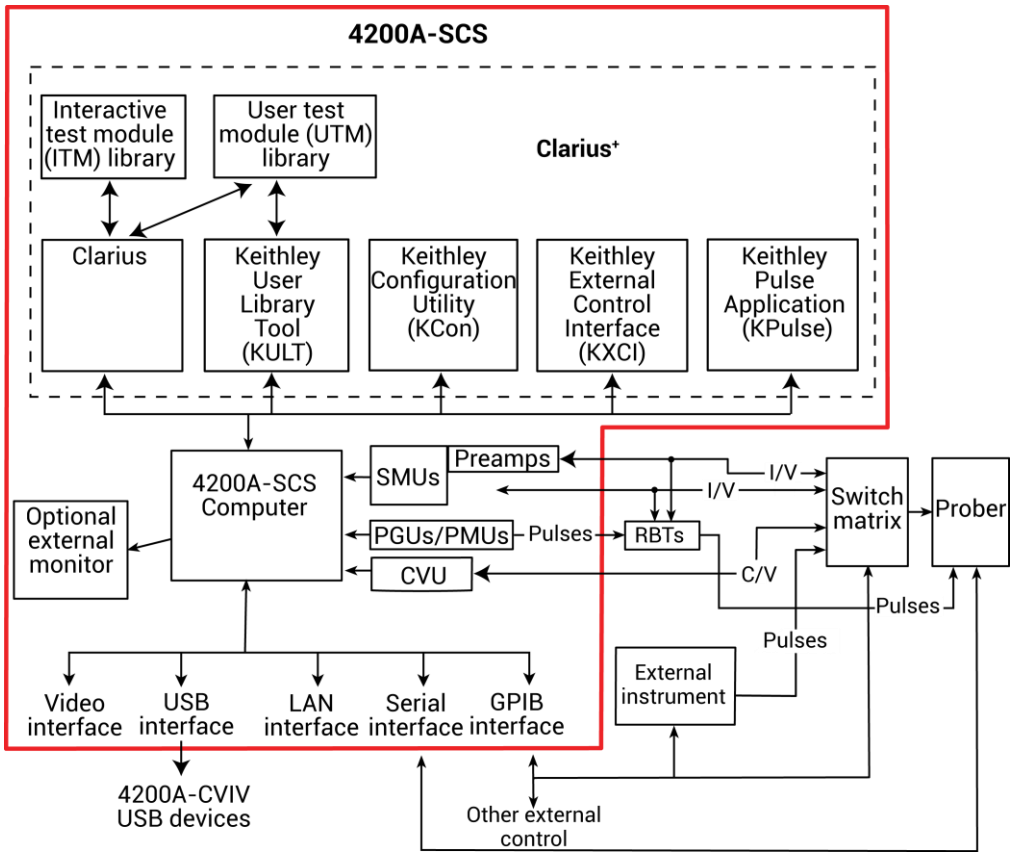
4200A-SCS system overview

The 4200A-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 Clarius. Clarius 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 4225-PMU Ultra-fast IV pulse-measure card. Tests requiring pulse source, but no corresponding pulse measurement, can use the 4220-PGU pulse-only card. One typical configuration with pulse source-measure capability would be a 4200A-SCS system that consists of four SMUs, two 4225-PMUs and four 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 4200A-SCS components and typical supported external components are illustrated in the following figure.

Figure 1: 4200A-SCS summary



Source-Measure Unit (SMU)

The fundamental instrument module used by the 4200A-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 and measure current or voltage
- Source current and measure voltage or 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 4200-SMU is a medium power (2 W) source-measure unit, and the 4210 is a high power (20 W) SMU. The following table lists the maximum limits of the two SMUs.

Source-measure units			
Model	**Maximum voltage**	**Maximum current**	**Maximum power**
4200-SMU	210 V	105 mA	2.2 W
4210-SMU	210 V	1.05 A	22 W

Preamplifier

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

If preamplifiers are ordered, the 4200A-SCS will be shipped from the factory with the preamplifiers installed on the rear panel of the mainframe. The preamplifiers 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 preamplifier is matched to the SMU to which it is connected. Therefore, if you disconnect the preamplifiers 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 (on page 3-16).

Pulse source-measure hardware

Keithley Instruments' pulse source-measure hardware for the 4200A-SCS includes one or more pulse Keithley pulse cards. Refer to Pulse measure and pulse generator units (on page 5-1) for details.

Ground unit (GNDU)

The ground unit on the rear panel of the 4200A-SCS provides a convenient method of making ground connections. This eliminates the need to use a SMU for this purpose. For details, refer to Ground unit (GNDU) overview (on page 3-25).

Capacitance-voltage (CVU) hardware

Keithley Instruments' capacitance-voltage hardware for the 4200A-SCS includes up to one capacitance-voltage unit (CVU). Refer to Multi-frequency capacitance-voltage unit (on page 4-1) for details.

Software features

CAUTION

When you start one of the Clarius+ applications for the first time, you must agree to the license agreement before continuing. If you do not respond with "Yes", your system will not function until you reinstall the software.

You can use the following applications to operate and maintain the 4200A-SCS.

- Clarius: This is the main 4200A-SCS parameter analysis application. You use Clarius to organize tests into individual projects. You then use Clarius to manage, execute, and review test results from the projects.
- Keithley Configuration Utility (KCon): KCon allows you to configure external GPIB instruments, switch matrices, and analytical probers connected to the 4200A-SCS. KCon also provides basic diagnostic and troubleshooting functions.
- Keithley User Library Tool (KULT): KULT allows you to create algorithms (user modules) in the C programming language and to then integrate these modules into Clarius. The modules can control internal 4200A-SCS instrumentation and external instrumentation. You also use KULT to create and manage libraries of user modules.
- Keithley External Control Interface (KXCI): KXCI allows you to use an external computer to remotely control 4200A-SCS SMUs over the GPIB bus or ethernet.
- Keithley Pulse Application (KPulse): A non-programming alternative to configure and control installed 4225-PMU or 4220-PGU pulse units. You can use it for quick tests that require minimal interaction with other 4200A-SCS test resources.

Embedded computer policy

CAUTION

If you install software that is not part of the standard application software for the 4200A-SCS, the non-standard software may be removed if the instrument is sent in for service. Back up the applications and any data related to them before sending the instrument in for service.

CAUTION

Do not reinstall or upgrade the Microsoft® Windows® operating system (OS) on any 4200A-SCS unless the installation is performed as part of authorized service by Keithley Instruments. Violation of this precaution will void the 4200A-SCS warranty and may render the 4200A-SCS unusable. Any attempt to reinstall or upgrade the 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 you must not attempt to reinstall or upgrade the operating system, you can restore the hard drive image (complete with the operating system) using the Acronis True Image OEM software tool, described in System-level backup and restore software (on page 12-8).

Section 2

Connections and configuration

In this section:

Introduction

This section contains information about connecting and configuring the 4200A-SCS.

NOTE

Details for the Keithley pulse cards are provided in the Pulse measure and pulse generator units (on page 5-1) section of this manual.

Basic source-measure connections

Basic information on connecting an external monitor, source-measure units (SMUs), the preamplifier, and the ground unit to devices under test (DUTs) is described in this section.

Maximum signal limits

⚠ WARNING

The 4200A-SCS is provided with an interlock circuit that must be positively activated in order for the high voltage output to be enabled. The interlock helps facilitate safe operation of the equipment in a test system. Bypassing the interlock could expose the operator to hazardous voltages that could result in personal injury or death.

⚠ WARNING

Asserting the interlock allows the SMU and preamplifier terminals to become hazardous, exposing the user to possible electrical shock that could result in personal injury or death. SMU and preamplifier 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 V, Category O.

CAUTION

The maximum allowed voltage between circuit COMMON and chassis ground is ±32 Vdc.
The maximum allowed voltages between the preamplifier signals are:
COMMON to chassis ground: 32 V_{peak}
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, and so forth.

- If the device is to be shielded (but not guarded), connect the DUT shield to COMMON (see the next figure).
- If the device is to be guarded, connect the DUT shield to GUARD (inner shield of triaxial cable; see the next figure).

⚠ WARNING

A safety shield must be used whenever hazardous voltages (>30 V_{RMS}, 42 V_{peak}) will be present in the test circuit. To prevent electrical shock that could cause injury or death, never use the 4200A-SCS in a test circuit without a properly installed and configured safety shield.

See Guarding (on page 3-32) for more information on the principles and advantages of guarding.

Figure 2: Device shielding

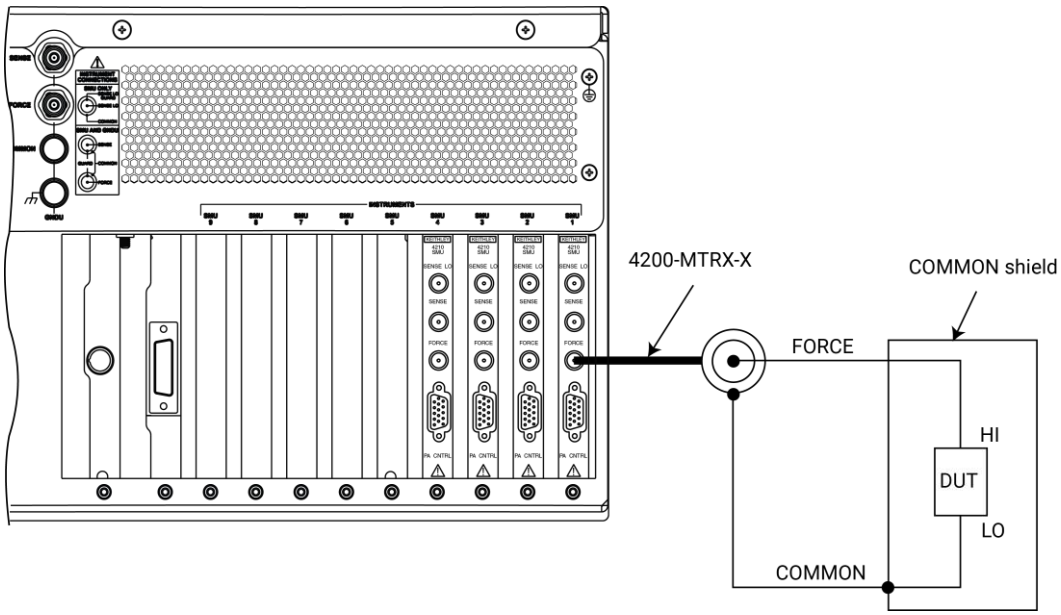


Figure 3: Device shielding basic circuit

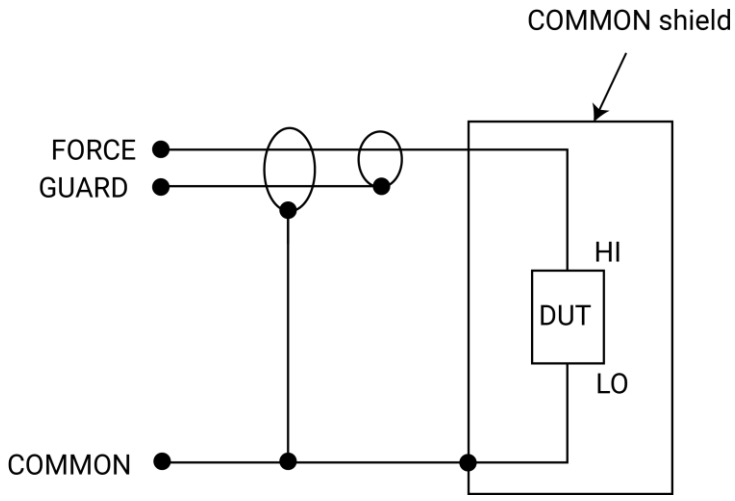


Figure 4: Device guarding

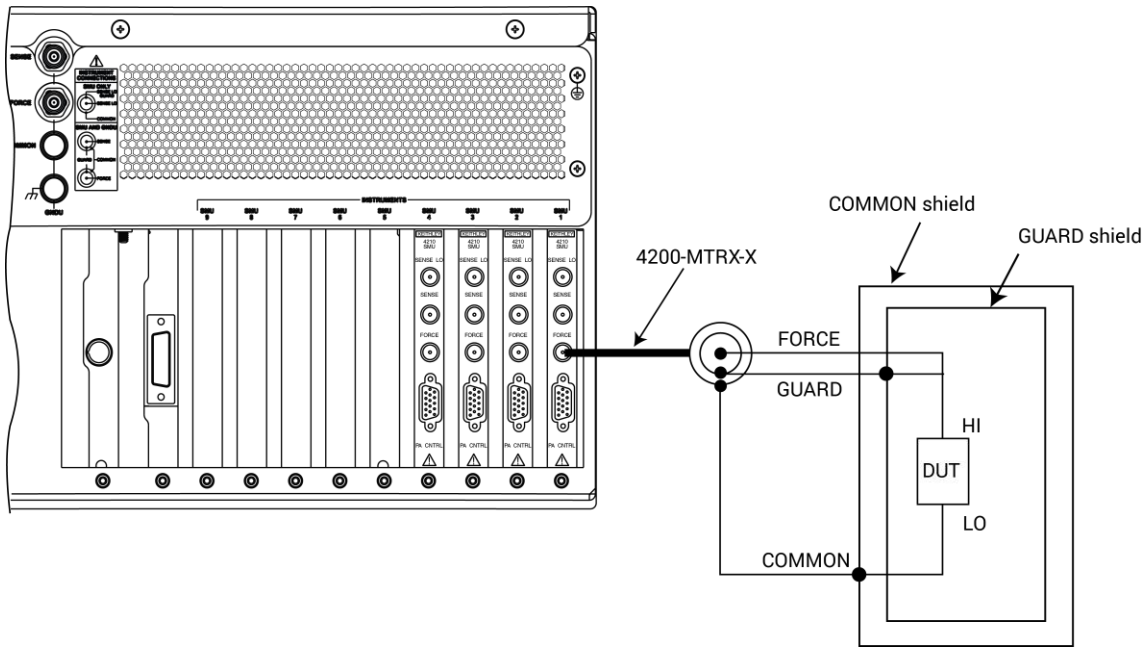
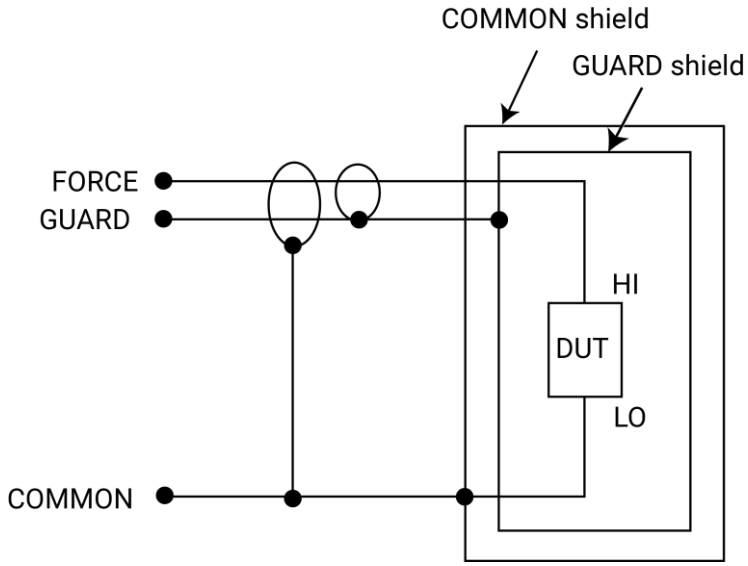


Figure 5: Device guarding basic circuit



Signal integrity

To maintain signal integrity, especially at low current levels, consider the following when making signal connections between the 4200A-SCS instrumentation and the device under test (DUT):

- Use only low-noise triaxial cables such as those provided with the SMU (4200-MTRX-X) and preamplifier (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.
- Avoid stresses in cables. Do not allow large portions to hang under their own weight. Place on a table or flat surface if possible. Avoid tight bends in the cables.

Refer to Maintenance (on page 13-1) or the *Keithley Instruments Low Level Measurements Handbook* for more information about measurement integrity.

SMU connections

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

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 (on page 2-10).

SMU local sense connections

The next figure shows typical SMU connections using local sensing. Use a triaxial cable such as the 4200-MTRX-X to 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 6: SMU local sense connections

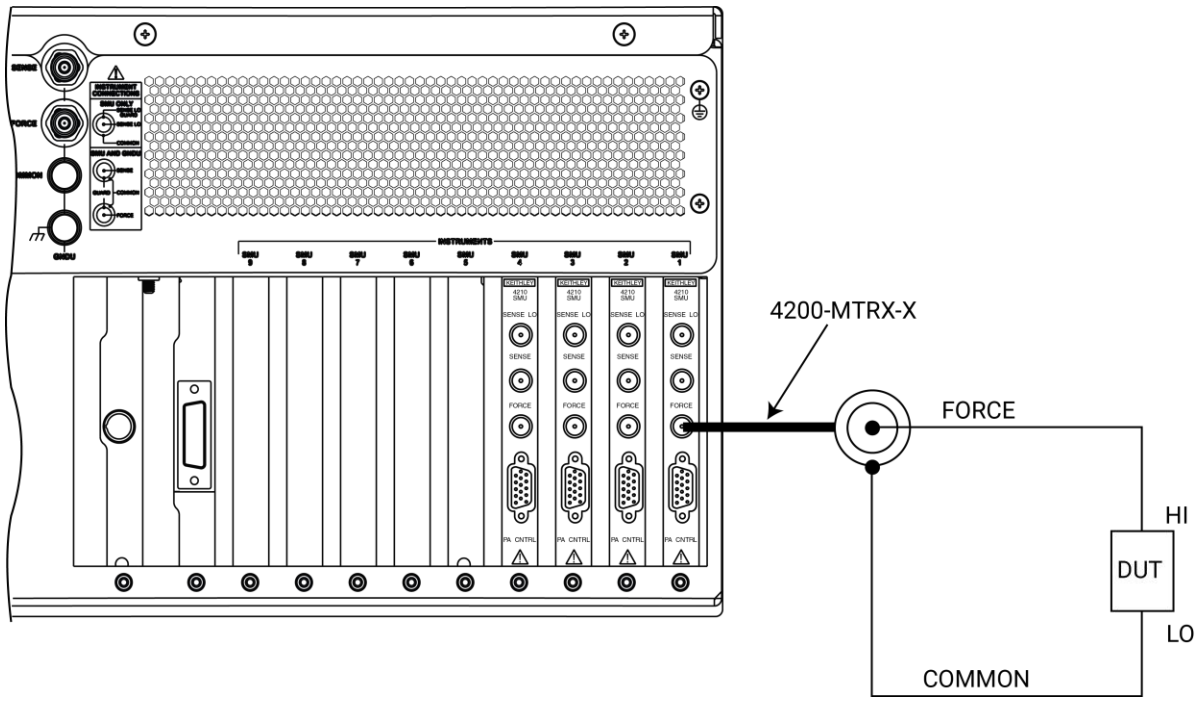
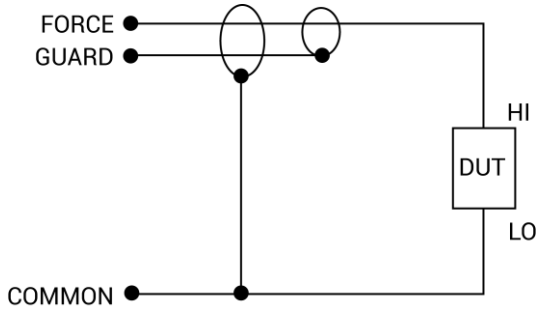


Figure 7: SMU local sense connections - equivalent circuit



Basic device connections for SMUs

The next figures show the basic device connections to the 4200A-SCS rear panel, independent of the device mounting, test fixtures, and probers.

Figure 8: Two-terminal device connections to SMUs and preamplifiers

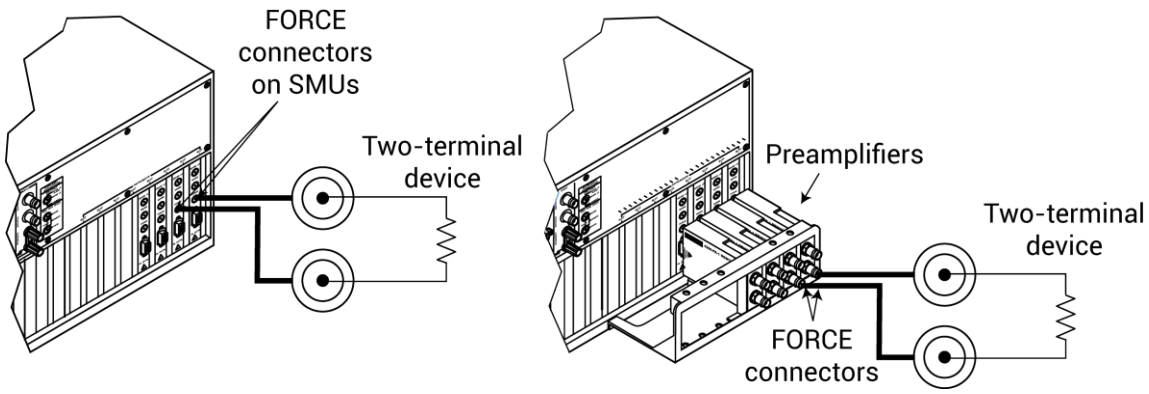


Figure 9: Three-terminal device connections to SMUs and preamplifiers

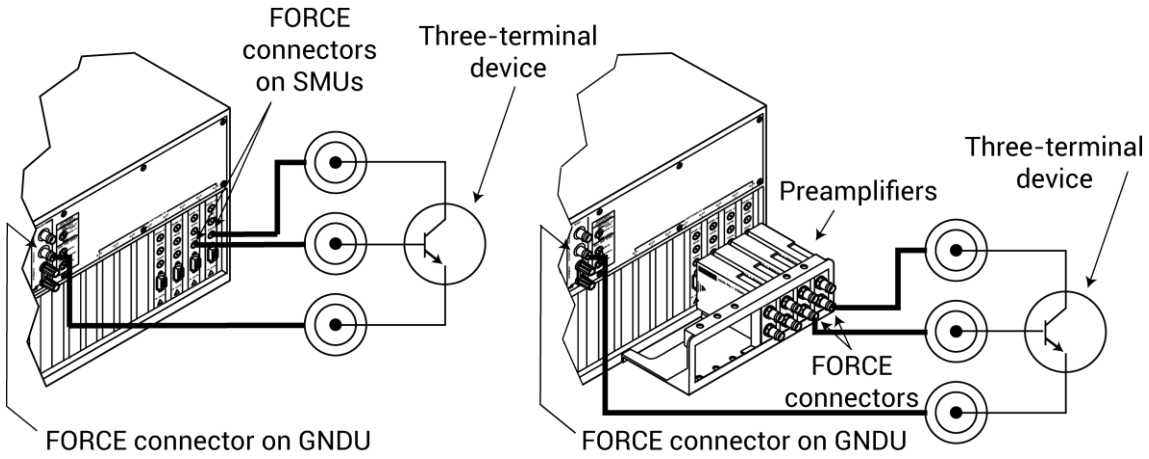
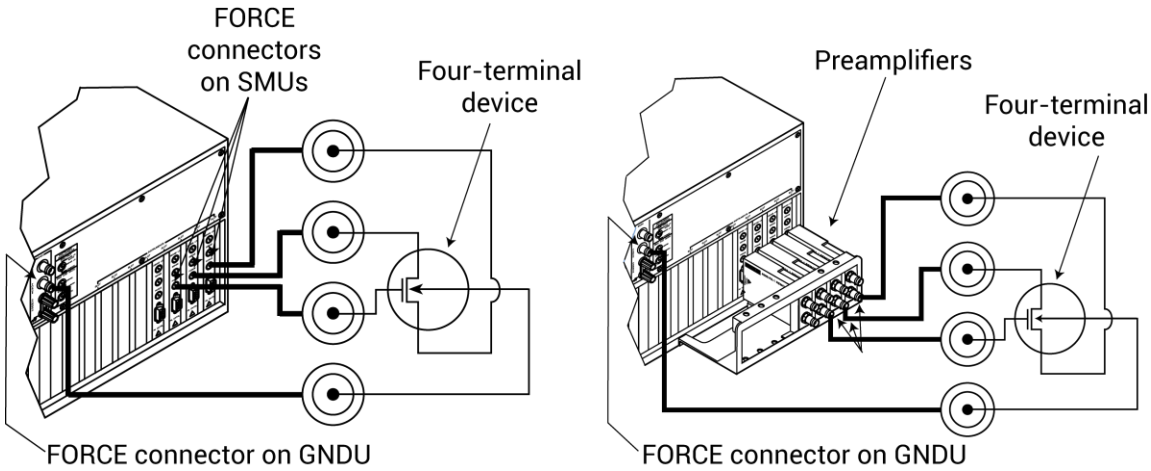


Figure 10: Four-terminal device connections to SMUs and preamplifiers



Preamplifier connections

NOTE

When using more than one preamplifier, use the ground unit for circuit COMMON connections instead of the outer shield of the preamplifier terminals (refer to Using the ground unit (on page 2-10)).

Connecting the preamplifier to the SMU

The preamplifier must be connected to the SMU before use. The connection method used depends on the mounting method (rear panel or remote). See Preamplifier mounting (on page 3-22) for details.

CAUTION

Turn off the system and disconnect the power cord before connecting or disconnecting the preamplifier. Failure to do so may result in SMU or preamplifier damage, possibly voiding the warranty.

Preamplifier local sense connections

The next figures show typical preamplifier connections using local sensing. Use a triaxial cable to make your connections as follows:

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

Figure 11: Preamplifier local sense connections

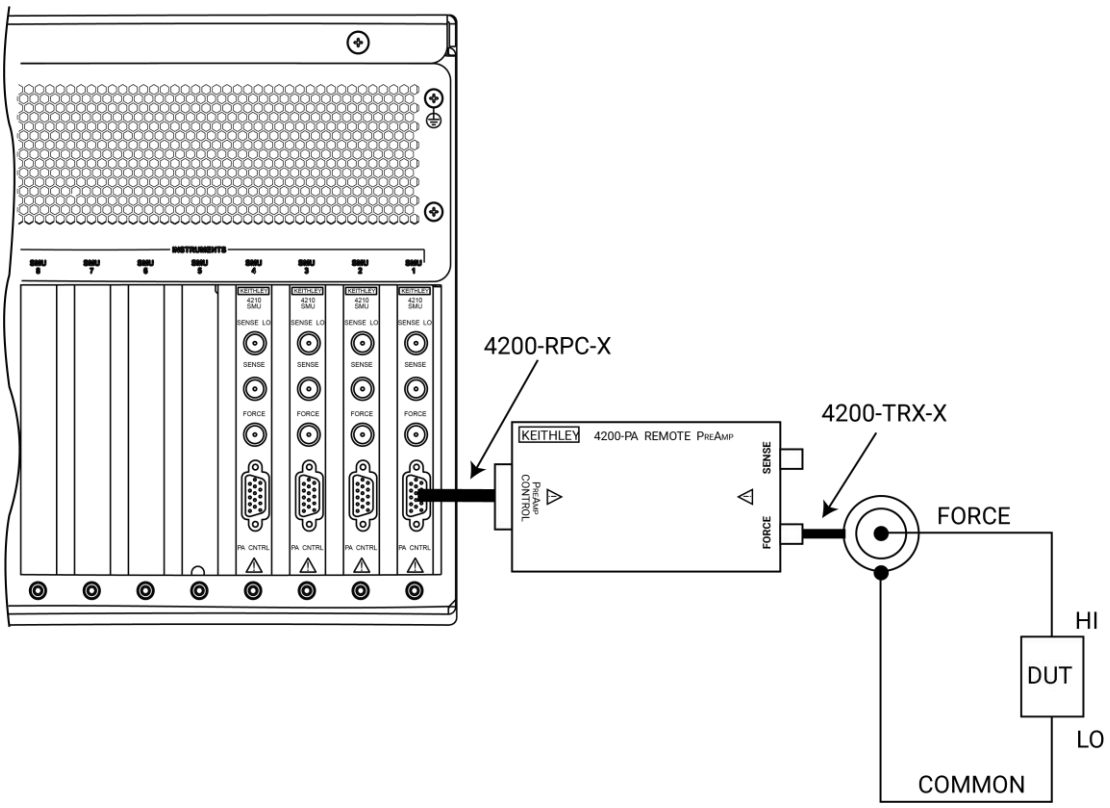
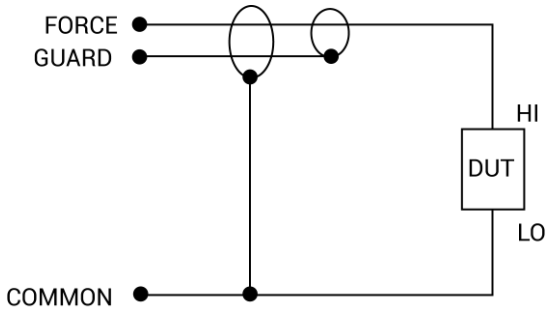


Figure 12: Basic preamplifier local sense connections



Using the ground unit

The ground unit (GNDU) provides convenient access to circuit COMMON through the GNDU FORCE terminal or the GNDU COMMON binding post terminal. The GNDU also has a SENSE terminal. The SENSE LO signal of each instrument installed in the 4200A-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

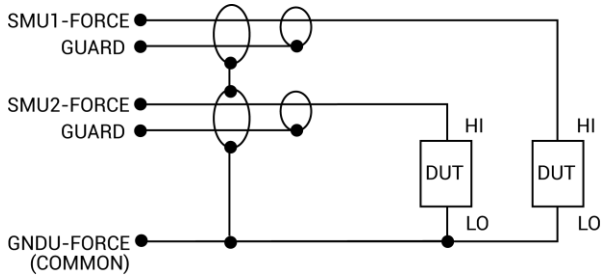
Although the ground unit is intended for circuit COMMON connections when using multiple SMUs, it can also be used for circuit COMMON connections when using only one SMU.

Ground unit and SMU local sense connections

The next figure 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 13: Ground unit and SMU local sense connections schematic

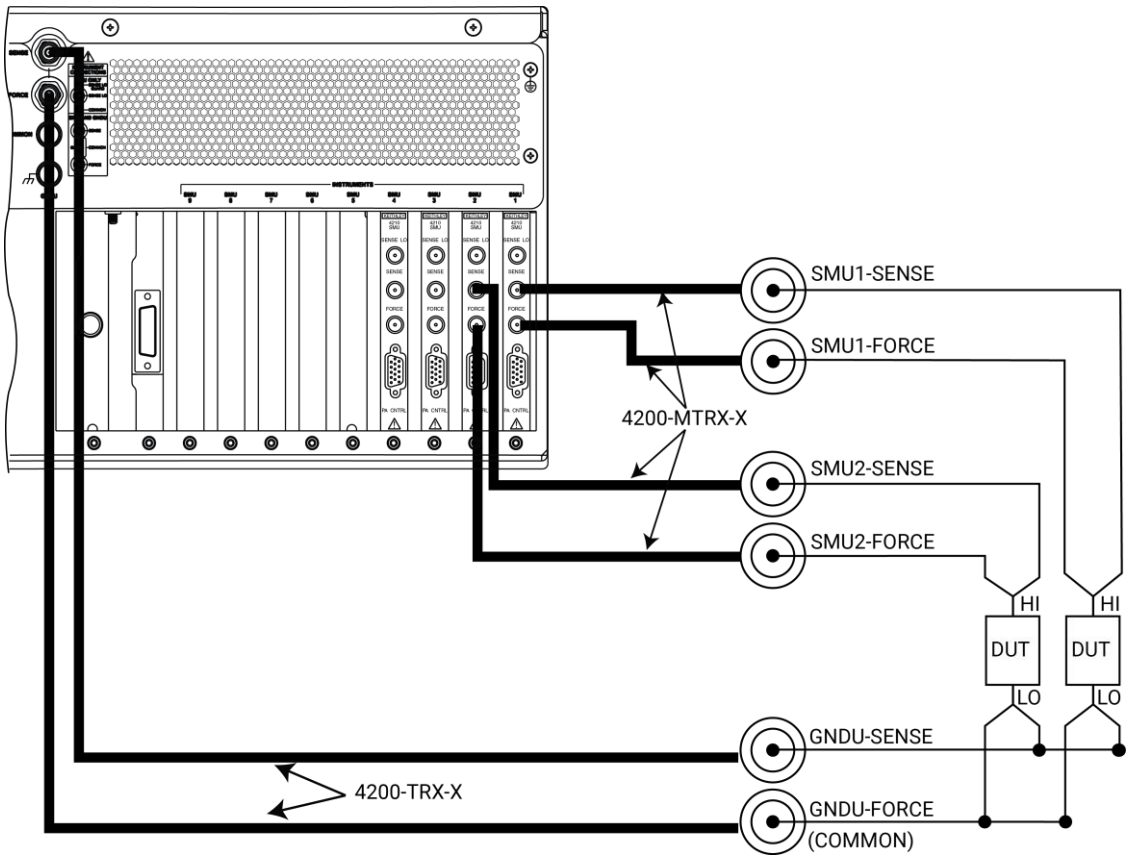


Ground unit and SMU remote sense connections

The next figure 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 14: Ground unit and SMU remote sense connections



Ground unit and preamplifier local sense connections

The next figure shows a typical local sense connection scheme using two preamplifiers, two DUTs, and the ground unit. Make connections as follows:

- Connect the two preamplifier 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 15: Ground unit and Preamplifier local sense connections

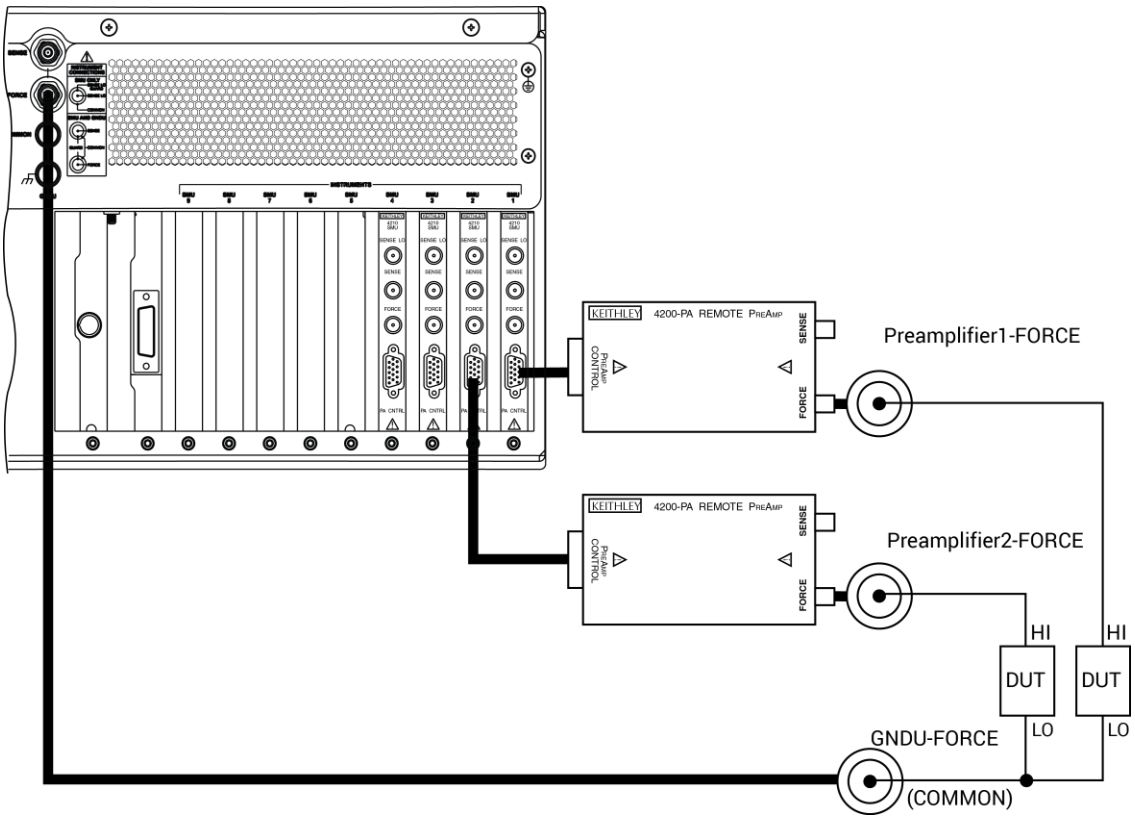
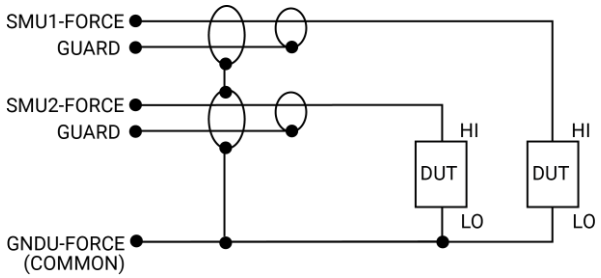


Figure 16: Ground unit and Preamplifier local sense connections schematic



Ground unit and preamplifier remote sense connections

The next figure shows a typical remote sense connection scheme using two preamplifiers, two DUTs, and the ground unit. Make connections as follows:

- Connect the preamplifier 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 the previous ground unit figures. Be sure to connect all your DUT LO terminals to the GNDU FORCE terminal (and SENSE terminal when using remote sensing).

Figure 17: Ground unit and preamplifier remote sense connections

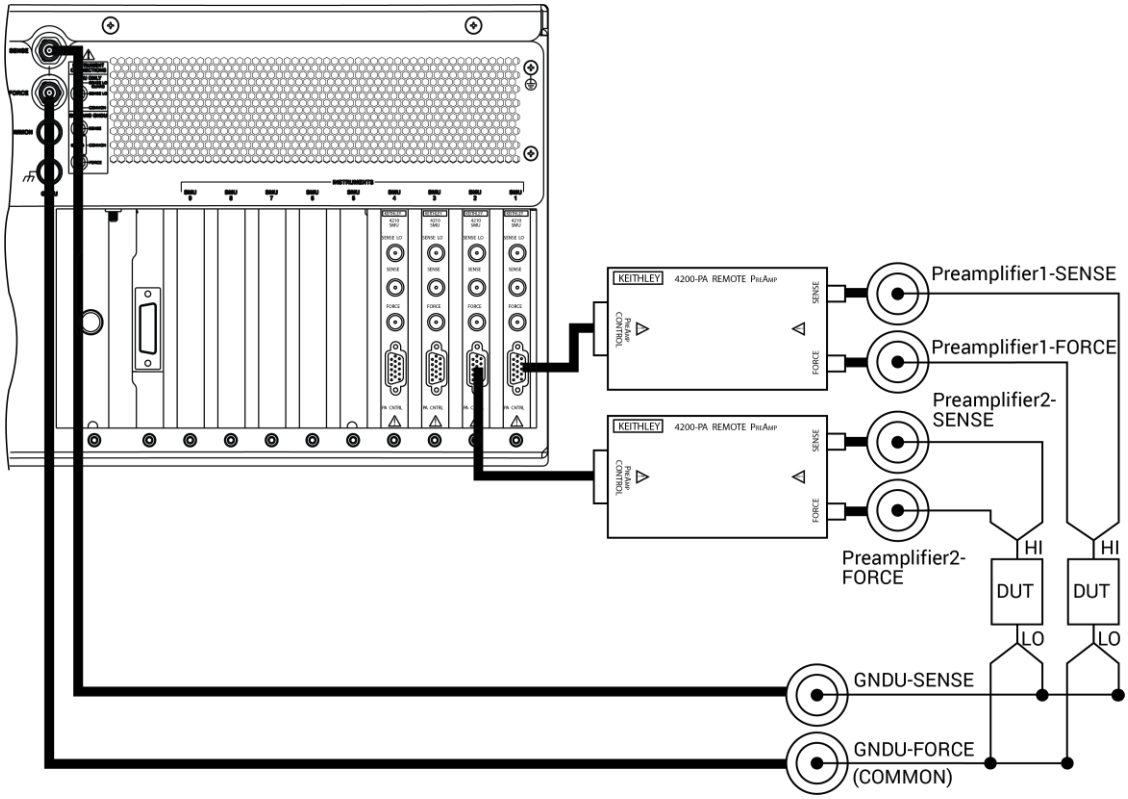
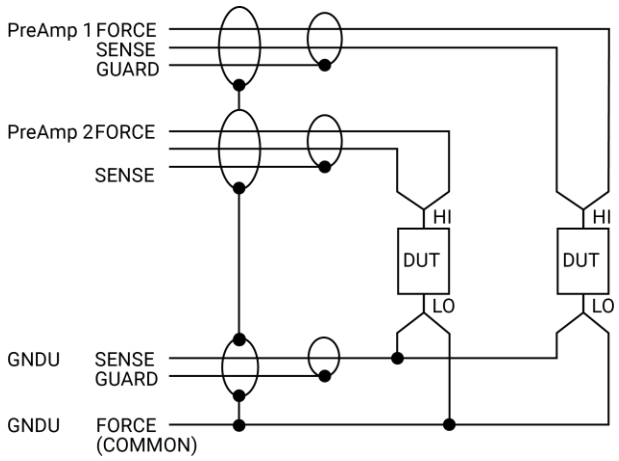


Figure 18: Ground unit and preamplifier remote sense connections schematic



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. The next figure 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 Clarius (on page 6-1) for more detailed instructions on configuring a SMU to provide a COMMON connection.

Figure 19: Typical SMU connections

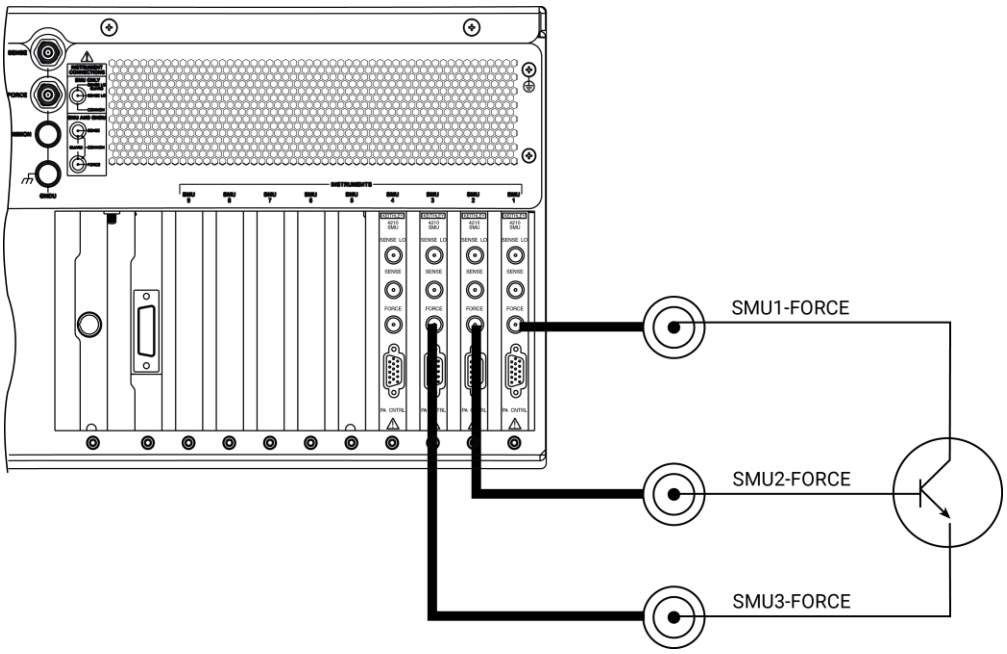
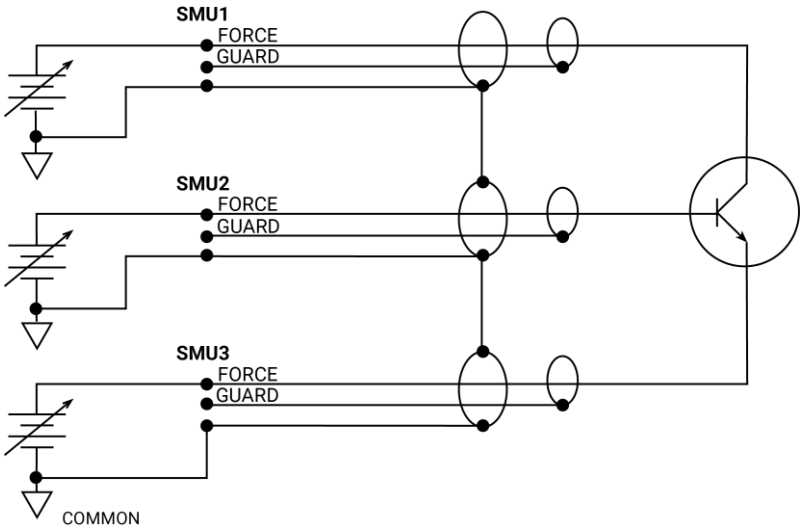


Figure 20: Typical SMU COMMON connections schematic



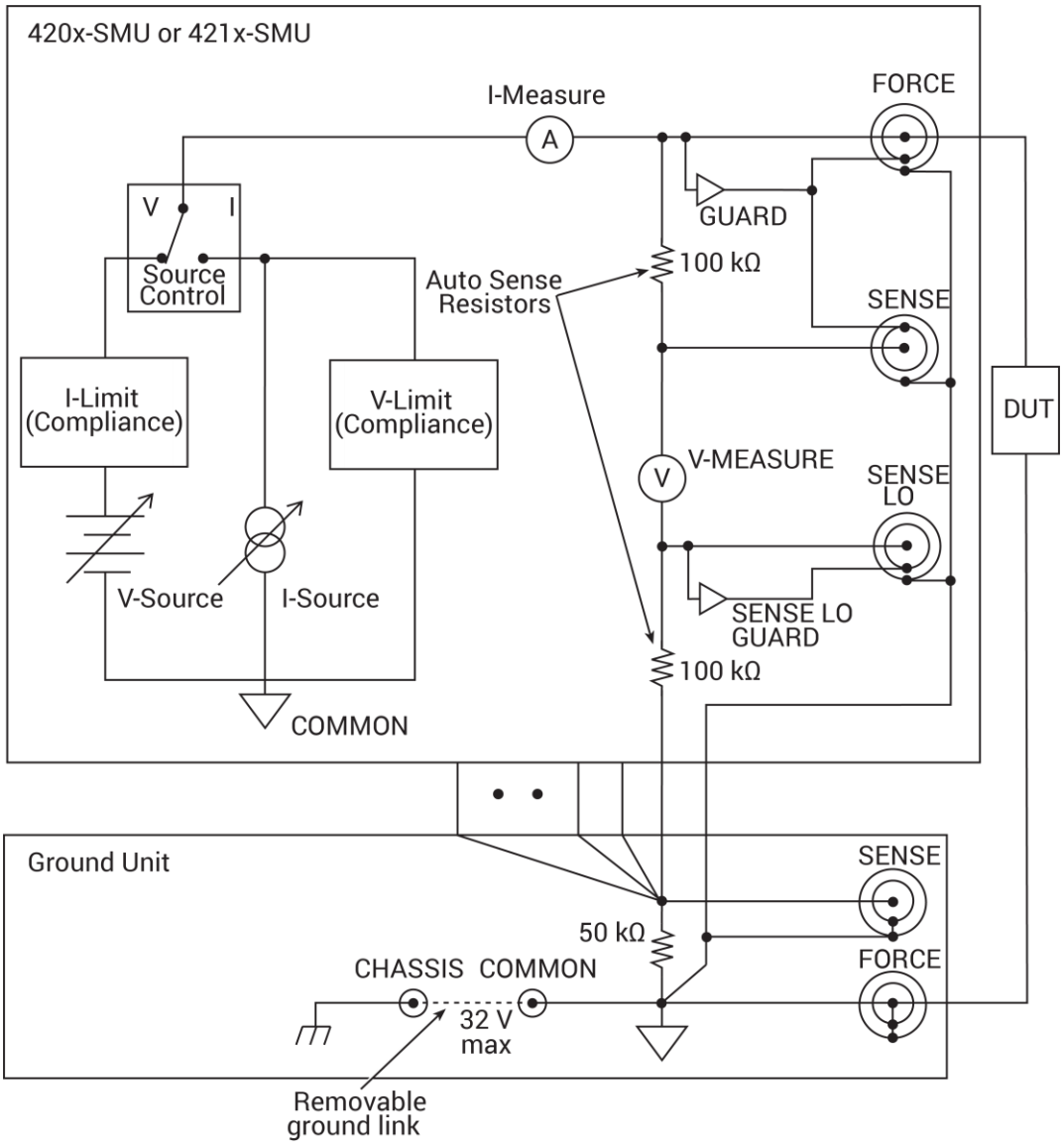
4200-SMU and 4210-SMU overview

The following paragraphs discuss the basic characteristics of both the 4200-SMU and 4210-SMU.

Basic SMU circuit configuration

The basic SMU circuit configuration is shown in the next figure. The SMU is essentially a voltage or current source in series with a current meter, connected in parallel with a voltage 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 (on page 2-1) for more detailed information.

Figure 21: Basic SMU source-measure configuration



SMU terminals and connectors

The locations and configuration of the 4200-SMU and 4210-SMU terminals are shown in the next figure. Basic information about these terminals is summarized below.

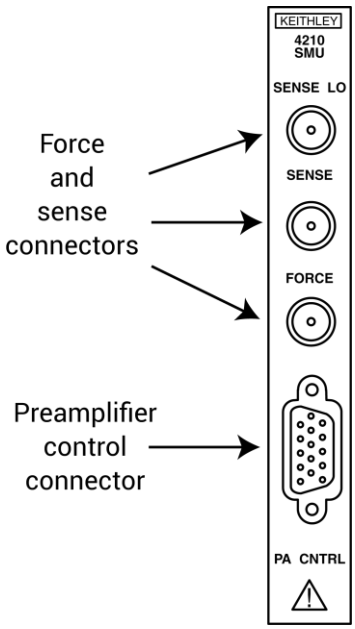
WARNING

Asserting the interlock allows the SMU and preamplifier terminals to become hazardous, exposing the user to possible electrical shock that could result in personal injury or death. SMU and preamplifier 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 V, Category O.

CAUTION

The maximum allowed voltage between COMMON and chassis ground is ±32 VDC.

Figure 22: 4200-SMU and 4210-SMU connectors



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
- The outer shield is circuit COMMON

Nominal internal autosense resistance appears between SENSE LO GUARD and COMMON.

NOTE

The remote sense capability of the ground unit should be used instead of the SENSE LO of a SMU. If it is necessary to use the SENSE LO terminal of a SMU, the SENSE LO terminals of all SMUs being used in a single 4200A-SCS should be connected to the DUT.

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.

- The center pin is SENSE
- The inner shield is GUARD
- The outer shield is circuit COMMON

Nominal internal autosense 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 are eliminated and the SMU will sense locally.

FORCE terminal

The FORCE terminal is a miniature triaxial connector used to apply the SMU FORCE signal to the DUT when a preamplifier is not being used.

- The center pin is FORCE
- The inner shield is GUARD
- The outer shield is circuit COMMON

PA CNTRL connector

The PA CNTRL (preamp control) terminal is a 15-pin D-sub connector that provides both power and signal connections to the 4200-PA remote preamp. The preamp can either be mounted and connected directly to the SMU, or it can be connected to the SMU using a cable (4200-RPC-X) when mounted remotely.

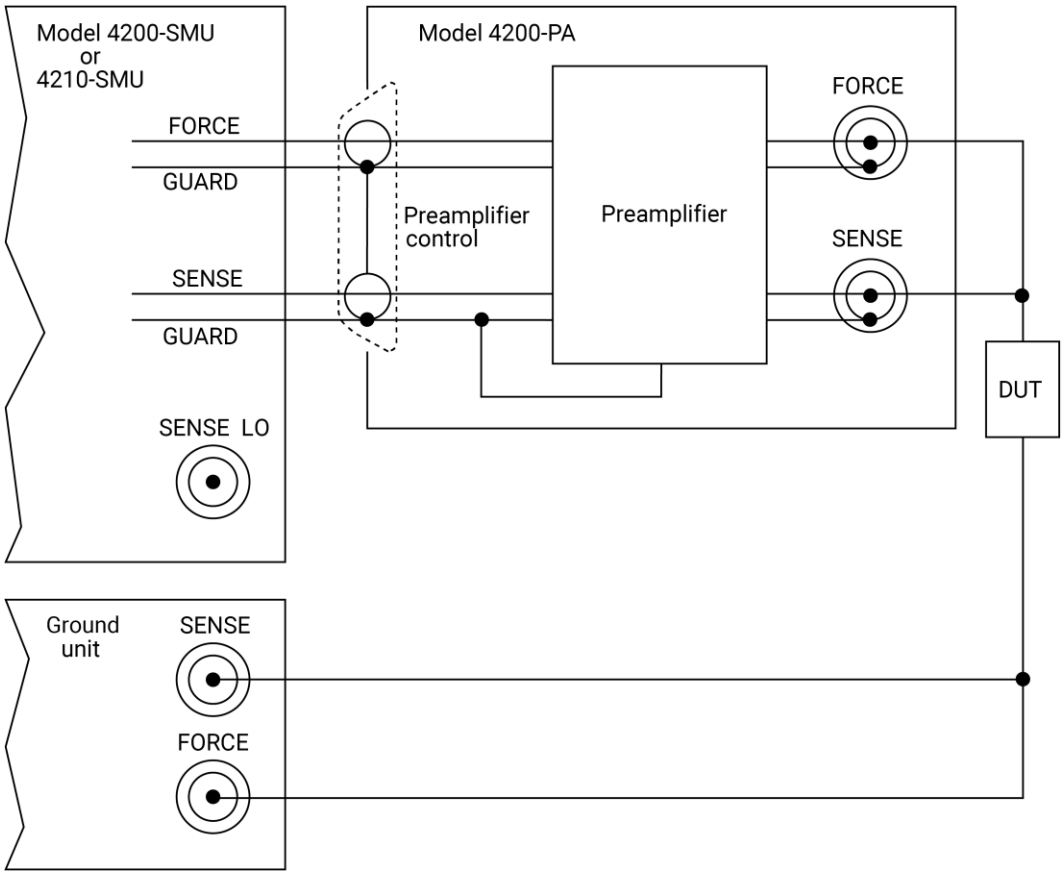
Basic SMU and preamplifier circuit configuration

Basic SMU and preamp circuit configuration is shown in the next figure. This configuration is similar to the SMU configuration discussed earlier, and the preamp, which adds low-current source-measure capabilities.

NOTE

The preamp FORCE terminal is connected to DUT HI, while DUT LO is connected to GROUND.

Figure 23: Basic SMU and preamplifier source-measure configuration



Test equipment connections

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

- Recommended connecting cables
- Switching system 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 preamplifier (4200-TRX-X) for all source-measure signal connections.

Refer to “Triaxial cables” in the *4200A-SCS User’s Manual* for a complete description of recommended triaxial cables. Refer to Pulse measure and pulse generator units (on page 5-1) and 4210-CVU connections (on page 4-7) for connection information.

NOTE

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

Test fixture and device under test (DUT) connections

The next table summarizes recommended Keithley Instruments test fixtures.

Test fixture	Description	DUT/fixture connections
8007	Semiconductor test fixture	DIP/triaxial cables
LR:8028	Component test fixture optimized for device testing up to 200 V/1 A	Mini jumpers

WARNING

Asserting the interlock allows the SMU and preamplifier terminals to become hazardous, exposing the user to possible electrical shock that could result in personal injury or death. SMU and preamplifier 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 V, Category O.

Test fixtures

There are two types of test fixtures for the 4200A-SCS:

- Low-voltage fixtures (less than ±20 V).
- High-voltage fixtures (greater than ±20 V), which require extra precautions to ensure there are no dangerous shock hazards.

WARNING

To avoid high voltage exposure that could result in personal injury or death, whenever the interlock of the 4200A-SCS is asserted, the FORCE and GUARD terminals of the SMUs and preamplifier should be considered to be at high voltage, even if they are programmed to a non-hazardous voltage current.

Testing with less than ±20 V with SMUs

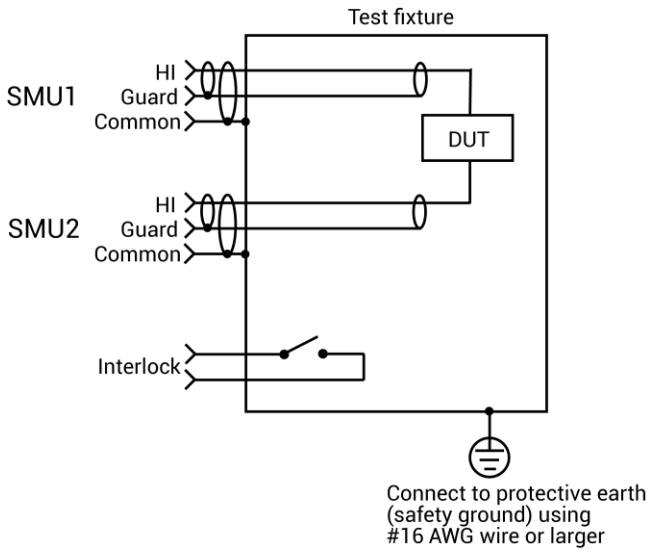
For testing discrete devices, you need a test fixture equipped with 3-lug triaxial connectors. This allows the 4200A-SCS to be connected to the discrete device.

The next figure shows a basic test fixture to test a two-terminal device. For best performance when testing with less than ±20 V, follow these standard industry practices:

- Use a metal test fixture
- Connect the metal fixture to COMMON
- Mount the DUT on high-resistivity terminals (for example, Teflon)
- Guarding will reduce leakage and parasite capacitance that degrades measurement quality

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. See the *Learning Center* on your 4200A-SCS for a copy.

Figure 24: Typical test fixture



NOTE

The 4200A-SCS functions on all current ranges and up to ±20 V without the interlock being asserted.

Testing with more than ±20 V

WARNING

Asserting the interlock allows the SMU and preamplifier terminals to become hazardous, exposing the user to possible electrical shock that could result in personal injury or death. SMU and preamplifier 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 V, Category O.

If voltages greater than ±20 V are required for testing, follow these steps:

- Add an interlock switch to the fixture to ensure that hazardous voltages are not present when the exterior enclosure of the test fixture is open.

NOTE

The 4200A-SCS voltage output will be higher when the exterior enclosure of the test fixture is closed.

- Connect the exterior enclosure to COMMON or safety ground using #16 AWG wire or greater.
- Ensure that the wiring (FORCE, GUARD, and SENSE) within the fixture does not electrically contact the exterior enclosure.

Prober connections

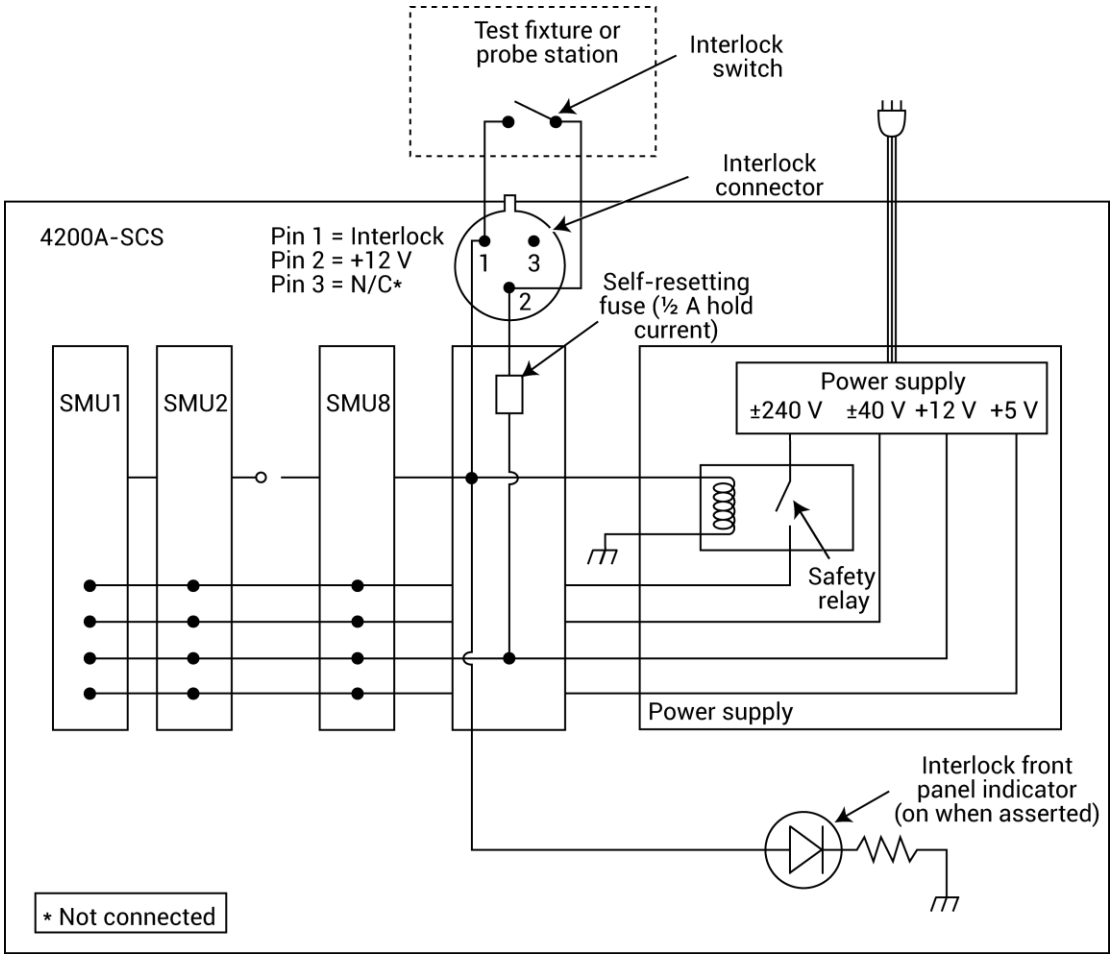
Refer to the prober appendices, starting with Set up a Probe Station (on page F-1), for details on how to enable and configure prober control for supported probers.

Configuring safety interlock operation

The next figure shows typical interlock connector wiring. Note that a normally open switch should be used. An open interlock condition occurs when the switch is open. To connect the interlock cable, see the “Getting started” section in the *4200A-SCS User’s Manual*.

The Keithley Instruments 8007 and LR:8028 test fixtures have a safety interlock connected to the instrument 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.

Figure 25: Interlock connector wiring



WARNING

Turning the 4200A-SCS output off does not place the instrument in a safe state (an interlock is provided for this function). Hazardous voltages may be present on all output and guard terminals. To prevent electrical shock that could cause injury or death, never make or break connections to the 4200A-SCS while the instrument is powered on. Turn off the equipment from the front panel or disconnect the main power cord from the rear of the 4200A-SCS before handling cables. Putting the equipment into an output-off state does not guarantee that the outputs are powered off if a hardware or software fault occurs. Precautions must be taken to prevent a shock hazard by surrounding the test device and any unprotected leads (wiring) with double insulation for 250 V, Category O.

Control and data connections

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

- GPIB connections
- Printer port connections
- LAN connections
- USB connections

GPIB connections

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

Recommended cables

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

Configuring IEEE-488 controller operation

The 4200A-SCS can be configured to operate either as a GPIB controller or GPIB subordinate. The 4200A-SCS acts as a GPIB controller when Clarius is running. Refer to Clarius (on page 6-2) for more information about Clarius. When operating as a controller, the 4200A-SCS reserves primary address 0, making that address unavailable to GPIB subordinate devices such as GPIB switch matrices, CV meters, and probe stations. Drivers for these and other instruments, typically integrated into semiconductor test systems, are included with the 4200A-SCS. These drivers, called user libraries, permit Clarius and the 4200A-SCS to control GPIB subordinate devices directly.

For instrumentation and equipment that is not supported by the standard user libraries, the open architecture of the 4200A-SCS allows you to create your own user libraries using the Keithley User Library Tool (KULT). Refer to Keithley User Library Tool (on page 8-1) for more information regarding the standard user libraries, KULT, and controlling external instrumentation.

Configuring IEEE-488 subordinate operation

The 4200A-SCS acts as a GPIB subordinate when the Keithley External Control Interface (KXCI) software is running. When KXCI is running, the 4200A-SCS can be controlled by an external computer using a command set nearly identical to the GPIB command set used to control a Keysight 4145B Semiconductor Parameter Analyzer. Refer to Keithley External Control Interface (KXCI) (on page 10-1) for detailed information regarding KXCI.

Pulse cards

The Keithley Instruments pulse cards are two-channel pulse generator instruments. There are two pulse generator instruments available for the 4200A-SCS:

- 4220-PGU High-Voltage Pulse Generator Unit (PGU)
- 4225-PMU Ultra-Fast IV Pulse Measure Unit (PMU)

Both cards offer:

- Two output channels
- Standard (2-level) pulse
- Segment Arb® waveform

Each output channel has two output ranges:

- 10 V (into high impedance, 5 V into 50 Ω)
- 40 V (into high impedance, 20 V into 50 Ω)

The 4220-PGU is a 2-channel voltage pulse generator. The 4225-PMU is also a 2-channel voltage pulse generator, but includes integrated simultaneous current and voltage measurement with two A/D converters for each channel.

Both can be isolated from the DUTs by a high-endurance output relay (HEOR). The HEOR is typically used for applications that require high-speed, high-volume switching of the output.

A pulse card can be programmed for continuous pulse output or set to output a finite number of pulses (burst or trig burst triggering modes). The pulse amplitude can be set from 100 mV to 40 V. The pulse period can be set from 20 ns to 1 s with a minimum pulse width of 10 ns. Transition times (pulse rise and pulse fall) can be set independently. Refer to Pulse card settings for details on all pulse card settings.

NOTE

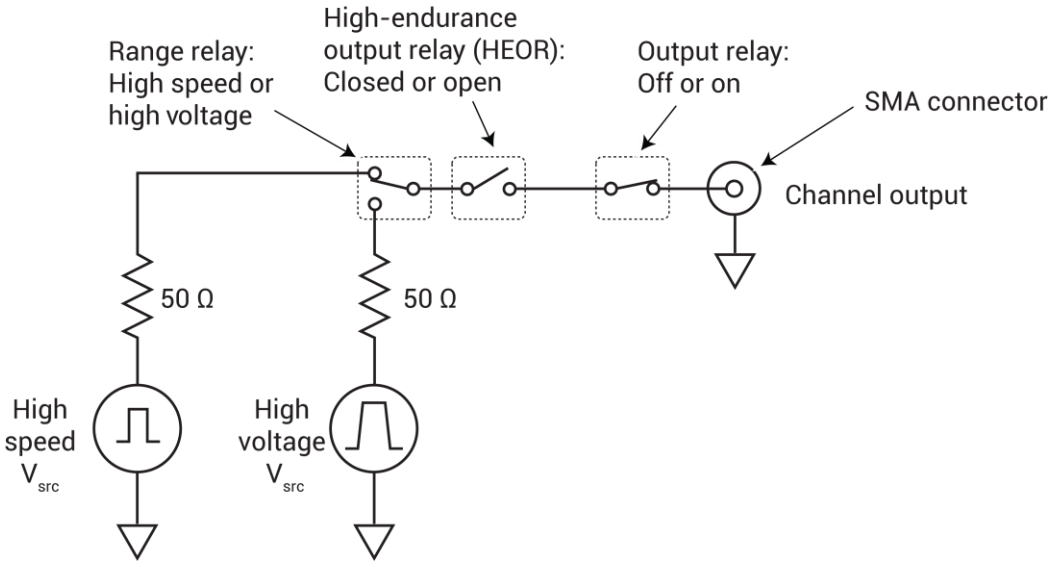
Pulse amplitude can be set as high as 80 V depending on the pulse high and low levels, pulse output range, and DUT load settings.

Refer to Pulse measure and pulse generator units (on page 5-1) for details on pulse card connectors and connections to the DUT.

The next figure shows a simplified schematic of the 4220-PGU pulse card single channel output. The range relay chooses between the high-speed and high-voltage output ranges. The schematic for the 4225-PMU is similar except it also includes measure circuitry for both current and voltage.

The HEOR provides fast, unlimited, open and close cycles for demanding tests such as flash memory endurance. See Segment Arb waveforms (on page 11-7) for details about the typical use of the HEOR, which is a solid-state relay (SSR) for connecting or disconnecting a pulse channel from a device terminal.

Figure 26: Simplified schematic of a 4220-PGU channel

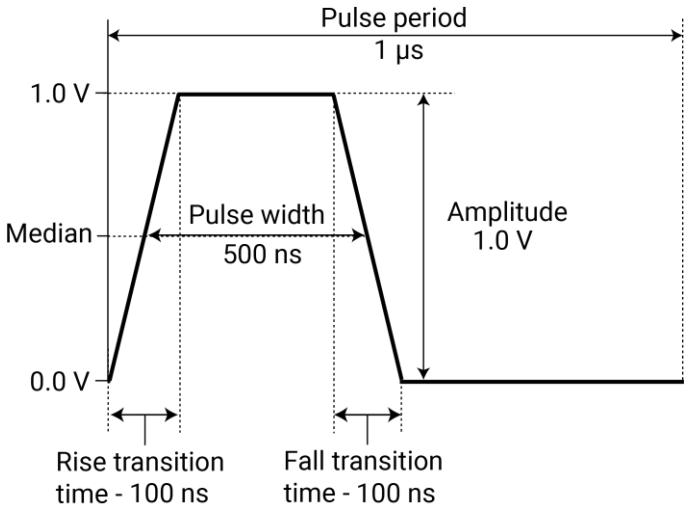


Standard pulse

Each channel of a pulse card can be configured for standard pulse output. The next figure shows an example of standard pulse output.

A pulse card is a dual-channel pulse generator. Each channel can output high speed (low voltage) or high voltage (medium speed) pulses. The basic pulse characteristics of the pulse card are listed in the data sheet.

Figure 27: Standard pulse example (pulse high = 1 V, pulse low = 0 V)



Configuring the system

You do not need to perform system configuration operations if you use only internal instruments, such as factory-installed SMUs, preamplifiers, and the Ground Unit (GNDU). The 4200A-SCS detects internal instruments and configures the system appropriately for local operation.

However, if you add supported external instruments such as switch matrices, the 4200A-CVIV Multi-Switch, external GPIB instruments, and probe stations, you must configure the system so that Clarius and KXCI can use these resources. Also, if you need remote operation of the 4200A-SCS through KXCI, you must further configure the system.

Perform these configurations using the Keithley Configuration Utility (KCon). Start the tool by selecting the KCon icon on your desktop.

Figure 28: KCon desktop icon

See Keithley Configuration Utility (KCon) (on page 7-1) for details on using KCon.

NOTE

If KCon is running, you cannot start Clarius or KXCI. If Clarius or KXCI is already running, you can start KCon but cannot save any system configuration changes that you may make.

Section 3

Source-measure hardware

In this section:

Source-measure units

This section provides information about the 4200-SMU, 4201-SMU, 4210-SMU, and 4211-SMU source-measure units and the related instruments, including:

- 420x-SMU and 421x-SMU overview (on page 3-1): Discusses 420x-SMU and 421x-SMU basic circuit configurations, operating boundaries, and connectors.
- Source-measure unit (SMU) with 4200-PA overview (on page 3-16): Details how the 4200-PA extends 420x-SMU and 421x-SMU dynamic range, and describes basic circuit configurations, operating boundaries, connectors, and mounting methods.
- Ground unit (GNDU) overview (on page 3-25): Provides basic information about using the ground unit.
- Source-measure concepts (on page 3-31): Provides information on guarding, remote sensing, sink operation, and sweep concepts.

4200-SMU and 4210-SMU overview

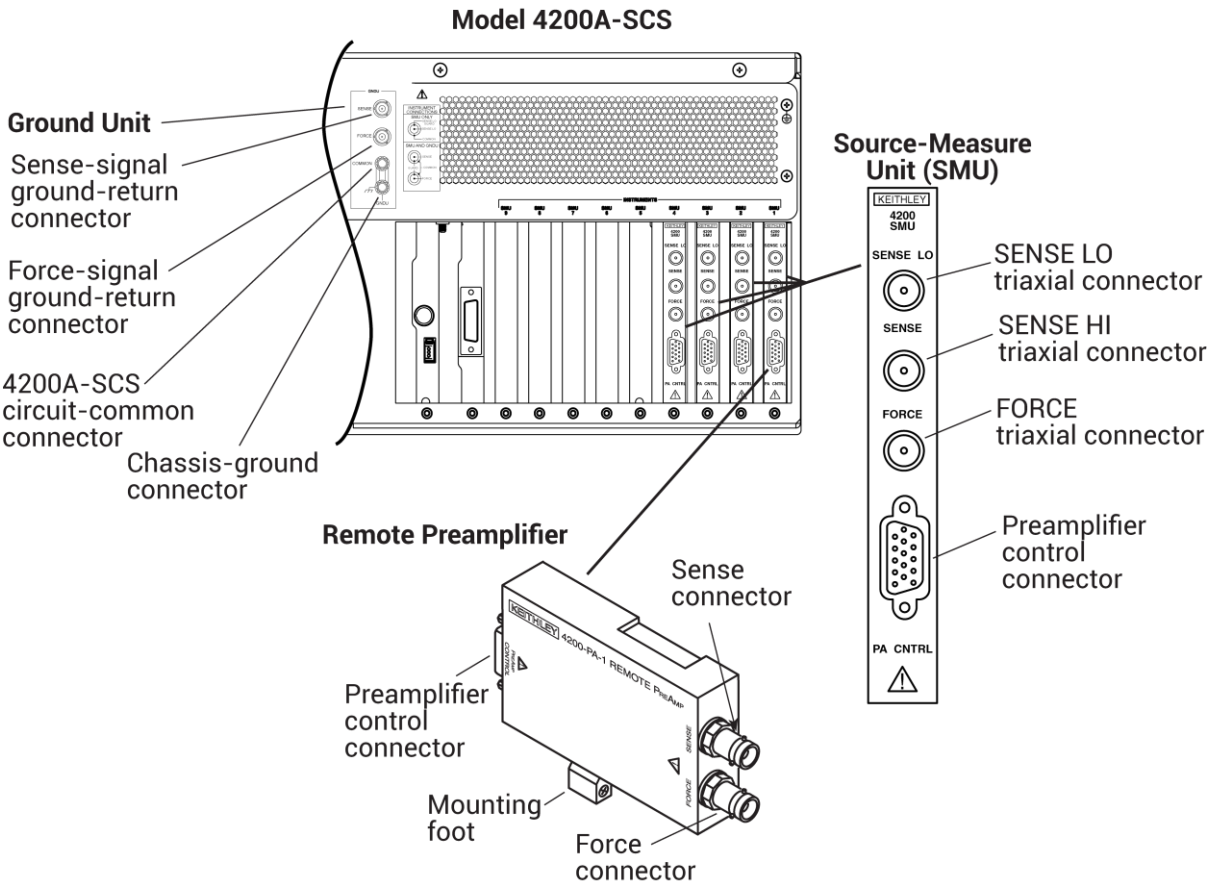
The following topics discuss these aspects of the 420x-SMU and 421x-SMU, with and without a 4200-PA:

- Basic SMU configuration
- Compliance limits
- Operating boundaries
- Terminals and connectors

Source-measure hardware overview

The 4200A-SCS mainframe accepts up to eight SMUs, each of which may be used with or without preamplifiers. Four of the SMUs may be high-power 421x-SMU models.

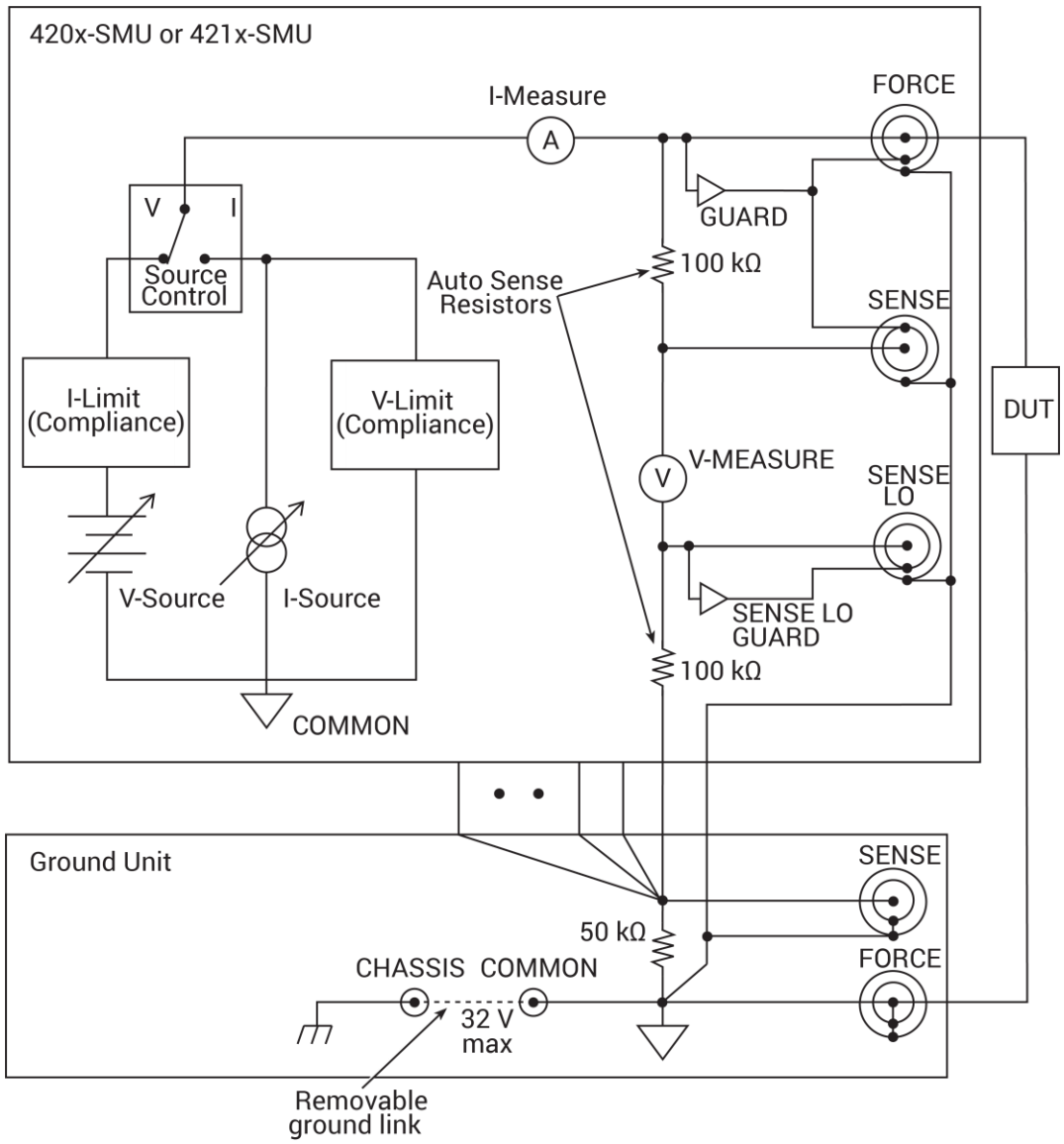
Figure 29: Source-measure hardware overview



Basic SMU circuit configuration

The basic SMU circuit configuration is shown in the next figure. The SMU is essentially a voltage or current source in series with a current meter, connected in parallel with a voltage 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 (on page 2-1) for more detailed information.

Figure 30: Basic SMU source-measure configuration



Compliance limits

You can set a limit that stops a SMU from sourcing a current or voltage that is more than that limit. This limit is called compliance and helps prevent damage to the device under test (DUT). The SMU will not exceed the maximum limit set for compliance. When a SMU is acting as a current source, the voltage is clamped at the compliance value; conversely, the current is clamped at the compliance value when the SMU is acting as a voltage source.

When a SMU reaches compliance, it continues to make measurements. However, the measurement stays at the value it was at when compliance occurred. For example, if you are sourcing voltage and the compliance is set to 100 mA, it continues to measure 100 mA after compliance is reached. The voltage, however, is not at the programmed value.

You can stop the test if the source reaches compliance. Refer to Compliance exit condition options (on page 6-117) for detail.

If you set a specific measurement range, compliance can also be restricted by the range. Compliance must be more than 11% of the measurement range. If not, an event is generated and the compliance setting is automatically changed to the maximum compliance value for the selected range. For example, if compliance is set to 1 V and the measurement range is 200 mV, output voltage will clamp at 210 mV. If you attempt to change compliance to a value that is not appropriate for the selected range, compliance is not changed and a warning is generated. You must change the range before you can select the new compliance value. If you set the measurement range to be automatically selected, the measurement range does not affect compliance.

The lowest allowable compliance is based on the load and the source value. For example, if you are sourcing 1 V to a 1 kΩ resistor, the lowest allowable current compliance is 1 mA (1 V/1 kΩ = 1 mA). Setting a compliance lower than 1 mA limits the source.

For another example, assume the following conditions:

- Current compliance: 10 mA
- Voltage sourced by the instrument: 10 V
- DUT resistance: 10 Ω

With a source voltage of 10 V and a DUT resistance of 10 Ω, the current through the DUT should be 10 V / 10 Ω = 1 A. However, because compliance is set to 10 mA, the current cannot exceed 10 mA, and the voltage across the resistance is limited to 100 mV. In effect, the 10 V voltage source is transformed into a 10 mA current source.

In steady-state conditions, the set compliance value restricts the instrument output unless there are fast transient load conditions.

When measurement autorange is disabled, the maximum and minimum compliance values cannot be set below the minimum value. When autorange is enabled, the programmed compliance value cannot be set below 10 nA when sourcing voltage, or below 20 mV when sourcing current.

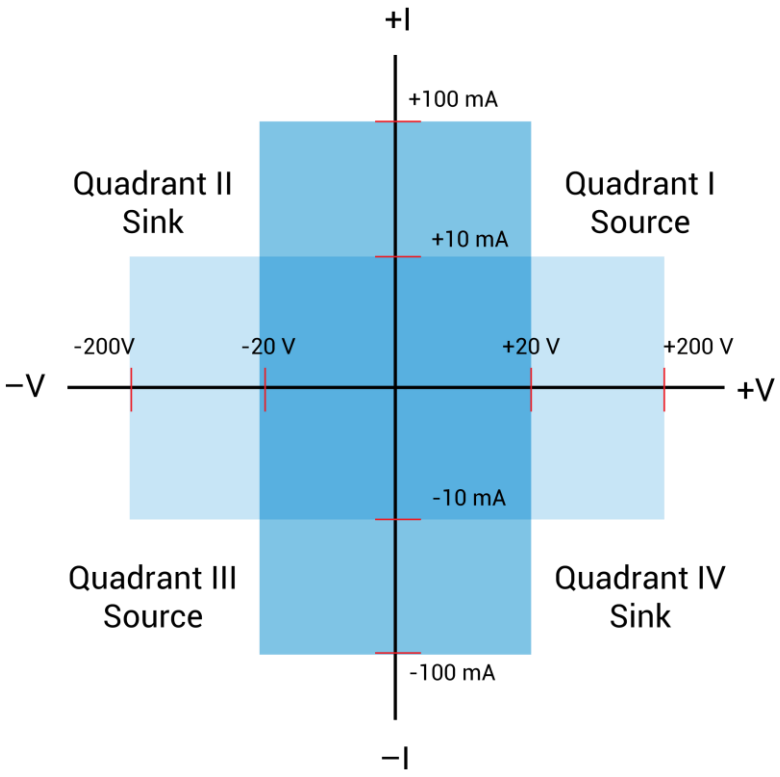
Source or sink

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 420x-SMU and 421x-SMU are shown in the next figures. 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.

4200-SMU source or sink

In the general operating boundaries in the next figure, the 100 mA, 20 V and 10 mA, 200 V magnitudes are nominal values. The actual maximum output magnitudes of the 420x-SMU are 105 mA, 21 V and 10.5 mA, 210 V. Note that the boundaries are not drawn to scale.

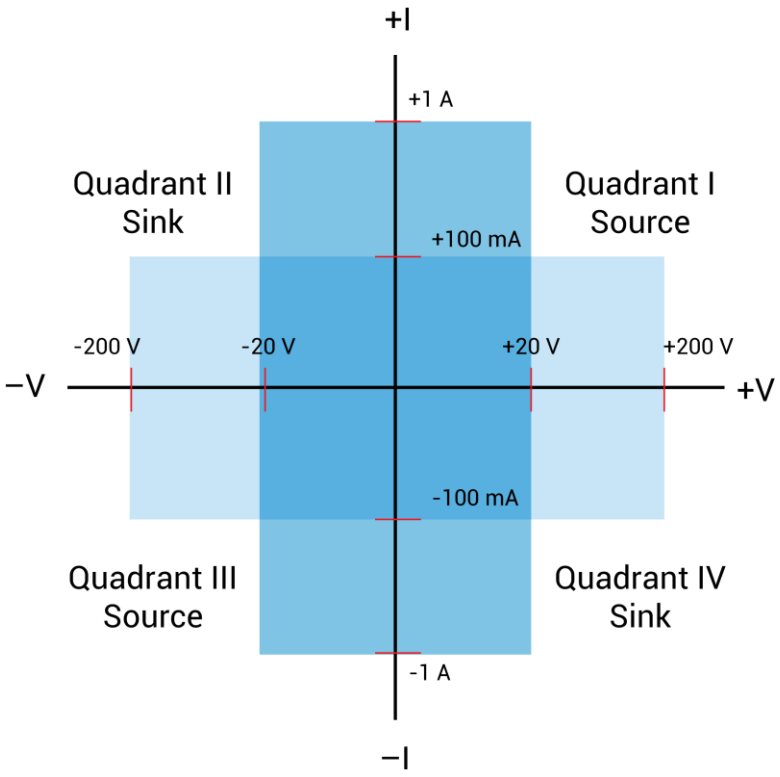
Figure 31: 420x-SMU and 4200-PA operating boundaries



4210-SMU source or sink

In the next figure, the 1A, 20 V and 100 mA, 200 V magnitudes are nominal values. The actual maximum output magnitudes of the 421x-SMU are 1.05 A, 21 V and 105 mA, 210 V. The boundaries are not drawn to scale.

Figure 32: 421x-SMU and 4200-PA operating boundaries



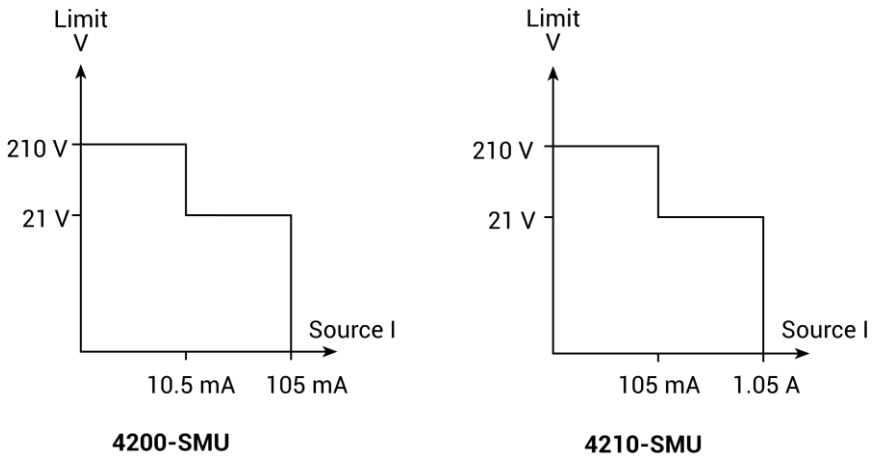
I-Source operating boundaries

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.

The next two figures show the operating boundaries for the I-Source. Only the first quadrant of operation is covered; operation in the other three quadrants is similar.

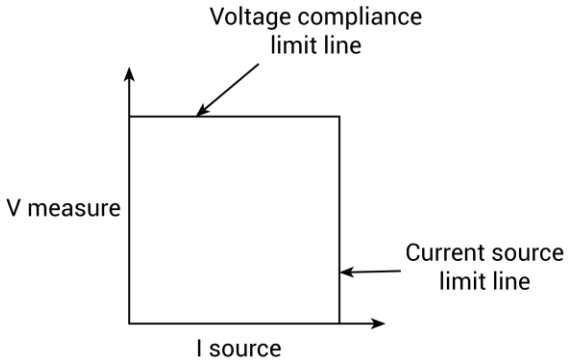
As shown in the next figure, the 420x-SMU can output up to 105 mA at 21 V, or 10.5 mA at 210 V. The 421x-SMU can output up to 1.05 A at 21 V, or 105 mA at 210 V.

Figure 33: 420x-SMU and 421x-SMU I-source output characteristics



The next figure 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.

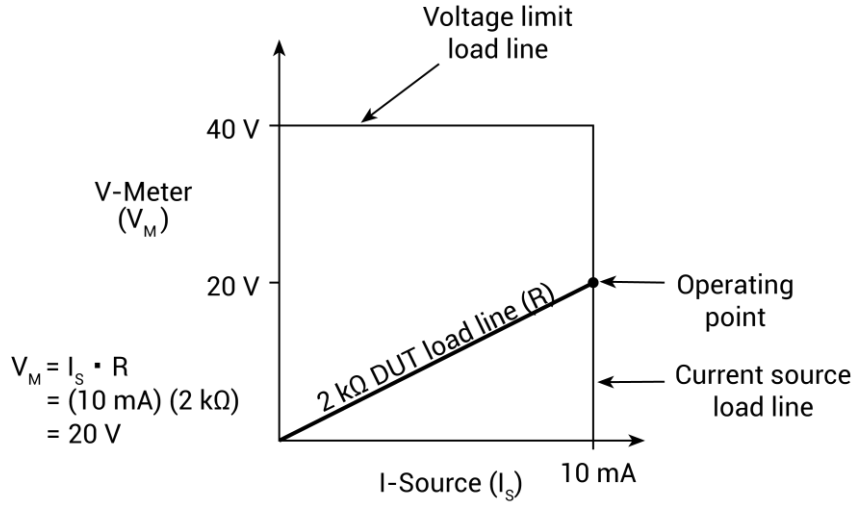
Figure 34: 420x-SMU and 421x-SMU I-source limit lines



I-Source operation examples

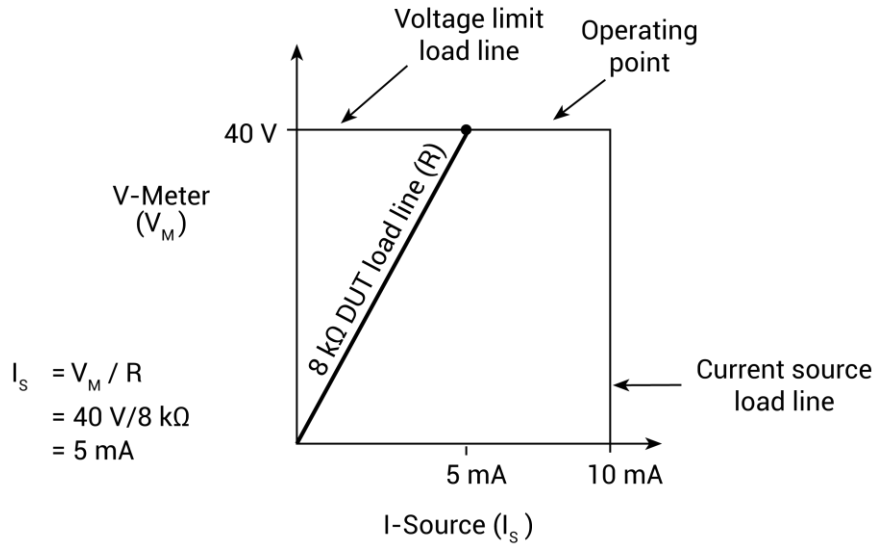
The next figures show operation examples for resistive loads that are 2 kΩ and 8 kΩ, respectively. For these examples, the SMU is programmed to source 10 mA and limit (compliance) 40 V. The SMU is sourcing 10 mA to the 2 kΩ load, and subsequently measures 20 V. As shown, the load line for 2 kΩ intersects the 10 mA current source line at 20 V, which is below the programmed voltage limit.

Figure 35: Current source normal operation



The following figure shows what happens if the resistance of the load is increased to 8 kΩ. The DUT load line for 8 kΩ intersects the 40 V voltage compliance limit line, placing the SMU in compliance. In compliance, the SMU cannot source its programmed current (10 mA). For the 8 kΩ DUT, the SMU only outputs 5 mA (at the 40 V limit).

Figure 36: Current source in compliance



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. However, 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.

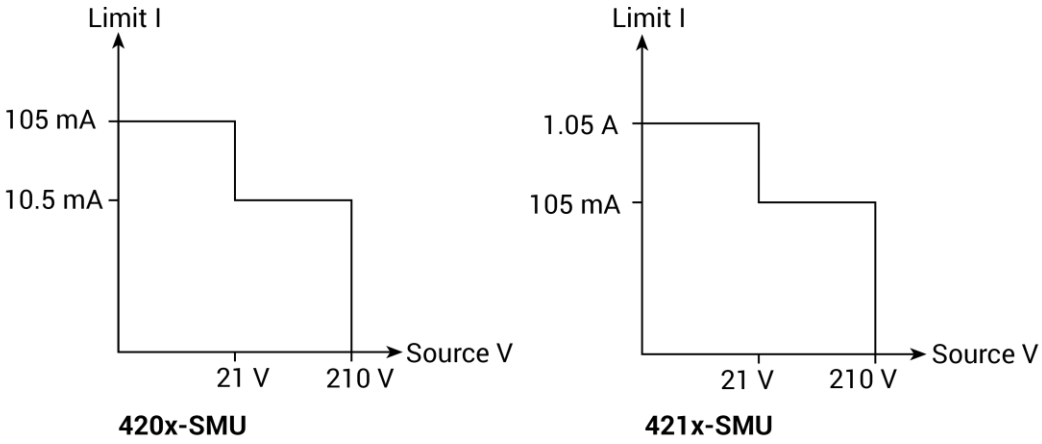
Regardless of the load, voltage will never exceed the programmed compliance of 40 V.

V-Source operating boundaries

The next two figures show the operating boundaries for the voltage source. Only the first quadrant of operation is covered; operation in the other three quadrants is similar.

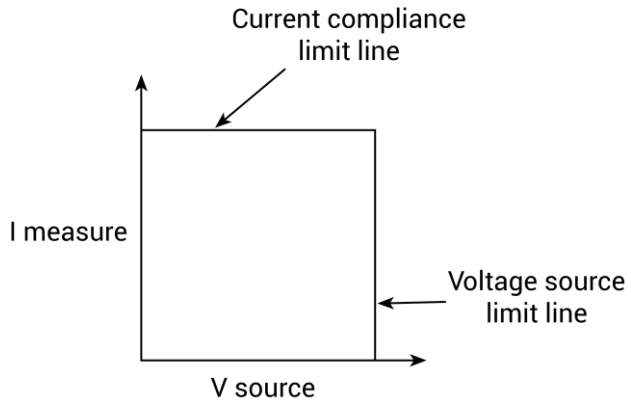
As shown in the next figure, the 420x-SMU can output up to 21 V at 105 mA, or 210 V at 10.5 mA. The 421x-SMU can output up to 21 V at 1.05 mA, or 210 V at 1.5 mA.

Figure 37: V-Source operating boundaries



The next figure shows the limit lines for the voltage 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 38: 420x-SMU and 421x-SMU V-Source limit lines

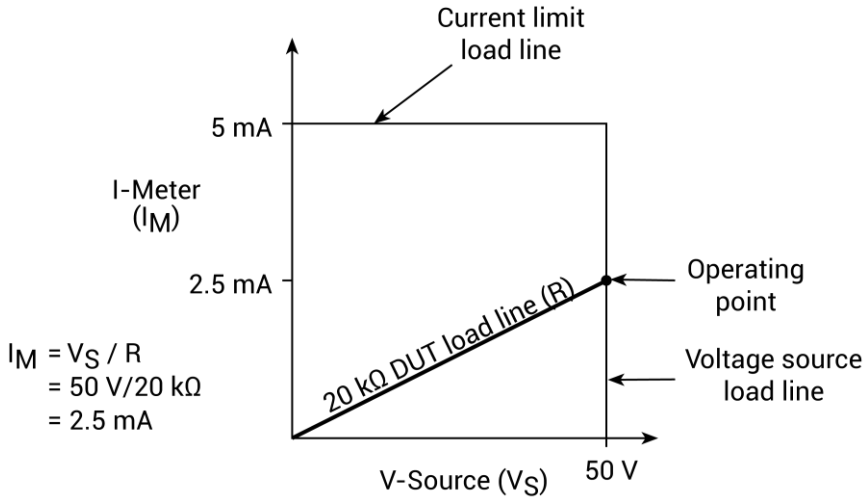


V-Source operation examples

The next figures show operation examples for resistive loads that are 20 kΩ and 8 kΩ, respectively. For these examples, the SMU is programmed to source 50 V and limit 5 mA. The SMU is sourcing 50 V to the 20 kΩ load and subsequently measures 2.5 mA, which is within the 5 mA programmed current limit.

As shown in the following figure, the load line for 20 kΩ intersects the 50 V voltage source line at 2.5 mA.

Figure 39: Normal voltage source operation

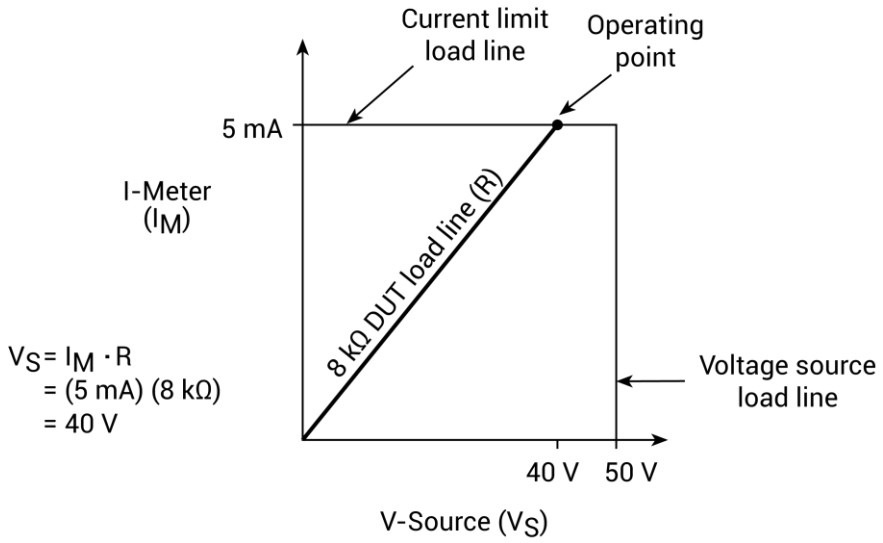


The next figure shows what happens if the resistance of the load is decreased to 8 kΩ. The DUT load line for 8 kΩ intersects the current compliance limit line, which places the SMU in compliance. In compliance, the SMU cannot source its programmed voltage (50 V). For the 8 kΩ DUT, the SMU will only output 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 about 50 V at 0 mA. However, 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 never exceeds the programmed compliance of 5 mA.

Figure 40: Voltage source in compliance



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. However, 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 420x-SMU and 421x-SMU terminals are shown in the next figure. Basic information about these terminals is summarized below. Refer to the Basic source-measure connections (on page 2-1) topic for additional information regarding SMU signal connections.

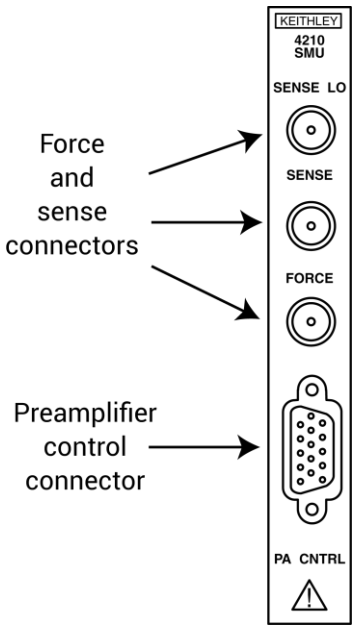
WARNING

Exposure to electrical shock could result in personal injury or death, therefore, precautions must be taken to prevent a shock hazard by surrounding the test device and any unprotected leads (wiring) with double insulation for 250 V, Category O. See IEC 61010-1 safety standards for details. When connecting to the 4200A-SCS SMU outputs, make sure to use devices and cables that have ratings for the sourced voltages. Otherwise, they will not properly insulate the external connections to the instrument and pose a shock hazard. SMU and preamplifier terminals should be considered hazardous even if the outputs are programmed to be low voltage.

CAUTION

The maximum allowed voltage between COMMON and chassis ground is ±32 VDC.

Figure 41: 420x-SMU and 421x-SMU connectors



FORCE terminal

The FORCE terminal is a miniature triaxial connector used to apply the SMU FORCE signal to the DUT when a preamplifier 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 preamplifier 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 a SMU. If it is necessary to use the SENSE LO of a SMU, the SENSE LO terminals of all SMUs being used in that 4200A-SCS should be connected to the DUT.

Preamplifier control connector

The preamplifier control (PA CNTRL) terminal is a 15-pin D-subminiature connector that provides both power and signal connections to the 4200-PA Remote Preamplifier. The preamplifier can either be mounted and connected directly to the SMU, or it can be connected to the SMU through a cable (4200-RPC-X) when mounted remotely. Refer to the Source-measure unit (SMU) with 4200-PA overview (on page 3-16) for more information on the preamplifier.

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

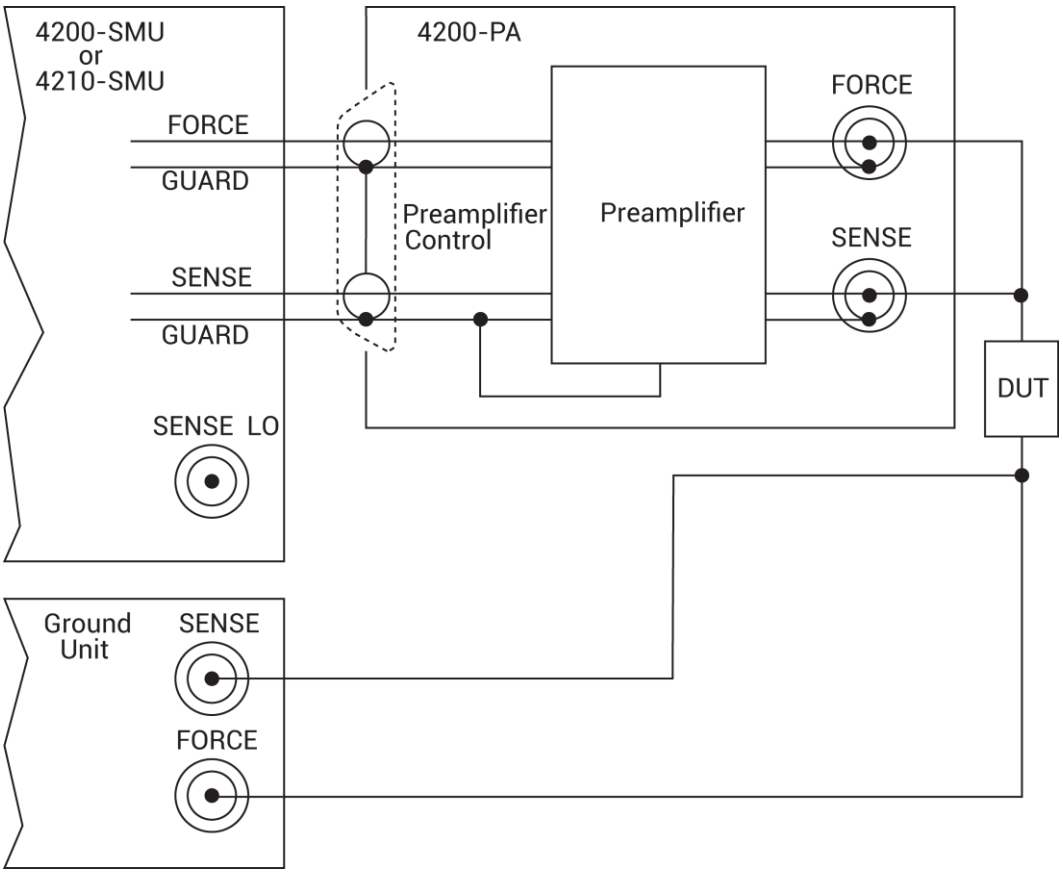
The following paragraphs discuss these aspects of the 4200-PA remote preamplifier:

- Basic circuit configuration
- Compliance limit
- Operating boundaries
- Connectors
- Preamplifier mounting

Basic SMU/preamplifier circuit configuration

The basic SMU and preamplifier circuit configuration is shown in the next figure. This configuration is similar to the SMU configuration discussed earlier, with the exception of the preamplifier, which adds low-current source-measure capabilities. Preamplifier FORCE terminal is connected to DUT HI, while DUT LO is connected to COMMON. See Basic source-measure connections (on page 2-1) in the *4200A-SCS User's Manual* for more detailed information.

Figure 42: Basic SMU and preamplifier source-measure configuration



Compliance limit for a SMU with a 4200-PA

A current limit can be programmed for a SMU with a 4200-PA when it is sourcing voltage. However, a voltage limit can be programmed when sourcing current. The compliance limit characteristics are the same for a SMU with a 4200-PA as for a SMU alone. Refer to Compliance limits (on page 3-4).

Using minimum compliance

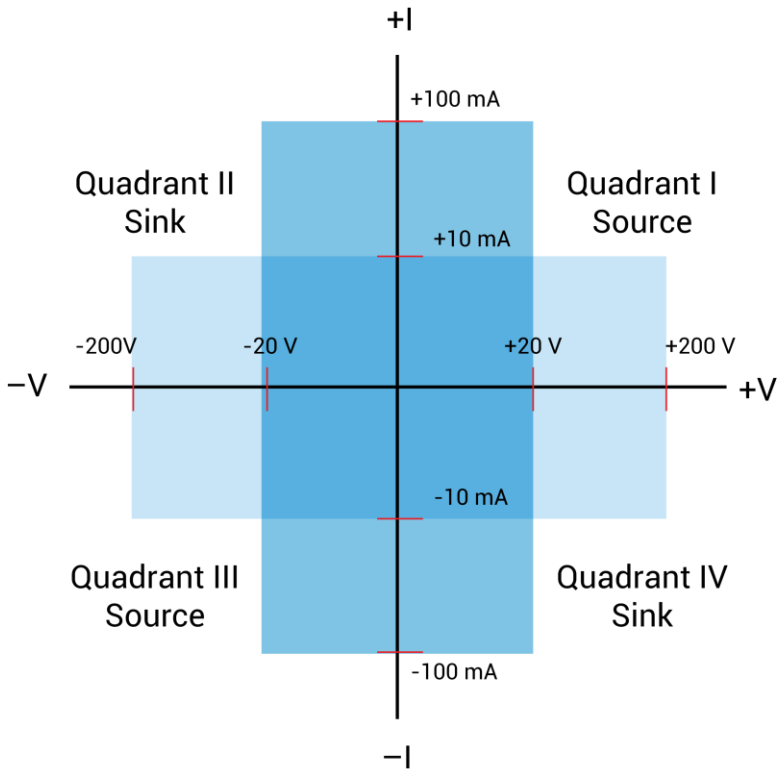
The minimum compliance value is particularly applicable when measurement autorange is disabled. When measurement autorange is disabled, the compliance value cannot be set below the minimum value that is specified in the previous tables indicating 4200-PA compliance limits. When autorange is enabled, the programmed compliance value cannot be set below 100 fA when sourcing voltage, or below 20 mV when sourcing current.

Operating boundaries

As with the SMUs alone, adding a 4200-PA preamplifier also allows operation in any of the four quadrants. The four quadrants of operation for the 4200-PA with the 4200-SMU and 4210-SMU are shown in the following figures.

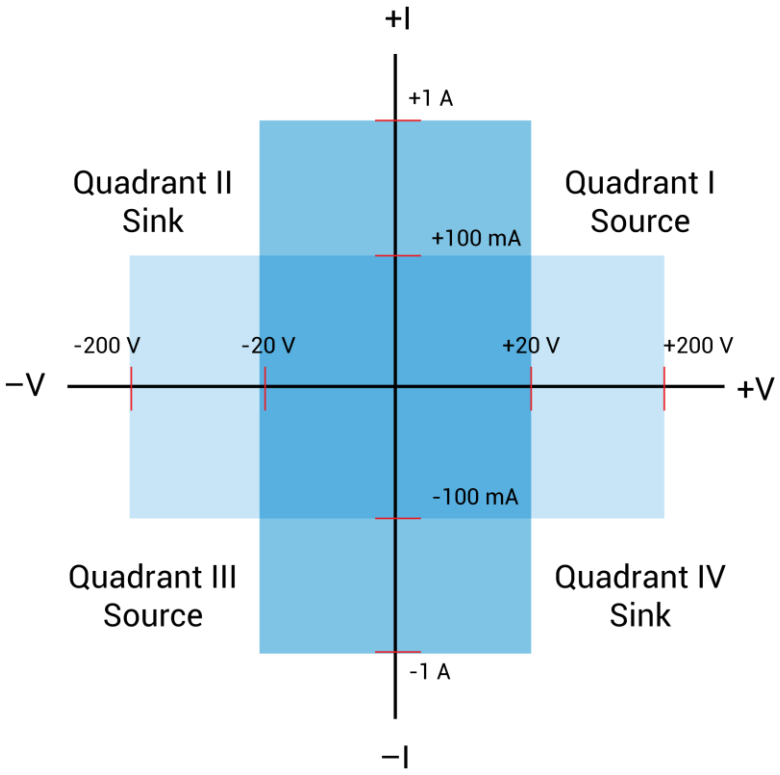
420x-SMU with 4200-PA: In the general operating boundaries in the next figure, the 100 mA, 20 V and 10 mA, 200 V magnitudes are nominal values. The actual maximum output magnitudes of the 4200-SMU and 4200-PA are 105 mA, 21 V and 10.5 mA, 210 V. Note that the boundaries are not drawn to scale.

Figure 43: 420x-SMU and 4200-PA operating boundaries



421x-SMU with 4200-PA: In the next figure, the 1 A, 20 V and 100 mA, 200 V magnitudes are nominal values. The actual maximum output magnitudes of the 421x-SMU and 4200-PA are 1.05 A, 21 V and 105 mA, 210 V. Note that the boundaries are not drawn to scale.

Figure 44: 421x-SMU and 4200-PA operating boundaries



Preamplifier terminals and connectors

The locations and configuration of the 4200-PA terminals are shown in the next figure. Basic information about these terminals is summarized below. Refer to the Basic source-measure connections (on page 2-1) for additional information regarding making preamplifier signal connections.

WARNING

Exposure to electrical shock could result in personal injury or death, therefore, precautions must be taken to prevent a shock hazard by surrounding the test device and any unprotected leads (wiring) with double insulation for 250 V, Category O. See IEC 61010-1 safety standards for details. When connecting to the 4200A-SCS SMU outputs, make sure to use devices and cables that have ratings for the sourced voltages. Otherwise, they will not properly insulate the external connections to the instrument and pose a shock hazard. SMU and preamplifier terminals should be considered hazardous even if the outputs are programmed to be low voltage.

CAUTION

Turn off the system and disconnect the power cord before connecting or disconnecting the preamplifier. Failure to do so may result in SMU or preamplifier damage, possibly voiding the warranty.

CAUTION

The maximum allowed voltages between the various preamplifier signals are as follows:
COMMON to chassis ground: 32 V_{peak}
GUARD to COMMON: 250 V_{peak}
SENSE or FORCE to GUARD: 40 V_{peak}

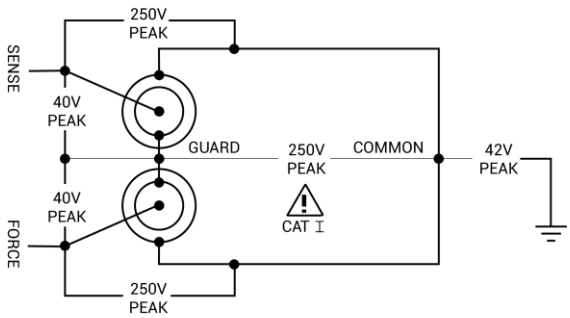
FORCE terminal

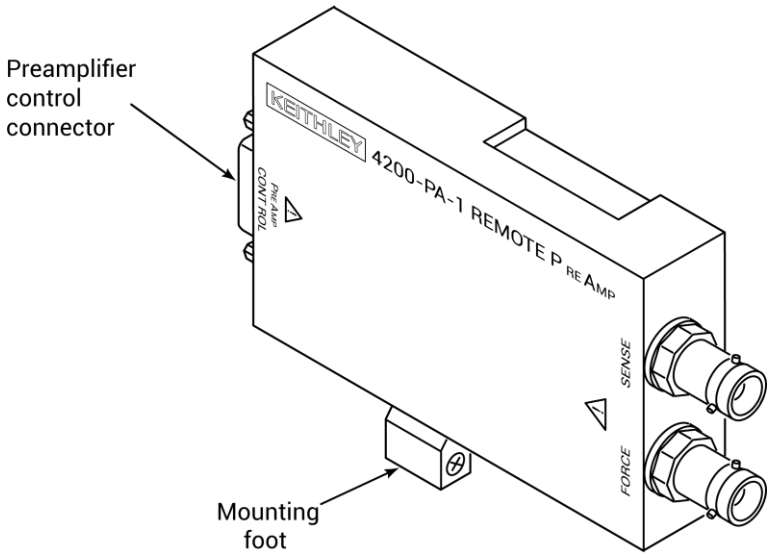
The FORCE terminal is a standard triaxial connector used as a return path for the SMU or preamplifier FORCE current. The center pin is FORCE, the inner shield is GUARD, and the outer shield is circuit COMMON.

NOTE

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

Figure 45: 4200-PA connectors





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/preamplifier measurements will be made relative to this DUT connection.

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.

Preamplifier CONTROL connector

The preamplifier CONTROL connector connects to the SMU PA CNTRL connector and provides both power and signal connections from the 420x-SMU or 421x-SMU to the 4200-PA preamplifier.

Preamplifier mounting

WARNING

Exposure to electrical shock could result in personal injury or death, therefore, precautions must be taken to prevent a shock hazard by surrounding the test device and any unprotected leads (wiring) with double insulation for 250 V, Category O. See IEC 61010-1 safety standards for details. When connecting to the 4200A-SCS SMU outputs, make sure to use devices and cables that have ratings for the sourced voltages. Otherwise, they will not properly insulate the external connections to the instrument and pose a shock hazard. SMU and preamplifier terminals should be considered hazardous even if the outputs are programmed to be low voltage.

CAUTION

Turn off the system and disconnect the power cord before connecting or disconnecting the preamplifier. Failure to do so may result in SMU or preamplifier damage, possibly voiding the warranty.

NOTE

As shipped, any 4200-PA units ordered with the 4200A-SCS are factory-mounted on the rear panel. Do not remove the preamplifiers from the 4200A-SCS unless they are to be mounted at a remote site.

The preamplifier may either be mounted directly to the 420x-SMU or 421x-SMU on the 4200A-SCS rear panel, or mounted and connected remotely.

Remote preamplifier mounting

The 4200-PA can be mounted remotely using one of the optional mounting kits. Follow the steps below to remotely mount and connect the preamplifier. Refer to the instructions provided with the particular remote mounting kit for detailed information on physically mounting the preamplifier module.

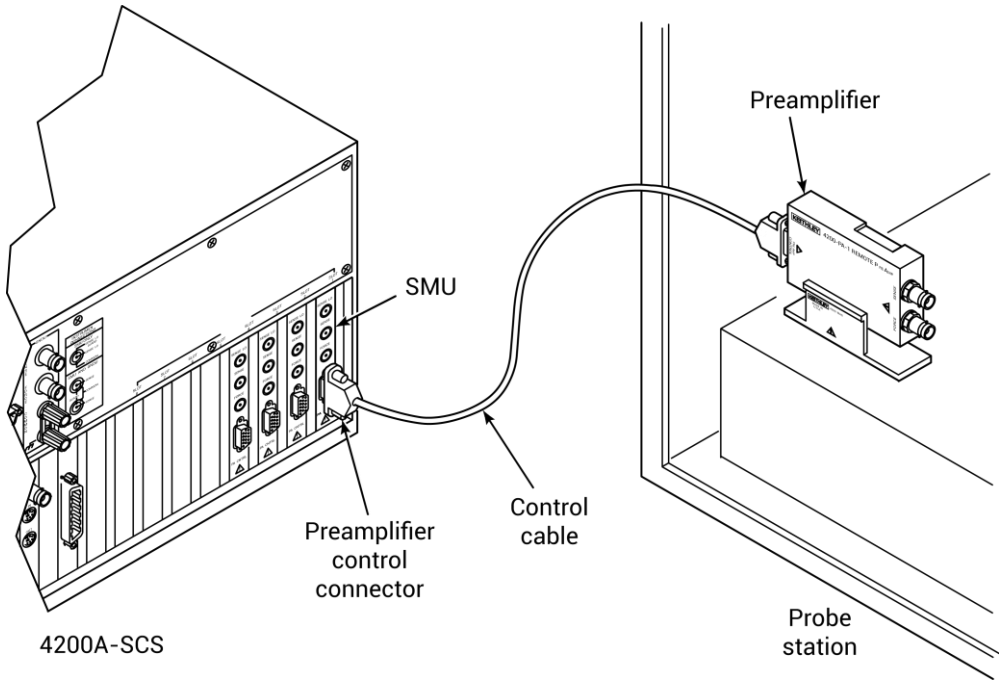
1. Ensure system power is turned off.
2. Disconnect the line cord.
3. Mount the preamplifier at the remote location using the appropriate mounting kit.
4. Connect the control/power cable between the preamplifier control connector on the preamplifier and the PA CNTRL connector on the SMU as shown in the next figure.

NOTE

The preamplifiers are matched to the SMUs that they were originally connected to. Ensure that each preamplifier is connected to its original SMU.

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

Figure 46: Typical preamplifier remote mounting



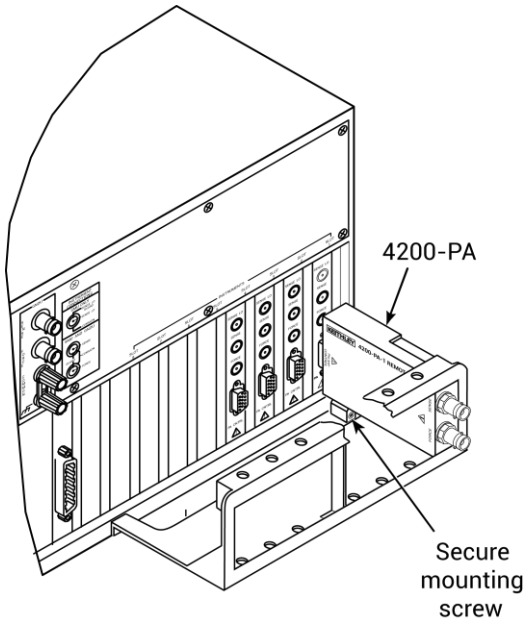
Rear panel mounting

A mounting foot secures the preamplifier to the rear panel. Also, a mounting bracket provides extra support for all the preamplifiers as shown in the next figure. If you remove the preamplifiers to mount them at a remote site, ensure that you install the screws in the chassis and retain the bracket for future use.

NOTE

The preamplifiers are matched to the SMUs that they were originally connected to. Ensure that each preamplifier is connected to its original SMU.

Figure 47: Preamplifier rear panel mounting



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. The next figure 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 Clarius (on page 6-1) for more detailed instructions on configuring a SMU to provide a COMMON connection.

Figure 48: Typical SMU connections

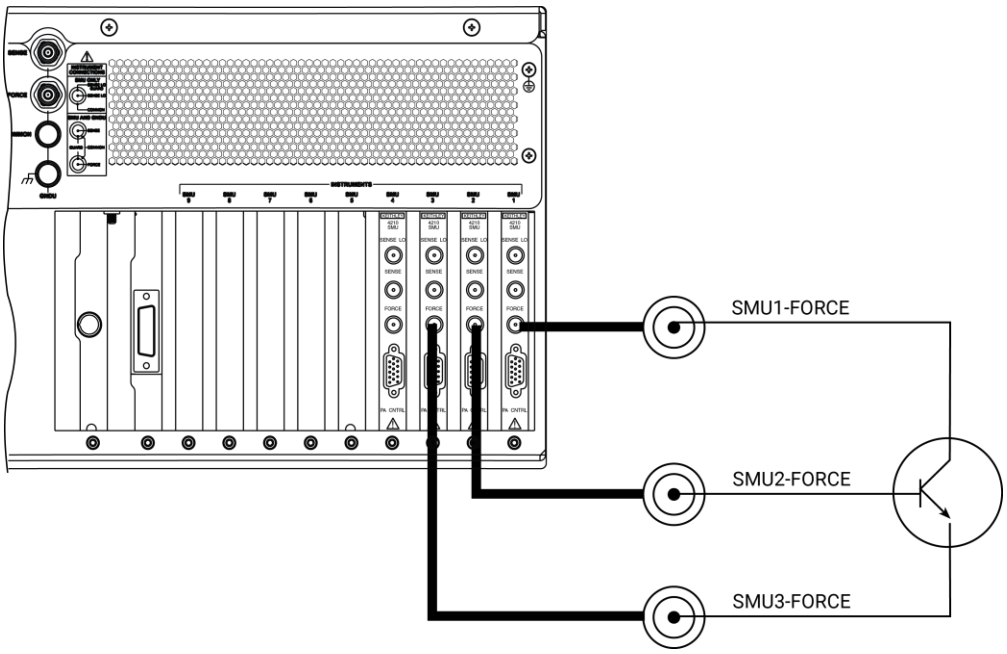
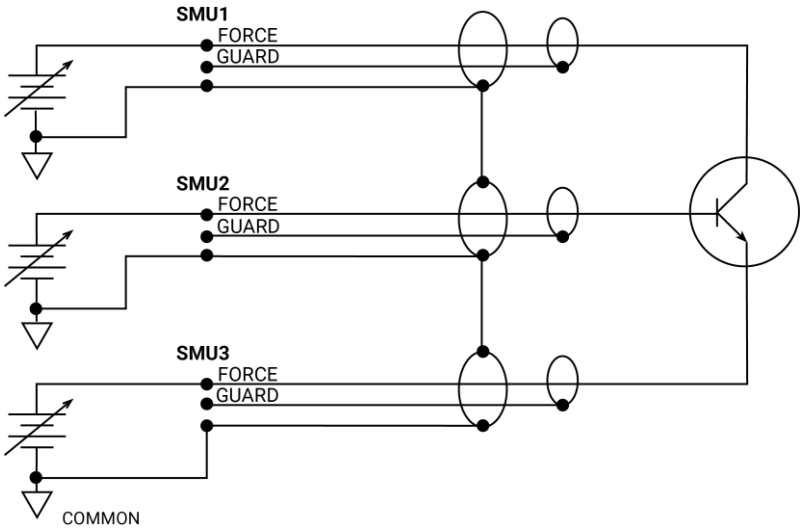


Figure 49: Typical SMU COMMON connections schematic



Ground unit (GNDU) overview

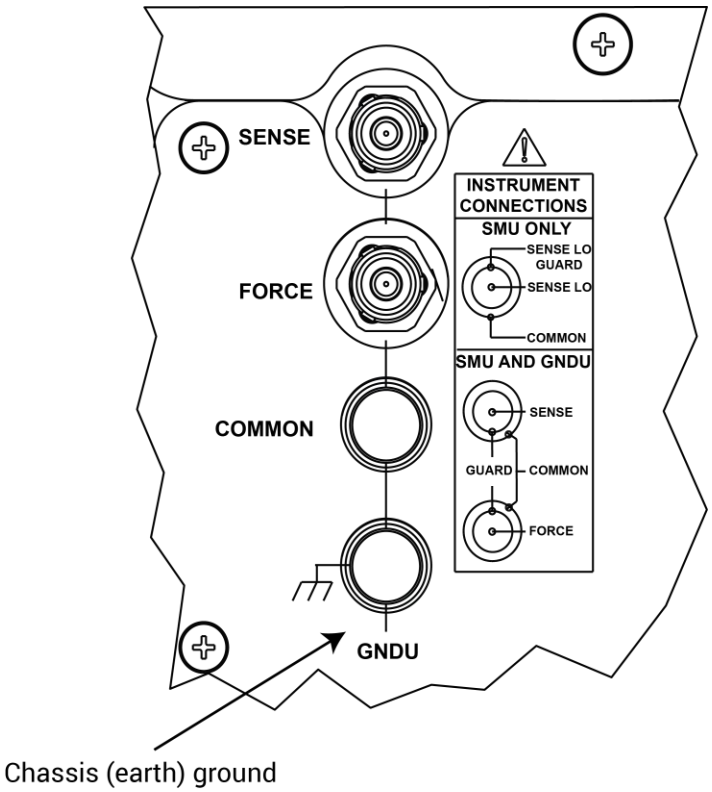
The following topics describe:

- Basic circuit configurations
- Connectors

Basic characteristics

The ground unit, shown in the next figure, provides convenient access to circuit COMMON, which is the measurement ground signal shared by all installed 4200A-SCS instrumentation. In addition, the GNDU SENSE terminal provides access to the SMU SENSE LO signals.

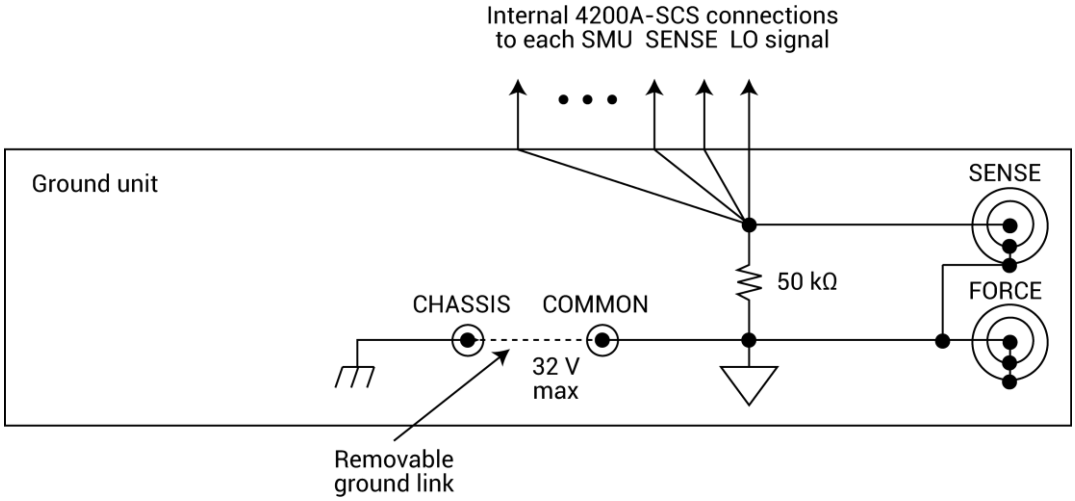
Figure 50: Ground unit (GNDU) connectors



Ground unit connections

The next figure 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 51: Ground unit connections



Ground unit DUT connections

The next figure shows the connections necessary to use the GNDU with a SMU to make full-kelvin remote sense measurements. Similarly, the following figure includes the preamplifier. As shown in these figures, the GNDU FORCE signal provides the return path for SMU or preamplifier FORCE current. See the Basic source-measure connections (on page 2-1) for detailed information on the ground unit, SMU, and preamplifier connections.

Figure 52: Full-Kelvin SMU - ground unit connections

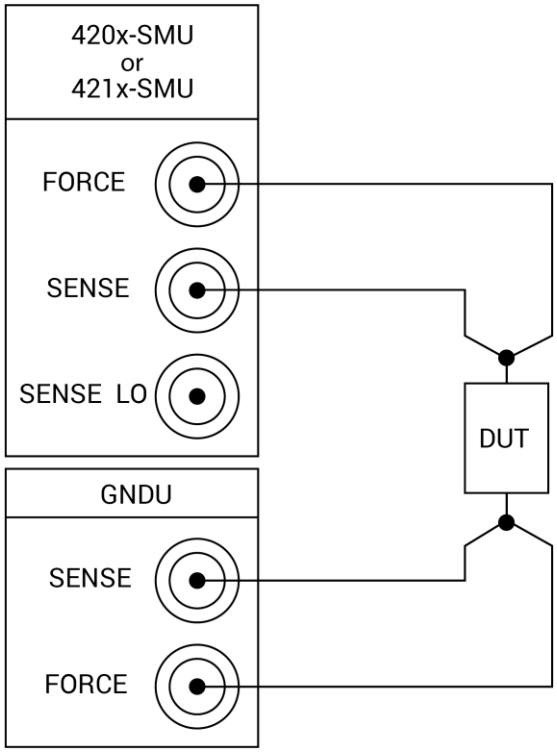
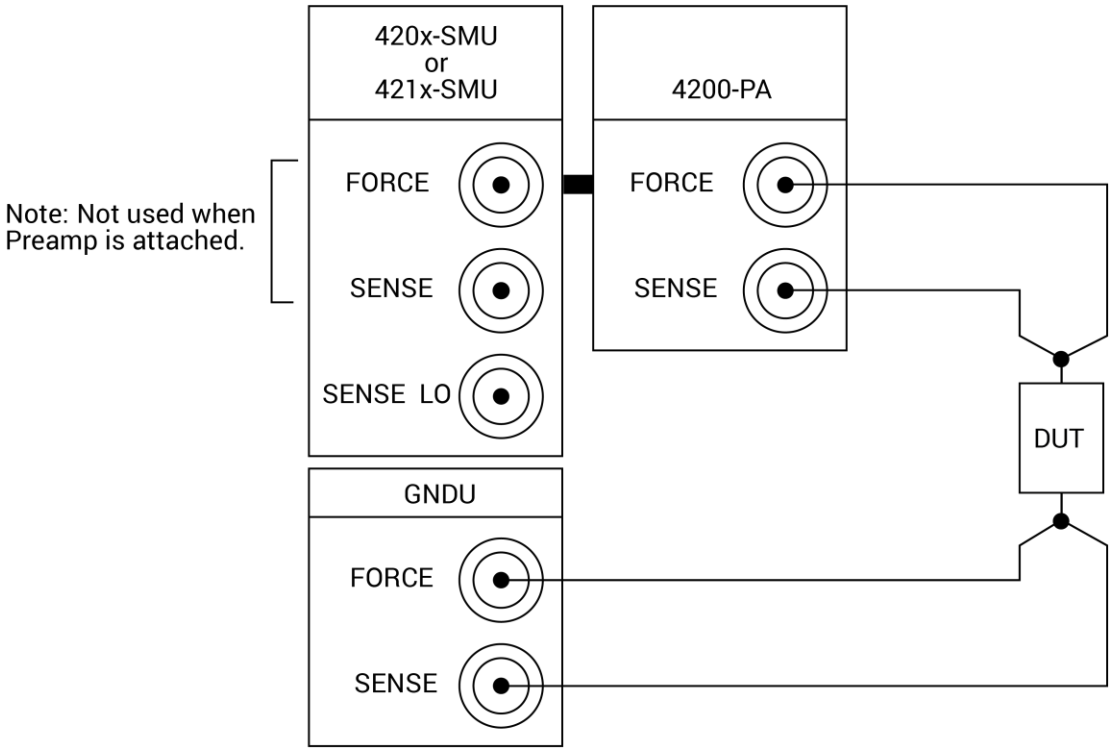


Figure 53: Full-Kelvin preamp - ground unit connections



Ground unit terminals and connectors

Refer to the Using the ground unit (on page 2-10) for the locations and configuration of the GNDU terminals. Basic information about these connectors is summarized below. Refer to the Basic source-measure connections (on page 2-1) for additional information regarding ground unit signal connections.

CAUTION

The maximum allowed voltage between circuit COMMON and chassis ground is ±32 VDC.

FORCE terminal

The FORCE terminal is a standard triaxial connector used as a return path for the SMU or preamplifier 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/preamplifier measurements will be made relative to this DUT connection.

COMMON terminal

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

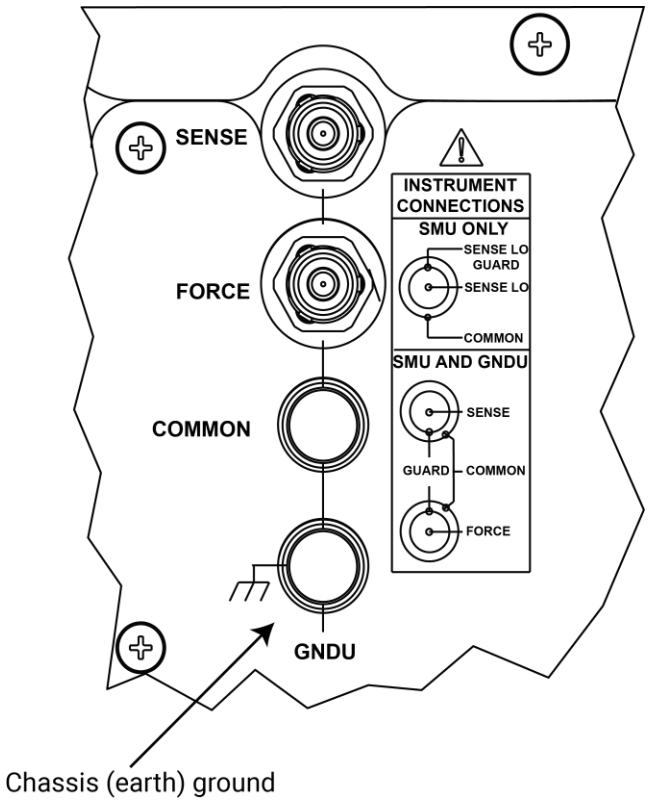
NOTE

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 Interference (on page 13-20) for details).

Chassis ground

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

Figure 54: Chassis ground



Source-measure concepts

- Guarding (on page 3-32): An overview of guarding, guarding concepts, guard connections, and test fixture guarding.
- Remote sensing (on page 3-39): Covers an overview of sensing, sensing concepts, and sense selection.
- Sink overview (on page 3-40): Provides an overview of sink operation and summarizes sink operating boundaries.
- Source-measure considerations (on page 3-42): Details various source-measure circuit configurations, including Source V, Measure V, and measure-only.
- Sweep concepts (on page 3-45): Covers the source-delay-measure cycle and provides an overview of sweep waveforms.

Guarding

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 preamplifier.

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

To avoid high voltage exposure that could result in personal injury or death, whenever the interlock of the 4200A-SCS is asserted, the FORCE and GUARD terminals of the SMUs and preamplifier should be considered to be at high voltage, even if they are programmed to a non-hazardous voltage current.

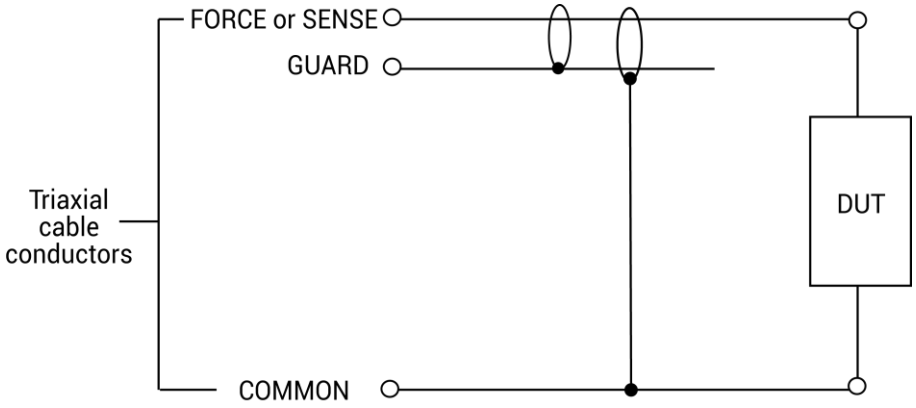
Guard connections

GUARD is available at the inner shield of the FORCE and SENSE triaxial connectors for both the SMU and the preamplifier, as shown in the next figure.

Figure 55: SMU and Preamplifier triaxial connectors

The next figure shows the triaxial cable connections to the device under test (DUT). Note that GUARD is not connected in this example, but it can be routed internally to a test fixture, as described in Test fixture guarding (on page 3-34).

Figure 56: GUARD connections

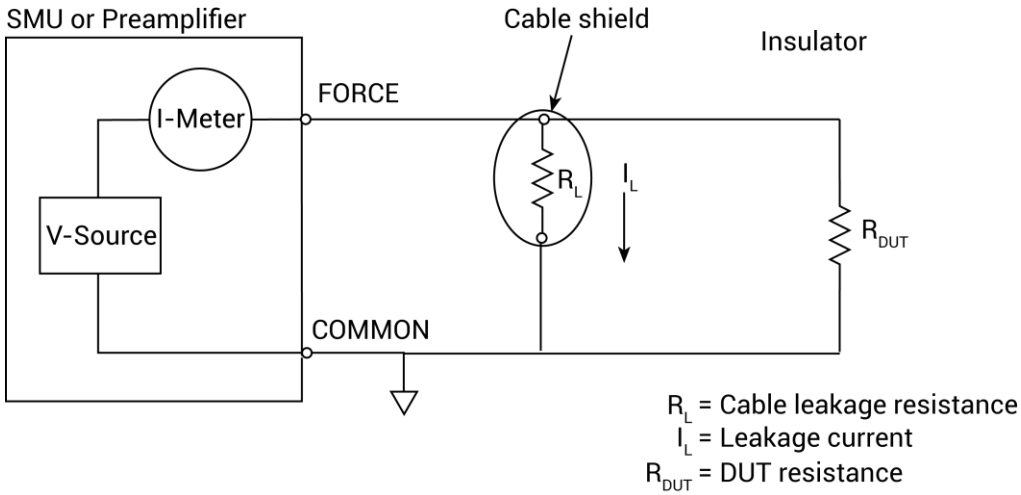


Guarding concepts

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

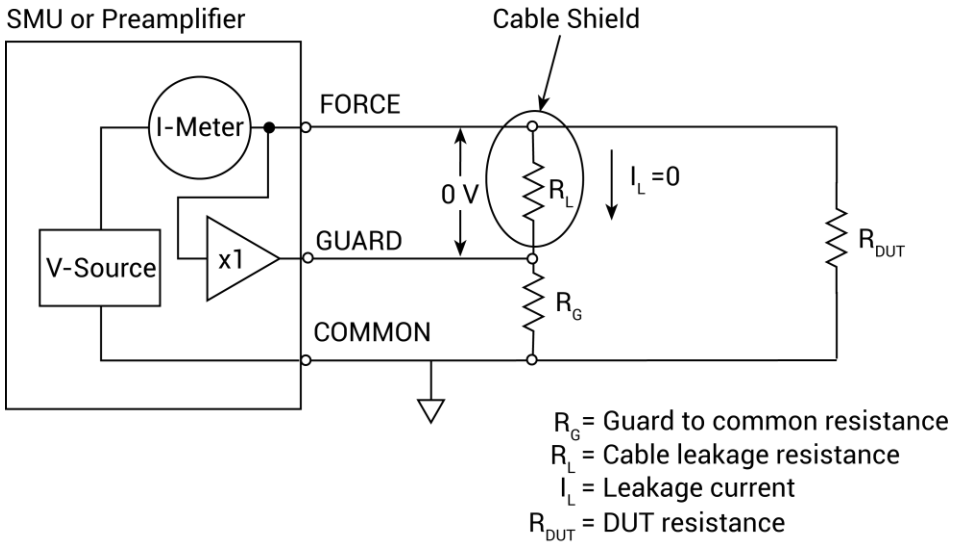
In the unguarded circuit of the following figure, the cable leakage resistance, R_L, is effectively in parallel with the DUT, creating an unwanted leakage current, I_L. This leakage current may seriously affect readings, particularly at low current levels.

Figure 57: Guarded circuit



In the guarded circuit of the next figure, 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 preamplifier.

Figure 58: Guarded circuit



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 triaxial 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 provide protection from shock hazards, an enclosure should be provided that surrounds all live parts.

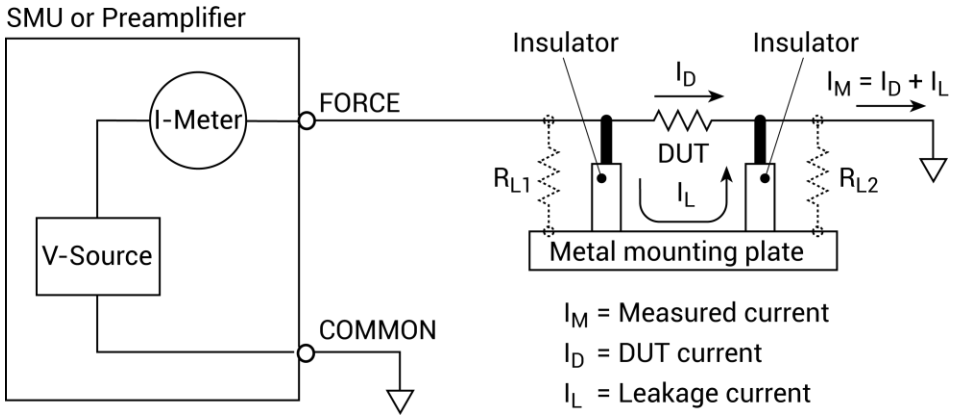
Nonconductive enclosures must be constructed of materials that are suitably rated for flammability and the voltage and temperature requirements of the test circuit. Connect the enclosure of all metal test fixtures to protective earth (safety ground). See your specific test fixture for information. Nonconductive test fixtures must be rated to double the maximum capability of the test equipment in the system.

For metallic enclosures, the test fixture chassis must be properly connected to protective earth (safety ground). A grounding wire (16 AWG or larger) must be attached securely to the test fixture at a screw terminal designed for safety grounding. The other end of the ground wire must be attached to a known protective earth (safety ground).

The following figures show how guard can eliminate leakage current through the insulators in a test fixture.

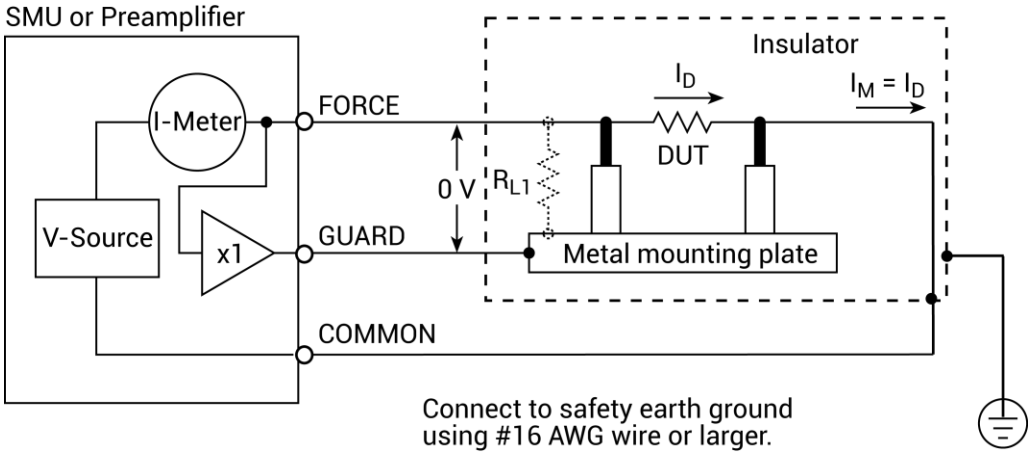
In the next figure, leakage current (I_L) flows through the insulators (R_{L1} and R_{L2}) to COMMON, adversely affecting the low-current (or high-resistance) measurement of the DUT.

Figure 59: Test fixture - unguarded



In the following figure, 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 preamplifier measures only the current through the DUT.

Figure 60: Test fixture - guarded



NOTE

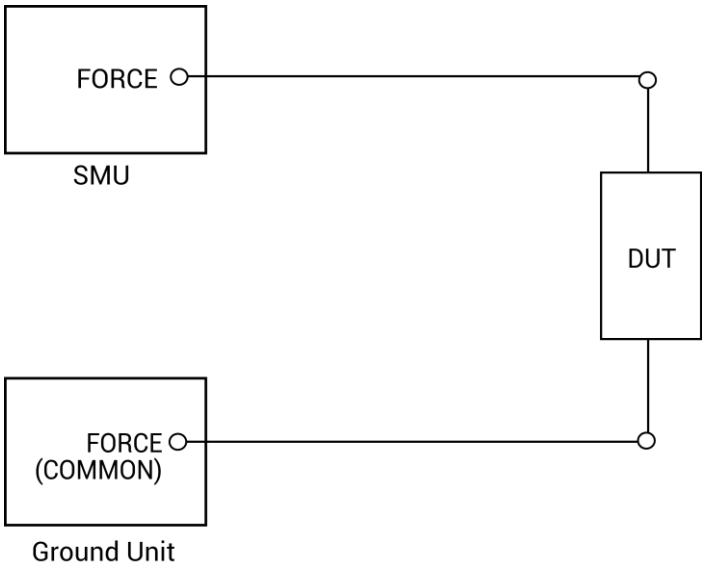
The guard signal has an output impedance of 100 kΩ. Therefore, it is effective only when connected to high-impedance loads.

Local and remote sensing

There are two types of sensing: local and remote.

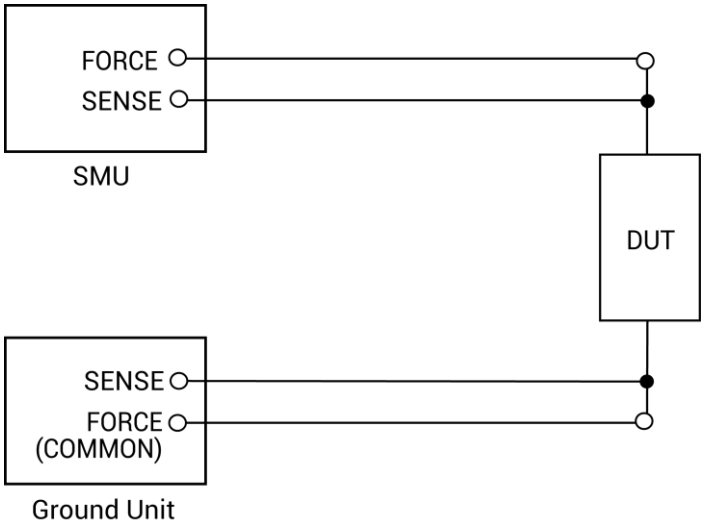
With local sensing, only two terminals are connected to the DUT: SMU FORCE and Ground Unit FORCE (COMMON), as shown in the following figure.

Figure 61: Local sensing



With remote sensing, both SENSE terminals are connected to the DUT, along with both FORCE terminals, as shown in the following figure.

Figure 62: Sensing overview



NOTE

See Basic source-measure connections (on page 2-1) for detailed information on various connection methods. Guard connections are not shown in these figures for the sake of clarity.

Sense selection

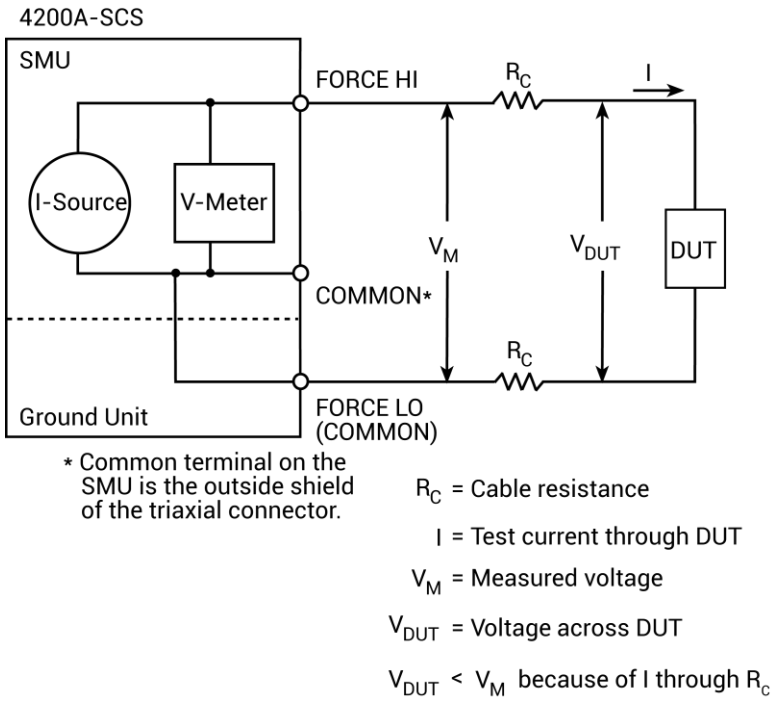
The sensing method is automatically selected depending on the connection method used. To use local sensing, connect only SMU FORCE and ground unit FORCE (COMMON) to the DUT (see previous figure "Local Sensing"). To use remote sensing, add the SENSE connections shown in the previous figure "Remote Sensing."

Local sensing

Measurements made on devices with impedances above approximately 1 kΩ are generally made using the local sensing method, as shown in the next figure. 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.

If you are measuring low-impedance DUTs, the local sensing method may give inaccurate results. The cable resistance (R_C) and 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 1 mΩ to 100 mΩ, so it may be difficult to get accurate local sensing measurements with DUT resistances below 100 Ω to 1 kΩ, depending on the magnitudes of the cable resistance and contact resistance. In these cases, you may need to use remote sensing.

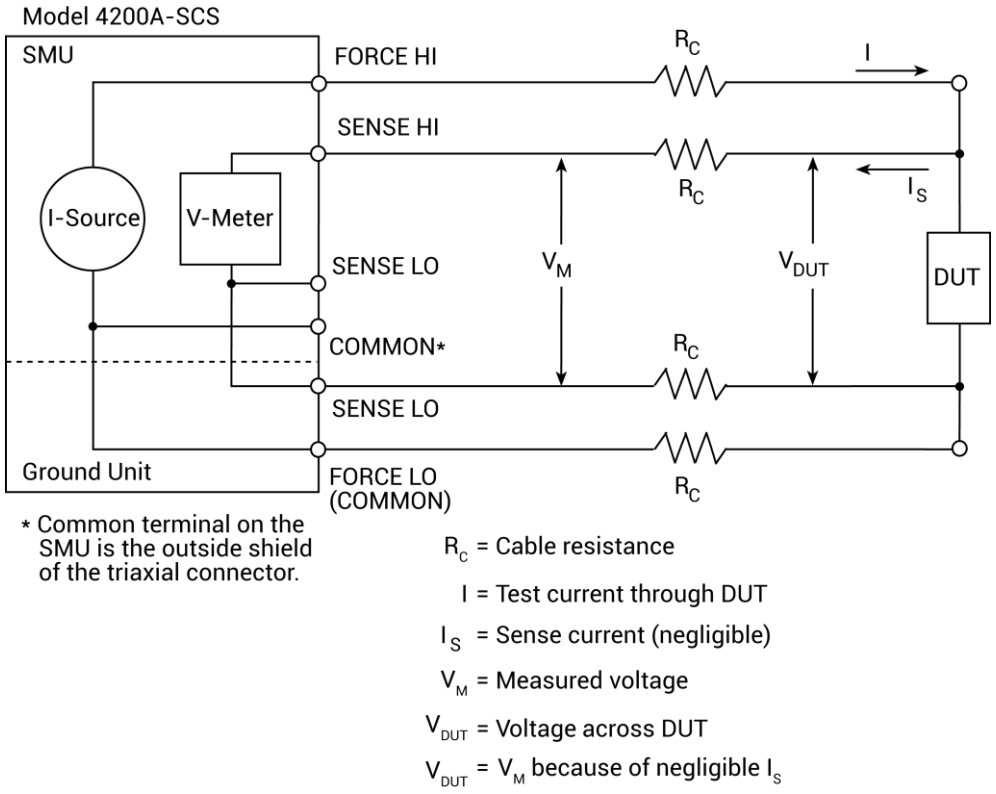
Figure 63: Local sensing



Remote sensing

The remote sensing method shown in the next figure 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. 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 64: Remote sensing



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 kΩ.
- Maximum V-Source or V-Measure accuracy are required

NOTE

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

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 +12 V 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).

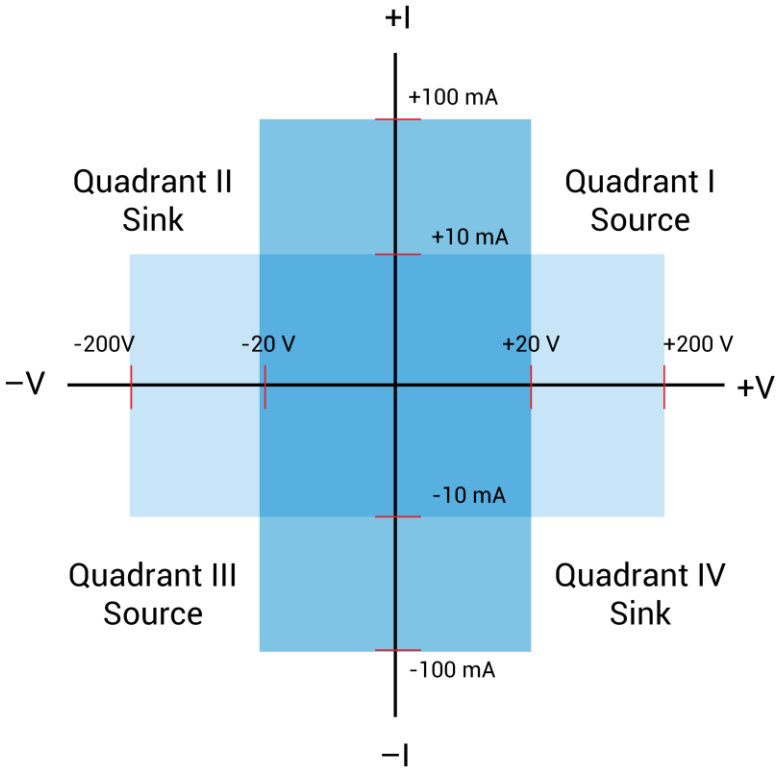
CAUTION

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 preamplifier due to excessive current that will flow into the unit.

4200-SMU sink boundaries

Nominal 420x-SMU boundaries are shown in the next figure. Note that actual boundaries are 210 V at 10.5 mA or 21 V at 105 mA.

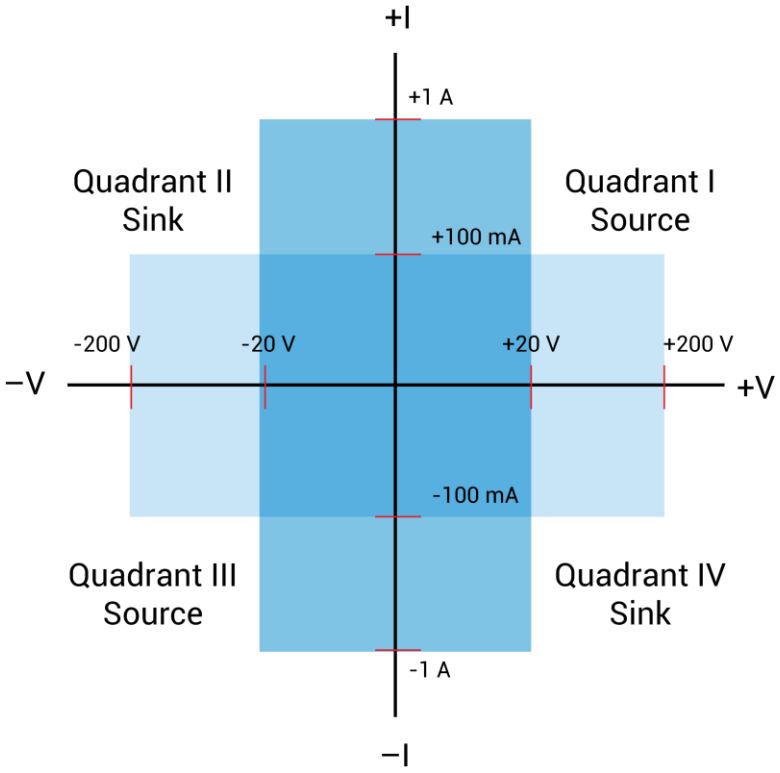
Figure 65: 420x-SMU and 4200-PA operating boundaries



4210-SMU sink boundaries

Nominal 421x-SMU sink boundaries are shown in the next figure. Actual boundaries are 210 V at 105 mA or 21 V at 1.05 A.

Figure 66: 421x-SMU and 4200-PA operating boundaries



Source-measure considerations

When configured to source current (I-Source), 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.

Source I, measure V or I

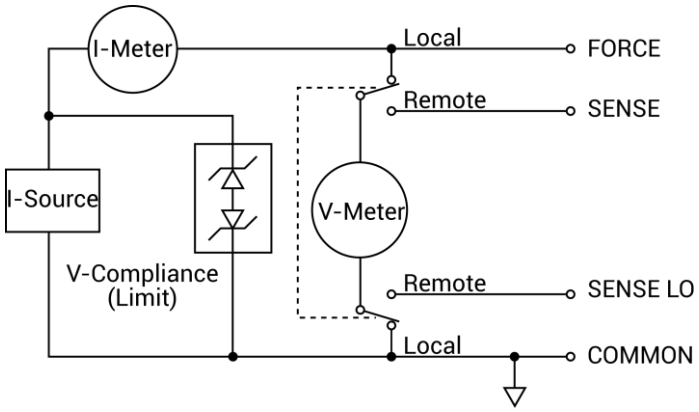
For voltage measurements, the SENSE selection (local or remote) determines where the measurement is made. In local SENSE, voltage is measured at the FORCE and COMMON terminals.

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 preamplifier and the DUT.

NOTE

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

Figure 67: Source I, measure V configuration

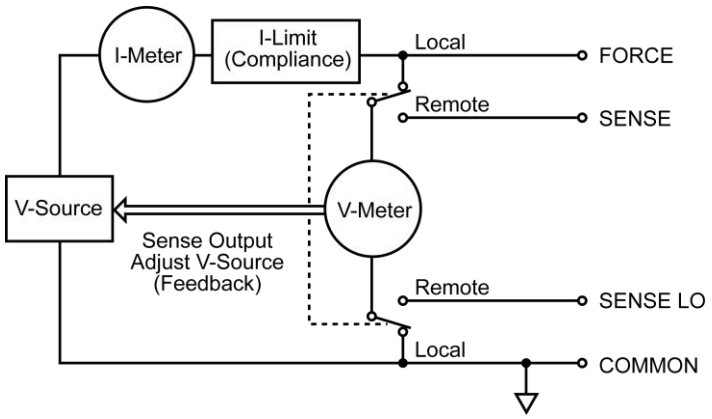


Source V, measure I or V

When configured to source voltage (V-Source) as shown in the next figure, 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.

Figure 68: Source V, measure I configuration



Measure only (V or I)

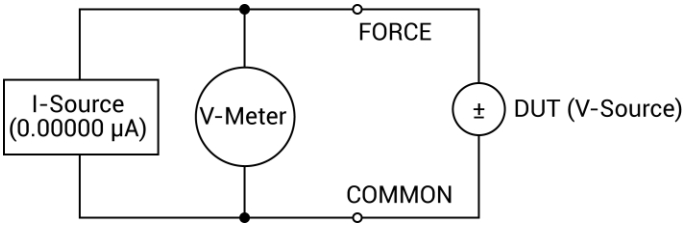
The next figures show the configurations for using the SMU exclusively as a voltmeter or ammeter. For both of these configurations, use local sensing.

CAUTION

For measure V, set the voltage compliance higher than the measured voltage. For measure I, set the current compliance higher than the measured current.

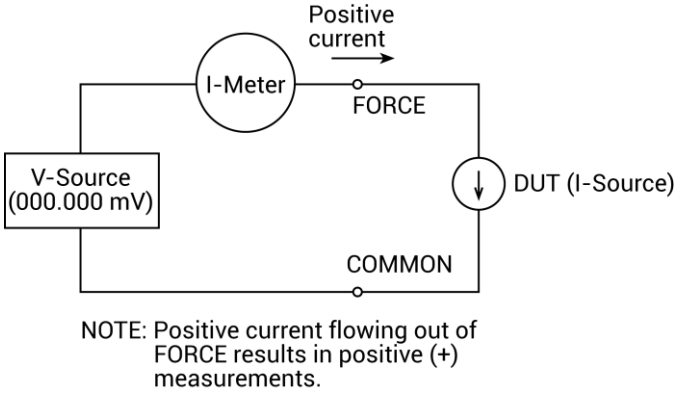
In the following figure, the SMU is configured to measure voltage only by setting it to source 0 A and measure voltage.

Figure 69: Measure voltage only



In the following figure, the SMU is configured to measure current only by setting it to source 0 V and measure current. Note that to get positive (+) readings, conventional current must flow from FORCE to COMMON.

Figure 70: Measure current only



Sweep concepts

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 the next figure) as part of a sweep (refer to Sweep waveforms (on page 3-46)).

Source-delay-measure cycle

During each SDM cycle, the following occurs:

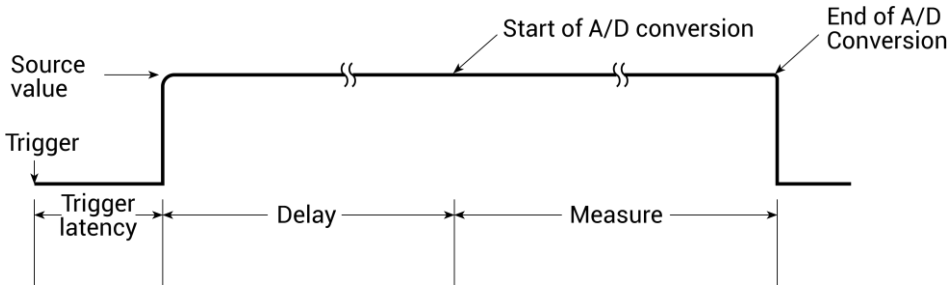
1. The source output level is set.
2. There is a wait for the source delay.
3. The measurement is made.

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 needed can be calculated or determined by trial and error. For purely resistive loads and at higher current levels, the sweep 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 71: SDM cycle



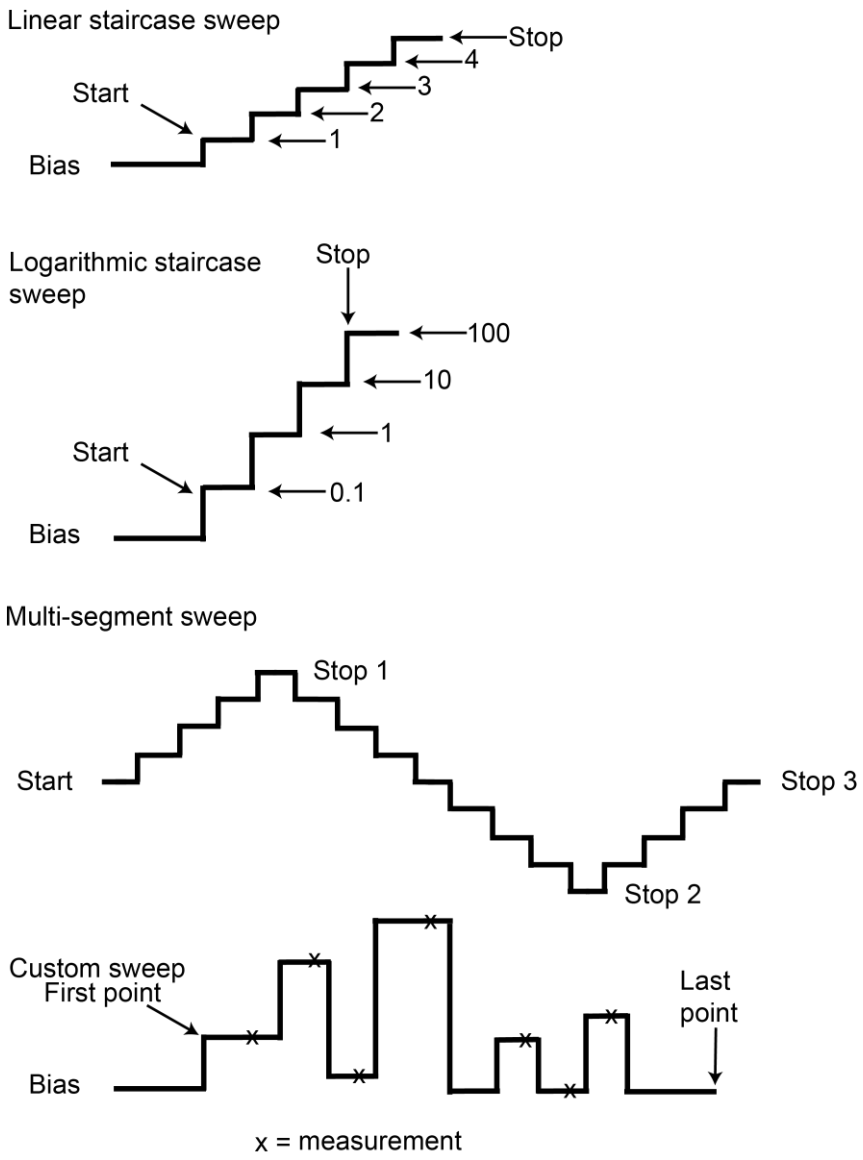
Sweep waveforms

There are four general sweep types: linear staircase, logarithmic staircase, multi-segment, and custom, as shown in the following figure. 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 multi-segment sweep increments through a series of segments with variable voltage or current steps. 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 is made at each step (level). The time spent at each step depends on how the SDM cycle is configured for aspects such as the sweep delay.

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

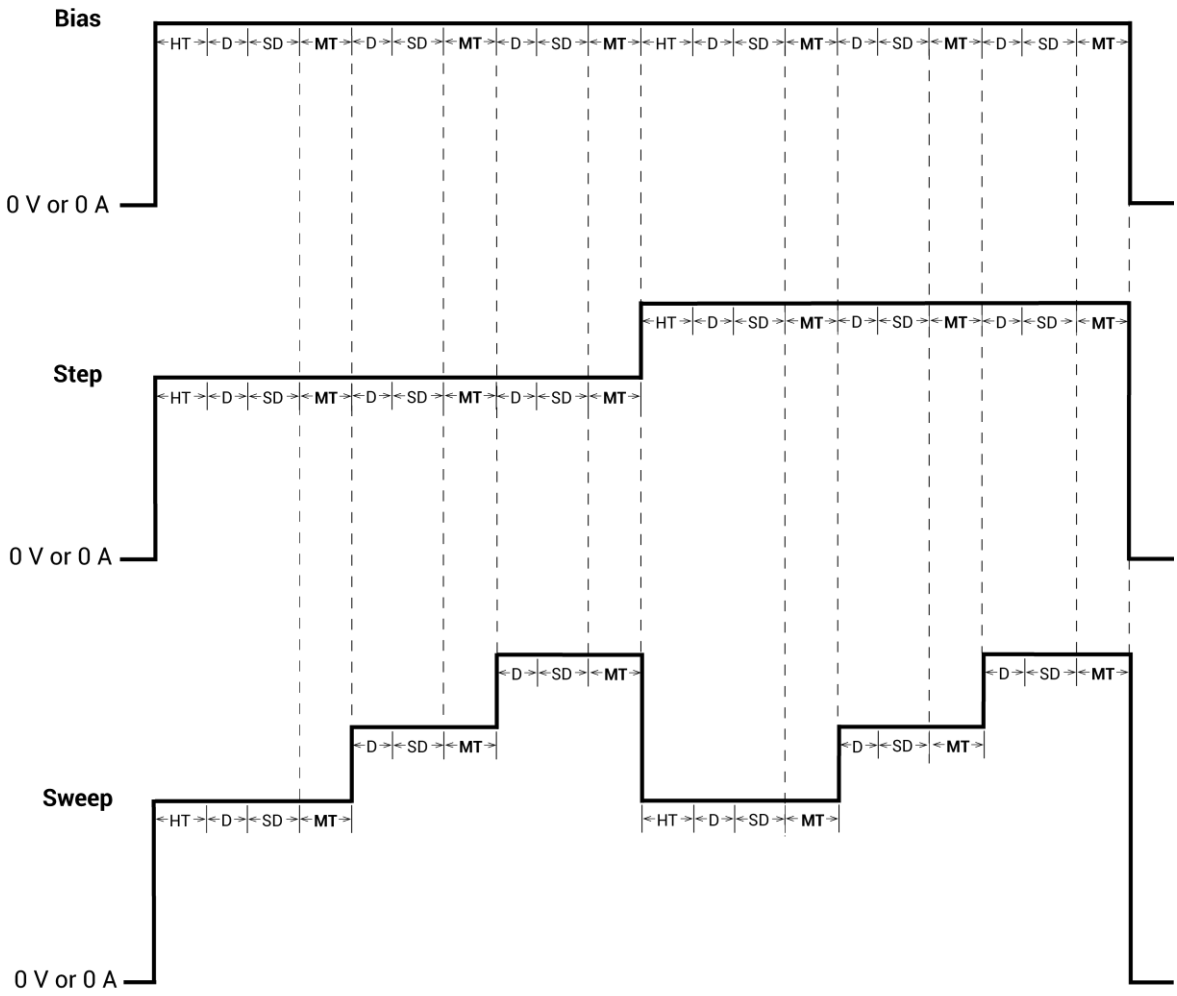
Figure 72: Sweep waveforms



Operation mode timing diagrams

The following figure shows source-measure timing for a test system using three SMUs. It shows basic timing between the three operation modes: sweep, step, and bias.

Figure 73: Sweeping Mode timing diagram



HT = Hold time
D = Delay (default delay x delay factor)
SD = Sweep delay
MT = Measure time

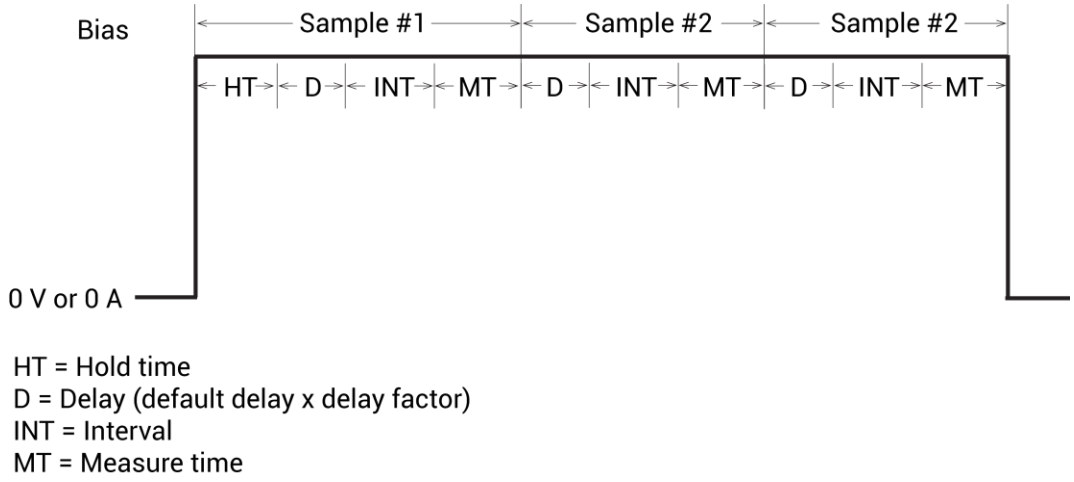
The timing elements act as follows:

- 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 Aperture 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.

Sampling Mode timing diagram

The following figure shows a timing diagram for the Sampling Mode.

Figure 74: Sampling Mode timing diagram



HT = Hold time
D = Delay (default delay x delay factor)
INT = Interval
MT = Measure time

A range-dependent delay (D) is automatically applied by a SMU before each measurement to allow for source settling. 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.

In sampling mode, all device terminals are set to a static operation mode: 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. You can set the delay to 0 s by setting the Delay Factor to 0.

The Measure Time (MT) is determined by the Filter Factor and the A/D Aperture 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.

Section 4

Multi-frequency capacitance-voltage unit

In this section:

Introduction

The 4210-CVU and 4215-CVU Multi-Frequency (1 kHz to 10 MHz) Capacitance-Voltage Unit are impedance measurement modules that can be installed in the 4200A-SCS.

The 4210-CVU operates from 1 kHz to 10 MHz. The AC test signal (10 mV_{RMS} to 100 mV_{RMS}) can be DC voltage biased from –30 V to +30 V. The 4215-CVU operates from 1 kHz to 10 MHz at 1 kHz resolution. The AC test signal (10 mV to 1 V RMS) can be DC voltage biased from –30 V to +30 V.

Measurement overview

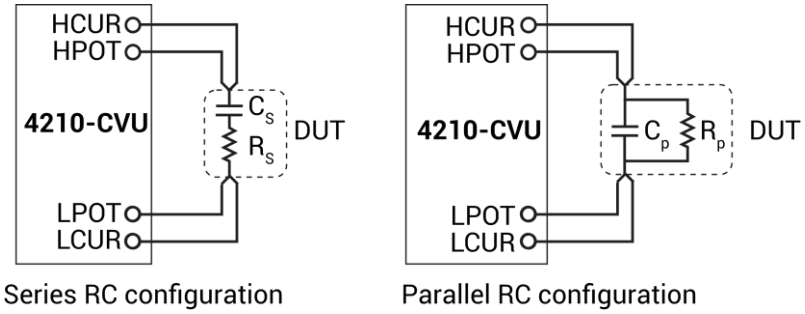
The 4210-CVU makes AC impedance measurements (Z_{DUT}) of the device under test (DUT) by sourcing an AC test voltage across the device and measuring the resulting AC current and AC voltage.

The AC current is measured as shown in Typical 4210-CVU test connections to a DUT (on page 4-8).

The HCUR/HPOT and LCUR/LPOT terminal pairs are interchangeable. Each provides the full capability of the other pair.

The simplified model of a device-under-test (DUT) is a resistor and a capacitor. As shown in the next figure, the 4210-CVU can measure the DUT as a series configuration of the resistor-capacitor (RC) or as a parallel RC configuration.

Figure 75: Measure models (simplified)



Measurement functions

The 4210-CVU can measure these parameters:

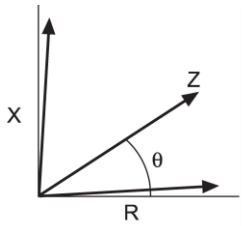
- Parallel capacitance and conductance (Cp-Gp)
- Parallel capacitance and dissipation factor (Cp-D)
- Series capacitance and resistance (Cs-Rs)
- Series capacitance and dissipation factor (Cs-D)
- Resistance and reactance (R-jX)
- Impedance and phase angle (Z-theta)
- Admittance and phase angle (Y-theta)

The time domain of the AC current and AC voltage must be processed into the frequency domain to produce the phasor form of the DUT impedance.

The capacitive impedance (and conductance) are calculated based on the measured AC impedance and phase.

The next figure shows the vector diagram and fundamental equations for impedance.

Figure 76: Vector diagram for impedance (Z)



$$|Z| = \sqrt{R^2 + X^2} \qquad R = Z\cos\theta$$

$$Z = R + jX \qquad X = Z\sin\theta$$

$$\theta = \arctan\left(\frac{X}{R}\right) \qquad Y = \frac{1}{Z} = (G + jB)$$

Z = Impedance
θ = Phase angle
R = Resistance
X = Reactance
Y = Admittance
G = Conductance
B = Susceptance

The capacitance is calculated from the capacitive impedance and the test frequency using the formula:

$$C_{DUT} = \frac{I_{DUT}}{2\pi f V_{AC}}$$

Where:

- C_{DUT} = Capacitance of the DUT (F)
- I_{DUT} = Current of the DUT (A)
- f = Test frequency (Hz)
- V_{AC} = Measured AC voltage (V)

The inductance (L) can be calculated from the reactance (X) and test frequency (f):

$$L = \frac{X}{2\pi f}$$

Test signal

The test signal can be set for the following frequencies:

For the 4210-CVU:

- 1 kHz to 10 kHz in 1 kHz increments
- 10 kHz to 100 kHz in 10 kHz increments
- 100 kHz to 1 MHz in 100 kHz increments
- 1 MHz to 10 MHz in 1 MHz increments

The AC signal output level can be set from 10 mV_{RMS} to 100 mV_{RMS} (1 mV_{RMS} resolution). The output impedance is 100 Ω (typical). The AC voltage measure range is 10 0 mV_{RMS}.

The ranges available to measure current are 1 µA, 30 µA, or 1 mA. With autorange selected, range selection is done automatically.

For the 4215-CVU:

- 1 kHz to 10 MHz in 1 kHz increments

The AC signal output level can be set from 10 mA to 1 V. The output impedance is 100 Ω (typical). The AC voltage measure range is 100 mV_{RMS}.

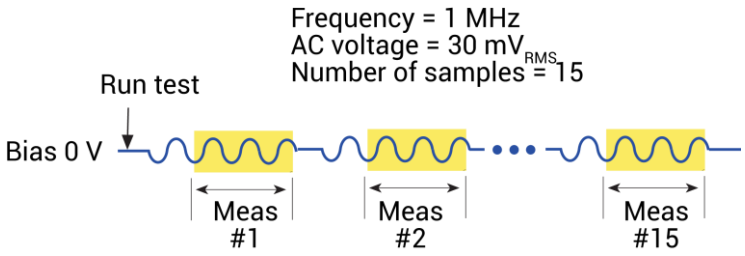
The ranges available to measure current are 1 µA, 30 µA, or 1 mA. With autorange selected, range selection is done automatically.

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).

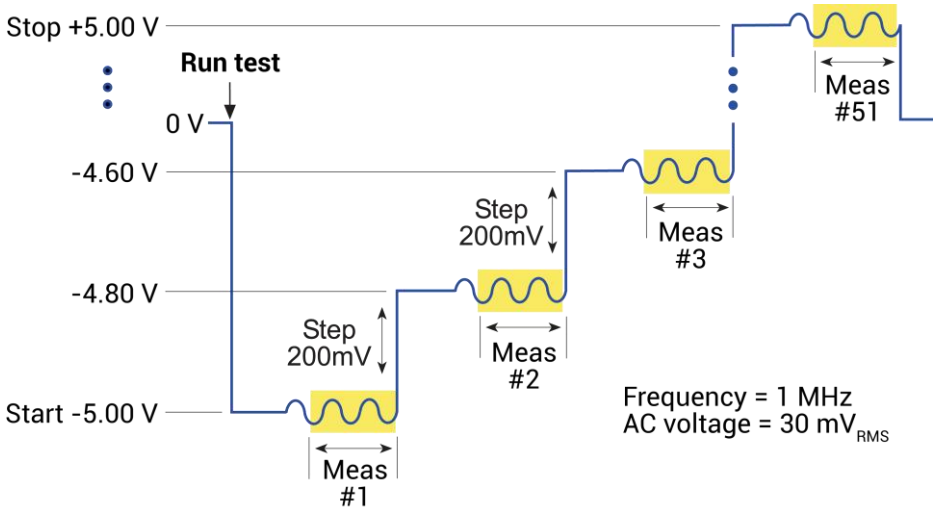
The next figure shows an example of DC bias waveform. In this example, the DC bias is set to 0 V, but you can set it to any valid DC bias level. You specify the number of measurements to make.

Figure 77: DC Bias waveform (example)



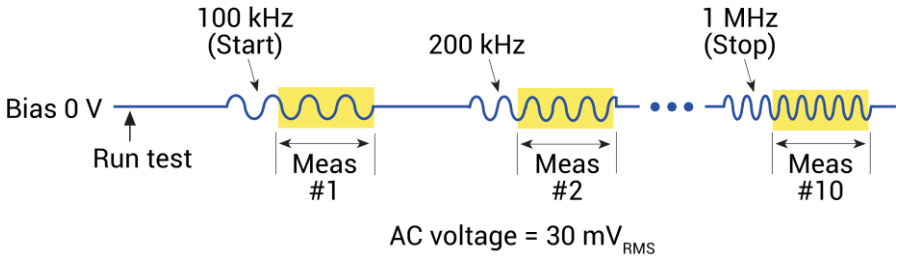
The next figure shows an example of DC voltage sweep. You specify the start voltage, stop voltage, and step voltage. The number of measure points is calculated by the 4210-CVU.

Figure 78: DC voltage sweep (example)



The next figure shows an example of a frequency sweep. You specify the start frequency and the stop frequency. The CVU calculates the number of measure points.

Figure 79: Frequency sweep (example)



If you are setting up a voltage list sweep, you specify the voltage levels for the sweep.

If you are setting up a voltage step frequency sweep, the sweep includes voltage stepping. A frequency sweep is performed for every voltage step point.

If you are setting up a frequency step voltage sweep, the sweep includes frequency stepping. A voltage sweep is performed for every frequency point.

NOTE

Refer to Operation Mode (CVU) (on page 6-61) for details on the bias and sweep functions.

Force-measure timing

Timing diagrams for the force-measure process for bias and sweep functions are shown in the next two topics.

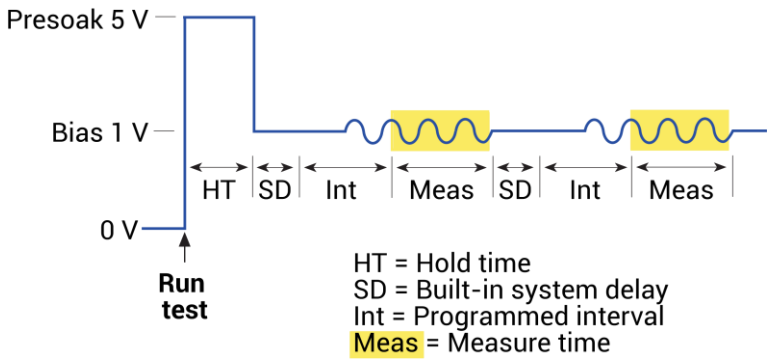
Bias function timing

You can apply DC voltage bias to the DUT (±30 V).

Timing for the force-measure process for a bias function is shown in the next figure. When the test is started, the following timing sequence takes place:

1. The DC source outputs the presoak voltage for the hold time.
2. The DC source goes to the DC bias voltage.
3. After the built-in system delay and time intervals, the CVU makes a measurement. The AC test signal is applied just before the start of the measurement. The AC drive is turned off after the measurement is completed. This step is repeated for every measurement.

Figure 80: Force-measure timing



Sweep function timing

Force-measure timing for a sweep function is similar to the timing for a bias function (shown in Bias function timing (on page 4-7)), with the following differences:

- The hold time is repeated at the beginning of each subsequent sweep step.
- A sweep delay is used in place of the interval.

CVU connections

The CVU is shipped with four red SMA cables (male-to-male, 100 Ω, 1.5 m). These characterized cables must be used for connection to the CVU order to achieve optimum performance.

If you have the 4210-CVU-Prober-Kit, you also have four 3 m SMA cables.

Connection notes

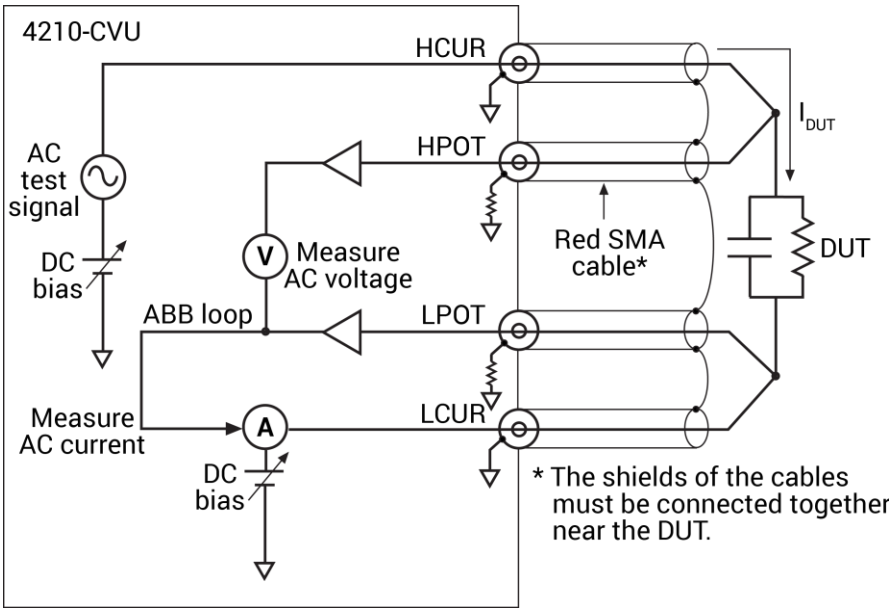
- Use only the supplied red SMA cables for connections to the 4210-CVU.
- Do not use a mix of cable lengths on different CVU terminals.
- Use the supplied torque wrench to tighten SMA cable connectors to 8 in. lb.
- In Clarius, the cable length setting of a test must match the length of the SMA cables used for your setup.
- If you need to run connection compensation, do it after setting up connections or after making changes to existing connections, but before running any tests. See the Connection compensation (on page 4-14) topic.
- 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.
- Use coaxial cables to extend SMA shielding to the DUT, then connect them together.

Typical CVU test connections to a DUT

The shields of the SMA cables must be connected together and extended as far as possible to the device under test (DUT), as shown in the following figure.

Use the supplied torque wrench to tighten the SMA connections to 8 in. lb.

Figure 81: Measurement circuit (simplified)

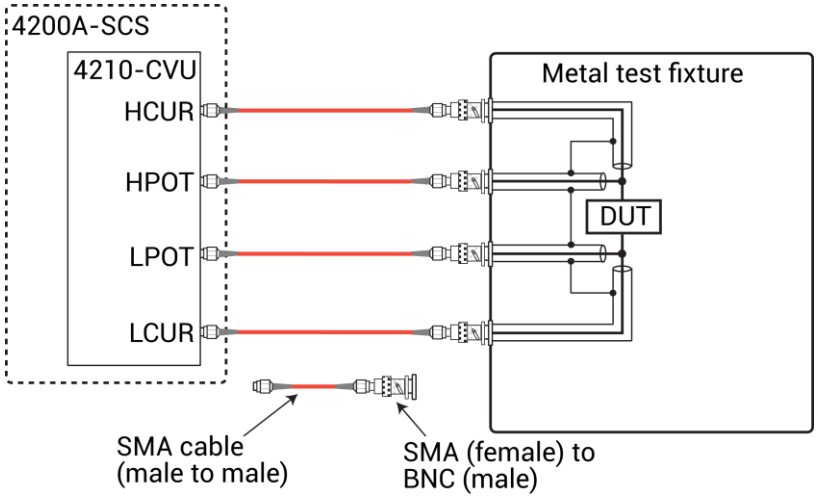


NOTE

Note that you can swap the HCUR and HPOT and LCUR and LPOT terminal functionality in Clarius.

The next figure shows typical connections to a DUT installed in a test fixture equipped with female BNC bulkhead connectors. 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. The cables and adapters shown are the ones supplied with the 4210-CVU.

Figure 82: Test connections to DUT

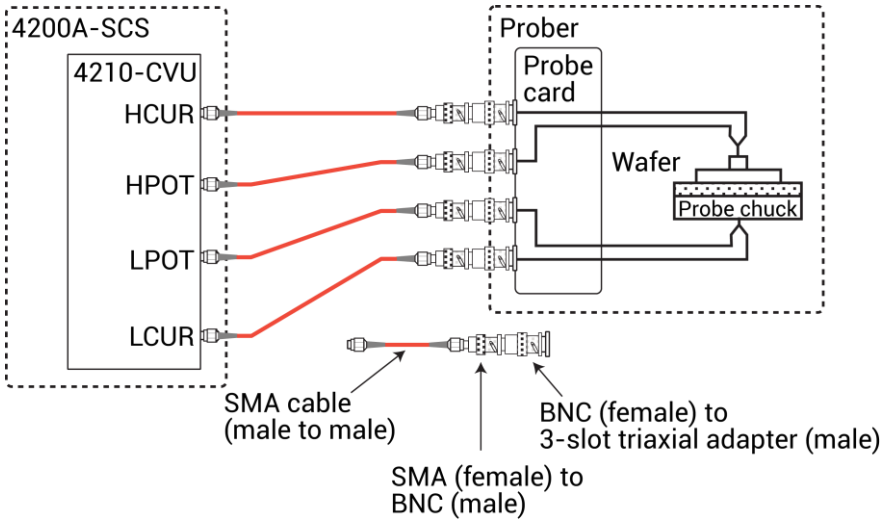


Test connections to a probe card

The 4210-CVU-Prober-Kit includes 3-meter SMA cables and connection accessories to connect the 4210-CVU to a probe card. Refer to the probe kit documentation for a list of the supplied items. The cables and adapters supplied with the CVU and the probe kit accommodate most connection requirements.

The next figure shows typical test connections to a probe card. The connections use triaxial female connectors. The probe kit includes two types of BNC to triaxial adapters. The 7078-TRX-BNC has the guard connected to the inner shield of the adapter. The 7078-TRX-GND has the guard disconnected. In most applications, the 7078-TRX-BNC is the preferred adaptor.

Figure 83: Typical connections to a probe card



NOTE

The shields of the SMA cables must be connected together and extended as far as possible to the DUT, as shown in Typical 4210-CVU test connections to a DUT (on page 4-8).

NOTE

If the probe card uses BNC (female) connectors, you do not need the BNC-to-triaxial adapters.

Typical CVU matrix card connections

In your project, you can automate the use of a CVU and other instrumentation using a switching matrix and actions to control the switching. When the project is run, the switching matrix automatically makes the required instrument connections for each test in the project.

The next figures show typical connections for a switch system using a Series 700 Switching System with the 7174A Matrix Card installed.

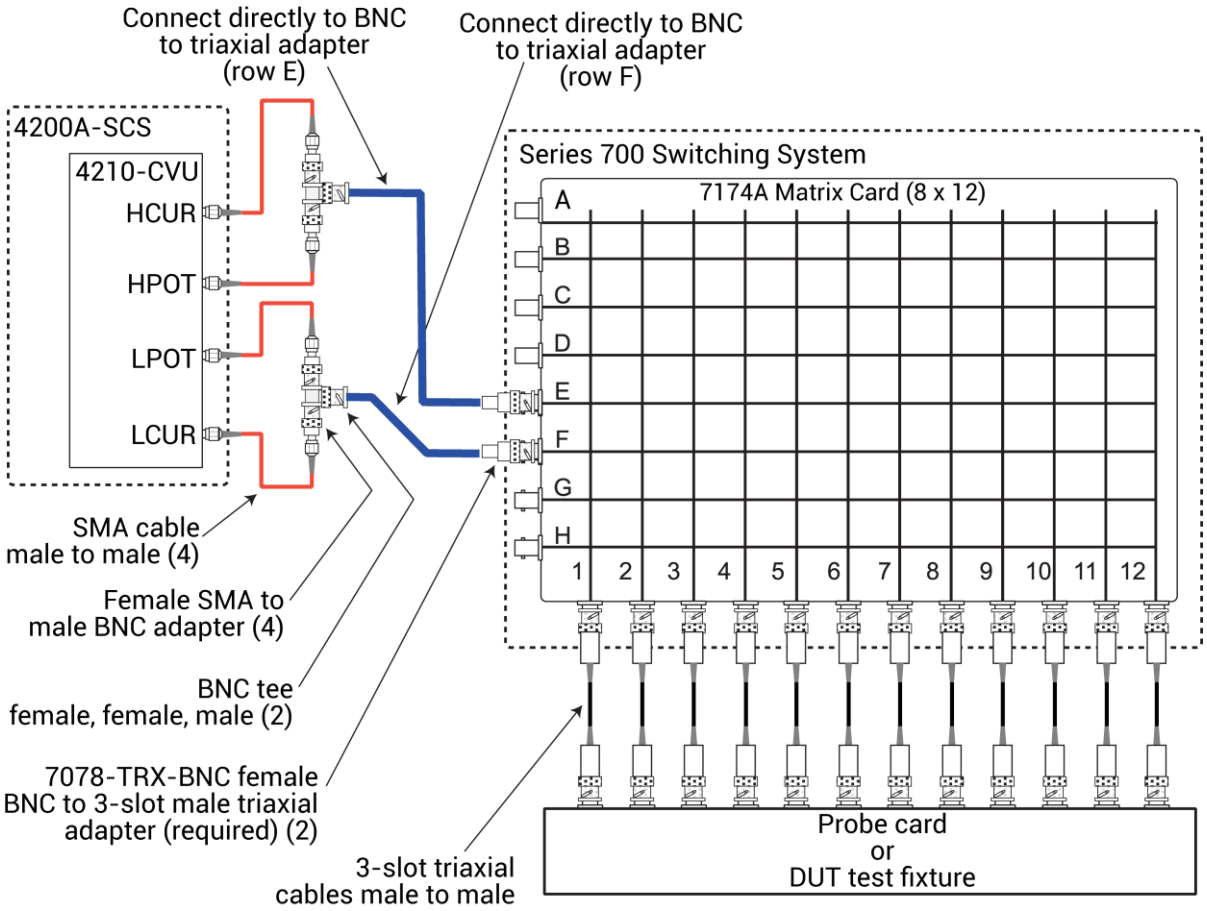
NOTE

You can also use the 7072 Matrix Card for C-V testing. However, you must use rows G and H and local (2-wire) sensing.

The SMA cables and adapters shown in the following drawings are supplied with the CVU or the 4210-CVU-Prober-Kit. The triaxial and BNC cables are not supplied. The prober kit includes two types of BNC-to-triaxial adapters that connect directly to the rows of the matrix. The 7078-TRX-BNC has the guard connected to the inner shield of the adapter. The 7078-TRX-GND has the guard disconnected.

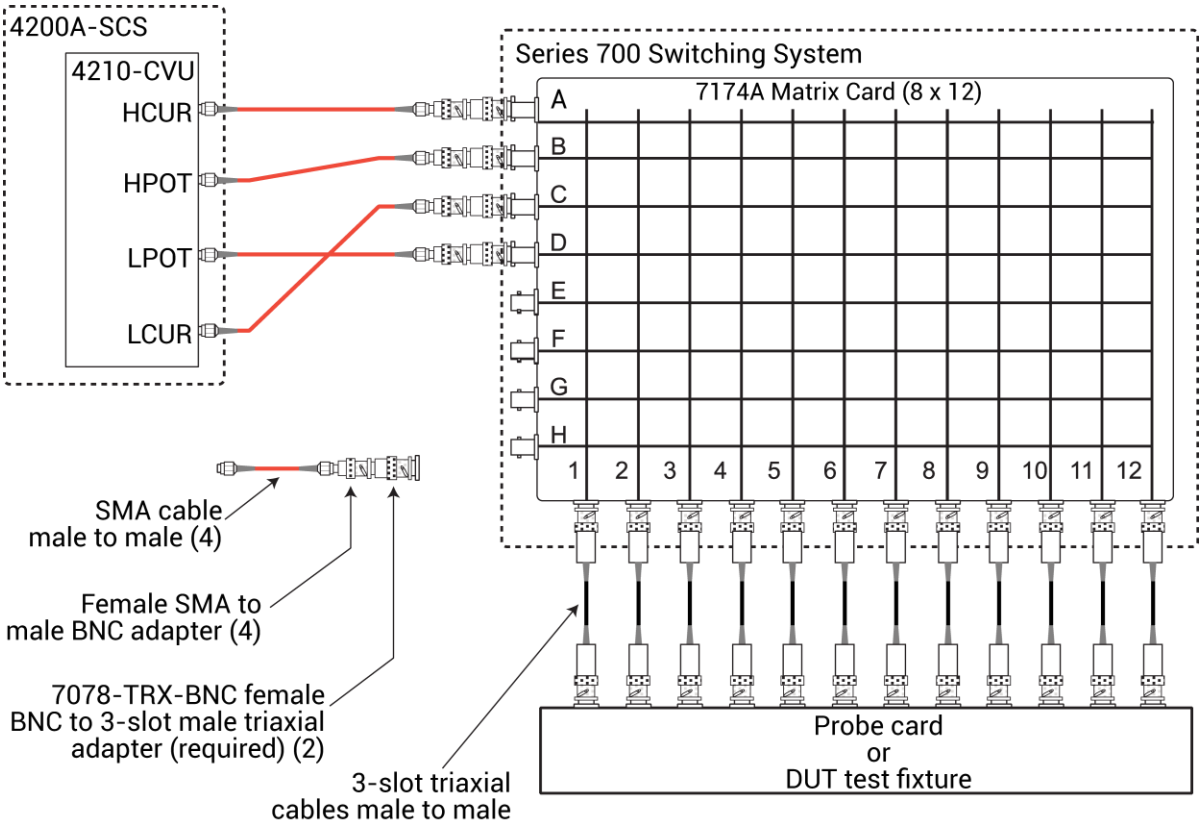
This figure shows connections for local (2-wire) sensing. It shows the 4210-CVU connected to rows E and F of the matrix. This is the connection scheme for the `cap-iv-cv-matrix` project. For details, see cap-iv-cv-matrix (on page 4-49).

Figure 84: Test connections for a switch matrix - local (2-wire) sensing



The following figure shows connections for remote (4-wire) sensing.

Figure 85: Test connections for a switch matrix - remote (4-wire) sensing



NOTE

The 7078-TRX-BNC adapters must be used in order to extend SMA shielding through the matrix card.

NOTE

The shields of the SMA cables must be connected together and extended as far as possible to the DUT, as shown in Typical 4210-CVU test connections to a DUT (on page 4-8).

Connection compensation

You can correct offset and gain errors caused by the connections between the CVU and the device under test (DUT) by using connection compensation. To use correction, you will:

- Generate connection compensation data for open, short, and load conditions.
- Enable CVU connection compensation.

When a test is run, the enabled compensation values are factored in by each measurement.

If open, short, or load compensation is disabled, those compensation values are not used by the test.

Once compensation values are stored, they are available to any project that uses a CVU.

NOTE

Update connection compensation any time the connection setup is changed or disturbed. Changes in temperature or humidity do not affect connection compensation.

NOTE

If the 4210-CVU is connected to a 4200A-CVIV Multi-Switch, run the `cvu-cviv-comp-collect` action. Refer to the *4200A-CVIV Multi-Switch User's Manual* for detail.

Use the following general guidelines to determine which correction needs to be done:

- **Open** correction: Offset correction for small capacitances (>1 MΩ, large impedance).
- **Short** correction: Offset correction for large capacitances (<10 Ω, small impedance).
- **Load** correction: Resistive load correction for gain error. Keithley recommends a load that is as close in impedance to the cabling system (100 Ω). The load must have an impedance versus frequency characteristic that is purely resistive over the frequency range of subsequent measurements.

Generate open connection compensation data

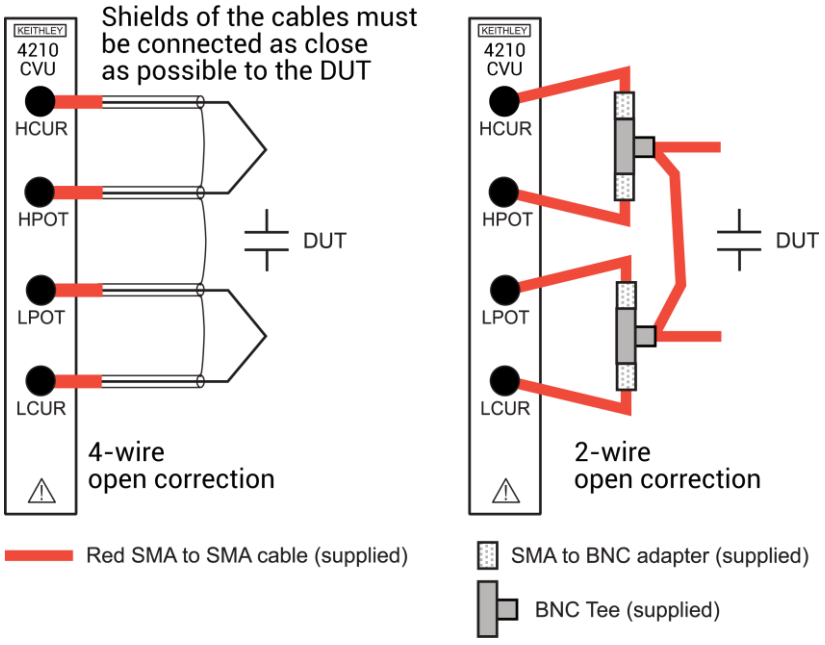
Open connection compensation is usually done to offset correction for small capacitances.

Open compensation is done with all the cables, adaptors, switch matrices, and other hardware connections connected to the device under test in the test circuit. The probes must be lifted up or the device must be removed from the test fixture.

To generate open connection compensation data:

1. Make the connections to the CVU, as shown in the following figure. For remote (4-wire) sensing, the shields of the four SMA cables must be connected as close as possible.

Figure 86: Connections for open connection compensation CVU

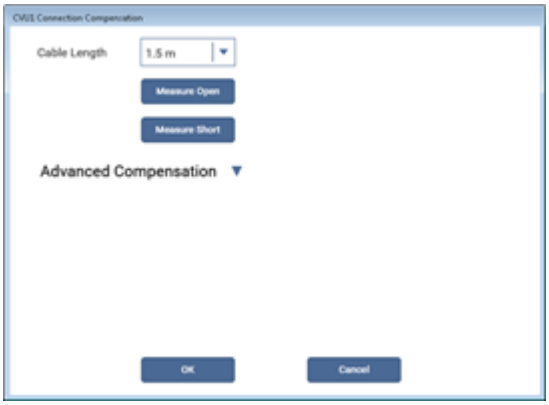


2. In Clarius, select **Tools**. The Clarius Tools dialog box opens.

Figure 87: Clarius Tools dialog box

3. Select **CVU Connection Compensation**.

Figure 88: CVU Connection Compensation dialog box



4. Select the **cable length**. You can select:
   - **0 m:** Use if measurements are made at the terminals of the CVU (no cables).
   - **1.5 m:** Use with the standard red SMA cables (part number CA-447A) that are supplied with the CVU.
   - **3 m:** Use with the red SMA cables (part number CA-446) that are supplied with the 4200-CVU-Prober-Kit. You can also use this setting if you are using a switching matrix.
   - **Custom:** Cable length coefficients are measured by the user using the Measure Custom Cable Length option under Advanced Compensation.

5. If you selected Custom cable length, select **Advanced Compensation** and select **Measure Custom Cable Length**. Follow the on-screen instructions.
6. If you are using a switching matrix, close the matrix switches that connect the CVU to the open. Refer to Switch matrix control (on page A-21).
7. In the Clarius CVU Connection Compensation dialog box, select **Measure Open**.
8. Follow the instructions.
9. Select **OK**.

Generate short connection compensation data

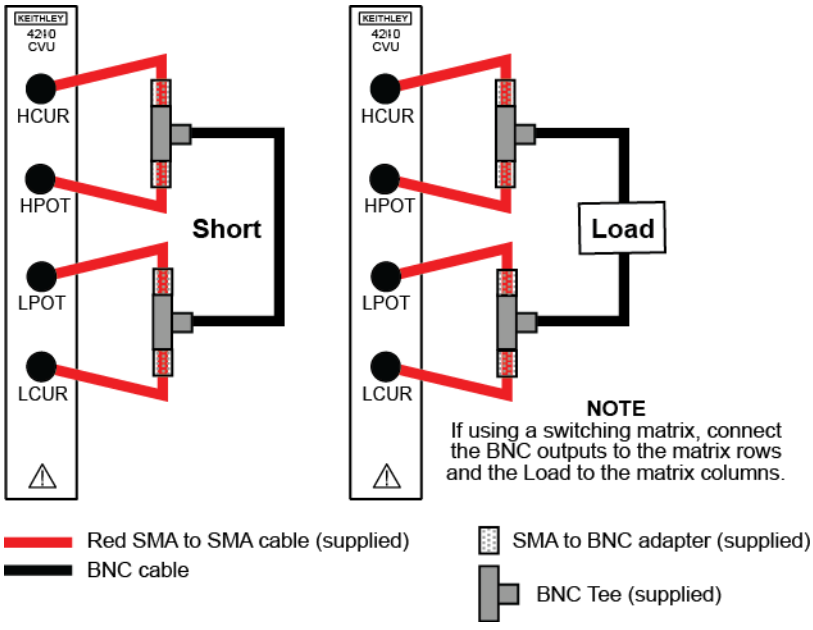
Short connection compensation is usually done to offset correction for large capacitances.

Short compensation is done by connecting all the CVU terminals directly together. A known short is connected to the CVU terminals through all the cables, adaptors, and probes that may be in the test circuit. You can make a short at the wafer level by shorting all probes together.

To generate short connection compensation data:

1. Make the connections to the CVU, as shown in the following figure. For remote (4-wire) sensing, the shields of the four SMA cables must be connected.

Figure 89: Connections for short and load connection compensation

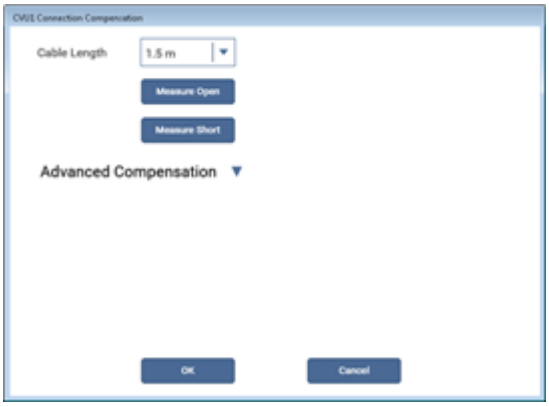


2. In Clarius, select **Tools**. The Clarius Tools dialog box opens.

Figure 90: Clarius Tools dialog box

3. Select **CVU Connection Compensation**.

Figure 91: CVU Connection Compensation dialog box



4. Select the **cable length**. You can select:
   - **0 m:** Use if measurements are made at the terminals of the CVU (no cables).
   - **1.5 m:** Use with the standard red SMA cables (part number CA-447A) that are supplied with the CVU.
   - **3 m:** Use with the red SMA cables (part number CA-446) that are supplied with the 4200-CVU-Prober-Kit. You can also use this setting if you are using a switching matrix.
   - **Custom:** Cable length coefficients are measured by the user using the Measure Custom Cable Length option under Advanced Compensation.

5. If you selected Custom cable length, select **Advanced Compensation** and select **Measure Custom Cable Length**. Follow the on-screen instructions.
6. If you are using a switching matrix, close the matrix switches that connect the CVU to the open. Refer to Switch matrix control (on page A-21).
7. In the Clarius CVU Connection Compensation dialog box, select **Measure Short**.
8. Follow the instructions.
9. Select **OK**.

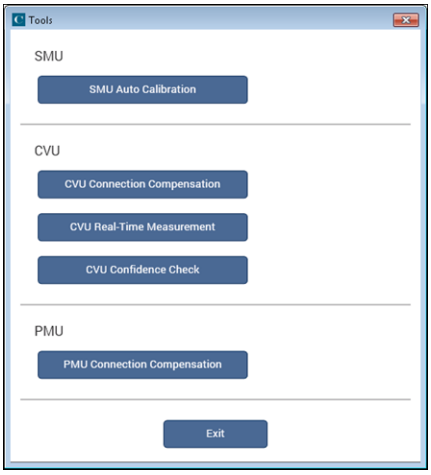
Generate load connection compensation data

Loads are reference resistors, typically 50 or 100 Ω or less, that must be resistive and constant over the entire frequency range (1 kHz to 10 MHz). A load is connected to the output terminals using all the cables, adaptors, probes and other hardware that will be in the test circuit.

To generate load correction data:

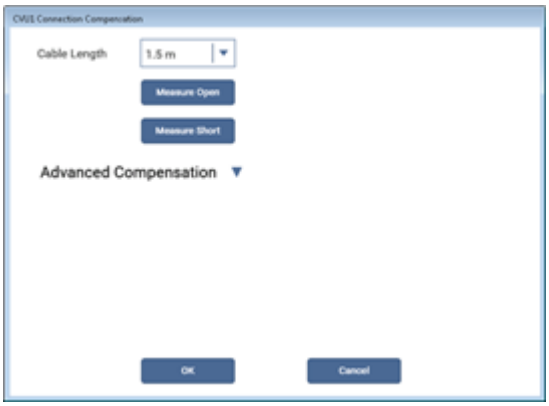
1. Make the connections to the CVU. See the Test connections for a switch matrix (on page 4-11).
2. In Clarius, select **Tools**. The Clarius Tools dialog box opens.

Figure 92: Clarius Tools dialog box



3. Select **CVU Connection Compensation**.

Figure 93: CVU Connection Compensation dialog box



4. Select the **cable length**. You can select:
   - **0 m:** Use if measurements are made at the terminals of the CVU (no cables).
   - **1.5 m:** Use with the standard red SMA cables (part number CA-447A) that are supplied with the CVU.
   - **3 m:** Use with the red SMA cables (part number CA-446) that are supplied with the 4200-CVU-Prober-Kit. You can also use this setting if you are using a switching matrix.
   - **Custom:** Cable length coefficients are measured by the user using the Measure Custom Cable Length option under Advanced Compensation.
5. If you selected Custom cable length, select **Advanced Compensation** and select **Measure Custom Cable Length**. Follow the on-screen instructions.
6. If you are using a switching matrix, close the matrix switches that connect the CVU to the open. Refer to Switch matrix control (on page A-21).
7. If it is not open, select **Advanced Compensation**.
8. Next to Measure Load, enter the value of the load in ohms.
9. Select **Measure Load**.
10. Follow the instructions.
11. Select **OK**.

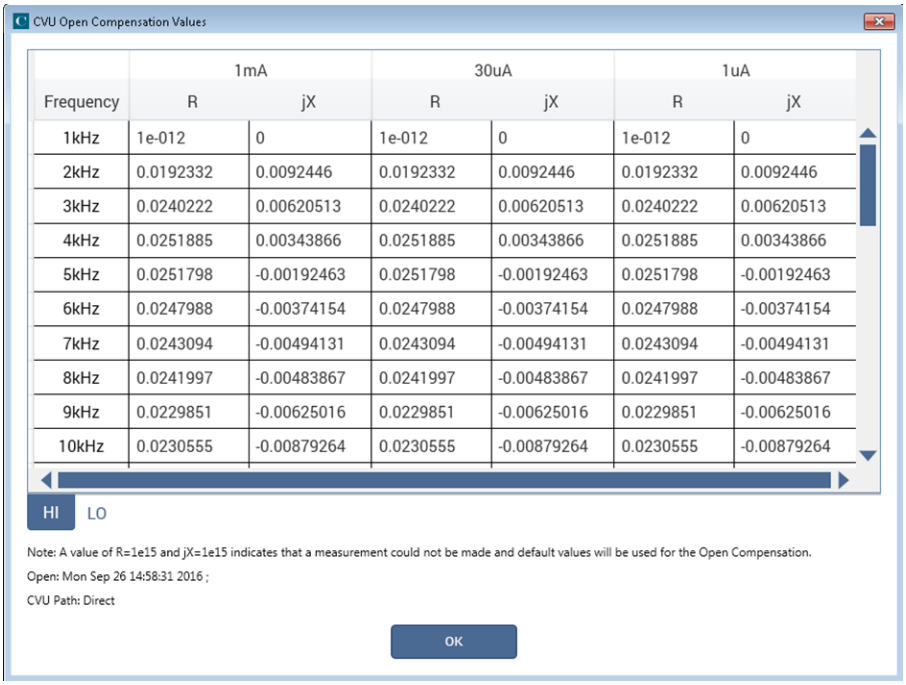
Compensation data

You can view the compensation data. Clarius lists R and jX compensation values for every test frequency and measurement range for open, short, and load.

To view the data generated by connection compensation:

1. In Clarius, select **Tools**.
2. Select **CVU Connection Compensation**.
3. Select **Advanced Compensation**.
4. Select the data you would like to display: **Open**, **Short**, or **Load**.
5. Select **View Compensation Data**.
6. Select the **HI** tab to review the high side values.
7. Select the **LO** tab to review the low side values.

Figure 94: Open compensation values example



Enable compensation

To use the values generated by connection compensation, you need to enable compensation for each test.

When compensation is enabled, the most recently acquired CVU compensation data is applied. Compensation values can be gathered using the CVU Connection Compensation option in Tools or through actions and user modules.

To enable compensation:

1. Select the test from the project tree.
2. Select **Configure**.
3. Select the terminal in the center pane.
4. In the right pane, select **Terminal Settings**.
5. Under Compensation, select the types of compensation as needed.
6. Make sure **Cable Length** is the same as the Cable Length setting in the Tools > CVU Connection Compensation dialog box that was used to generate connection compensation data.

Figure 95: Enable connection compensation

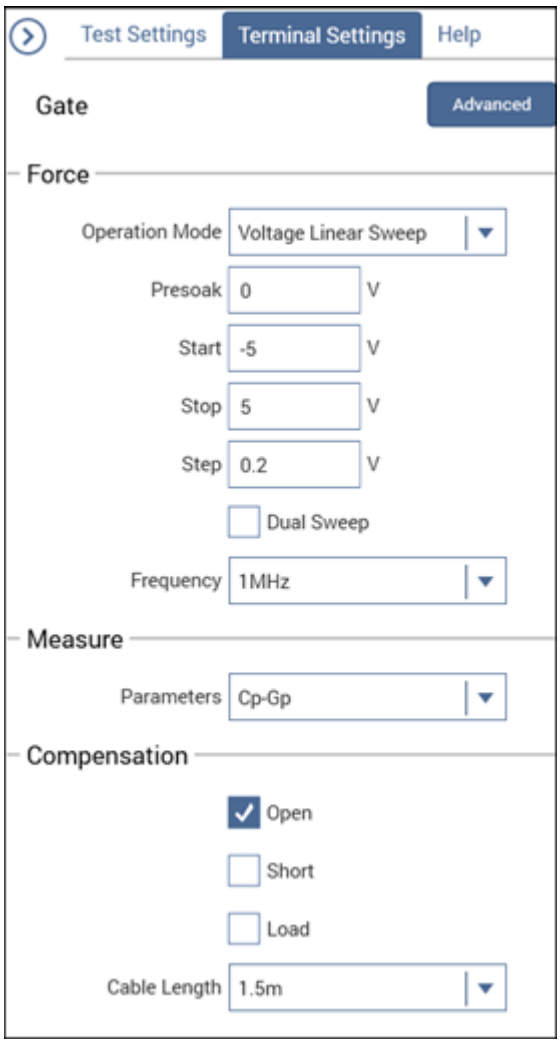


Figure 96: Enable connection compensation - 4215

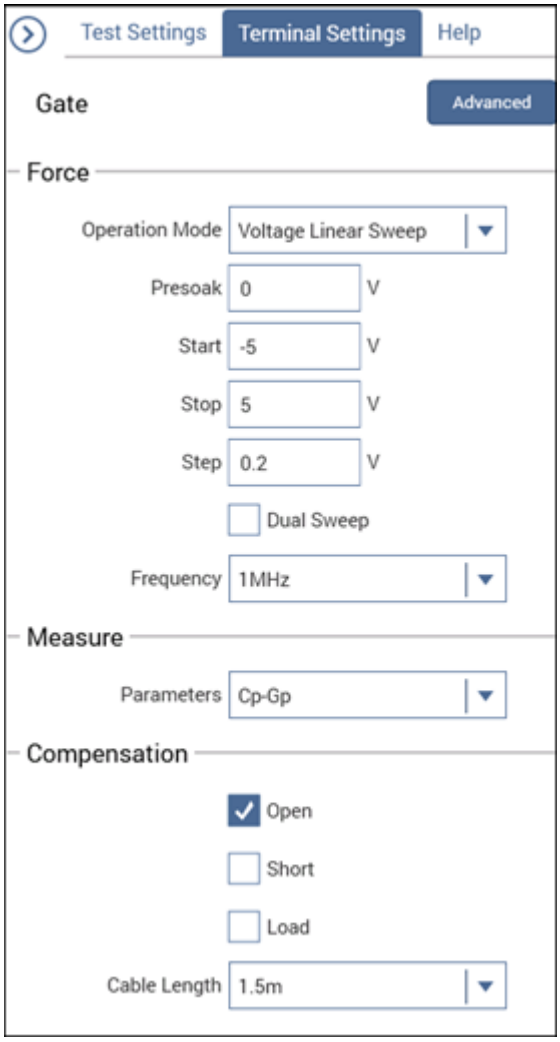


ABB unbalance errors

The CVU uses the autobalancing bridge (ABB) technique to achieve accurate impedance measurements. The ABB creates a virtual ground at the DUT to minimize measurement error. Every CVU measurement is made with ABB active. The ABB always attempts to lock the low side of the DUT to virtual ground.

If the ABB fails to lock, the measurement is made, but may be out of specification. If this occurs, the returned data is flagged and shown in yellow on a blue background on the Analyze sheet.

Most common reasons that the ABB fails to lock are:

- The cable lengths on the CVU terminals are not the same
- HPOT or LPOT terminals were disconnected
- Excessive noise on the LPOT terminal
- High frequency sources
- Physical cable lengths do not match the cable length set in Clarius
- Improperly torqued SMA cables
- Sub-optimal I_{range} setting
- Too much parasitic load on the low side of the DUT

You can use Confidence Check (on page 4-27) to help troubleshoot ABB errors.

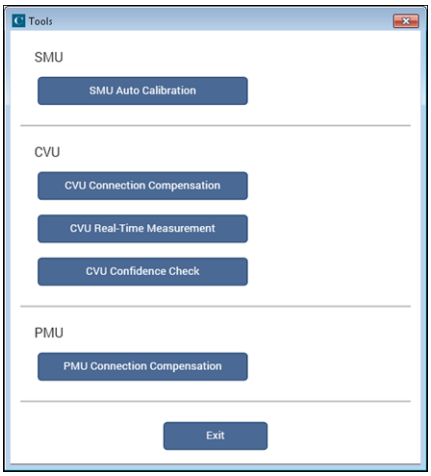
CVU Real-Time Measure Mode

The CVU Real-Time Measure Mode provides a direct real-time user interface to the CVU to help you set up and debug your system. For example, you can use it to confirm that contact has been made with the pads on a wafer. The measurements are independent of the open and short confidence checks.

To make real-time measurements:

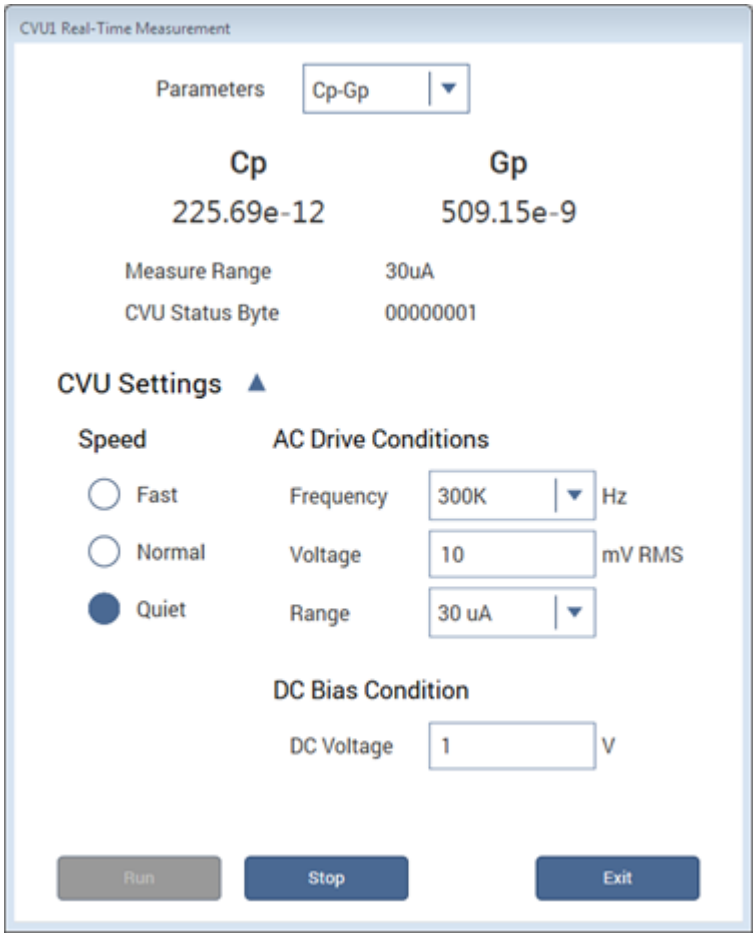
1. In Clarius, select **Tools**. The Clarius Tools dialog box opens.

Figure 97: Clarius Tools dialog box



2. Select **CVU Real-Time Measure Mode**.

Figure 98: Real-Time Measurement dialog box



3. Select the parameters for which you want to return results.
4. Set the Speed, AC Drive Conditions, and DC Bias Conditions for the conditions you want to test.
5. Select **Run**. The results for the selected parameters are displayed at the top of the dialog box.

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 CVU is connected to the DUT, the Confidence Check displays the measured readings in real time in the Messages area of Clarius.

An open or short confidence check makes a measurement on the high and the low sides of the test circuit.

Run an open check and short check

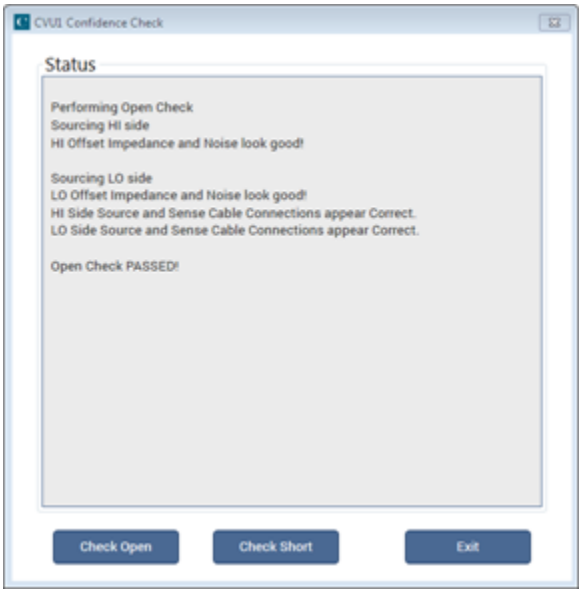
To do a CVU confidence check:

1. If you are using a switching matrix, connect the switching matrix to the CVU and DUT or the short as explained in Test connections for a switch matrix (on page 4-11). For the short check, close the matrix switches to connect the CVU to the DUT or short. For the open check, also close the matrix switches, but lift the probes or disconnect the DUT.
2. In Clarius, select **Tools**. The Clarius Tools dialog box opens.

Figure 99: Clarius Tools dialog box

3. Select **CVU Confidence Check**.

Figure 100: CVU Confidence Check dialog box



4. Select **Check Open** or **Check Short**.
5. Follow the instructions and click **OK**.
6. When the check is complete, the dialog box displays the results of the test. If the test failed, the results include suggestions for troubleshooting.

4210-CVU project example

The following example shows you how to use the 4210-CVU in a project to make a capacitance measurement.

The example assumes you have a 4210-CVU connected to a test fixture as described in Typical 4210-CVU test connections to a DUT (on page 4-8).

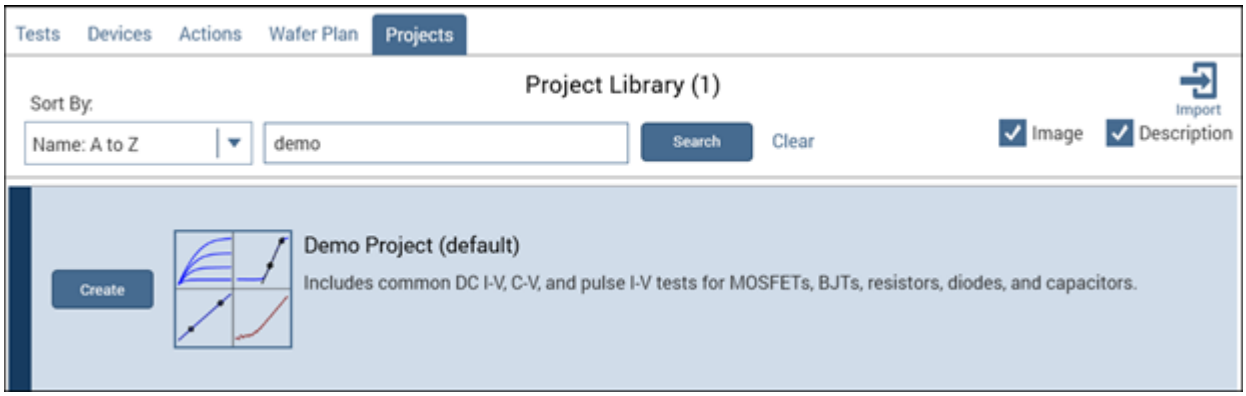
In this test, you sweep the DC bias from –5 V to 5 V in 0.2 V steps, with a 1 MHz capacitance measurement made at each step.

Select a project

Select a project:

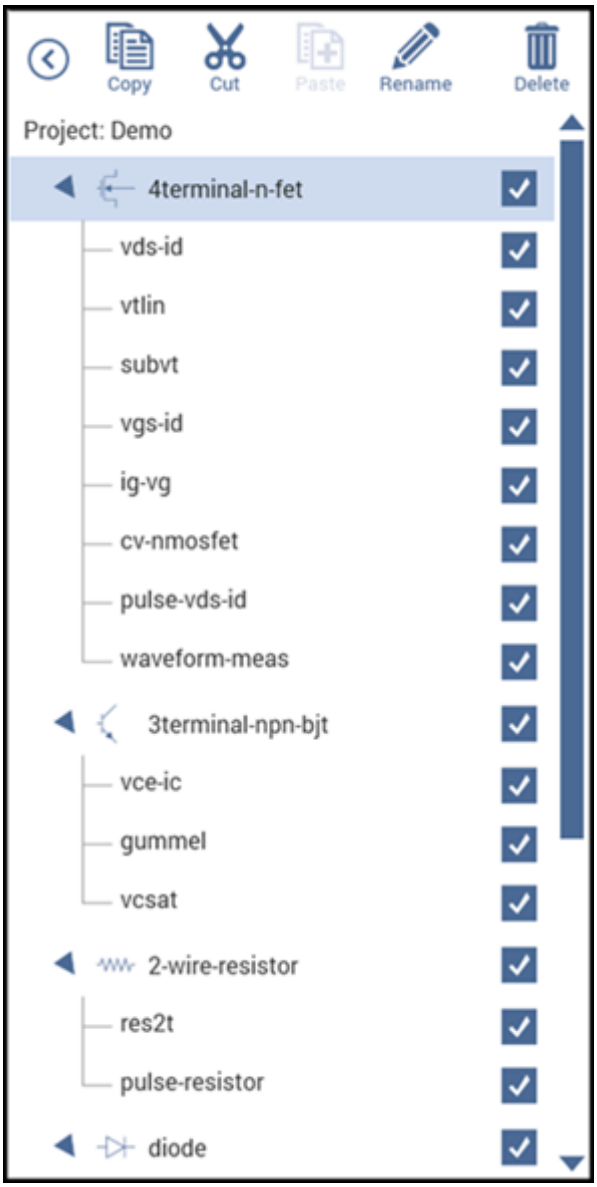
1. Start **Clarius**.
2. Do Connection compensation (on page 4-14) to correct for offset and gain errors caused by the connections.
3. Choose **Select**.
4. Select **Projects** to display the Project Library in the center pane.
5. Search for **demo**. The Demo Project is displayed, as shown in the next figure.

Figure 101: Demo Project selected



6. Select **Create**. You are prompted to replace the existing project.
7. Select **Yes**. The project is displayed in the project tree.

Figure 102: Demo Project in the project tree



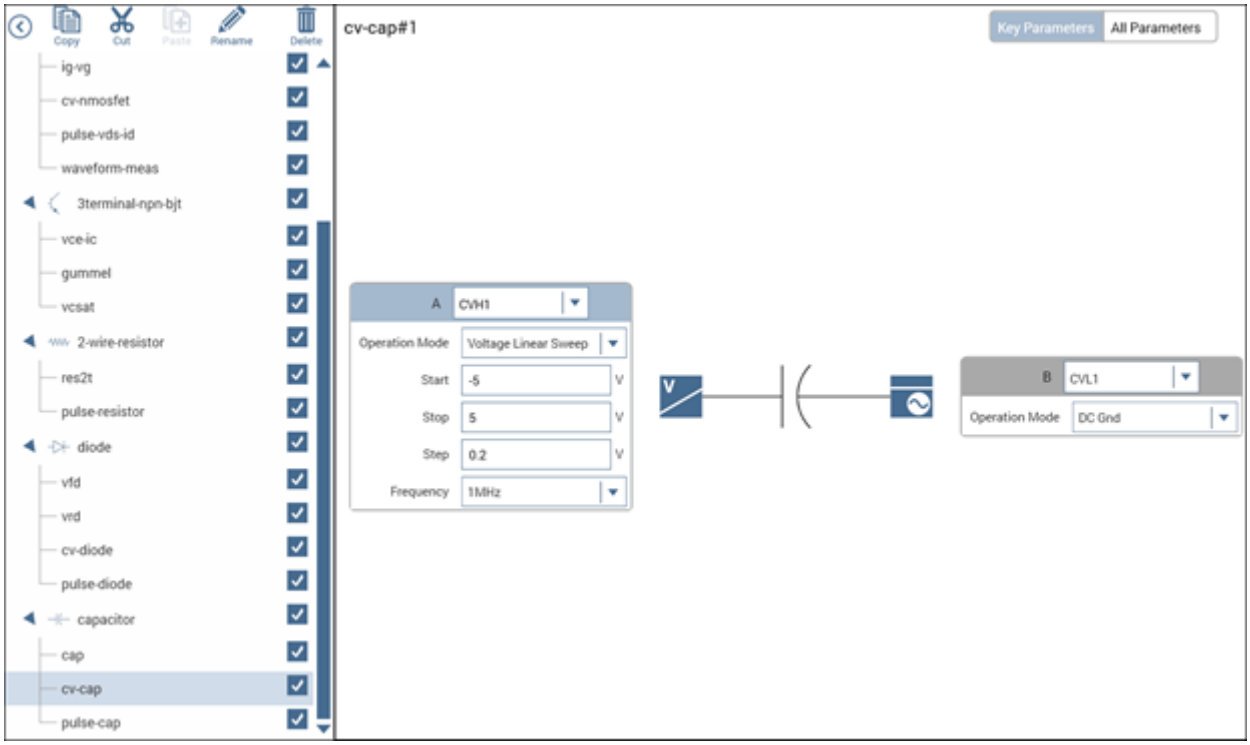
Configure the test

This example uses the `cv-cap` test, which measures the capacitance as a function of a linear voltage sweep of a capacitor.

Configure the cv-cap test:

1. In the project tree, select the **cv-cap** test.
2. Select **Configure**. The key parameters are displayed in the center pane.

Figure 103: cv-cap key parameters



3. Verify the settings for CVH1:
   - Operation Mode: Voltage Linear Sweep
   - Start: –5 V
   - Stop: 5 V
   - Step: 0.2 V
   - Frequency: 1 MHz

Run the test and review results

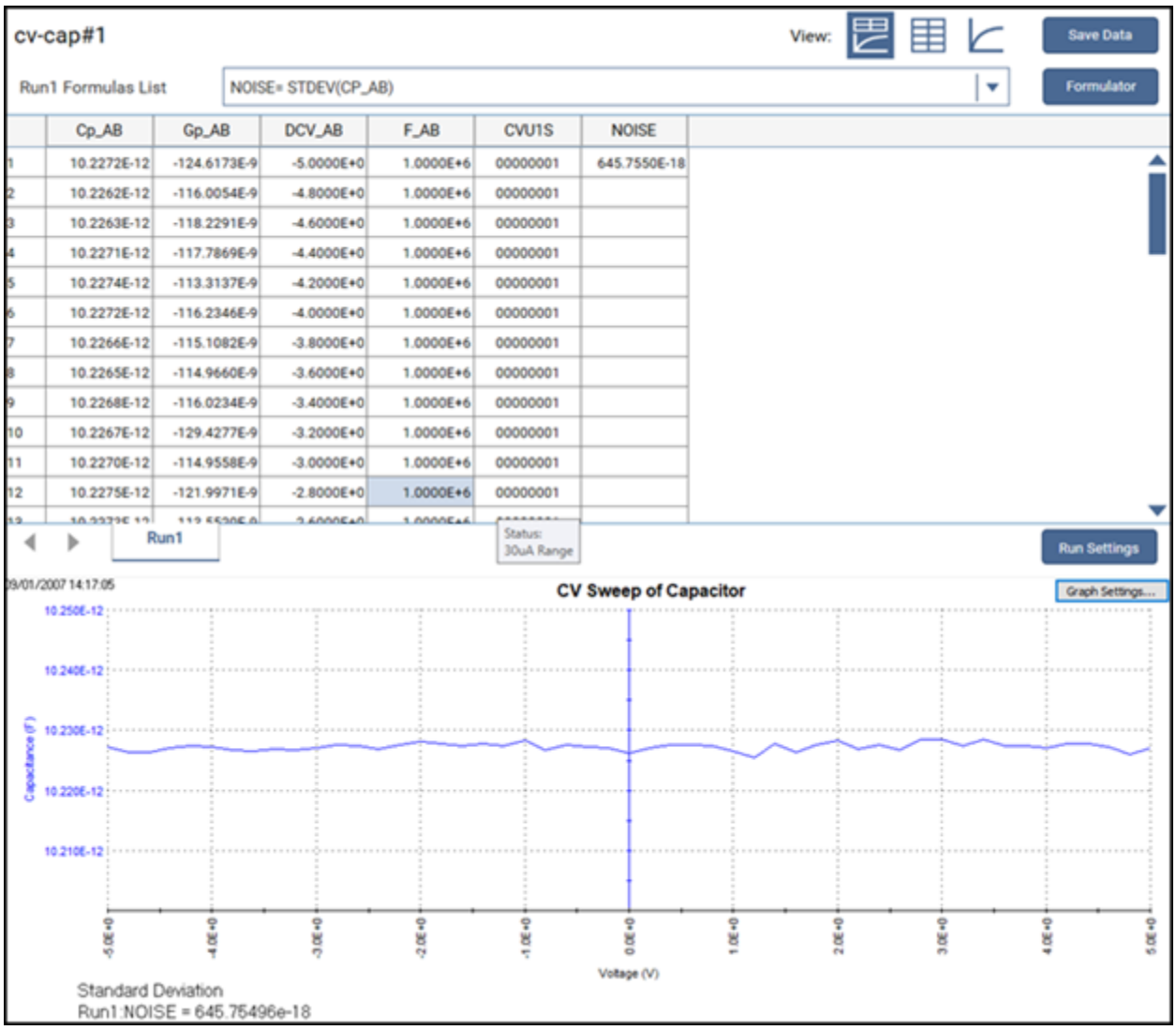
To run the cv-cap test and review the results:

1. Select **Run**.
2. Select **Analyze**. The test results are shown as data in a spreadsheet and on the graph, as shown in the figure below.
3. To export the data, select **Save Data**.
4. To save the information in the Run sheet, select **Save Sheet**. The information is saved in a Microsoft® Excel® spreadsheet format.
5. To save the information in the graph, select the **Graph File Format** and select **Save Graph1**. The graph is saved into selected file type.

Note that the extracted NOISE parameter is displayed on the graph.

For information on changing how the graph is displayed, refer to Change the display of the graph (on page 6-259).

Figure 104: cv-cap test results



Timing diagrams

The following topics describe typical test setups with the related timing diagrams.

CVU using voltage bias

For this test, make the following test settings:

- Speed: Normal
- Report Timestamps selected
- Mode: Sampling
- Interval: 0.25 s
- Number of Samples: 10
- Hold Time: 1 s

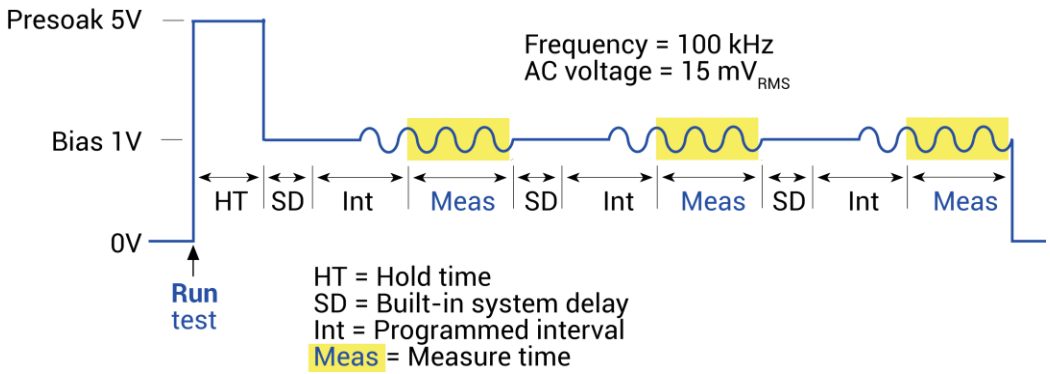
Make the following Terminal Settings for CVH1:

- Operation Mode: Voltage Bias
- Presoak: 5 V
- DC Bias: 1 V
- Frequency: 100 kHz
- Parameters: Cp-Gp
- Compensation as needed.

When this test is run, the following force-measure sequence occurs:

1. The DC source goes to the Presoak voltage of 5 V for the hold time (1 s).
2. The DC source goes to the DC bias voltage of 1 V.
3. After the built-in system delay and interval (0.25 s), the 4210-CVU makes 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. This step is repeated for every sample.

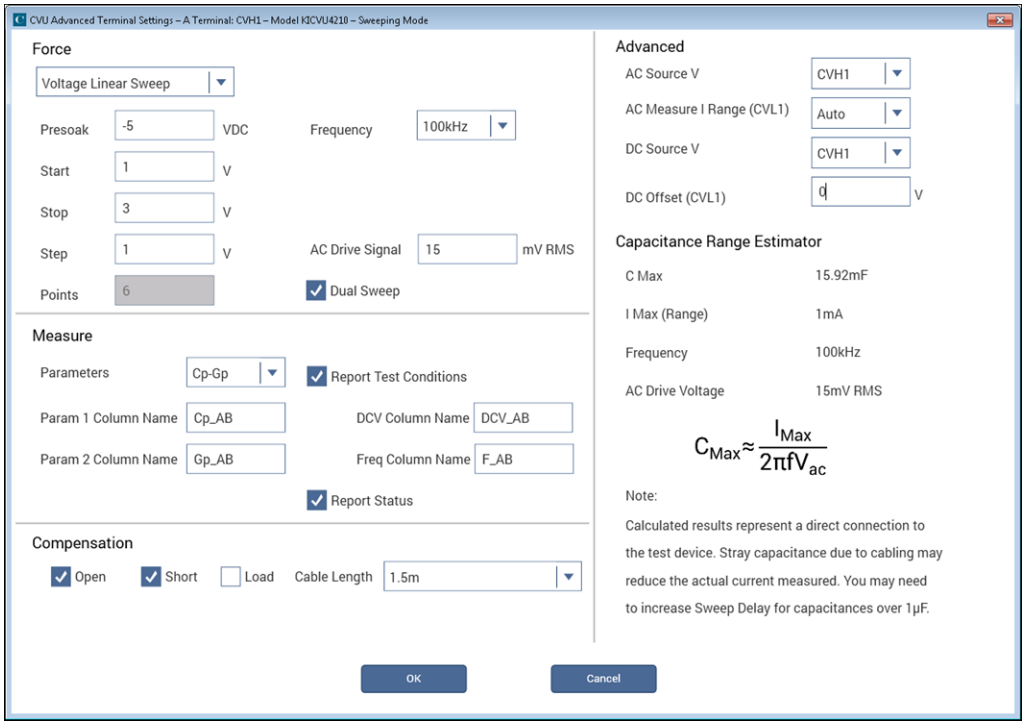
Figure 105: CVU Voltage Bias output



CVU using voltage sweep

The following figure shows the Advanced terminal settings for a voltage linear sweep. This sweep measures Cp-Gp.

Figure 106: CVU voltage sweep



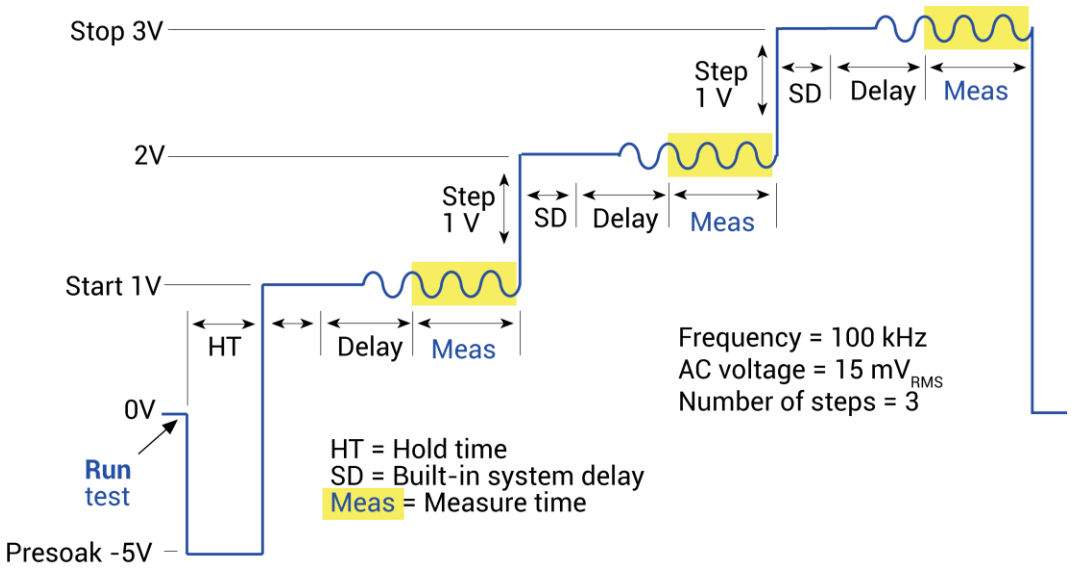
When this test is run, the following force-measure sequence occurs:

1. The DC source goes to the presoak voltage of -5 V for the hold time.
2. The DC bias voltage goes to the first step of the sweep (1 V).
3. After the built-in system delay and sweep delay, the 4210-CVU makes a measurement. The AC test signal is applied before the start of the measurement. AC drive is turned off after the measurement is made.
4. Steps 2 and 3 are repeated for the 2 V and 3 VDC bias voltage steps. The sweep delay repeats at the beginning of each subsequent step.

The sweep delay and hold time are set from the Test Settings.

To do a dual CVU voltage sweep, select **Dual Sweep** in the Terminal Settings pane. If Dual Sweep is selected, after the stop step is measured, the sweep continues in the reverse direction. For the settings shown in the above figure, the dual sweep steps through 1 V, 2 V, 3 V, 3 V, 2 V, and 1 V. The number of measurements doubles to six.

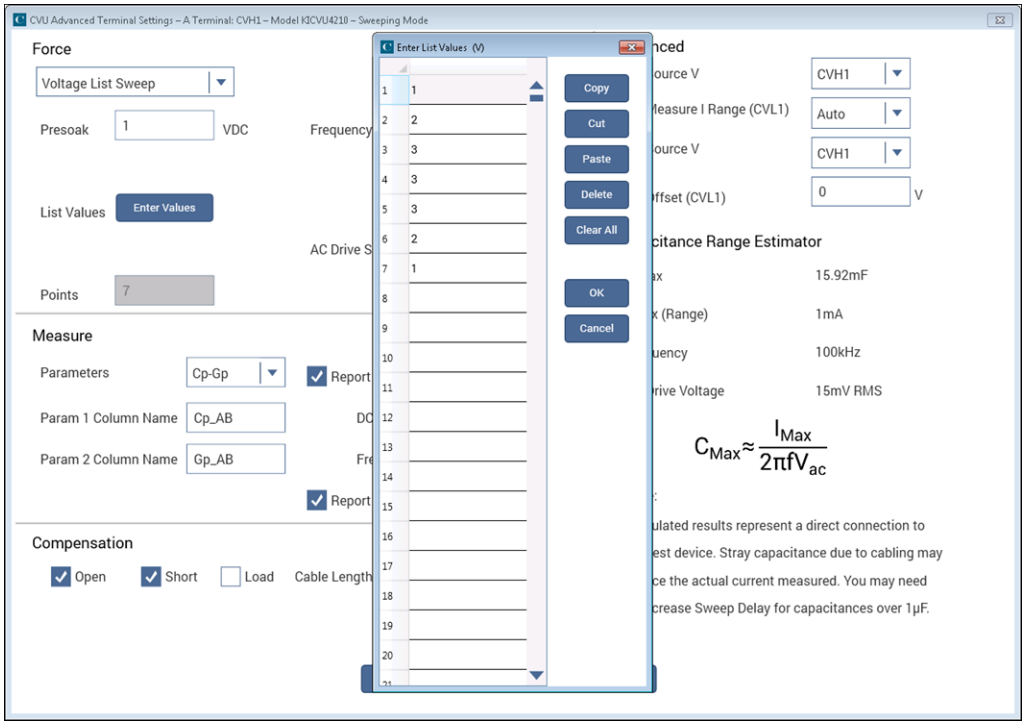
Figure 107: CVU voltage sweep output



CVU using voltage list sweep

This figure shows an example of the Advanced Terminal Settings dialog box when CVU Voltage List Sweep is selected as the operation mode, set to measure Cp-Gp.

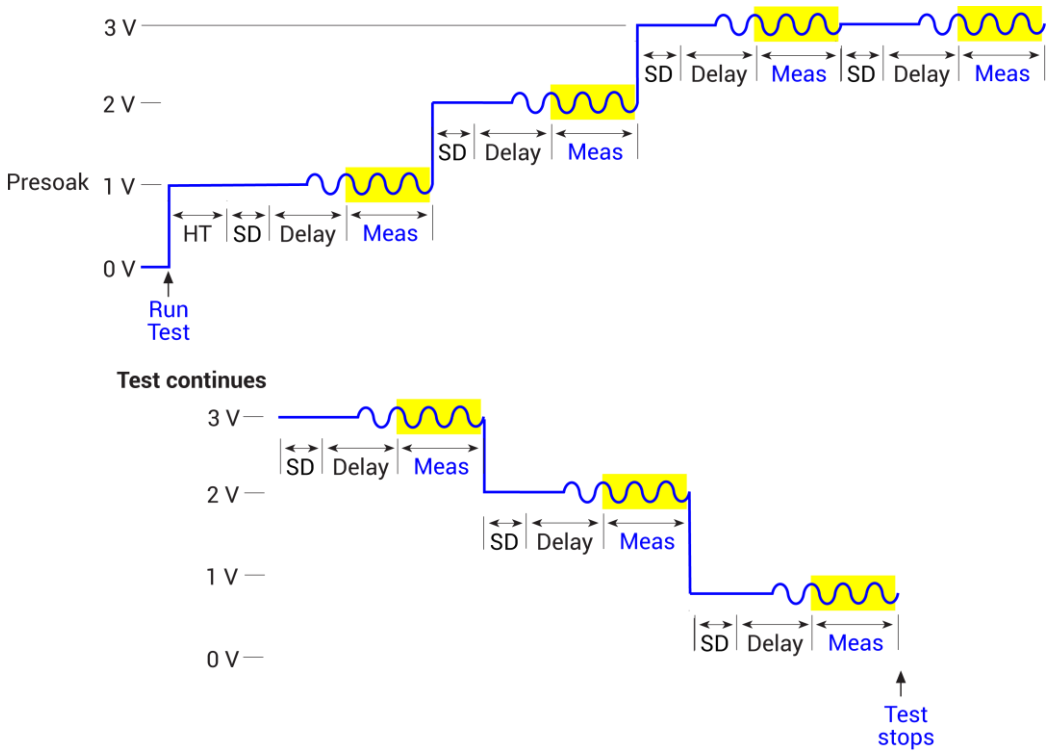
Figure 108: CVU Voltage List Sweep



When this test is run, this sequence occurs:

1. The DC source goes to the presoak voltage for the hold time. Typically, the presoak voltage is at the same potential as the first point in the sweep to allow the device to charge up to equilibrium before measurements begin.
2. The DC bias goes to the first sweep point voltage.
3. After the built-in system delay and sweep delay, the 4210-CVU makes 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. Steps 2 and 3 are repeated for remaining DC bias voltages. The sweep delay repeats at the beginning of each step.

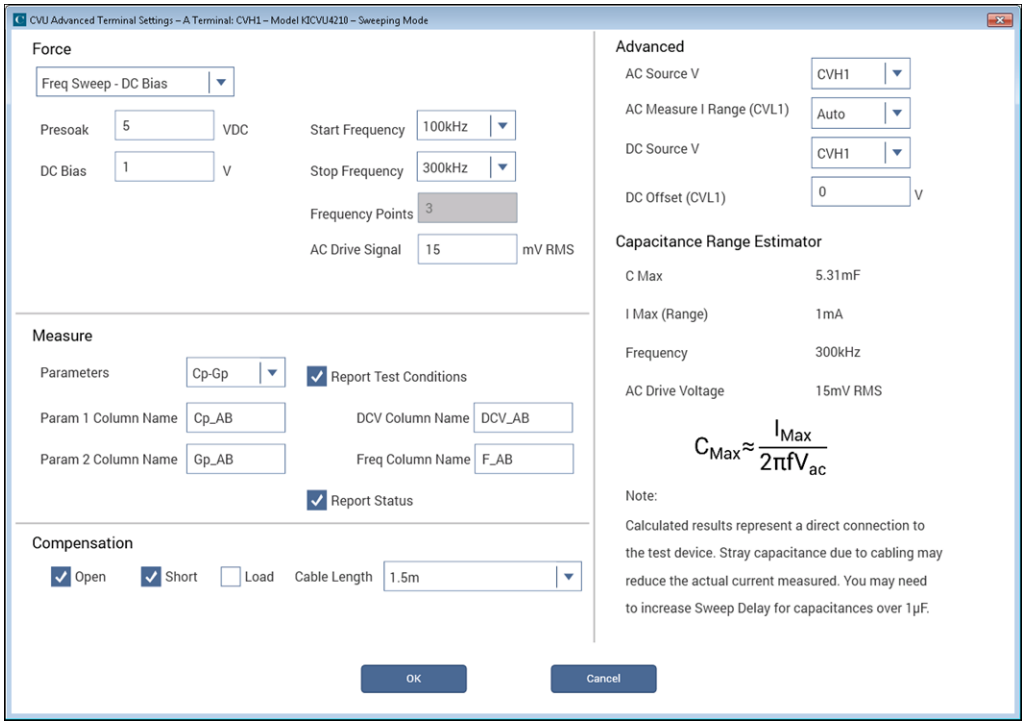
Figure 109: CVU Voltage List Sweep output



CVU using frequency sweep - DC bias

The following shows an example of the Advanced Terminal Settings dialog box set to Freq Sweep - DC Bias.

Figure 110: CVU Frequency Sweep - DC bias

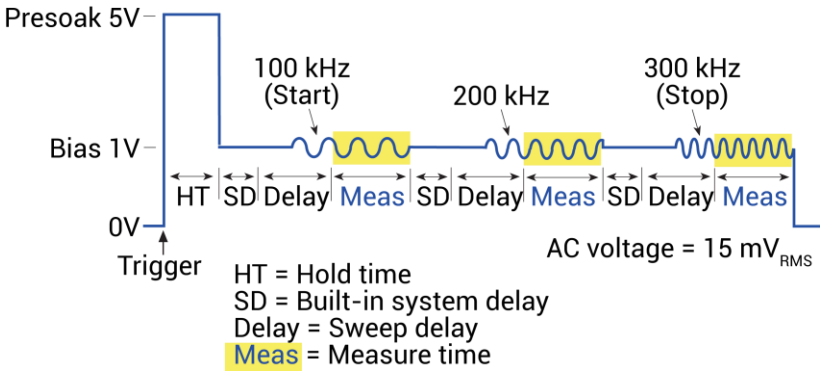


When this test is run, the following force-measure sequence occurs:

1. The DC source goes to the Presoak voltage of 5 V for the hold time.
2. The DC bias goes to 1 V for the system delay and sweep delay times and remains on during the frequency sweep.
3. The 4210-CVU makes a measurement for the first frequency point (100 kHz). The AC test signal is applied before the start of the measurement. AC drive is turned off after the measurement is made.
4. Step 3 is repeated for the other frequency points. The system delay and sweep delay are repeated for each subsequent measurement.

The sweep delay and hold time are set in the Test Settings pane.

Figure 111: CVU Frequency Sweep (bias) output



CVU frequency sweep - DC step

The following figure shows an example of the Advanced Terminal Settings dialog box for a test set to Frequency Sweep - DC Step. The test is set to measure Cp-Gp.

Figure 112: CVU (4210) Freq Sweep - DC Step

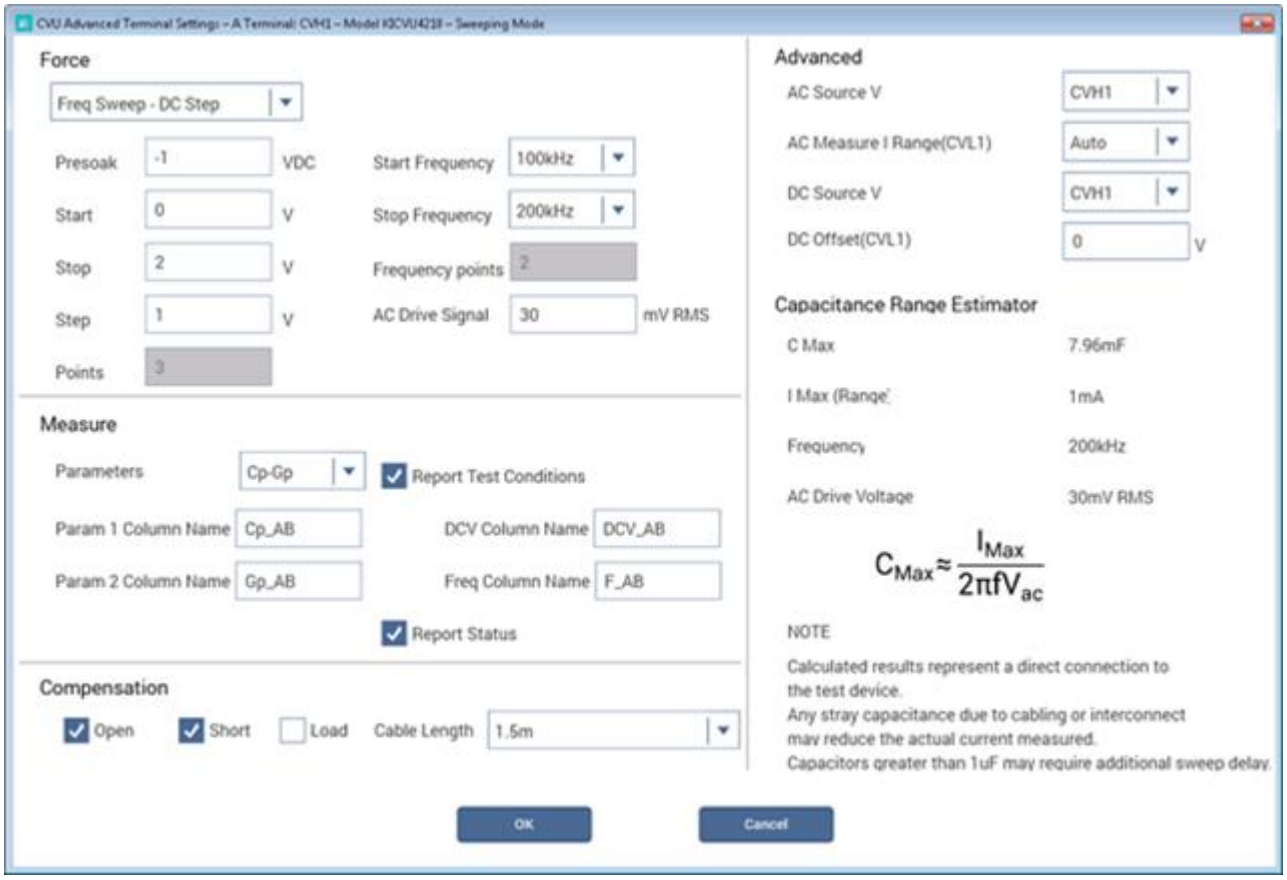
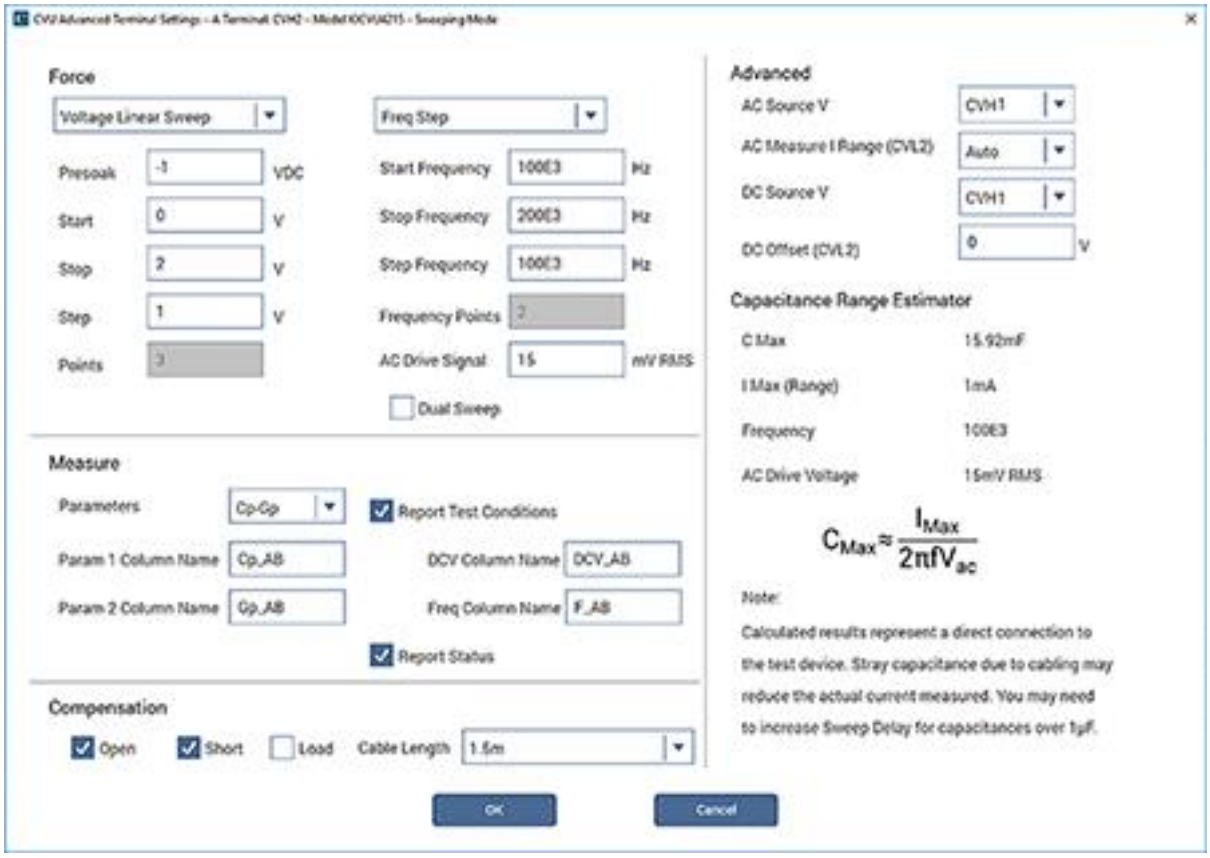


Figure 113: CVU (4215) Freq sweep - DC step

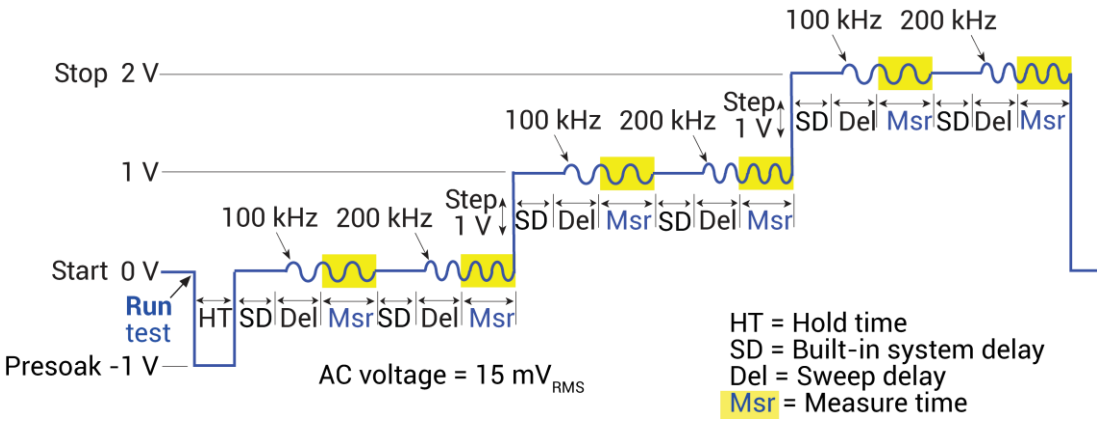


When this test is run, the following source-measure sequence occurs:

1. The DC source goes to the presoak voltage of –1 V.
2. After the hold time, DC bias goes to 0 V.
3. After the system delay and the sweep delay, the 4210-CVU makes a measurement for the 100 kHz frequency point. The AC signal is applied before the start of the measurement.
4. After another system delay and sweep delay, a measurement is made for the 200 kHz frequency point.
5. DC bias goes to 1 V.
6. Steps 3 and 4 are repeated.
7. DC bias goes to 2 V.
8. Steps 3 and 4 are repeated.

The sweep delay and hold time are set in the Test Settings pane for the test.

Figure 114: CVU Frequency Sweep (step) output



CVU DC sweep - frequency step

The following figure shows an example of the Advanced Terminal Settings dialog box for a test set to DC Sweep - Frequency Step. The test is set to measure Cp-Gp.

Figure 115: CVU DC sweep - Freq Step

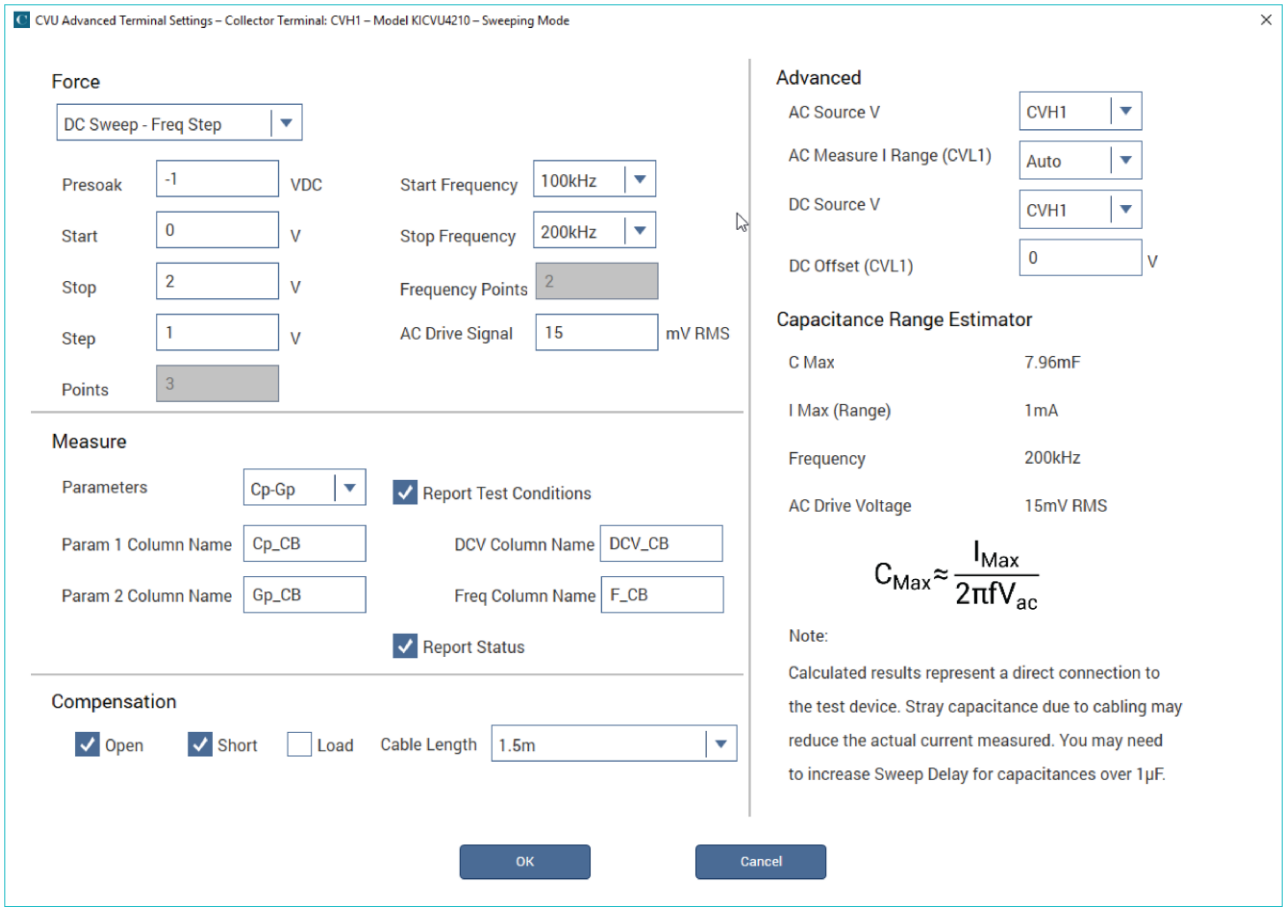
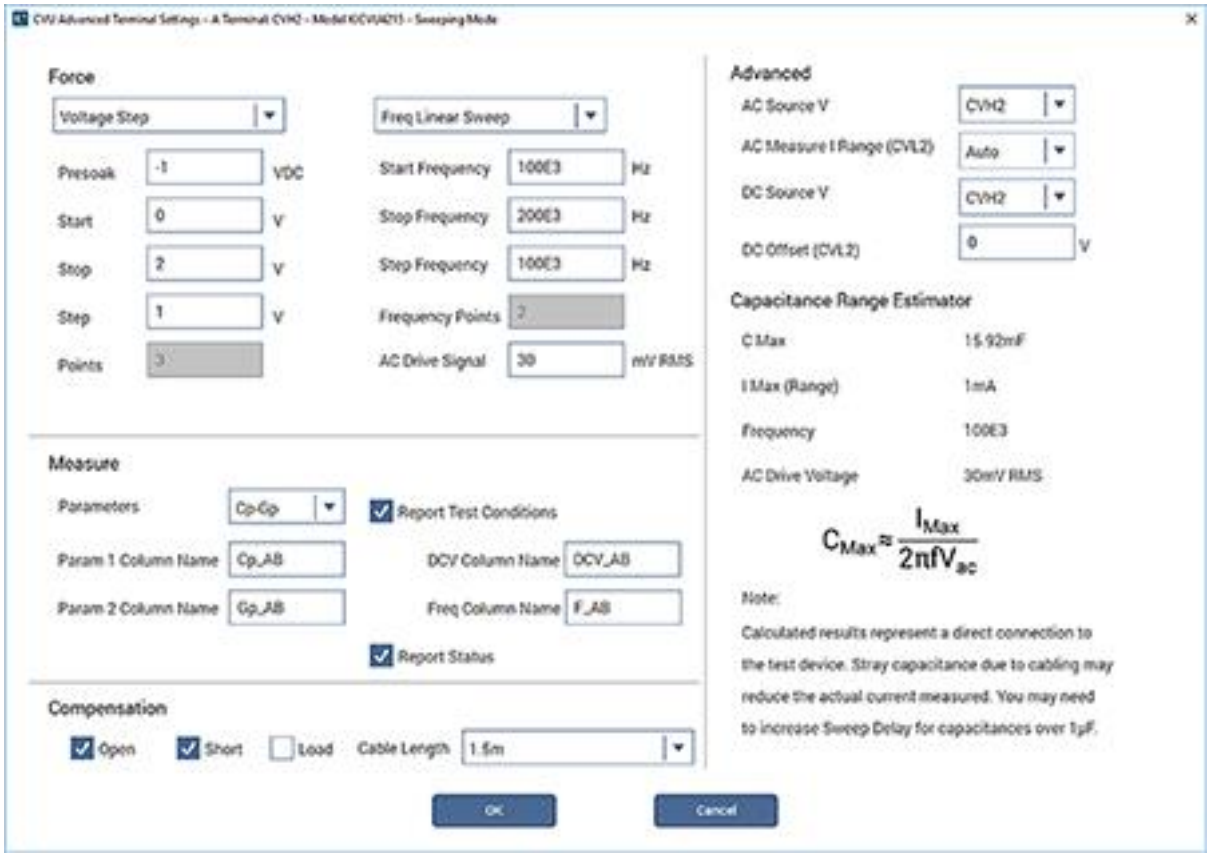


Figure 116: CVU (4215) DC sweep - Freq Step

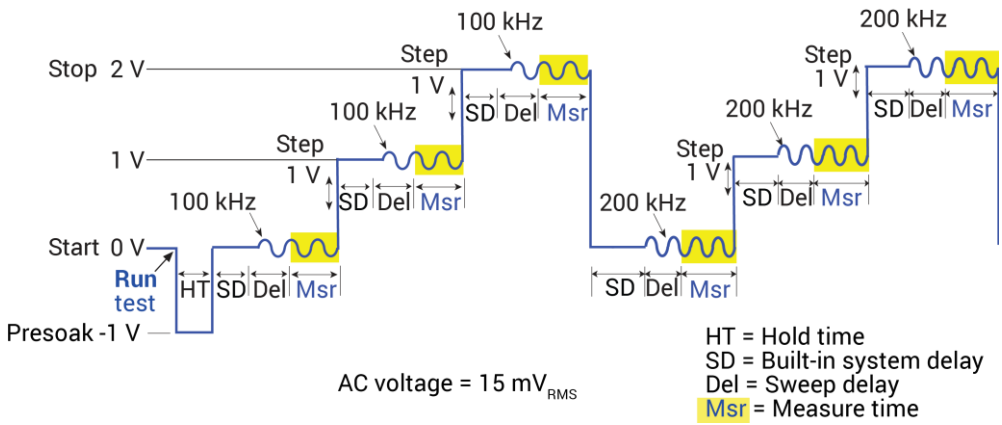


When this test is run, the following source-measure sequence occurs:

1. The DC source goes to the presoak voltage of -1.
2. After the hold time, DC bias goes to 0 V.
3. The 100 kHz frequency step point is set. The AC signal is applied before the start of the measurement.
4. After the system delay and sweep delay, a measurement is made for the 0 VDC sweep point.
5. After another system delay and sweep delay, a measurement is made for the next DC sweep point. This is repeated until the last sweep point measurement is taken.
6. The frequency goes to 200 kHz.
7. Steps 4 and 5 are repeated.

The sweep delay and hold time are set in the Test Settings pane for the test.

Figure 117: CVU DC Sweep - Freq Step output

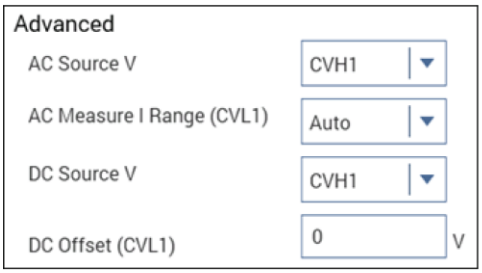


CVU Terminal Settings Advanced settings and circuits

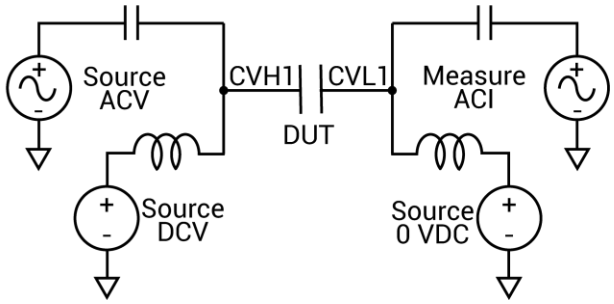
You can apply AC drive voltage and DC bias voltage to either the CVH1 terminal or the CVL1 terminal. To change the options, select the **Terminals Settings** pane, then select **Advanced.**

By default, AC source voltage is applied to the CVH1 terminal and the current measurement is made at the CVL1 terminal. Also by default, the DC source voltage (bias) is applied to the CVL1 terminal. The settings and test circuits are shown in the following figures.

Figure 118: AC source and DC source applied to CVH1

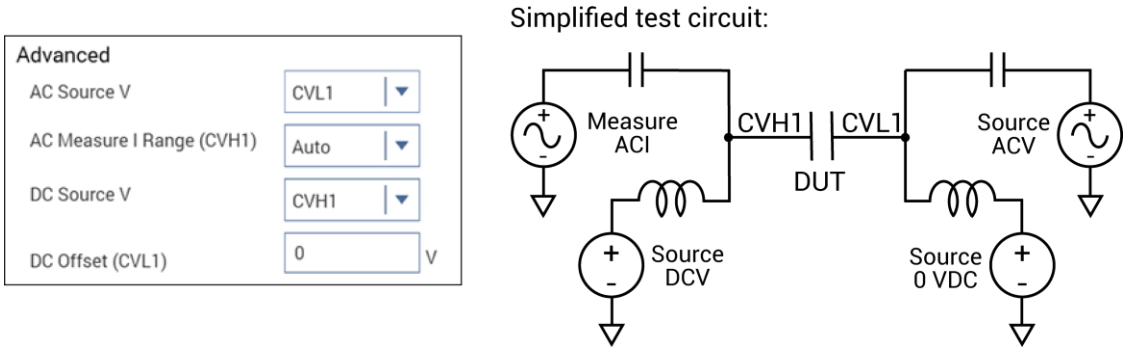


Simplified test circuit:



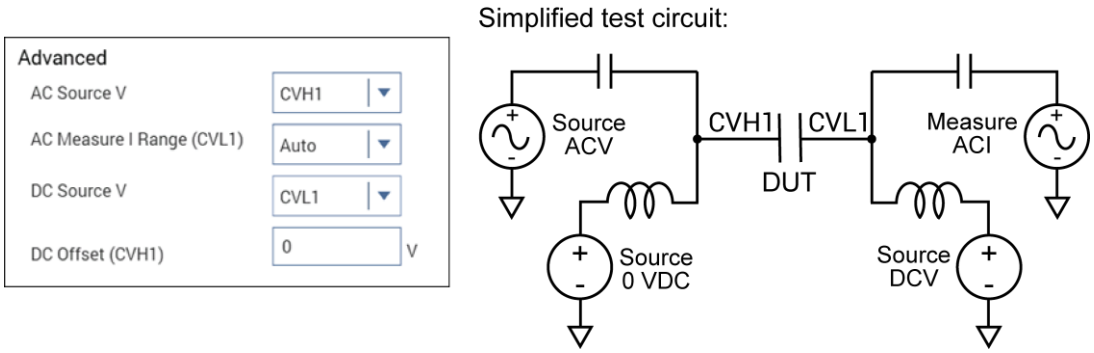
You can change the configuration to source AC voltage to CVL1 and DC source voltage to CVH1 while measuring AC current at CVH1, as shown in the following figure.

Figure 119: AC source applied to CVL1 and DC source applied to CVH1



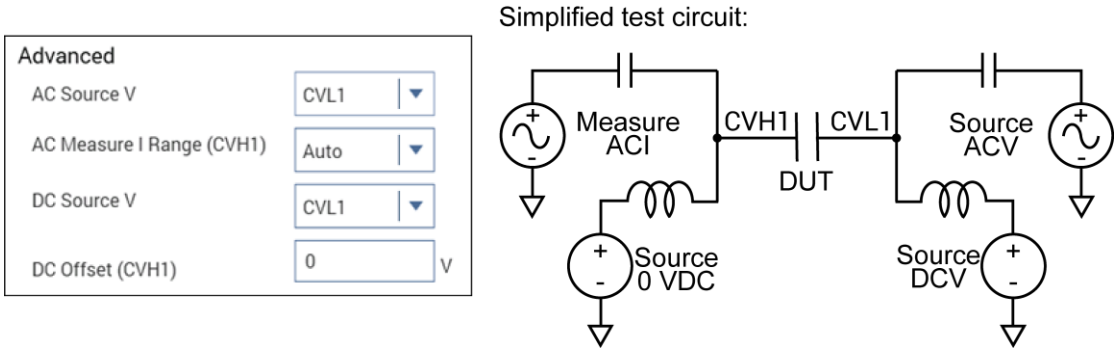
Another configuration sources AC drive voltage to the CVH1 terminal and sources DC bias voltage to the CVL1 terminal. AC current is measured at CVL1.

Figure 120: ACV source applied to CVH1 and DC bias applied to CVL1



The following settings source AC drive voltage and DC bias voltage to the CVL1 terminal. AC current is measured at CVH1.

Figure 121: ACV source and DCV bias applied to CVL1



C-V projects

A project contains tests for a testing application. The projects that support 4210-CVU testing are summarized in the next table.

42XX-CVU projects

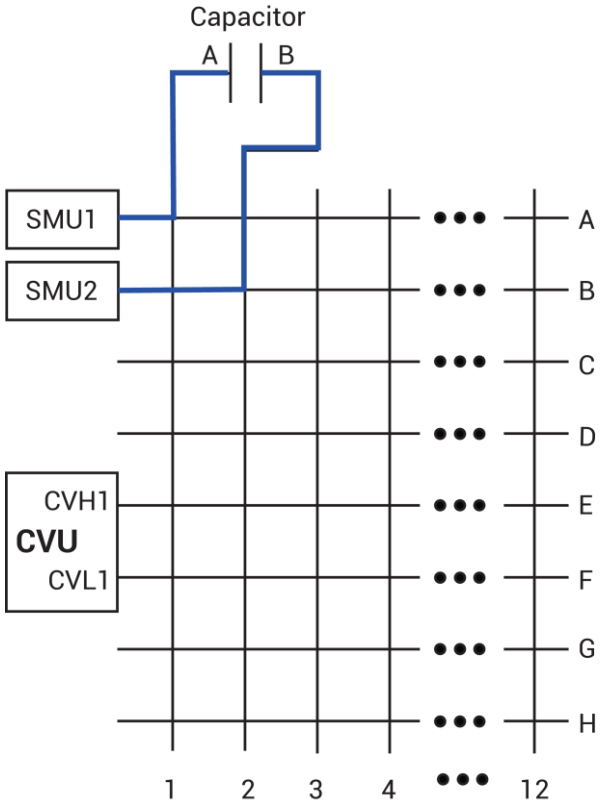
Projects	Description
cap-iv-cv-matrix (on page 4-49)	Combines a 4200-SMU, 4210-CVU, and matrix switching in one project. Closes switches on the matrix card to connect the SMUs to a device to measure I-V on a capacitor. Then it opens and closes another set of switches to make C-V measurements on a capacitor.
cvu-bjt (on page 4-55)	Measures capacitance (0 V bias) between two terminals of a bipolar junction transistor (BJT): CCB, CCE, and CBE.
cviv-bias-highv (on page 4-61)	Contains tests for making high-voltage C-V measurements on a Zener diode, MOS capacitor, capacitor, and a Schottky diode. These tests use the 4200A-CVIV bias tees to combine signals from a CVU and a SMU. The CVU measures the capacitance and the SMU supplies the DC bias.
cvu-bjt-cviv (on page 4-59)	This project has test modules that measure the capacitance as a function of time at 0 V between the terminals of a BJT: collector-base, collector-emitter, and base-emitter. You can measure this capacitance at 0 V or as a function of an applied voltage. The 4200A-CVIV switches the CVU to the BJT terminals.
cvu-highv (on page 4-60)	Contains tests for making high-voltage C-V measurements on a Zener diode, MOS capacitor, capacitor, and a Schottky diode.
cvu-moscap (on page 4-63)	Measures 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.
moscap-lifetime (on page 4-89)	Determines 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.
mosfet-cviv (on page 4-96)	Demonstrates how you can use a 4200A-CVIV Multi-Switch to automate I-V and C-V testing of a MOSFET. When the project is run, the 4200A-CVIV connects four SMUs to the MOSFET. The SMUs perform I-V tests on the MOSFET. The 4200A-CVIV then connects the CVU to the MOSFET. The CVU measures the gate to drain/source/bulk capacitance as a function of the DC voltage.
cap-measurements (on page 4-97)	Performs both a C-V sweep and a C-f sweep on a metal-insulator-metal (MIM) capacitor. Standard deviation is calculated.

Projects	Description
mosfet-cviv-cv-bias-tees (on page 4-62)	This project contains test modules that perform high-voltage CV measurements in the off-state on a 3-terminal, n-channel MOSFET.
mosfet-cviv-cv-bias-tees-400V (on page 4-62)	This project contains test modules that perform high-voltage CV measurements in the off-state on a 3-terminal, n-channel MOSFET. This isa 400 V version of mosfet-cviv-cv-bias-tees.
mosfet-dc-pulse-iv-sweeps (on page 4-62)	This project contains test modules that perform DC and pulse I-V vds-id sweeps on a MOSFET using the 4225-RPMs and displays data on the project Analyze pane.
moscap-mobile-ion (on page 4-100)	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.
mosfet (on page 4-109)	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 a test that calculates and plots the doping concentration as a function of depletion depth.
Nanowire tests (on page 4-118)	These tests perform C-V sweeps on a two-terminal nanowire device. The tests are similar but use different drive frequencies. The tests generate capacitance versus voltage graphs.
diode (on page 4-121)	This project contains DC I-V, C-V, and pulse I-V tests for a PN junction. Measures the capacitance of a PN junction or Schottky diode as a function of the DC bias voltage across the device. Has tests that include a basic C-V sweep, $1/C^2$ versus V, and a doping profile.
solarcell (on page 4-127)	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, maximum current, maximum voltage, short-circuit current, open-circuit voltage, and efficiency). The SMU also makes a reverse biased I-V measurement and performs C-V and C-f sweeps.
default (on page 4-135)	Summarizes the C-V tests that have been added to the default project. C-V testing for an n-MOSFET, diode, and a capacitor.
cntfet-characterization	This project contains DC I-V, pulsed I-V, and C-V tests for a carbon nanotube FET (CNTFET).
gate-charge (on page 4-137)	Measures the gate voltage as a function of gate charge of a transistor.

Capacitor I-V and C-V Measurements with Series 700 Project (cap-iv-cv-matrix)

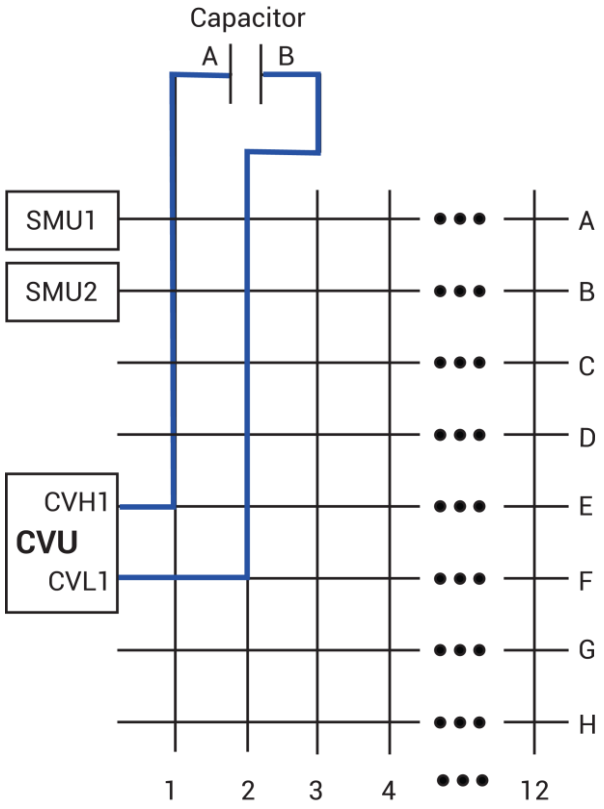
This project demonstrates how to use a Series 700 Switching System to automate I-V and C-V testing of a capacitor. When you run the project, the switching matrix connects two 4200-SMUs to the capacitor, as shown in the following figure, and I-V measurements are made.

Figure 122: Switching matrix signal path for I-V testing



The switching matrix then connects the 4210-CVU to the capacitor, as shown in the following figure, and C-V measurements are made.

Figure 123: Switching matrix signal path for C-V testing



This project is set up for use with a Series 700 with a 7174A matrix card. You must configure the switching system and the matrix card in KCon (see Add an external instrument (on page 7-4) in the KCon section). Make sure sensing is set to Local (2-wire).

NOTE

You can also use the 7072 matrix card for this project. However, you must use Rows G and H with the 7072.

cap-iv-cv-matrix project summary

This project tests a capacitor by measuring leakage current (using a 4200-SMU) and capacitance (using the 4210-CVU). These tests generate graphs for current versus time and capacitance versus time. The testing sequence is:

1. All switches for the matrix are opened.
2. Switches for the matrix card are closed to connect the SMUs to a capacitor (see A in the next figure). 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 B in the next figure).
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.

cap-iv-cv-matrix connections

The next figure shows the basic test configuration. Details on 4210-CVU connections are provided in 4210-CVU connections (on page 4-7). See Connections and configuration (on page 2-1) for details on 4200-SMU connections. Details on CVU connections to the matrix card are provided in Test connections for a switch matrix (on page 4-11).

Use only the supplied (red) 100 Ω SMA cables for connections from the matrix card (rows E and F) to the 4210-CVU. Be sure that all used SMA cables are the same length.

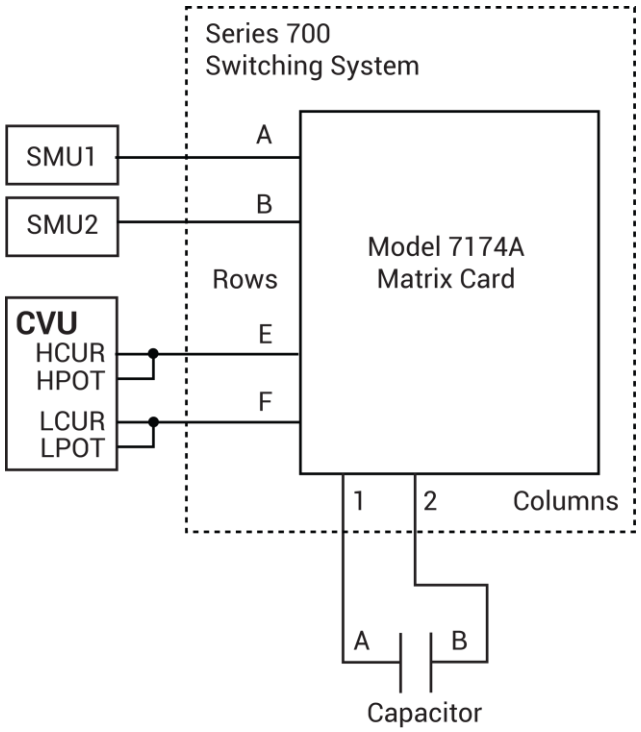
Use triaxial cables to connect the matrix cards (columns 1 and 2) to the capacitor or prober card.

Use the supplied triaxial cables to connect the SMUs to the matrix card (rows A and B).

NOTE

After making or changing connections, be sure to use the Confidence Check diagnostic tool and do connection compensation tests. Refer to Confidence Check (on page 4-27) and Connection compensation (on page 4-14) for details.

Figure 124: Simplified configuration to use a switching matrix for I-V and C-V testing



Formulas and constants

This project uses two formulas with no constants:

- AVG_CAP: Calculates the average capacitance in farads (F).
  AVG_CAP = AVG(CP_AB)
- STD_DEV: Calculates the standard deviation of the capacitance measurements.
  STD_DEV = STDEV(CP_AB)

connect test

The `connect` test controls the matrix card switches. It connects the 4200-SMUs to the capacitor.

The Configure settings are:

- The OpenAll parameter is selected. This opens all matrix switches at the beginning of the test sequence.
- The SMU1 parameter is set for pin 1. This connects SMU1 to column 1 of the matrix card. As shown in cap-iv-cv-matrix connections (on page 4-51), column 1 is connected to terminal A of the capacitor.
- The SMU2 parameter 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).

iv-cap test

In this test, 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 SMUs are:

- 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 tree, select `iv-cap` and then select Configure. From the Configure pane, parameters are set from the Test Settings pane and the Terminal Settings pane.

Analyze sheet

The test data is displayed in the Analyze sheet:

- Time: Timestamp of the measurement.
- AI: Measured current.

connect-cv test

This test controls the matrix card switches. It is used to disconnect the SMUs and then connect the 4210-CVU to capacitor.

Configure settings are:

- The OpenAll parameter is selected. This opens all switches after the `iv-cap` test is finished.
- The CVH1 parameter 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 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).

cv-capacitor test

This test 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.

Analyze sheet

The test data is displayed in the Analyze sheet:

- 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.
- CVU1S: Status code for each measurement. For details, see Measurement status (on page 6-243).
- AVG_CAP: Formulator calculation result.
- STD_DEV: Formulator calculation result.

NOTE

AB = Terminal A to Terminal B.

BJT Capacitance Tests (cvu-bjt)

The internal capacitance measurements on bipolar junction Transistors (BJTs) can affect the gain and frequency response of devices. This project includes test modules that measure the capacitance as a function of time at 0 V between the terminals of a BJT. Measurements are made between the collector-base, collector-emitter, and base-emitter. You can measure capacitance at 0 V or as a function of an applied voltage that you specify.

This project includes the following tests:

- `c-cb0`: Measures the capacitance as a function of time between the collector and base.
- `c-ce0`: Measures the capacitance as a function of time between the collector and emitter.
- `c-be0`: Measures the capacitance as a function of time between the base and emitter.

cvu-bjt connections

The next figures show the basic test configurations. Note that the untested terminal must be connected to the outer 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.

Refer to Typical 4210-CVU test connections to a DUT (on page 4-8) for connection details. Use only the supplied (red) 100 Ω SMA cables for connections to the 4210-CVU. Be sure that all SMA cables are the same length (1.5 m or 3 m).

NOTE

After making or changing connections, be sure to use the Confidence Check diagnostic tool and do connection compensation tests. Refer to Confidence Check (on page 4-27) and Connection compensation (on page 4-14) for details.

Figure 125: Basic configurations for BJT c-cb0 test

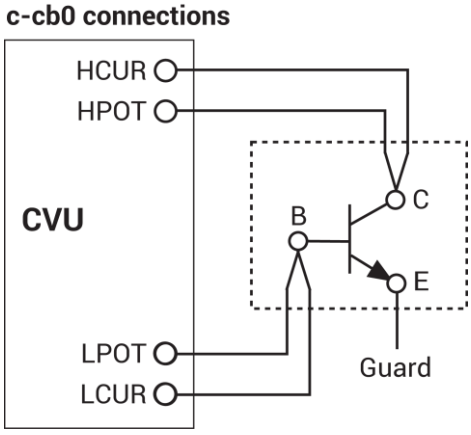


Figure 126: Basic configurations for BJT c-ce0 test

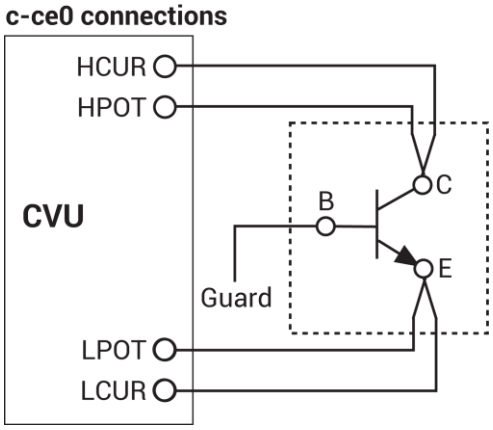
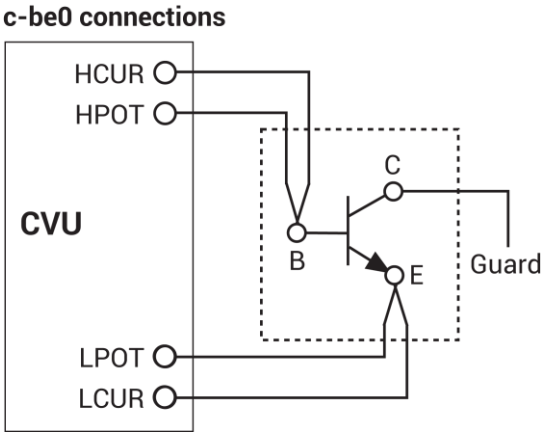


Figure 127: Basic configurations for BJT c-be0 test



Formulas and constants

This project uses two formulas with no constants:

- AVG_CAP: Calculates the average capacitance in farads (F). The formula is:
  AVG_CAP = AVG(CP_AB)
- STD_DEV: Calculates the standard deviation of the capacitance measurements. The formula is:
  STD_DEV = STDEV(CP_AB)

c-cb0 test

This test measures the capacitance as a function of time between the collector and base terminals of a BJT at 0 V. The results (C versus t) are then plotted on a graph. This test also calculates the average capacitance and standard deviation.

Analyze sheet

Test data is displayed in the Analyze sheet:

- Time: Measured parallel capacitance.
- Cp_CB: Measured parallel capacitance.
- Gp_CB: Measure conductance.
- DCV_CB: Forced DC bias voltage.
- F_CB: Forced test frequency.
- CVU1S: Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see Measurement status (on page 6-243).
- AVG_CAP: Calculated; the average capacitance in farads (F).
- STD_DEV: Calculated; the standard deviation of the capacitance measurements.

c-ce0 test

This test measures the capacitance as a function of time between the collector and emitter terminals of a BJT at 0 V. The results (C versus t) are plotted on a graph. This test also calculates the average capacitance and standard deviation.

Analyze sheet

Test data is displayed in the Analyze sheet:

- 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.
- CVU1S: Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see Measurement status (on page 6-243).
- STD_DEV: Calculated value; the standard deviation of the capacitance measurements.
- AVG_CAP: Calculated value; the average capacitance in farads (F).

c-be0 test

This test measures the capacitance as a function of time between the base and emitter terminals of a BJT at 0 V. The results (C versus t) are plotted on a graph. This test also calculates the average capacitance and standard deviation.

Analyze sheet

Test data is displayed in the Analyze sheet:

- Time: Timestamp for each measurement.
- Cp_BE: Measured parallel capacitance.
- Gp_BE: Measured conductance.
- DCV_BE: Forced DC bias voltage.
- F_BE: Forced test frequency.
- CVU1S: Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see Measurement status (on page 6-243).
- AVG_CAP: Calculated value; the average capacitance in farads (F).
- STD_DEV: Calculated value; the standard deviation of the capacitance measurements.

BJT I-V and C-V Tests Using 4200A-CVIV Multi-Switch Project (cvu-bjt-cviv)

This project has test modules that measure the capacitance as a function of time at 0 V between the terminals of a BJT: collector-base, collector-emitter, and base-emitter. You can measure this capacitance at 0 V or as a function of an applied voltage. The 4200A-CVIV switches the CVU to the BJT terminals.

This project includes several actions that perform CVU connection compensation through the 4200A-CVIV Multi-Switch using a user-defined configuration. They are:

- CVU Connection Compensation Using 4200A-CVIV (cvu-cviv-comp-collect-cb)
- CVU Connection Compensation Using 4200A-CVIV (cvu-cviv-comp-collect-ce)
- CVU Connection Compensation Using 4200A-CVIV (cvu-cviv-comp-collect-be)

The project also contains the following actions and tests:

- 4200A-CVIV Configure (cviv-configure-collector-base): Uses the 4200A-CVIV Multi-Switch to connect SMUs and the CVU to a device.
- BJT Capacitance collector-base (c-cb0): This test measures the capacitance as a function of time between the collector and base terminals of a BJT at 0 V. The results (C versus t) are then plotted on a graph. This test also calculates the average capacitance and standard deviation.
- 4200A-CVIV Configure (cviv-configure-collector-emitter): This action uses the 4200A-CVIV Multi-Switch to connect SMUs and the CVU to a device.
- BJT Capacitance collector-emitter (c-ce0): This test measures the capacitance as a function of time between the collector and emitter terminals of a BJT at 0 V. The results (C versus t) are plotted on a graph. This test also calculates the average capacitance and standard deviation.
- 4200A-CVIV Configure (cviv-configure-base-emitter): This action uses the 4200A-CVIV Multi-Switch to connect SMUs and the CVU to a device.
- BJT Capacitance base-emitter (c-be0): This test measures the capacitance as a function of time between the base and emitter terminals of a BJT at 0 V. The results (C versus t) are plotted on a graph. This test also calculates the average capacitance and standard deviation.

For additional information on using the 4200A-CVIV, refer to *Application Note: Making C-V and I-V Measurements Using the 4200A-CVIV Multi-Switch and 4200A-SCS Parameter Analyzer*.

High-Voltage C-V Tests Project (cvu-highv)

This project contains tests for making high-voltage C-V measurements on a Zener diode, MOS capacitor, capacitor, and a Schottky diode.

The 4205-RBT is included in the 4200-CVU-PWR C-V Power Package.

The tests in this project include:

- Zener Diode High-Voltage C-V Sweep (hvcv-zener): Enables high-voltage C-V measurements on a Zener diode using the CVU, SMU, and 4205-RBT remote bias tee.
- MOS Capacitor High-Voltage C-V Sweep (hvcv-moscap): Enables a high-voltage C-V sweep on a MOS capacitor using the CVU, SMU, and the 4205-RBT remote bias tee.
- Capacitor High-Voltage Bias Capacitance Measurements (200vbias): Makes a high-voltage C-V measurement on a capacitor using the CVU, SMU, and the 4205-RBT remote bias tee.
- Capacitor High-Voltage C-V Sweep (200vsweep): This test enables a high-voltage C-V sweep on a capacitor using the CVU, SMU, and the 4205-RBT remote bias tee.
- Schottky Capacitor 400 V C-V Sweep (400vsweep): This test enables a high-voltage C-V sweep on a Schottky diode using the CVU, SMU, and the 4205-RBT remote bias tee.

For additional information on high-voltage measurements, refer to *Application Note 2972: Using the Model 4200-CVU-PWR C-V Power Package to Make High Voltage and High Current C-V Measurements.*

High Voltage C-V Tests Using 4200A-CVIV Bias Tee Project (cviv-bias-highv)

This project contains tests for making high-voltage C-V measurements on a Zener diode, MOS capacitor, capacitor, and a Schottky diode. These tests use the 4200A-CVIV bias tees to combine signals from a CVU and a SMU. The CVU measures the capacitance and the SMU supplies the DC bias.

The tests in this project include:

- Zener Diode High-Voltage C-V Sweep (hvcv-zener) - Enables high-voltage C-V measurements on a Zener diode using the CVU, SMU, and either the 4200A-CVIV multi-switch or 4205-RBT remote bias tee.
- MOS Capacitor High-Voltage C-V Sweep (hvcv-moscap) - Enables a high-voltage C-V sweep on a MOS capacitor. Common parameters are derived in the Formulator, including the oxide thickness, flatband voltage, threshold voltage, and doping concentration.
- Capacitor High-Voltage Bias Capacitance Measurements (200vbias) - Enables a high-voltage capacitance measurement on a capacitor using a SMU, CVU, and the 4200A-CVIV or 4200-CVU-PWR (two 4205-RBTs).
- Capacitor High-Voltage C-V Sweep (200vsweep) - Enables a high-voltage C-V sweep on a capacitor using a SMU, CVU, and the 4200A-CVIV or 4200-CVU-PWR (two 4205-RBTs).
- Schottky Diode 400V C-V Sweep (400vsweep) - Enables a high-voltage C-V sweep on a Schottky diode using a SMU, CVU, and the 4200A-CVIV or 4200-CVU-PWR (two 4205-RBTs).

For additional information on high-voltage measurements, refer to the following Application Notes:

- Switching Between C-V and I-V Measurements Using the 4200A-CVIV Multi-Switch and 4200A-SCS Parameter Analyzer
- Switching Between C-V and I-V Measurements Using the 4200A-CVIV Multi-Switch and 4200A-SCS Parameter Analyzer (Japanese)
- Using the 4200A-CVIV Multi-Switch to Make High Voltage and High Current C-V Measurements

MOSFET 3-Terminal C-V Tests Using the 4200A-CVIV Bias Tees (mosfet-cviv-cv-bias-tees and mosfet-cviv-cv-bias-tees-400V)

This project contains test modules that perform high-voltage C-V measurements in the off-state on a 3-terminal, n-channel MOSFET. The mosfet-cviv-cv-bias-tees-400V test lets you make C-V measurements up to 400 V.

The tests in this project include:

- MOSFET Capacitance Gate-Source (cgs) - Enables high-voltage C-V measurements at the gate and source terminals while sweeping the drain terminal. This test uses the BiasT: SMU AC Gnd at the drain terminal for the AC signals.
- MOSFET Capacitance Drain-Source (cds) - Enables high-voltage C-V measurements at the drain and source terminals using the BiasT: SMU LO I CV HI and BiasT: SMU LO I CV LO modes. The gate is at BiasT: SMU AC Gnd with a 0 V bias.
- MOSFET Reverse Transfer Capacitance (crss) - Enables high-voltage C-V measurements at the drain and gate terminals using the BiasT: SMU LO I CV HI and Bias T: SMU LO I CV LO modes. The source is at BiasT: SMU AC Gnd with a 0 V bias. This test is also known as Cgd.
- MOSFET Input Capacitance (ciss) - Sweeps high-voltage at the drain and the source terminals. Connects both the drain and the source to BiasT: SMU LO I CV HI. The gate is set to Bias T: SMU LO I CV LO.
- MOSFET Output Capacitance (coss) - Enables high-voltage C-V measurements at the drain terminal. Connects both the gate and the source to BiasT: SMU LO I CV LO. The gate is set to BiasT: SMU LO I CV HI.

For additional information on high-voltage measurements, refer to *Application Note: Using the 4200A-CVIV Multi-Switch to Make High Voltage and High Current C-V Measurements.*

MOSFET DC and Pulse I-V Drain Family of Curves (mosfet-dc-pulse-iv-sweeps)

This project contains test modules that perform DC and pulse I-V vds-id sweeps on a MOSFET using the 4225-RPMs and displays data on the project Analyze pane.

The tests in this project include:

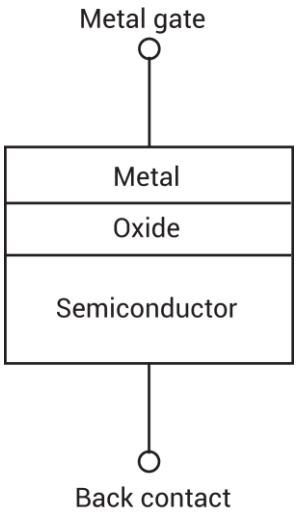
- MOSFET Drain Family of Curves (vds-id) - Generates a Vds-Id test on a MOSFET by stepping the gate voltage and sweeping the drain voltage while measuring the drain current.
- MOSFET Pulse I-V Drain Family of Curves (pulse-vds-id) - Generates a pulse I-V drain family of curves on a MOSFET.

MOS Capacitor C-V Project (cvu-moscap)

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.

An example of the construction of a MOS capacitor is shown in the next figure. As shown, the MOS capacitor is 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 128: MOS capacitor

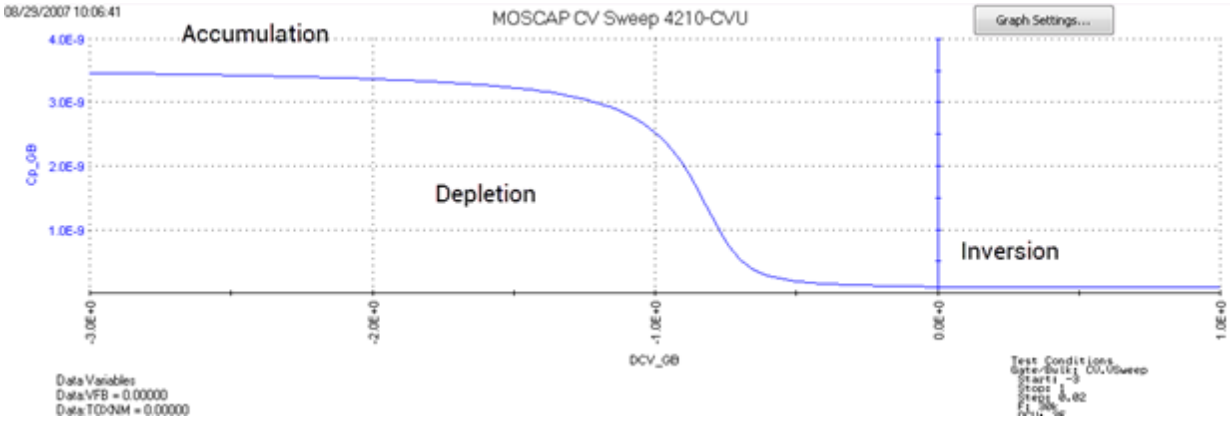


The most important property of the MOS capacitor is that its capacitance changes with changes to applied voltage. As a result, the modes of operation of the MOS capacitor change as a function of the applied voltage. As a DC sweep voltage is applied to the gate, it causes the device to pass through accumulation, depletion, and inversion.

MOS capacitor C-V curves

The next figure 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 capacitor.

Figure 129: C-V curve example for p-type MOS-C



The C-V curve for an n-type MOS capacitor is analogous to a p-type, except that:

- The majority carriers are electrons.
- The n-type MOS capacitor curve shape is essentially a mirror image of the p-type MOS capacitor curve shape.
- The accumulation region occurs at positive polarities.
- The inversion region occurs at negative polarities.

Accumulation region

For a p-type MOS capacitor, 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 capacitor, 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 capacitor, as the gate voltage moves toward positive values, the MOS capacitor 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 4210-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 capacitor, as the gate voltage increases beyond the threshold voltage, dynamic carrier generation and recombination move toward net carrier generation. Although the average net concentration of carriers in a semiconductor is stable at equilibrium, carrier generation and recombination occur dynamically.

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 high frequency-CV (HF-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.

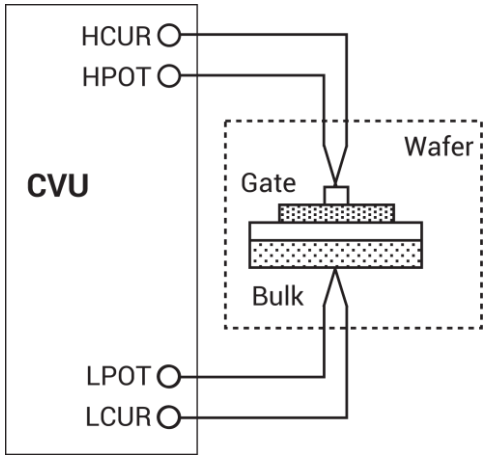
NOTE

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.

cvu-moscap connections

The next figure shows the basic test configuration for testing a MOS capacitor.

Figure 130: Simplified configuration to test MOS capacitor



4210-CVU connections (on page 4-7) provides details on connections to a semiconductor wafer. Use only the supplied (red) 100 Ω SMA cables for connections to the 4210-CVU. Be sure that all SMA cables are the same length.

NOTE

After making or changing connections, be sure to use the Confidence Check diagnostic tool and do connection compensation tests. Refer to Confidence Check (on page 4-27) and Connection compensation (on page 4-14) for details.

cvu-moscap project formulas

Formulas and user-defined constants that are used for the application tests are summarized (in alphabetical order) in the following.

Formula: AR

Formula name: AR (aR)

Units: None

Description: Intermediate parameter for calculation of corrected capacitance.

Formulator entry:

AR = GP_GB-(GP_GB^2 + (2*PI*F_GB*CP_GB)^2)*RS

Simplified equation:

$$a_R = G - \left(G^2+(2\pi fC)^2 R_s\right)$$

Formula: BEST_HI

Formula name: 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.

Formulator entry:

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)

Formula: BEST_LO

Formula name: BEST_LO

Units: None

Description: Index from DEPTHM array that is three Debye lengths from the surface.

Formulator entry:

BEST_LO = FINDD(DEPTHM, 3*DEBYEM, 2)

Formula: CADJ

Formula name: CADJ (C_{ADJ})

Units: F

Description: Corrected capacitance by compensating series resistance.

Formulator entry:

CADJ = ((GP_GB^2 + (2*PI*F_GB*CP_GB)^2)*(CP_GB)/(AR^2 + (2*PI*F_GB*CP_GB)^2)

Simplified equation:

$$C_{ADJ} = \frac{(G^2+(2\pi fC)^2)C}{a_R^2+ 2\pi fC^{\ 2}}$$

Formula: CFB

Formula name: CFB (C_{FB})

Units: F

Description: Flatband capacitance.

Formulator entry:

CFB = (COX*ES*AREA/(DEBYEM*1E2))/(COX+(ES*AREA/(DEBYEM*1E2)))

Simplified equation:

$$C_{FB} = \frac{C_{OX}\left(\frac{\varepsilon_S A}{\lambda}\right)(1 \times E^2)}{C_{OX}+\left(\frac{\varepsilon_S A}{\lambda}\right)(1 \times E^2)}$$

Formula: CMIN

Formula name: CMIN

Units: F

Description: Minimum capacitance.

Formulator entry:

CMIN = MIN(MAVG(CADJ, 10))+1E-15

Formula: COX

Formula name: COX (C_{OX})

Units: F

Description: Oxide capacitance, usually set to maximum capacitance in accumulation.

Formulator entry:

COX = MAX(MAVG(CADJ, 10))+1E-15

Formula: DEBYEM

Formula name: DEBYEM (λ)

Units: m

Description: Debye length (in meters).

Formulator entry:

DEBYEM = SQRT(ES*K*TEMP/(ABS(N90W)*Q^2))*1E-2

Simplified equation:

$$\lambda = \left(\frac{\varepsilon_S kT}{q^2 N_X}\right)^{1/2} (1xE^{-2})$$

Formula: DEPTHM

Formula name: DEPTHM (W)

Units: m

Description: Depletion depth (in meters).

Formulator entry:

DEPTHM = 1E-2*AREA*ES*(1/CADJ-1/COX)

Simplified equation:

$$W = A\varepsilon_S \left(\frac{1}{C} - \frac{1}{C_{OX}}\right)(1xE^{-2})$$

Formula: INVCSQR

Formula name: INVCSQR

Units: $1/F^2$

Description: Inverse square of capacitance.

Formulator entry:

INVCSQR = 1/(MAVG(CADJ, 5))^2

Simplified equation:

$$INVCSQR = \frac{1}{C^2}$$

Formula: MAXINVSQR

Formula name: MAXINVSQR

Units: $1/F^2$

Description: Finds row position of maximum point on $1/C^2$ curve.

Formulator entry:

MAXINVSQR = MAXPOS(INVCSQR)

Formula: N90W

Formula name: N90W

Units: None

Description: Doping density at 90% of maximum depletion depth.

Formulator entry:

N90W = AT(NDOPING, FINDD(DEPTHM, 0.9*MAX(DEPTHM), 2))

Formula: NAVG

Formula name: NAVG (N_{AVG})

Units: None

Description: Average doping calculated between index BEST_HI and BEST_LO.

Formulator entry:

NAVG = AVG(SUBARRAY(NDOPING, COND(BEST_HI, BEST_LO, BEST_HI, BEST_LO), COND(BEST_HI, BEST_LO, BEST_LO, BEST_HI)))

Formula: NDOPING

Formula name: NDOPING (N)

Units: $1/cm^3$

Description: Doping density.

Formulator entry:

NDOPING = ABS((-2)/(AREA^2*Q*ES)/(DELTA(INVCSQR)/DELTA(DCV_GB)))

Simplified equation:

$$N(W) = \frac{2}{q\varepsilon_S A^2 \left[\frac{d\left(\frac{1}{C^2}\right)}{dV}\right]}$$

Formula: NSLOPE

Formula name: NSLOPE

Units: None

Description: Finds slope of $1/C^2$ curve.

Formulator entry:

NSLOPE = LINFITSLP(DCV_GB, INVCSQR, VFBPOS, MAXINVSQR)

Formula: NSUB

Formula name: NSUB

Units: $1/cm^3$

Description: Calculated substrate doping concentration.

Formulator entry:

NSUB = 2/(NSLOPE*Q*ES*AREA^2)

Formula: PHIB

Formula name: PHIB (φ_B)

Units: V

Description: Bulk potential.

Formulator entry:

PHIB = (-1)*K*TEMP/Q*LN(ABS(N90W)/NI)*DOPETYPE

Simplified equation:

$$\phi_B = \frac{kT}{q} Ln\left(\frac{N_{BULK}}{N_i}\right) (DopeType)$$

Formula: QEFF

Formula name: QEFF (Q_{EFF})

Units: C/cm^2

Description: Effective oxide charge.

Formulator entry:

QEFF = COX*(WMS-VFB)/AREA

Simplified equation:

$$Q_{EFF} = \frac{C_{OX}(W_{MS} - V_{FB})}{A}$$

Formula: RS

Formula name: RS (R_S)

Units: Ω

Description: Series resistance calculated from capacitance.

Formulator entry:

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:

$$R_S = \frac{\left(\frac{G}{2\pi fC}\right)^2}{\left[1+\left(\frac{G}{2\pi fC}\right)^2\right]G}$$

Formula: TOXNM

Formula name: TOXNM (T_{OX})

Units: nm

Description: Calculated thickness of oxide (in nanometers).

Formulator entry: TOXNM = (1E7*AREA*EOX)/COX

Simplified equation:

$$T_{OX} = \left(\frac{A\varepsilon_{OX}(1xE^7)}{C_{OX}}\right)$$

Formula: VFB

Formula name: VFB (V_{FB})

Units: V

Description: Flatband voltage. Once CFB (CFB) is derived, V_{FB} is interpolated from the closest V_{GS} values.

Formulator entry:

VFB = AT(DCV_GB,FINDD(CADJ, CFB, 2))

Formula: VFBPOS

Formula name: VFBPOS

Units: None

Description: Finds row position of flatband voltage.

Formulator entry:

VFBPOS = FINDD(DCV_GB, VFB, 2)

Formula: VTH

Formula name: VTH (V_{TH})

Units: V

Description: Threshold voltage

Formulator entry:

VTH = VFB+DOPETYPE*(AREA/COX*SQRT(4*ES*Q*ABS(N90W*PHIB)) + 2*ABS(PHIB))

Simplified equation:

$$V_{TH} = \left[\pm \frac{A}{10^{12} C_{OX}} \sqrt{4\,\varepsilon_S\, q |N_{BULK}| |\phi_B|} + 2|\phi_B| \right] + V_{FB}$$

Formula: WMS

Formula name: WMS (W_{MS})

Units: V

Description: Work function difference between metal and semiconductor (W_M and W_S are defined in the constants area of the Formulator).

Formulator entry:

WMS = WM-(WS+(EBG/2)-PHIB)

Simplified equation:

$$W_{MS} = W_M - \left[W_S + \frac{E_G}{2} - \phi_B \right]$$

cvu-moscap project constants

The user-defined constants that are used for the application tests are summarized (in alphabetical order) in the next tables.

Constant	Default value	Units	Description
AREA	0.010404	cm^2	Gate area of device
DOPETYPE	1	none	1 = P-type, –1 = N-type
EBG	1.12	eV	E_{BG} - Semiconductor energy gap
EOX	3.4e-013	F / cm	ε_{OX} - Permittivity of oxide
ES	1.03e-012	F / cm	ε_S - Semiconductor permittivity
NI	1.45e+10	cm^{-3}	N_I - Intrinsic carrier concentration
TEMP	300	K	Test temperature
WM	4.15	V	W_M - Metal work function
WS	4.05	V	W_S - Silicon electron affinity

moscap-cvsweep test

This test makes a capacitance measurement at each step of a user-configured linear voltage sweep. Using the acquired C-V data, the Formulator calculates parameters, including oxide capacitance (C_{OX}), oxide thickness (T_{OX}), flatband capacitance (C_{FB}), flatband voltage (V_{FB}), doping density, depletion depth (DEPTHM), Debye length, threshold voltage (V_{TH}), and the effective oxide charge (Q_{EFF}).

The series resistance (R_S) is also calculated from the capacitance (in strong accumulation) and the conductance. The corrected capacitance (C_{ADJ}) is calculated by compensating for the series resistance.

From the acquired data, a capacitance versus voltage graph is generated.

moscap-cvsweep Analyze sheet

The test data is displayed in the Analyze sheet:

- 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.

moscap-c-2vsv test

This test performs a C-V sweep and displays the inverse squared capacitance ($1/C^2$) as a function of the gate voltage (V_G). This sweep can yield important information about doping profile because the substrate doping concentration (N_{SUB}) is inversely related to the reciprocal of the slope of the $1/C^2$ versus V_G curve. 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. The doping concentration is the result of the NSUB Formulator calculation.

In the following equation, *N* (NDOPING) is related to the reciprocal of the slope of the $1/C^2$ versus V_G curve.

$$N(W) = \frac{2}{q\varepsilon_S A^2 \left[\frac{d\left(\frac{1}{C^2}\right)}{dV}\right]}$$

Where:

- N(W) = 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 (V)

moscap-c-2vsv Analyze sheet

The test data is displayed in the Analyze sheet:

- 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.

moscap-dopingprofile test

This test generates a doping profile, which is a plot of the doping concentration versus depletion depth. The difference in capacitance at each step of the gate voltage is proportional to the doping concentration. The depletion depth is computed from the high-frequency capacitance and oxide capacitance at each measured value of the gate voltage. The results are plotted on the graph.

The doping concentration (*N*) is calculated in the `moscap-c-2vsv` test. The depletion depth (*W*), which is 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. 386; see References (on page 4-88)). The program computes each *W* element of the calculated data array as shown below:

$$W = A\varepsilon_S\left(\frac{1}{C} - \frac{1}{C_{OX}}\right)(1xE^{-2})$$

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)
- $1xE^{-2}$ = units conversion from cm to m

moscap-dopingprofile Analyze sheet

The test data is displayed in the Analyze sheet:

- Cp_GB: Measured parallel capacitance.
- Gp_GB: Measured conductance.
- DCV_GB: Forced DC bias voltage.
- F_GB: Forced test frequency.
- CVU1S: Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see Measurement status (on page 6-243).
- Formulas: Formulator calculation results.

NOTE

GB = gate-to-bulk.

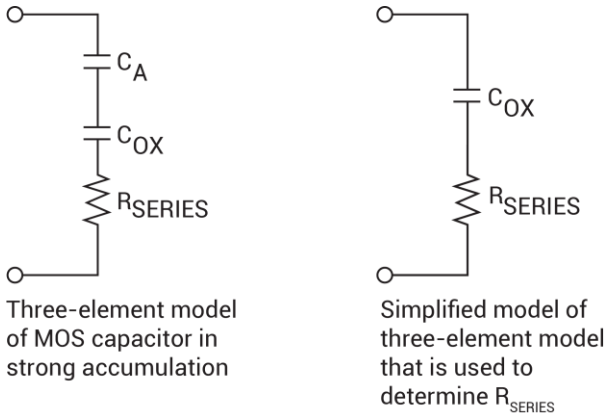
Compensating for series resistance

After generating a C-V curve, you may need to compensate the measurements 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 fabrication plants, the substrate bulk resistance is 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 the next figure.

In this model, C_{OX} is the oxide capacitance while CA is the capacitance of the accumulation layer. The series resistance is represented by R_{SERIES}.

Figure 131: Simplified model to determine series resistance



Three-element model of MOS capacitor in strong accumulation

Simplified model of three-element model that is used to determine R_{SERIES}

The corrected capacitance (C_{ADJ}) and corrected conductance (G_{ADJ}) are calculated from the formulas shown below (from Nicollian and Brews, p. 222-264; see References (on page 4-88)).

$$C_{ADJ} = \frac{(G^2+(2\pi fC)^2)C}{a_R^2+ 2\pi fC^{\ 2}}$$

$$G_{ADJ} = \frac{(G^2+(2\pi fC)^2)Q_R}{a_R^2+(2\pi fC)^2}$$

Where:

$$a_R = G - (G^2+(2\pi fC)^2 R_s)$$

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 Clarius
- 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:

$$R_S = \frac{\left(\frac{G}{2\pi fC}\right)^2}{\left[1+\left(\frac{G}{2\pi fC}\right)^2\right]G}$$

Where:

- R_S = series resistance
- G = measured conductance
- C = measured parallel model capacitance (in strong accumulation)
- f = test frequency as set in Clarius

NOTE

The above equations for compensating for series resistance require that the 4210-CVU is using the parallel model (Cp-Gp).

In this project, these formulas have been added into the Clarius Formulator so the capacitance and conductance can be automatically compensated for the series resistance.

Extracting MOS device parameters from C-V measurements

The following topics describe how MOS device parameters are extracted from C-V measurements.

Oxide thickness

For a relatively thick oxide (>50Å), 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 capacitor 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:

$$T_{OX} = \left(\frac{A\varepsilon_{OX}(1xE^{7})}{C_{OX}}\right)$$

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 x E^7 = units conversion from cm to nm

NOTE

Oxide thickness calculations based on C-V measurements can be very precise. 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 below. 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 (on page 4-88)).

The Debye equation calculates the Debye length parameter (λ) that is used in the flatband capacitance equation. 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 NA (bulk-doping concentration for a p-type acceptor material), or a user-supplied ND (bulk doping concentration for an n-type donor material).

NOTE

The flatband capacitance method is invalid when the interface trap density (DIT) becomes very large (10^{12} to 10^{13} or greater). However, the method should give satisfactory results for most users. Those dealing with high DIT values should consult the appropriate literature for a more suitable method.

Use the next two equations to calculate the flatband capacitance:

$$C_{FB} = \frac{C_{OX}\left(\frac{\varepsilon_S A}{\lambda}\right)(1 \times E^2)}{C_{OX} + \left(\frac{\varepsilon_S A}{\lambda}\right)(1 \times E^2)}$$

Where:

- CFB = flatband capacitance (F)
- C_{ox} = oxide capacitance (F)
- ε_s = permittivity of the substrate material (F/cm)
- A = gate area (cm^2)
- 1 x E^2 = units conversion from m to cm
- λ = extrinsic Debye length, which is calculated as follows:

$$\lambda = \left(\frac{\varepsilon_S kT}{q^2 N_X}\right)^{1/2} (1xE^{-2})$$

Where:

- λ = extrinsic Debye length
- ε_s = permittivity of the substrate material (F/cm)
- kT = thermal energy at room temperature (293 K) (4.046 x 10^{-21}J)
- q = electron charge (1.60219 x 10^{-19} C)
- Nx = *N* at 90% W_{MAX} or N90W (refer to Nicollian and Brews; see References (on page 4-88)) or, when input by the user, $N_X = N_A$ or $N_X = N_D$
- 1 x 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 capacitor to form a MOSFET, a p-type MOS capacitor becomes an n-type MOSFET, also called an n-channel MOSFET. Conversely, an n-type MOS capacitor becomes a p-channel MOSFET.

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:

$$V_{TH} = \left[\pm \frac{A}{10^{12} C_{OX}} \sqrt{4 \varepsilon_S q |N_{BULK}| |\phi_B|} + 2|\phi_B| \right] + V_{FB}$$

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} coulombs)
- N_{BULK} = bulk doping (cm^{-3})
- ϕ_B = bulk potential (V)
- V_{FB} = flatband voltage (V)

The physical meaning of the threshold voltage is the same for both a MOS capacitor 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:

$$V_{TH} = V_{FB} \pm \left[\frac{A}{C_{OX}} \sqrt{4 \varepsilon_S q |N_{BULK} \phi_B|} + 2|\phi_B| \right]$$

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 substrate material (F/cm)
- q = electron charge (1.60219×10^{-19} coulombs)
- N_{BULK} = bulk doping (cm^{-3}); note that the Formulator name for N_{BULK} is N90W
- ϕ_B = bulk potential (V); note that the Formulator name for ϕ_B is PHIB

The bulk potential is calculated as follows:

$$\phi_B = \frac{kT}{q} \mathrm{Ln}\left(\frac{N_{BULK}}{N_i}\right) \text{ (DopeType)}$$

Where:

- ϕ_B = bulk potential (V); note that 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} coulombs)
- N_{BULK} = bulk doping (cm^{-3}); note that the Formulator name for N_{BULK} is 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 that 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 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:

$$W_{MS} = W_M - \left[W_S + \frac{E_G}{2} - \phi_B\right]$$

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)

In tests, the values for W_M, W_S, and E_G are listed in the Formulator as constants. You can change the values depending on the type of materials.

For silicon, silicon dioxide, and aluminum:

$$W_{MS} = 4.1 - \left[4.15 + \frac{1.12}{2} - \phi_B\right]$$

$$W_{MS} = -0.61 + \phi_B$$

$$W_{MS} = -0.61 - \left(\frac{kT}{q}\right)\ln\left(\frac{N_{BULK}}{n_i}\right) \quad (DopeType)$$

Where:

- 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 = is +1 for p-type materials and -1 for n-type materials; the value for DopeType is changed in the Constants area of the Formulator

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.

For the same gate and n-type silicon ($N_{BULK} = 10^{16} cm^{-3}$), W_{MS} = –0.27 V.

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}):

$$Q_{EFF} = Q_F + Q_M + Q_{OT}$$

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 (on page 4-88)). 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:

$$V_{FB} - W_{MS} = -\frac{Q_{EFF}}{C_{OX}}$$

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:

$$Q_{EFF} = \frac{C_{OX}(W_{MS} - V_{FB})}{A}$$

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 0.01 cm^2, 50 pF, p-type MOS capacitor 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 2.5×10^{-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:

$$N_{EFF} = \frac{Q_{EFF}}{q}$$

Where:

- N_{EFF} = effective oxide charge density (cm^2)
- Q_{EFF} = effective oxide charge (C)
- q = electron charge (1.60219×10^{-19} C)

References

Nicollian, E.H. and Brews, J.R., MOS Physics and Technology, Wiley, New York (2003).

Sze, S.M., Physics of Semiconductor Devices, 2nd edition. Wiley, New York (1985).

MOS Capacitor Lifetime Test Project (moscap-lifetime)

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.

The C-V sweep derives the oxide capacitance (C_{OX}), the minimum capacitance (C_{MIN}), and the doping concentration (N_{AVG} and N_{BULK}). The capacitance-time (C-t) sweep is then 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.

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. The generation rate is graphed as a function of the calculated depletion depth.

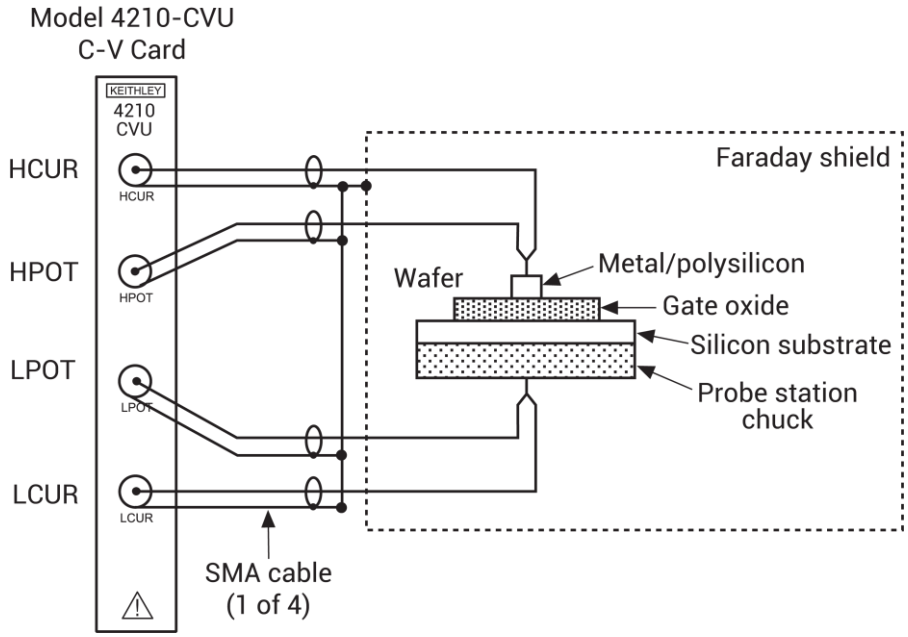
moscap-lifetime connections

The next figure shows the basic test configuration; 4210-CVU connections (on page 4-7) provides connection details. Use only the supplied (red) 100 Ω SMA cables for connections to the 4210-CVU. Be sure that all used SMA cables are the same length.

NOTE

After making or changing connections, be sure to use the Confidence Check diagnostic tool and do connection compensation tests. Refer to Confidence Check (on page 4-27) and Connection compensation (on page 4-14) for details.

Figure 132: Basic configuration for lifetime testing



moscap-lifetime formulas and constants

Tests in this project use the same formulas as the ones for the `cvu-moscap` project. It also uses additional formulas (used by the `gni-w-wf` test) and user-defined constants, which are summarized in the following tables. The formulas and constants are set from the Formulator for the tests.

Note that results of the C_{OX}, C_{MIN}, and N_{AVG} formulas (used by the `c-v` test) are needed for the `gni-w-wf` test. You must manually input the results into the Formulator for the `gni-w-wf` test.

When configuring the project, 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.

Details for referenced equations are in the documentation for Generation velocity and generation lifetime (on page D-73).

Formula: GNI

Formula name: GNI (G/ni)

Units: s^{-1}

Description: Generation rate.

Formulator entry:

GNI = -(ES*AREA*NAVG*COX)*DIFF(MAVG(1/CP^2, 5), TIME)/NI

Simplified equation:

$$\frac{G}{ni} = \frac{\varepsilon_S A N_{AVG} C_{OX}}{2}\left[\frac{\frac{1}{C^2_{t(i+1)}} - \frac{1}{C^2_{t(i-1)}}}{n_i t_{int}}\right]$$

Formula: WF

Formula name: WF (WF)

Units: cm

Description: Equilibrium inversion depth.

Formulator entry:

WF = ES*AREA*(1/MIN(CP_GB)-1/COX)

Simplified equation:

$$w_F = \varepsilon_S A\left(\frac{1}{C_{MIN}} - \frac{1}{C_{OX}}\right)$$

Formula: WWF

Formula name: WWF (W-WF)

Units: cm

Description: Depletion depth (W-WF computation).

Formulator entry:

WWF = ES*AREA*(1/CP_GB-1/COX)-WF

Simplified equation:

$$w - w_F = \varepsilon_S A\left(\frac{1}{C_{ti}} - \frac{1}{C_{OX}}\right) - w_F$$

Constants

Constant	Default value	Units	Description
AREA	0.010404	cm^2	Gate area of device
DOPETYPE	1	none	`1` = P-type, `-1` = N-type
EBG	1.12	eV	E_{BG} - Semiconductor energy gap
EOX	3.4e-13	F / cm	ε_{OX} - Permittivity of oxide
ES	1.03e-12	F / cm	ε_S - Semiconductor permittivity
NI	1.45e+010	cm^{-3}	N_I - Intrinsic carrier concentration
TEMP	300	K	Test temperature
WM	4.15	V	W_M - Metal work function
WS	4.05	V	W_S - Silicon electron affinity

c-v test

This test generates a C-V sweep on a MOS capacitor and derives C_{ox}, C_{min}, and N_{avg}. These parameters are input into the MOS Capacitor Zerbst C-t Sweep (C-t) test and the MOS Capacitor Generation Rate versus Depletion Depth (GNI_W-WF) test using the Formulator.

c-v Analyze sheet

Test data is displayed in the Analyze sheet:

- 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.

c-t test

This test drives a MOS capacitor into accumulation by applying a negative hold voltage for a period (hold time). The bias voltage is then reversed to drive the capacitor into depletion. While in depletion, a series of capacitance measurements are made at a set time interval. The capacitance versus time measurements are plotted on a graph.

When configuring the test, you need to set the DC bias hold voltage (for accumulation) and the DC bias voltage level (for depletion). You also set the hold time for the bias hold voltage (in accumulation).

c-t Analyze sheet

The test data is displayed in the Analyze sheet:

- 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.
- CVU1S: Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see Measurement status (on page 6-243).
- Formulas: Formulator calculation results.

NOTE

GB = Gate-to-bulk.

gni-w-wf test

This test is similar to the `c-t` test because it creates a Zerbst plot by biasing the device in accumulation for a specified period. However, it calculates the generation rate (GN_I) and the depletion depth (WW_F) for every measurement, and then generates a GN_I versus WW_F Zerbst plot.

This test creates a Zerbst plot by first biasing the device in accumulation for a specified 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 ($w\text{-}w_F$). You must manually 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 $w\text{-}w_F$ 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:

$$\frac{G}{ni} = \frac{\varepsilon_S A N_{AVG} C_{OX}}{2}\left[\frac{\frac{1}{C^2_{t(i+1)}} - \frac{1}{C^2_{t(i-1)}}}{n_i t_{int}}\right]$$

Where:

- G/n_i = 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 Analyze sheet) when the C-V test is run. The value for this parameter must be input into the Formulator for the `gni-w-wf` test. You can input a known value of N_{AVG} into the Formulator instead.

The equilibrium inversion depth (w_F) is calculated as:

$$w_F = \varepsilon_S A\left(\frac{1}{C_{MIN}} - \frac{1}{C_{OX}}\right)$$

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 Analyze sheet) when the C-V 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:

$$w - w_F = \varepsilon_S A\left(\frac{1}{C_{ti}} - \frac{1}{C_{ox}}\right) - w_F$$

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 Analyze sheet) 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.

gni-w-wf test procedure

1. Run the C-V test.
2. Get the calculation results for the C_{OX}, C_{MIN} and N_{AVG} formulas from the Analyze sheet.
3. Input the values for C_{OX}, C_{MIN}, and N_{AVG} into the Formulator for the `gni-w-wf` test (for example, C_{OX} = 145E-12).
4. Run the `gni-w-wf` test to generate the Zerbst plot.

NOTE

If the value for the N_{AVG} in the Formulator is known, it can be used instead of the one derived from the C-V test.

gni-w-wf Analyze sheet

Test data is displayed in the Analyze sheet:

- 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.
- CVU1S: Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see Measurement status (on page 6-243).
- Formulas: Formulator calculation results.

NOTE

GB = gate-to-bulk.

MOSFET I-V and C-V Tests Using 4200A-CVIV Multi-Switch Project (mosfet-cviv)

This project demonstrates how you can use a 4200A-CVIV Multi-Switch to automate I-V and C-V testing of a MOSFET. When the project is run, the 4200A-CVIV connects four SMUs to the MOSFET. The SMUs perform I-V tests on the MOSFET. The 4200A-CVIV then connects the CVU to the MOSFET. The CVU measures the gate to drain/source/bulk capacitance as a function of the DC voltage.

Capacitor Measurements (cap-measurements)

This project performs a capacitance-voltage (C-V) sweep and a capacitance-frequency (C-f) 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.

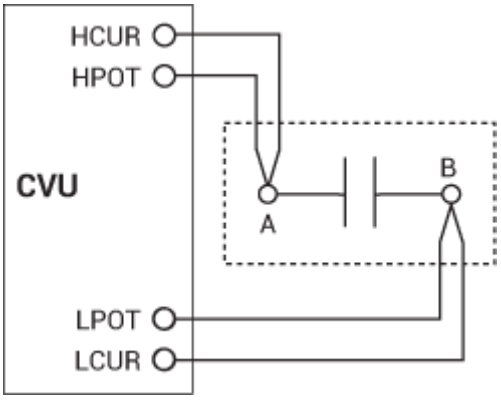
cvu-capacitor connections

The test configuration is shown in the next figure. Use only the supplied (red) 100 Ω SMA cables for connections to the 4210-CVU. Be sure that all SMA cables are the same length.

NOTE

After making or changing connections, be sure to use the Confidence Check diagnostic tool and do connection compensation tests. Refer to Confidence Check (on page 4-27) and Connection compensation (on page 4-14) for details.

Figure 133: Basic configuration to test a MIM capacitor



Formulas and constants

This project uses one formula with no constants.

- NOISE: Calculates the standard deviation of the capacitance measurements:
  NOISE = STDEV(CP_AB)

cv-10 pF test

This test performs a voltage sweep that measures capacitance on each step and generates a C versus V graph. It also calculates noise.

Analyze sheet

Test data is displayed in the Analyze sheet:

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 (on page 6-243).
NOISE	Formulator calculation result.

NOTE: AB = Terminal A to Terminal B.

cf-10 pF test

This test performs a frequency sweep that measures capacitance at each frequency point and generates a C versus F graph.

Analyze sheet

Test data is displayed in the Analyze sheet:

Cp_AB	Measured parallel capacitance.
Gp_AB	Measured conductance.
DCV_AB	Forced DC bias voltage.
F_AB	Forced test frequency.
CVU1S	Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status (on page 6-243).

NOTE: AB = Terminal A to Terminal B.

Interconnect Capacitance C-V Sweep test (cv-sweep)

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 (<1 pF).

This test uses a voltage sweep to measure capacitance at every step of the sweep and generates a capacitance versus voltage graph. It also calculates noise.

This test module makes a C-V sweep. The measurements are set to Quiet mode because the measurements are very sensitive.

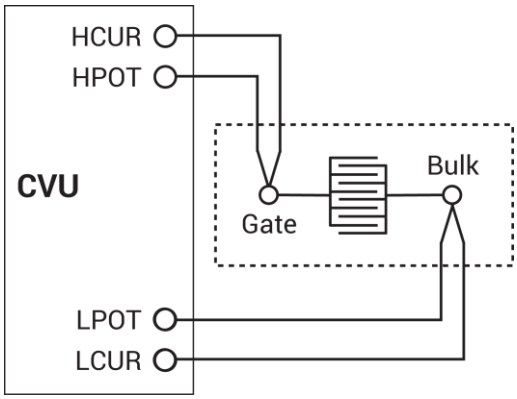
cv-sweep connections

The test configuration in the next figure shows the schematic representation of interconnect capacitance on a wafer. For details on connections to the wafer, see 4210-CVU connections (on page 4-7). Use only the supplied (red) 100 Ω SMA cables for connections to the 4210-CVU. Be sure that all used SMA cables are the same length.

NOTE

After making or changing connections, be sure to use the Confidence Check diagnostic tool and do connection compensation tests. Refer to Confidence Check (on page 4-27) and Connection compensation (on page 4-14) for details.

Figure 134: Basic configuration to test interconnect capacitance



Formulas and constants

This project uses one formula (no constants):

- NOISE: Calculates the standard deviation of the capacitance measurements.
  NOISE = STDEV(CP_GB)

Analyze sheet

Test data is displayed in the Analyze sheet:

- Cp_GB: Measured parallel capacitance.
- Gp_GB: Measured conductance.
- DCV_GB: Forced DC bias voltage.
- F_GB: Forced test frequency.
- CVU1S: Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see Measurement status (on page 6-243).
- NOISE: Formulator calculation result.

NOTE

GB stands for gate-to-bulk.

MOS Capacitor Mobile Ion Project (moscap-mobile-ion)

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 (*Vfb)* is used to determine the amount of charge. The flatband voltage is measured both at room temperature and after the device has been at an elevated temperature for enough time 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)

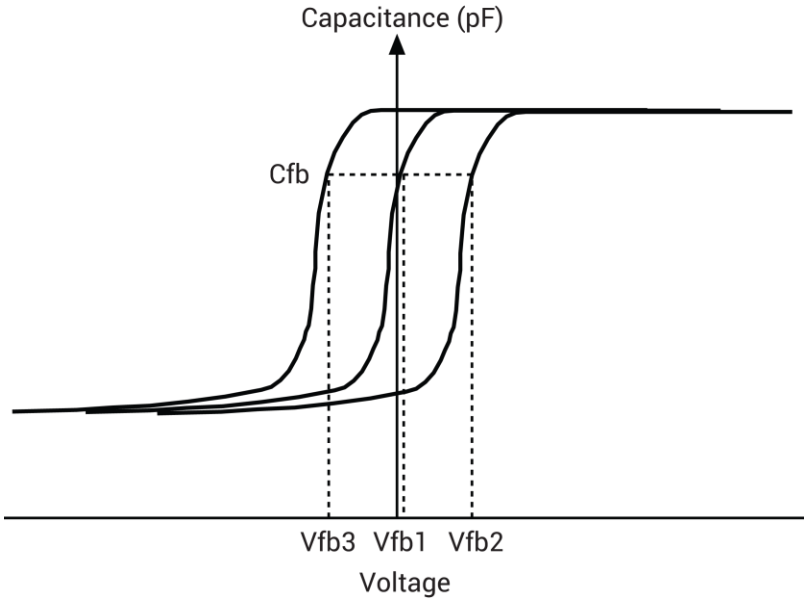
This semi-automatic project is executed from the subsite level. The oxide thickness and flatband voltage extracted from the data in the tests are sent to the subsite level Analyze sheet after the tests are executed. The calculated value of the mobile ion charge and concentration can be found in the Calc tab of the subsite Analyze sheet.

Before executing this test, you need to input the gate area of the device into the Constants area of the Formulator for the `CV-Vfb1`, `CV-Vfb2`, and `CV-Vfb3` test modules. Also, in the `CV-Vfb1` test, you must update the gate area in the formula area of the Formulator. The gate area formula to update is called `AREA_GATE` and is used in the calculation of the mobile ion charge in the subsite Calc sheet.

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 the next figure.

Figure 135: Flatband voltage shift caused by mobile charges in the oxide



6. The mobile ion charge is calculated as follows:

$$Q_m = \frac{|V_{fb3} - V_{fb2}| C_{OX}}{A}$$

Where:

- Q_m = mobile ion charge (C)
- V_{fb2} = flatband voltage measured second time after temperature stress (V)
- V_{fb3} = 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:

$$N_{mi} = \frac{Q_m}{q}$$

Where:

- N_{mi} = mobile ion concentration
- Q_m = mobile ion charge (C)
- q = electron charge

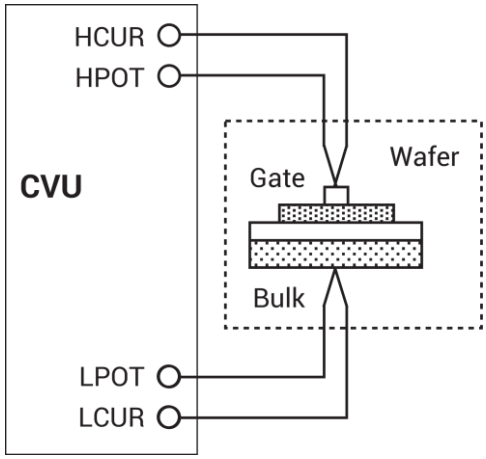
moscap-mobile-ion connections

The next figure shows the basic test configuration. Refer to 4210-CVU connections (on page 4-7) for details on connections to a semiconductor wafer. Only use the supplied (red), same length 100 Ω SMA cables for connections to the 4210-CVU.

NOTE

After making or changing connections, be sure to use the Confidence Check diagnostic tool and do connection compensation tests. Refer to Confidence Check (on page 4-27) and Connection compensation (on page 4-14) for details.

Figure 136: Basic configuration for mobile ion testing



Formulas and constants

This project uses formulas and constants that are used by the `cvu-moscap` project. The formulas and constants are summarized in cvu-moscap project formulas and constants (on page 4-67).

The `cv-vfb1` test also uses the following formula for gate area:

AREA_GATE = 2

Run the subsite

This semi-automatic test is executed from the `mobileion` subsite in the project. The test is run from this subsite because the oxide thickness and flatband voltage extracted from the data in the tests is sent to the subsite Analyze sheet after the tests are executed. The calculated value of the mobile ion charge and concentration is shown in the Calc tab of the Analyze sheet for the subsite.

Before executing this subsite, you need to input the gate area of the device into the Constants tab of the Formulator for the `cv-vfb1`, `cv-vfb2`, and `cv-vfb3` test modules. Also, for the `cv-vfb1` test, you must also update the `AREA_GATE` formula in the Formulator. This formula is used in the calculation of the mobile ion charge in the subsite Calc sheet.

This project must be run from the `mobileion` subsite level.

cv-vfb1 test

This test performs a C-V sweep on the MOS capacitor and extracts flatband capacitance, flatband voltage, and oxide capacitance. The oxide capacitance and the flatband voltage are sent to the Analyze sheet of the subsite. To view the values that are used in the mobile ion calculation, select the Calc tab in the subsite Analyze sheet.

cv-vfb1 Analyze sheet

The test data is displayed in the Analyze sheet:

- Time: Timestamp for each measurement (if Report Timestamps is selected).
- Cp_GB: Measured parallel capacitance.
- Gp_GB: Measured conductance.
- DCV_GB: Forced DC bias voltage.
- F_GB: Forced test frequency.
- Formulas: Formulator calculation results.

If rows are highlighted in blue, a fault occurred. For details, see Measurement status (on page 6-243).

NOTE

GB = gate-to-bulk.

bias-pos test

This test applies a positive DC bias voltage to the MOS capacitor. The voltage continues to be applied as the device under test (DUT) is heated in the next action, `hotchuck`.

No graph is generated for this test.

Formulator formulas and constants

Formulas are not used for this test.

bias-pos Analyze sheet

The test data is displayed in the Analyze sheet:

- 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.
- CVU1S: Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see Measurement status (on page 6-243).

NOTE

GB = gate-to-bulk.

hotchuck action

This action prompts you to turn on the hot chuck to a specified temperature and then to cool down the sample. After the temperature stress, select **OK** to generate the C-V sweep in the following test. The bias voltage is output until you select **OK**.

cv-vfb2 test

This test performs a C-V sweep on the sample after it has been heated and then cooled. The oxide capacitance and the flatband voltage (V_{fb}) are sent to the Analyze sheet for the subsite. Select the Calc tab of the Analyze sheet to review the values that are used in the mobile ion calculation.

cv-vfb2 Analyze sheet

The test data is displayed in the Analyze sheet:

- Time: Timestamp for each measurement (if Report Timestamps is selected).
- Cp_GB: Measured parallel capacitance.
- Gp_GB: Measured conductance.
- DCV_GB: Forced DC bias voltage.
- F_GB: Forced test frequency.
- Formulas: Formulator calculation results.

If rows are highlighted in blue, a fault occurred. For details, see Measurement status (on page 6-243).

NOTE

GB = gate-to-bulk.

bias-neg test

This test applies a negative DC bias voltage to the MOS capacitor. The voltage continues to be applied to the sample as the DUT is heated in the next test module, `hotchuck`.

Formulator formulas and constants

Formulas are not used for this test.

bias-neg Analyze sheet

The test data is displayed in the Analyze sheet:

- 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.
- CVU1S: Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see Measurement status (on page 6-243).

NOTE

GB = gate-to-bulk.

hotchuck action

This action prompts you to turn on the hot chuck to a specified temperature and then to cool down the sample. After the temperature stress, select **OK** to generate the C-V sweep in the following test. The bias voltage is output until you select **OK**.

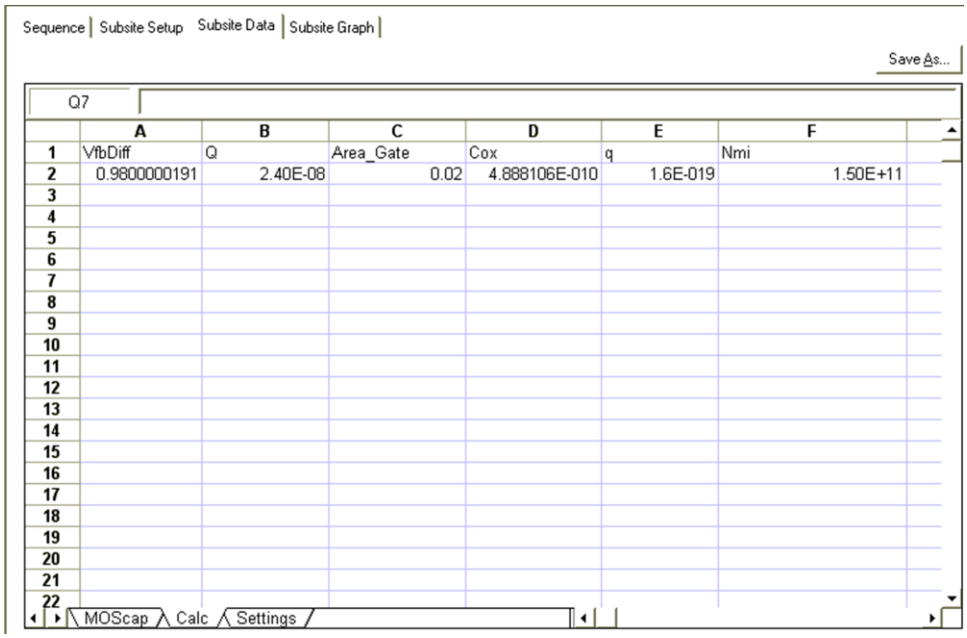
cv-vfb3 test

This test performs a C-V sweep on the sample after it has been heated and then cooled. The oxide capacitance and the flatband voltage (V_{fb}) are sent to the Analyze sheet for the subsite. View the Calc tab in the Analyze sheet to review the values that are used in the mobile ion calculation.

To open the Calc sheet for the subsite:

1. In the project tree, select **mobileion**.
2. Select **Analyze**.
3. Select the **Calc** tab.

Figure 137: Calc sheet for the subsite level (MobileIon)



cv-vfb3 Analyze sheet

The test data is displayed in the Analyze sheet:

- Time: Timestamp for each measurement (if Report Timestamps is selected).
- Cp_GB: Measured parallel capacitance.
- Gp_GB: Measured conductance.
- DCV_GB: Forced DC bias voltage.
- F_GB: Forced test frequency.
- Formulas: Formulator calculation results.

If rows are highlighted in blue, a fault occurred. For details, see Measurement status (on page 6-243).

NOTE

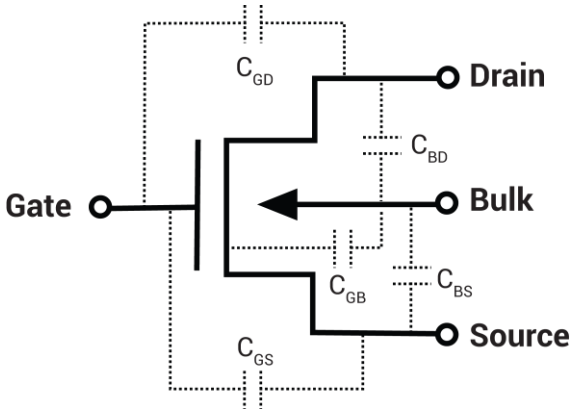
GB = gate-to-bulk.

MOSFET Project (mosfet)

This project contains DC I-V, C-V, and pulse I-V tests for a MOSFET. It makes C-V measurements to explore the basic operation and parameters of the device. Since the high-frequency operation and switching speed of a MOSFET are dependent on the capacitance of the device, capacitance measurements are often made to various parts of the device. 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.

Capacitance measurements are made to various parts of the device, as shown in the following figure.

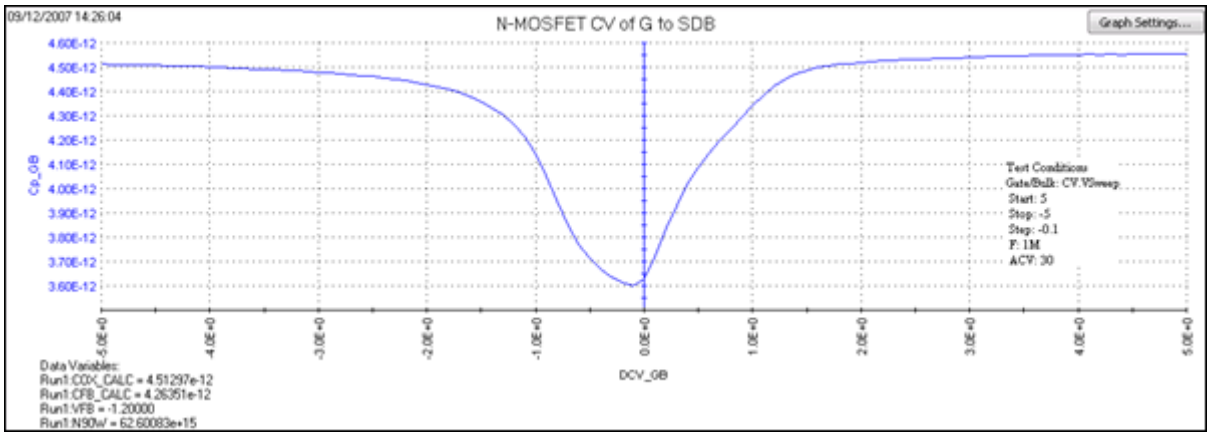
Figure 138: N-MOSFET with terminal capacitances



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` test generates a capacitance versus voltage graph. It 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 the next figure.

Figure 139: N-channel MOSFET g-to-sdb graph



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.

The oxide capacitance (C_{OX}) is usually set to the maximum capacitance in accumulation, and is calculated by the C_{OX} formula in the Formulator.

Oxide thickness is calculated by the T_{OX} formula in the Formulator.

The `mosfet-dopingprofile` test performs a C-V sweep on the two-terminal MOSFET. It generates a doping concentration versus depletion depth graph. The doping concentration (*N*) is calculated and plotted as a function of depletion depth. Depletion depth is calculated by the DEPTHM formula in the Formulator.

Doping density is calculated from the measured capacitance and the voltage. It is calculated by the NDOPING formula in the Formulator.

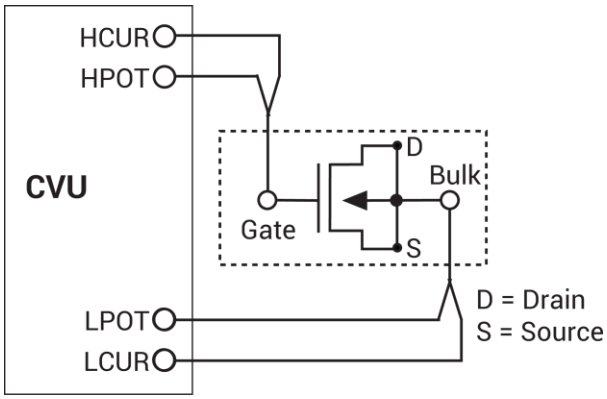
mosfet connections

The next figure shows the basic test configuration for MOSFET testing (for details, see 4210-CVU connections (on page 4-7)). As shown, 2-wire sense connections are used with the source, drain, and bulk terminals tied together. Use only the supplied CA-446A or CA-447A red 100 Ω SMA cables for connections to the 4210-CVU. Be sure that all used SMA cables are the same length.

NOTE

After making or changing connections, be sure to use the Confidence Check diagnostic tool and do connection compensation tests. Refer to Confidence Check (on page 4-27) and Connection compensation (on page 4-14) for details.

Figure 140: Two-terminal C-V test configuration for a MOSFET



mosfet formulas and constants

Formulas and user-defined constants that are used for the application tests are summarized (in alphabetical order) in the next topics. The formulas and constants are set from the Formulator settings for the tests.

Formula: AR

Formula name: AR (*aR*)

Units: None

Description: Intermediate parameter for calculation of corrected capacitance.

Formulator entry:

AR = GP_GB-(GP_GB^2 + (2*PI*F_GB*CP_GB)^2)*RS

Simplified equation:

$$a_R = G - \left(G^2+(2\pi fC)^2 R_s\right)$$

Formula: CADJ

Formula name: CADJ (*CADJ*)

Units: F

Description: Corrected capacitance by compensating series resistance.

Formulator entry:

CADJ = ((GP_GB^2 + (2*PI*F_GB*CP_GB)^2)*CP_GB)/(AR^2 +(2*PI*F_GB*CP_GB)^2)

Simplified equation:

$$C_{ADJ} = \frac{\left(G^2+(2\pi fC)^2\right)C}{a_R^2+ 2\pi fC^{\ 2}}$$

Formula: CFB_CALC

Formula name: CFB_CALC (*CFB*)

Units: F

Description: Flatband capacitance.

Formulator entry:

CFB_CALC = (COX_CALC*ES*AREA/(DEBYEM*1E2))/(COX_CALC+(ES*AREA/(DEBYEM*1E2)))

Simplified equation:

$$C_{FB} = \frac{C_{OX}\left(\frac{\varepsilon_S A}{\lambda}\right)(1 \times E^2)}{C_{OX} + \left(\frac{\varepsilon_S A}{\lambda}\right)(1 \times E^2)}$$

Formula: DEBYEM

Formula name: DEBYEM (λ)

Units: m

Description: Debye length (in meters).

Formulator entry:

DEBYEM = SQRT(ES*K*TEMP/(ABS(N90W)*Q^2))*1E-2

Simplified equation:

$$\lambda = \left(\frac{\varepsilon_S kT}{q^2 N_X}\right)^{1/2}(1 \times E^{-2})$$

Formula: DEPTHM

Formula name: DEPTHM (*W)*

Units: m

Description: Depletion depth (in meters).

Formulator entry:

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:

$$W = A\varepsilon_S\left(\frac{1}{C} - \frac{1}{C_{OX}}\right)(1xE^{-2})$$

Where:

- W = depletion depth (m)
- A = gate area (cm^2)
- εS = permittivity of the substrate material (F/cm)
- C = measured capacitance (F)
- C_{OX} = oxide capacitance (F)
- 1 x E^{-2} = units conversion from cm to m

Formula: INVCSQR

Formula name: INVCSQR

Units: $1/F^2$

Description: Inverse square of capacitance.

Formulator entry:

INVCSQR = 1/CADJ^2

Simplified equation:

$$INVCSQR = \frac{1}{C^2}$$

Formula: N90W

Formula name: N90W

Units: None

Description: Doping density at 90% of maximum depletion depth.

Formulator entry:

N90W = AT(NDOPING, FINDD(DEPTHM, 0.9*MAX(DEPTHM), 2))

Formula: NDOPING

Formula name: NDOPING (*N*)

Units: $1/cm^3$

Description: Doping density.

Formulator entry:

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:

$$N(W) = \frac{2}{q\varepsilon_S A^2 \left[\frac{d\left(\frac{1}{C^2}\right)}{dV}\right]}$$

Formula: PHIB

Formula name: PHIB (φ_B)

Units: V

Description: Bulk potential.

Formulator entry:

PHIB = (-1)*K*TEMP/Q*LN(ABS(N90W)/NI)*DOPETYPE

Simplified equation:

$$\phi_B = \frac{kT}{q} Ln\left(\frac{N_{BULK}}{N_i}\right) (DopeType)$$

Formula: RS

Formula name: RS (*RS*)

Units: Ω

Description: Series resistance calculated from capacitance.

Formulator entry:

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:

$$R_S = \frac{\left(\frac{G}{2\pi fC}\right)^2}{\left[1+\left(\frac{G}{2\pi fC}\right)^2\right]G}$$

Formula: TOXNM

Formula name: TOXNM (*TOX*)

Units: nm

Description: Calculated thickness of oxide.

Formulator entry:

TOXNM = (1E7*AREA*EOX)/COX_CALC

Simplified equation:

$$T_{OX} = \left(\frac{A\varepsilon_{OX}(1xE^7)}{C_{OX}}\right)$$

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 x E^7 = units conversion from cm to nm

Formula: VFB

Formula name: VFB (*VFB*)

Units: V

Description: Flatband voltage. Once CFB (*CFB*) is derived, *VFB* is interpolated from the closest *VGS* values.

Formulator entry:

VFB = AT(DCV_GB,POSVFB)

Constants for the mosfet project

Constant	Default Value	Units	Description
AREA	0.00012	cm^2	Gate area of device
DOPETYPE	1	none	1 = P-type, -1 = N-type
EOX	3.4e-013	F/cm	ε_{OX} - Permittivity of oxide
ES	1.04e-12	F/cm	ε_S - Semiconductor permittivity
Q	1.60218e-019	C	Charge on an electron
TEMP	293	K	Test temperature

g-to-sdb and mosfet-dopingprofile Analyze sheet

The test data is displayed in the Analyze sheet:

- 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.

Nanowire tests

You can use C-V measurements on semiconductor nanowires and nanowire-based devices 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, <1 pF. As a result, using proper techniques to reduce parasitic capacitance from affecting measurement accuracy is important.

These tests perform C-V sweeps on a two-terminal nanowire device. The tests are similar but use different drive frequencies. The tests generate capacitance versus voltage graphs.

cvu-nanowire connections

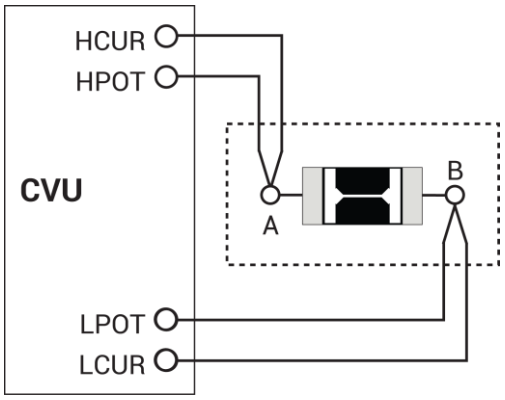
The next figure shows the basic test configuration. Refer to 4210-CVU connections (on page 4-7) for details. The `cv-gd` test is used to test a nanowire on a wafer. The `cv-sd` test can be used to test a nanowire on a wafer or a discrete nanowire device.

Use only the supplied (red) 100 Ω SMA cables for connections to the 4210-CVU. Be sure that all SMA cables used for the project are the same length.

NOTE

After making or changing connections, be sure to use the Confidence Check diagnostic tool and do connection compensation tests. Refer to Confidence Check (on page 4-27) and Connection compensation (on page 4-14) for details.

Figure 141: Basic configuration to test a nanowire device



Formulas and constants

This project uses two formulas, described in the following topics. There are no constants.

Formula: CAVG

Formula name: CAVG

Units: F

Description: Calculates the average capacitance.

Formulator entry:

CAVG = AVG(CP_AB)

Formula: STD_DEV

Formula name: STD_DEV

Units: None

Description: Calculates the standard deviation of the capacitance measurements.

Formulator entry:

STD_DEV = STDEV(CP_AB)

cv-gd and cv-sd tests

`cv-gd` test: This test measures C-V (gate to drain at 1 MHz) on a nanowire device on a wafer or a discrete nanowire device. It generates a C versus V graph.

`cv-sd` test: This test measures C-V (source to drain at 100 kHz) on a nanowire device on a wafer or a discreet nanowire device. This test generates a C versus V graph and calculates average capacitance and standard deviation.

NOTE

Both tests are configured the same except for the drive frequency.

cv-gd and cv-sd Analyze sheet

The test data is displayed in the Analyze sheet:

- Cp_AB: Measured parallel capacitance.
- Gp_AB: Measured conductance.
- DCV_AB: Forced DC bias voltage.
- F_AB: Forced test frequency.
- CVU1S: Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see Measurement status (on page 6-243).
- Formulas: Formulator calculation results (`cv-sd` test only).

NOTE

AB = Terminal A to Terminal B.

Diode Project (diode-project)

This project contains DC I-V, C-V, and pulse I-V tests for a PN junction.

The `diode` project measures 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, the effects of the space charge region spreading into the more highly doped area can be ignored.

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 the next figures, showing a reverse-biased Schottky diode and a reverse-biased p+n junction.

Figure 142: Reverse-biased Schottky diode

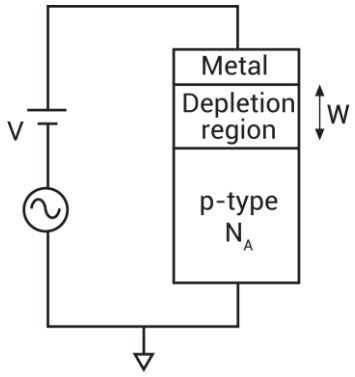
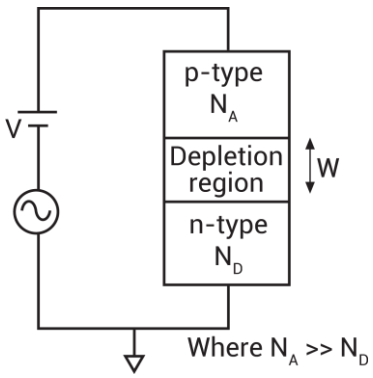


Figure 143: Reverse-biased p+n junction



Since the junction can be modeled after a parallel plate capacitor, the junction capacitance is calculated as follows:

$$C = \frac{\varepsilon_S A}{W}$$

Where:

- C = junction capacitance (F)
- εS = semiconductor permittivity (1.034e-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:

$$W = \frac{\varepsilon_S A}{C}$$

From the measured capacitance and the voltage, the doping density can be calculated as follows:

$$N(W) = \frac{2}{q\varepsilon_S A^2 \left[\frac{d\left(\frac{1}{C^2}\right)}{dV}\right]}$$

Where:

- N(W) = doping density (cm^{-3})
- A = area of junction (cm^2)
- C = junction capacitance (F)
- εS = semiconductor permittivity (1.034e-12 F/cm for silicon)
- q = electron charge (1.60219e-19 C)
- V = junction voltage

diode connections

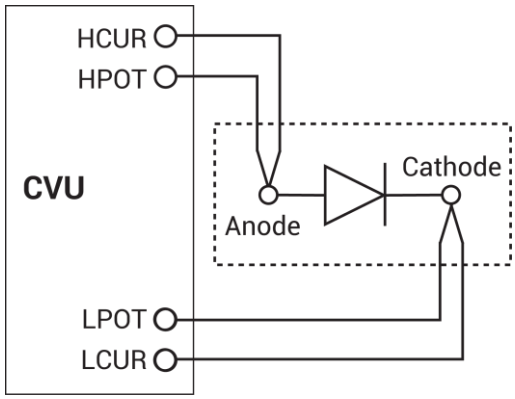
The next figure shows the basic test configuration. Refer to Test connections for a probe card (on page 4-10) for the connections to a semiconductor wafer.

Use only the supplied (red) 100 Ω SMA cables for connections to the 4210-CVU. Be sure that all SMA cables that are used in the setup are the same length.

NOTE

After making or changing connections, be sure to use the Confidence Check diagnostic tool and do connection compensation tests. Refer to Confidence Check (on page 4-27) and Connection compensation (on page 4-14) for details.

Figure 144: Basic configuration to test a semiconductor junction



diode project formulas and constants

Formulas and user-defined constants that are used for the tests are listed in the next topics. The formulas and constants are set from the Formulator for each test.

Formula: DEPTH

Formula name: DEPTH (W)

Units: cm

Description: Width of depletion (junction) region.

Formulator entry:

DEPTH = (ES*AREA)/(CP_AC)

Simplified equation:

$$W = \frac{\varepsilon_S A}{C}$$

Formula: INV_C

Formula name: INV_C (C)

Units: 1/F

Description: Inverse capacitance.

Formulator entry: INV_C = 1/(CP_AC)^2

Simplified equation:

$$C = \frac{1}{C^2}$$

Formula: DOPING

Formula name: DOPING (N)

Units: $1/cm^3$

Description: Majority carrier concentration.

Formulator entry:

DOPING = ABS(2/Q/ES/AREA^2/DIFF(INV_C, DCV_AC))

Simplified equation:

$$N(W) = \frac{2}{q\varepsilon_S A^2 \left[\frac{d\left(\frac{1}{C^2}\right)}{dV}\right]}$$

Constants for the diode project

Constant	Default Value	Units	Description
AREA	0.0001	cm^2	Gate area of device
ES	1.034e-12	F/cm	ε_S - Semiconductor permittivity

cvsweep-diode test

This test measures the PN junction capacitance as a function of the DC bias voltage while the junction is reverse biased. The test makes a capacitance measurement at each step of a user-configured linear voltage sweep. From the acquired C-V data, the test uses the Formulator to calculate parameters.

cvsweep-diode Analyze sheet

The test data is displayed in the Analyze sheet:

- 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

diode c-2vsv test

This test displays the data as a $1/C^2$ versus V curve. You can derive the doping density (N) from the slope of this curve because N is linearly related to the capacitance.

You can derive the built-in potential (φ_B) of the diode from the intersection of the $1/C^2$ curve and the horizontal axis. Use the graph option Linear Line Fit to derive both the doping density (N) and the built-in voltage on the x-axis.

c-2vsv-diode test Analyze sheet

The test data is displayed in the Analyze sheet:

- 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.

diode-dopingprofile test

This test performs a C-V sweep and uses the measurements for doping profiling. At each voltage step, the doping density and corresponding depth (W) are calculated and plotted on the graph.

diode-dopingprofile Analyze sheet

The test data is displayed in the Analyze sheet:

- 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.

Solar Cell Project (solarcell)

To determine the electrical characteristics of a photovoltaic (PV) cell, a 4200-SMU and 4210-CVU are used to perform I-V, C-V, and C-F sweeps. Graphs are generated from the collected current, capacitance, frequency, and voltage data.

Formulator calculations are made to determine the following electrical characteristics of the PV cell:

- **I-V testing:** The Formulator calculates 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 calculates inverse capacitance (INV_C) and the doping density (N).

I-V testing generates the following graphs:

- Current versus voltage graph for a forward-biased PV cell. Formulator formulas calculate the electrical characteristics of the cell.
- Current versus voltage graph for a reversed-biased PV cell.

C-V testing generates the following graphs:

- Capacitance versus voltage graph for a reversed-biased PV cell.
- A $1/C^2$ (inverse capacitance) and capacitance versus voltage graph for a PV cell. $1/C^2$ and doping density are calculated by the Formulator.
- Capacitance versus frequency graph.

The testing process is typically repeated using different light intensities and temperature conditions.

solarcell connections

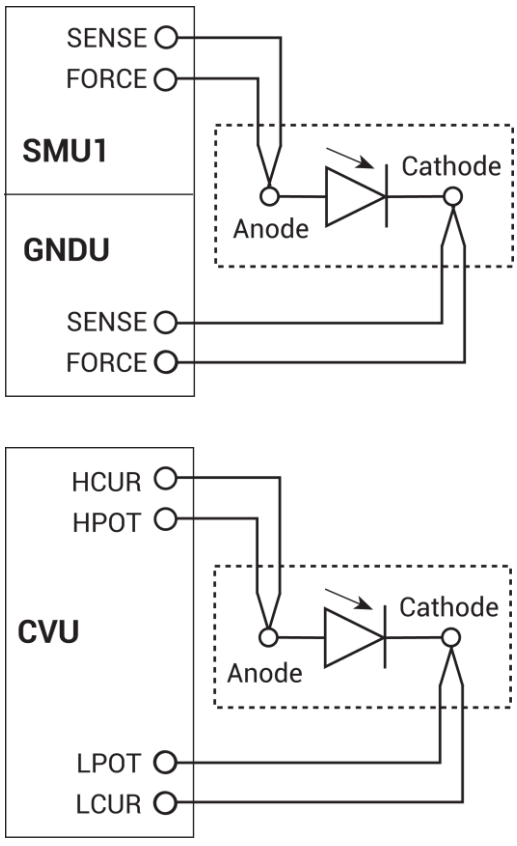
The next figures show the basic test configurations for this project:

- **I-V testing**: The basic configuration shows four-wire sensing using a 4200-SMU or 4210-SMU. Four-wire sensing is needed to achieve optimum performance. See Source-Measure Hardware (on page 3-1) for details on 4200-SMU connections.
- **C-V Testing**: See 4210-CVU connections (on page 4-7) for details on connections. Use only the supplied (red) 100 Ω SMA cables for connections to the 4210-CVU. Be sure that all used SMA cables are the same length.

NOTE

After making or changing connections, be sure to use the Confidence Check diagnostic tool and do connection compensation tests. Refer to Confidence Check (on page 4-27) and Connection compensation (on page 4-14) for details.

Figure 145: Basic configurations to test PV cells for I-V testing



solarcell formulas and constants

Formulas and user-defined constants that are used for the tests are listed in alphabetical order in the next topics. The formulas and constants are set from the Formulator for the tests.

Formula: CURR

Formula name: CURR

Units: A

Description: PV cell current. The absolute value of the current output.

Formulator entry:

CURR = ABS(ANODEI)

Formula: FF

Formula name: FF

Units: None

Description: Fill factor.

Formulator entry:

FF = (IMAX*VMAX)/(ISC*VOC)

Simplified equation:

$$FF = \frac{I_{MAX}V_{MAX}}{I_{SC}V_{OC}}$$

Formula: IMAX

Formula name: IMAX

Units: A

Description: Cell current where the power output of the cell is greatest (PMAX).

Formulator entry:

IMAX = ABS(AT(ANODEI, MAXPOS(P)))

Formula: INV_C2

Formula name: INV_C2 (C)

Units: 1/F2

Description: Inverse capacitance.

Formulator entry:

INV_C2 = 1/CP_AC^2

Simplified equation:

$$C = \frac{1}{C^2}$$

Formula: ISC

Formula name: ISC (I_{SC})

Units: A

Description: Short circuit current. ISC is the point in I-V data where V = 0.

Formulator entry:

ISC = ABS(AT(ANODEI, FINDD(ANODEV, 0, FIRSTPOS(ANODEV))))

Formula: N

Formula name: N ($N(_a)$)

Units: $1/cm^3$

Description: Doping density.

Formulator entry:

N = ABS(2/Q/ES/AREA^2/DIFF(INV_C2, DCV_AC))

Simplified equation:

$$N(a) = \frac{2}{\left| q\varepsilon_S A^2 \frac{d\left(\frac{1}{C^2}\right)}{dV} \right|}$$

Where:

- $N_{(a)}$ = 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)

Formula: P

Formula name: P

Units: W

Description: Power.

Formulator entry:

P = CURR*ANODEV

Simplified equation:

$$P = I \times V$$

Where:

- I = Cell current
- V = Cell voltage

Formula: PMAX

Formula name: PMAX (P_{MAX})

Units: W

Description: Maximum power point.

Formulator entry:

PMAX = MAX(P)

Simplified equation:

$$P_{MAX} = I_{MAX} \times V_{MAX}$$

Formula: VMAX

Formula name: VMAX

Units: V

Description: Cell voltage where the power output of the cell is greatest (PMAX).

Formulator entry:

VMAX = AT(ANODEV, MAXPOS(P))

Formula: VOC

Formula name: VOC (V_{OC})

Units: V

Description: Open circuit voltage. VOC is the point in I-V data where I = 0.

Formulator entry:

VOC = AT(ANODEV, FINDU(ANODEI, 0, LASTPOS(ANODEI)))

Constants for the solarcell project

Constant	Default value	Units	Description
AREA	8	cm^2	Gate area of device
ES	1.034e-12	F/cm	Semiconductor permittivity (εs)

fwd-ivsweep test

NOTE

Make sure the PV cell is connected to the 4200-SMU (see Connections (on page 4-128)).

This test uses a 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 makes 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 2 mV steps. A total of 151 current measurements are made.

fwd-ivsweep Analyze sheet

The test data is displayed in the Analyze sheet:

- AnodeI: Measured current.
- AnodeV: Forced voltage.
- Formulas: Formulator calculation results.

rev-ivsweep test

NOTE

Make sure the solar cell is connected to the 4200-SMU (see Connections (on page 4-128)).

This test uses a SMU to perform a reversed-bias voltage sweep. 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 made.

rev-ivsweep Analyze sheet

The test data is displayed in the Analyze sheet:

- AnodeI: Measured current.
- AnodeV: Forced voltage.

cvsweep-solarcell test

NOTE

Make sure the solar cell is connected to the 4210-CVU (see 4210-CVU connections (on page 4-7)).

This test uses a CVU to perform a capacitance voltage sweep on a solar cell. Capacitance is measured on each step of the sweep. A C versus V graph is generated from the collected data.

cvsweep-solarcell Analyze sheet

The test data is displayed in the Analyze sheet:

- Cp_AC: Measured parallel capacitance.
- Cp_AC: Measured conductance.
- DCV_AC: Forced DC bias voltage.
- F_AC: Forced test frequency.
- CVU1S: Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see Measurement status (on page 6-243).

NOTE

AC = Anode to cathode.

c-2vsv-solarcell test

NOTE

Make sure the solar cell is connected to the 4210-CVU (see 4210-CVU connections (on page 4-7)).

This test uses a CVU to perform a capacitance-voltage sweep on a solar 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.

c-2vsv-solarcell Analyze sheet

The test data is displayed in the Analyze sheet:

- Cp_AC: Measured parallel capacitance.
- Gp_AC: Measured conductance.
- DCV_AC: Forced DC bias voltage.
- F_AC: Forced test frequency.
- CVU1S: Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see Measurement status (on page 6-243).
- Formulas: Formulator calculation results.

NOTE

AC = Anode to cathode.

cfsweep test

NOTE

Make sure the PV cell is connected to the 4210-CVU (see 4210-CVU connections (on page 4-7)).

This test performs a frequency sweep, measuring capacitance at each frequency point, and then generates a C versus F graph.

cfsweep Analyze sheet

The test data is displayed in the Analyze sheet:

- Cp_AC: Measured parallel capacitance.
- Gp_AC: Measured conductance.
- DCV_AC: Forced DC bias voltage.
- F_AC: Forced test frequency.
- CVU1S: Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see Measurement status (on page 6-243).

NOTE

AC = Anode to cathode.

Demo project (default)

The tests in this project represent the most common device tests.

The tests in the default project that use the CVU are `cv-nmosfet`, `cv-diode`, and `cv-cap`.

cv-nmosfet test

Measures the capacitance as a function of the gate voltage between the gate terminal and the drain, source, and bulk terminals tied together. Several parameters are extracted, including the flatband capacitance, doping density, flatband voltage, and oxide thickness.

cv-diode test

Measures the junction capacitance as a function of an applied voltage sweep. The depletion depth (W) and the doping density (N) are calculated as a function of the C-V data.

cv-cap test

Measures the capacitance as a function of voltage on a capacitor. The noise is calculated using the standard deviation of the data.

Carbon Nanotube Transistor Characterization Project (cntfet-characterization)

This project contains DC I-V, pulsed I-V, and C-V tests for a carbon nanotube FET (CNTFET).

The DC I-V tests include measurements for drain current versus drain voltage (V_{ds-Id}) at stepped gate voltages and drain current versus gate voltage (V_{gs-Id}) at a constant drain voltage.

The pulsed I-V tests include pulsed V_{ds-Id} and pulsed V_{gs-Id}.

The C-V test includes a C-V sweep test that measures the gate-to-drain capacitance as a function of the gate voltage of a CNTFET.

The tests include:

- CNTFET Drain Family of Curves (vds-id-cntfet): This test generates the standard family of drain current versus drain voltage curves on a FET. For each gate voltage step, the test sweeps the drain voltage and measures the resulting drain current. This test uses either three or four SMUs that are connected to the gate, drain, source, and bulk terminals of the FET.
- CNTFET Drain Current versus Gate Voltage (vgs-id-cntfet): This test sweeps the gate voltage (V_g) and measures the resulting drain current (I_d) while the drain voltage (V_d) is kept constant.
- CNTFET Capacitance versus Voltage Sweep (cvsweep-cntfet): Measures the gate-to-drain capacitance as a function of the gate voltage of a carbon nanotube FET. The test is made at a constant test frequency.
- CNTFET Pulsed Drain Family of Curves (pulsed-vds-id): Uses CH1 and CH2 of a PMU to generate a pulse I-V drain family of curves. CH1 outputs a pulse step output to the gate. CH2 outputs a pulsed drain voltage sweep and measures the drain current.
- CNTFET Single Pulse I-V (cnt-pulse): The waveform capture mode of the PMU shows the time-based response of the drain current and drain voltage of a CNTFET. CH1 outputs a single pulse to the gate. CH2 captures the transient response of the drain current and drain voltage.

MOSFET Gate Charge Project (gate-charge)

This test measures the gate voltage (Vg) as a function of gate charge (Q) of a transistor. Vgs, Vd, and Id can be measured as a function of time.

This project includes:

- Gate charge test (gate-charge): Measures the gate voltage as a function of gate charge (Q) of a transistor. Vgs, Vd, and Id can be measured as a function of time.

For additional information on high-voltage measurements, refer to *Application Note: Measuring MOSFET Gate Charge with the 4200A-SCS Parameter Analyzer*.

Section 5

Pulse measure and pulse generator units

In this section:

Models 4220-PGU and 4225-PMU

The 4220-PGU Pulse Generator Unit and 4225-PMU Pulse Measure Unit are high-speed pulse generator cards for the 4200A-SCS. In this section, the 4220-PGU is referred to as a "PGU," and the 4225-PMU is referred to as a "PMU." The PGU provides pulse output only; the PMU provides both pulse output and pulse measurement. The PGU and PMU have similar pulse output characteristics.

NOTE

A 4225-PMU can be ordered with one or two 4225-RPM Remote Preamplifier/Switch Modules. In this section, the 4225-RPM is referred to as an "RPM." The RPM is a remote amplifier or a switch. For example, it is a preamplifier for the PMU to provide additional low-current measurement ranges. It is also used as a switch for the 4225-PMU, 4200-SMU, and 4210-CVU. See RPM input, output, and top panels (on page 5-8) for details about the RPM.

LPT functions that pertain to the PGU and PMU are documented in LPT commands for PGUs and PMUs (on page 14-98).

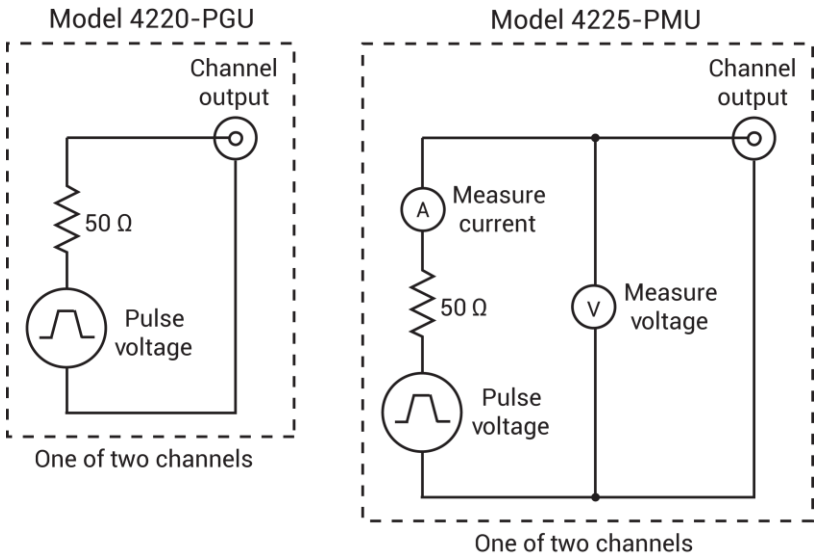
KPulse: The Keithley Pulse (KPulse (on page 11-1)) application supports the PGU.

NOTE

The pulse source-measure concepts covered in this section apply to the PGU and PMU.

The simplified circuits of the 4220-PGU and 4225-PMU pulse generators are shown in the next figure.

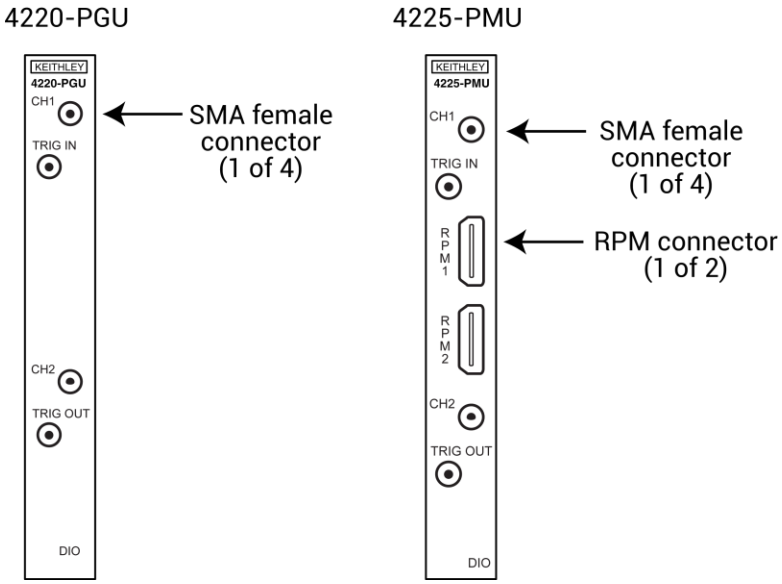
Figure 146: Simplified circuits of the PGU and PMU



PGU and PMU connectors

The connectors for the PGU and PMU pulse cards are shown in the next figure.

Figure 147: 4220-PGU and 4225-PMU connectors



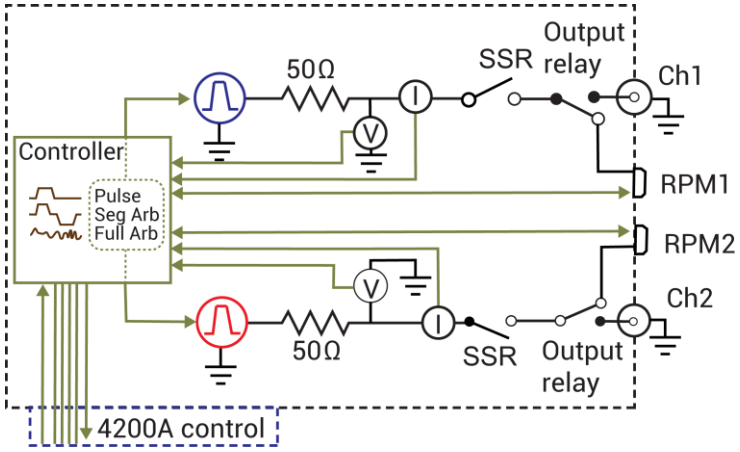
NOTE

Use the RPM connectors on the 4225-PMU to connect it to the 4225-RPM (on page 5-8).

PMU block diagram

The next figure 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 148: 4225-PMU block diagram



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. The test modes for standard pulse are pulse I-V and waveform capture. See Pulse parameter definitions (on page 5-51) for the standard (two-level) pulse mode.
- Segment Arb® waveform: 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 includes sequencing and sequence looping (see seg_arb_sequence (on page 14-140) and seg_arb_waveform (on page 14-144). Also see Segment Arb waveform (on page 5-72) for details on parameters.
- Full-arb pulse mode: For this multi-level pulse mode, the waveform consists of a number of user-defined points (see arb_array (on page 14-148) and arb_file (on page 14-149)). Also see Full Arb waveform (on page 14-148) for details on parameters.

KPulse supports all of these pulse modes.

Pulse measurement types (PMU)

The next table summarizes pulse measurement types that are available through LPT commands for the 4225-PMU. See Measurement types (on page 5-91) for detailed information.

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 the Spot mean measurements example figure; Measurement types (on page 5-91)).
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 Measurement types (on page 5-91) Waveform measurements and Waveform measurements (with pre-data and post-data) figures).
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.

Measure modes

The measure modes for the PMU are discrete pulses and average pulses.

Discrete pulses

For pulse I-V (spot mean), the averaged voltage or current readings for every sampled pulse period are acquired.

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 Analyze sheet.

NOTE

See Spot mean discrete readings (on page 5-92) for details on the discrete pulses measure mode for pulse I-V. For UTM programming, the discrete pulse measure mode is called spot mean discrete. The pulse_meas_sm (on page 14-119) function is used to select the discrete acquisition type.

See Waveform discrete readings (on page 5-95) 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 (on page 14-123) function is used to select the discrete acquisition type.

Average pulses

For pulse I-V (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.

NOTE

See Spot mean average readings (on page 5-93) for details on the average pulses measure mode for pulse I-V. For UTM programming, the average pulse measure mode is called spot mean average. The pulse_meas_sm (on page 14-119) function is used to select the average acquisition type.

See Waveform average readings (on page 5-95) 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 (on page 14-123) 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 ITMs so that less than one million samples will be acquired.

Pulse I-V (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, Clarius automatically lowers the sampling rate. For pulse I-V (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.

Model 4225-RPM

The 4225-RPM is a remote amplifier/switch that is used as a current preamplifier for the 4225-PMU. The RPM provides additional high-speed, low-current measurement ranges.

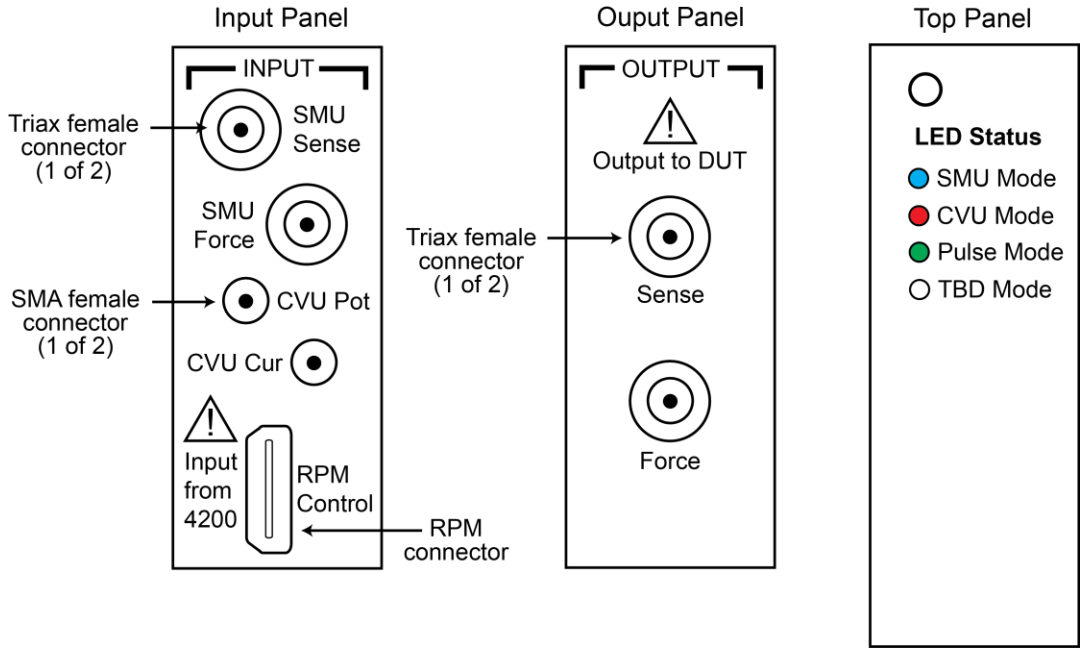
The RPM can also be used as a switch for the 4200-SMU and 4210-CVU. See Use the RPM as a switch (on page 5-11).

RPM input, output, and top panels

The input, output, and top panels for the RPM are shown in the next figure. The RPM connector on the input panel connects to one of the RPM connectors (channel 1 or channel 2) on the 4225-PMU. The RPM also has input connectors for a 4200-SMU (source-measure unit) and a 4210-CVU (capacitance-voltage unit).

The next figure 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 4200A-SCS is indicated by the Operate light on the front of the 4200A-SCS chassis. During normal Clarius operation, only the red, green, or blue lights 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 and green near the end of the process. During firmware upgrade of the 4225-RPM, the RPM LED is blue at the start, and changes to green for the remainder of the process.

Figure 149: 4225-RPM



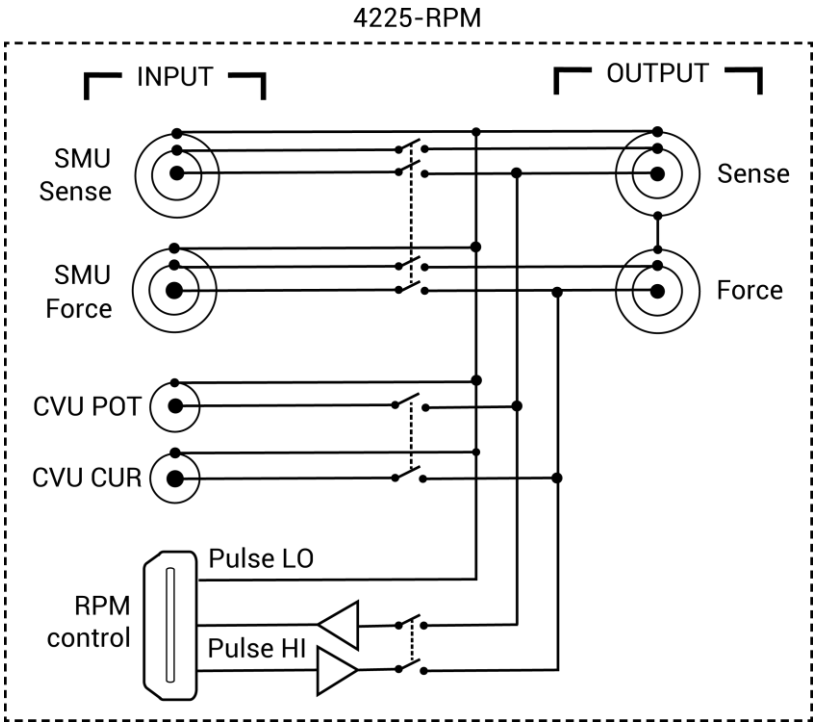
RPM wiring diagram

The internal wiring diagram of the RPM is shown in the next figure.

Signals from the 4200A-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 (on page 5-11) for more information on switching.

The LEDs on the top panel (see the previous figure) 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.

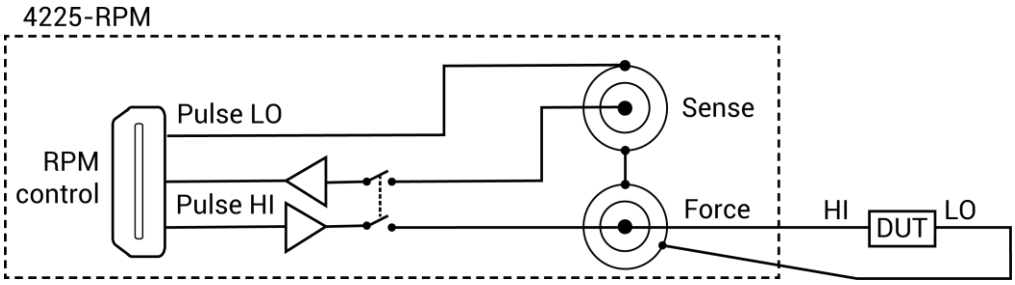
Figure 150: Wiring diagram of the RPM



RPM diagrams for local and remote sensing

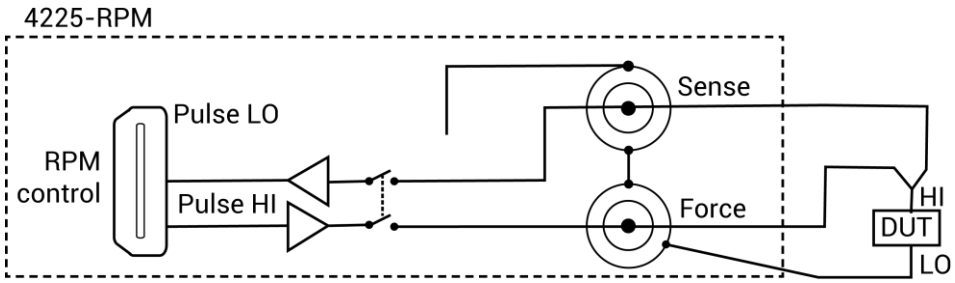
The next figure shows the diagram for local sensing. The center conductor of the Force triaxial connector is connected to the high side of the device under test (DUT) while the outer shield is connected to DUT LO. The Sense connector is not used.

Figure 151: RPM wiring diagram for local sensing



The next figure shows the diagram for remote sensing. Both Sense and Force are connected to DUT HI.

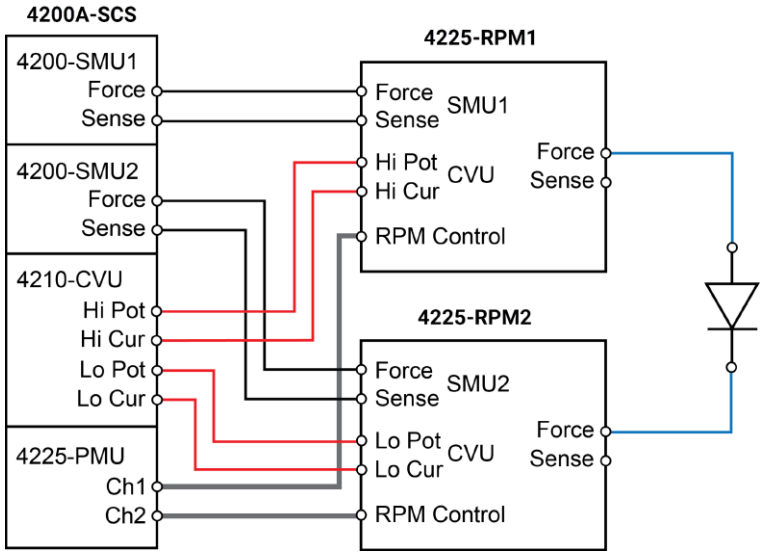
Figure 152: Diagram for remote sensing



Using the RPM as a switch

You can use the RPM to switch a PMU, CVU, or SMU to a DUT terminal. The RPM wiring diagram (on page 5-9) figure shows the switches. The next figure 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 153: Test configuration for using RPM as a switch



Both the red cables (supplied with the CVU) and the blue cables (supplied with the optional 4210-MMPC cable kits) are 100 Ω. You can do remote sensing (on page 5-25) using the optional 4210-MMPC cable kits with the RPMs.

Control RPM switching

Before using an RPM, configure the 4200A-SCS by doing the steps in Update the RPM configuration (on page 7-12). This properly associates the instruments connected to each RPM.

There are two methods to control RPM switching:

- ITM operation using automatic switching (after doing the steps in Update the RPM configuration (on page 7-12))
- UTM testing from within the user module, use the LPT function rpm_config (on page 14-139)

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.

Waveform capture

The maximum number of rows that can appear in the Analyze sheet 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 is any wider, the sample rate is automatically lowered by Clarius 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.

Connections

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.

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
- 4225-RPM connections

There are two optional prober cable kits that are available from Keithley Instruments to provide connections to a DUT:

- 4210-MMPC-S: Use this cable kit with the Suss Micro Tec PA200/300 series prober.
- 4210-MMPC-C: Use this cable kit with the Cascade Microtech 12000 series prober (manipulator type DCM-200 series).

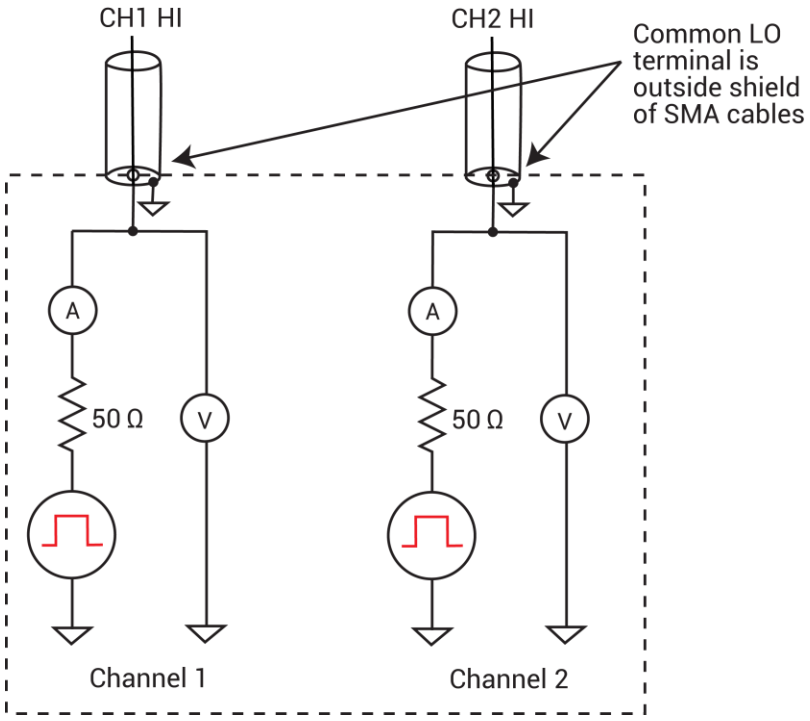
NOTE

For details on using these prober cable kits, refer to Keithley document number PA-1000 for the Suss prober cable kit and PA-1001 for the Cascade prober cable kit.

PMU common connections

Common LO for the PMU is the outer shells of the two SMA connectors. With an SMA cable connected (see next figure), common LO is the outside shield of the cable.

Figure 154: 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 the next figure and the following figure for a graphic definition of the loop area for the coaxial 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 LO) from all PMU channels as close as possible to the DUT. You reduce inductance by minimizing the loop area of the shield connections. The figure in Using a SMA to SSMC adapter cable to connect pulse card to DUT (on page 5-22) and the Local sensing (on page 5-24) figures 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 (on page 5-53)).

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 in. (20 cm) SMA cables. Always use the 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 (on page 5-14)).
- Minimize cable length (see Cable length (on page 5-15)).
- 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

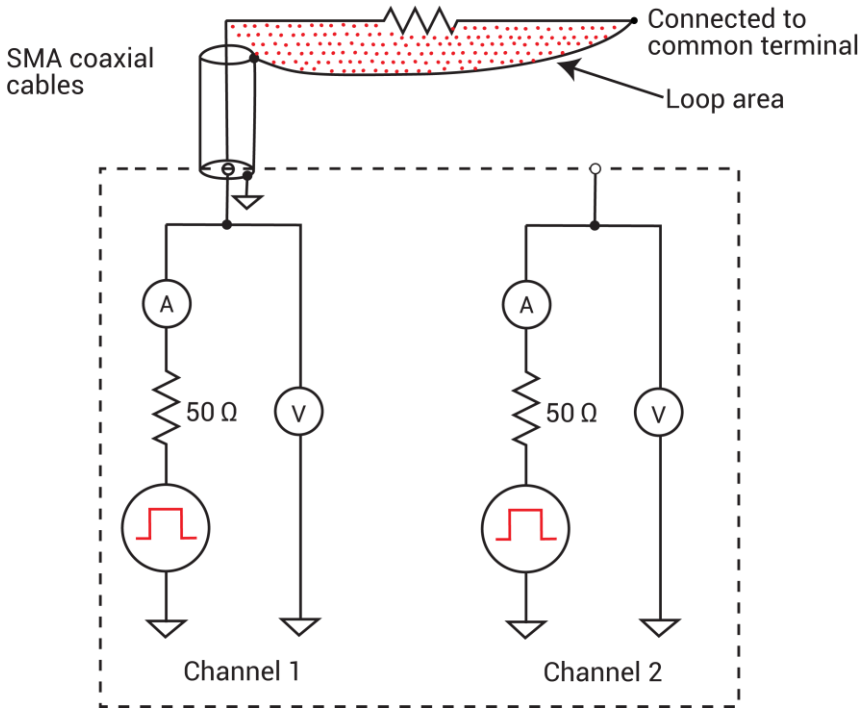
When possible, avoid pulse connections to the prober chuck. If unavoidable, use these guidelines when connecting to the prober chuck:

- 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, refer to Two-terminal device connections (on page 5-17), using figure "Two-terminal device connections to a PMU using both channels" as a guide.
- For a four-terminal device, use the Four-terminal device connections (on page 5-20) to two PMUs figure, or the Local sensing (on page 5-24) four-terminal figure as a guide (as applicable). This cabling approach permits the low-side measurement approach described in PMU capacitive charging/discharging effects (on page 5-53).

Two-terminal device connections

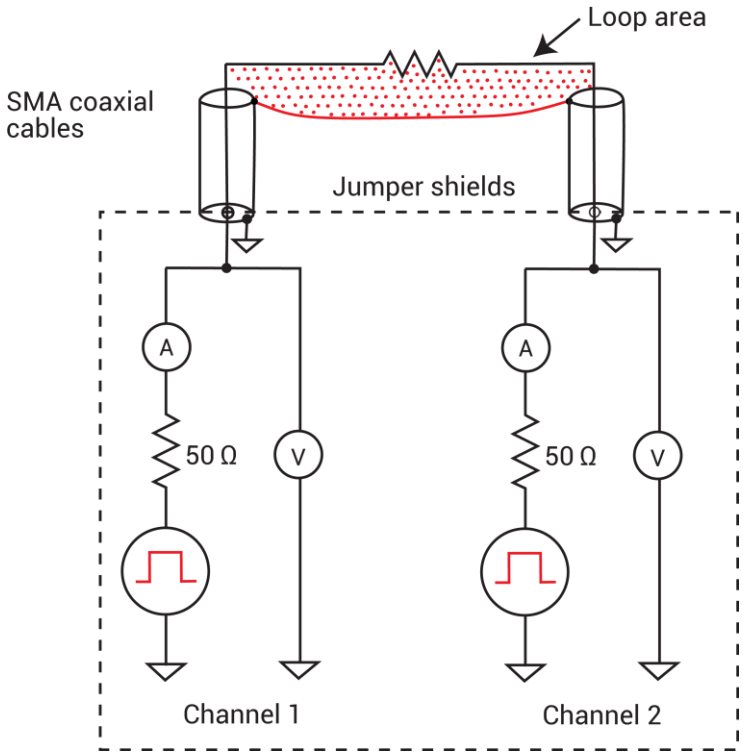
The next figure shows connections to a two-terminal device using a single channel. Connect one end of the device to the center conductor of Ch 1 and connect the other side to pulse card common (outside shield of the SMA cable).

Figure 155: 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 the next figure. 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 (on page 5-53)).

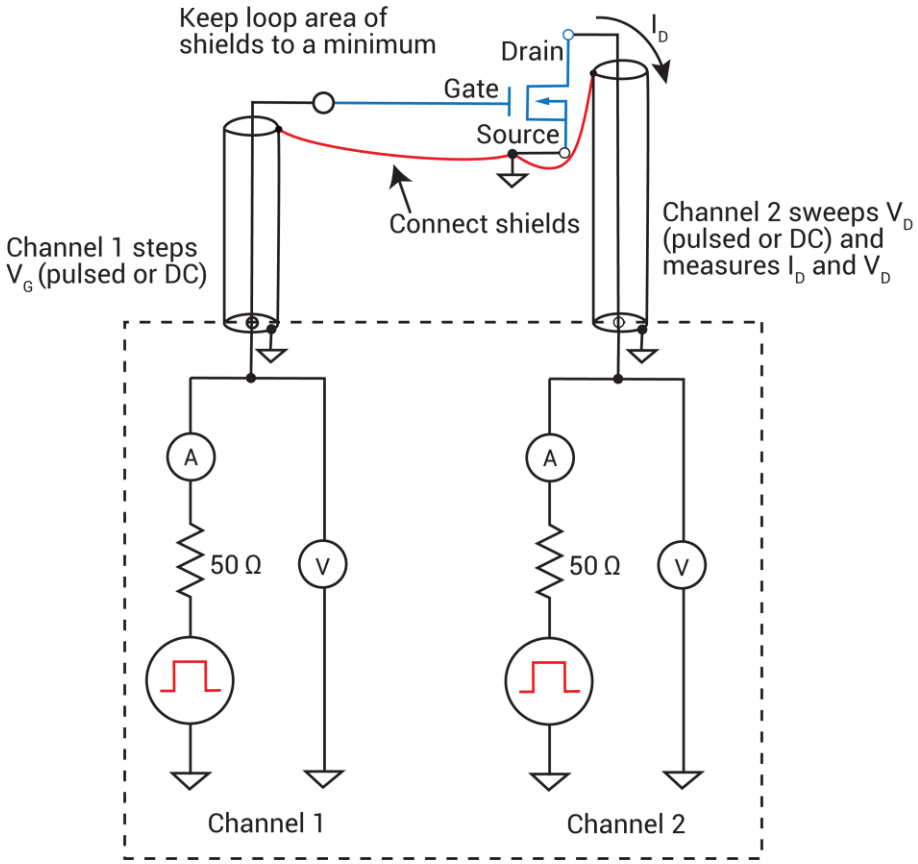
Figure 156: Two-terminal device connections to a PMU using both channels



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 the next figure. 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 157: Three-terminal device connections to a PMU using both channels

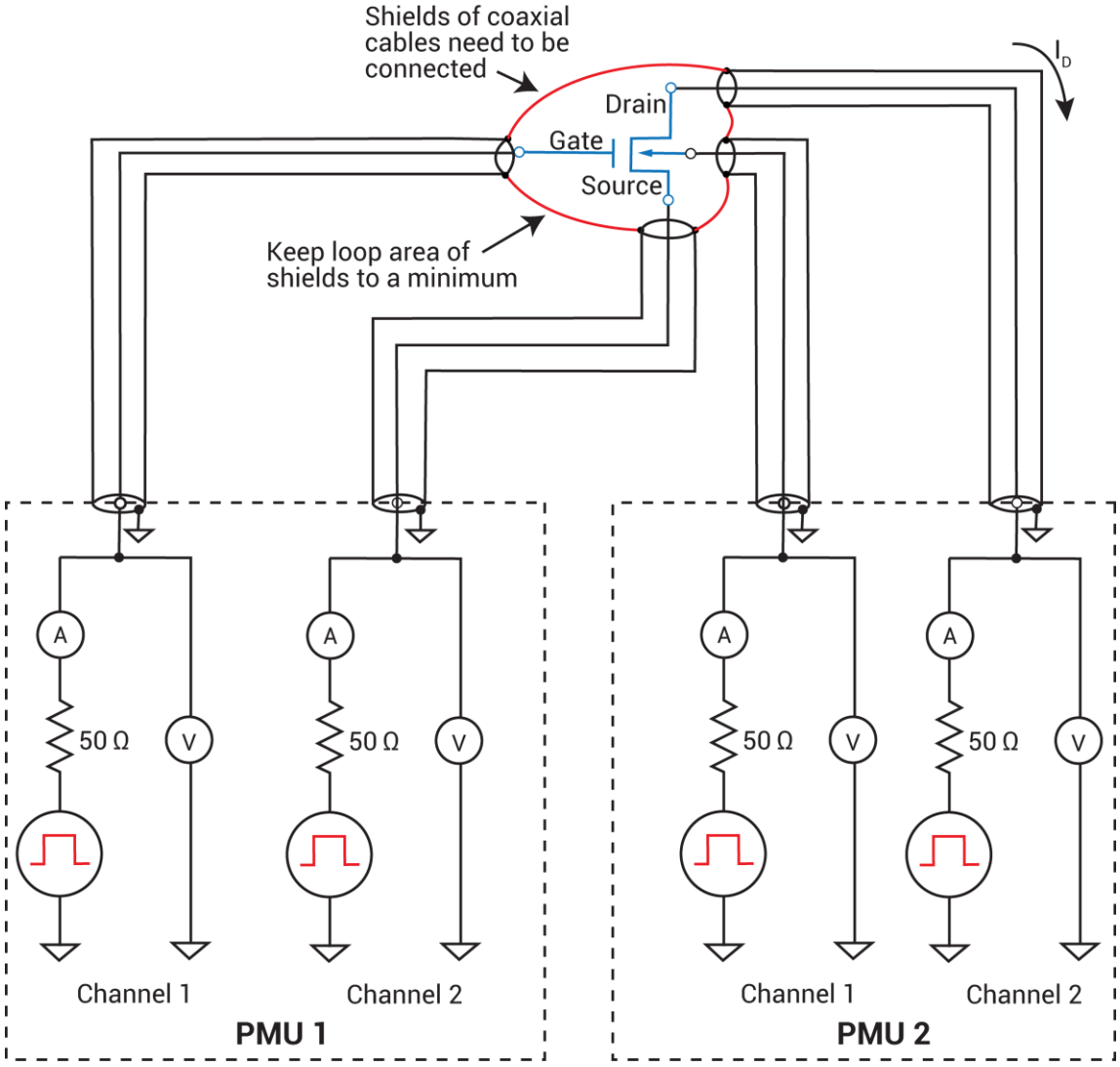


If ultra-fast I-V sourcing and measuring is required at each device terminal, then a second PMU is required for the source terminal. Up to four PMUs (eight channels) can be installed in one 4200A-SCS mainframe.

Four-terminal device connections

To test a four-terminal device, two PMUs are usually required. The next figure shows the four PMU channels connected to a four-terminal MOSFET. This configuration enables you 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 closely as possible to the device under test.

Figure 158: Four-terminal device connection to two PMUs



Connections to prober or test fixture bulkhead connectors

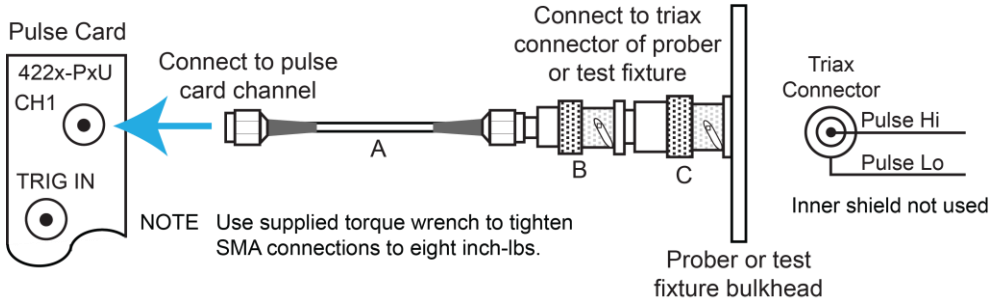
The 4200-PMU-Prober-Kit (available from Keithley Instruments) is a collection of standard and custom connectors and accessories used to connect the pulse generator to a common variety of probe stations. This kit can also be used for pulse card connections to a test fixture.

The next figure shows an example of how a PMU or PGU can be connected to a triaxial 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 159: Pulse card connections to triax prober or test fixture



A. White SMA cable (2 m/6.5 ft, included with the PGU and PMU)

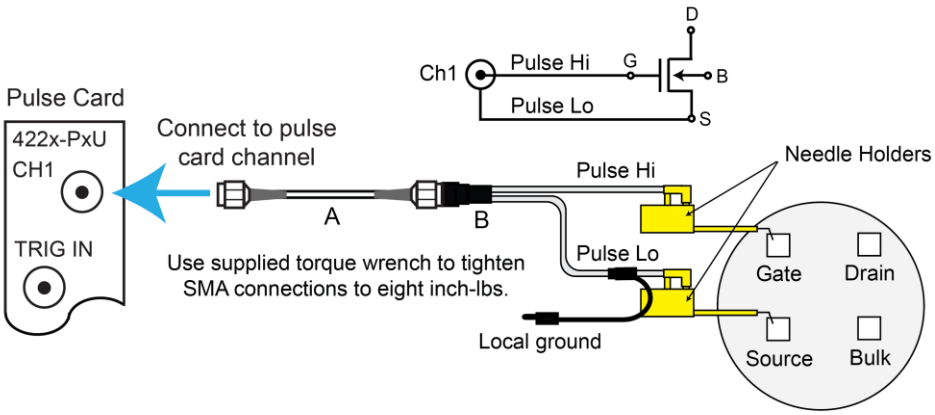
B. SMA female-to-BNC male (included with the 4200-Prober-Kit)

C. BNC female-to-three-lug triax male (included with the 4200-Prober-Kit))

Using an SMA to SSMC adapter cable to connect pulse card to DUT

The next figure shows an example of how to make pulse card connections to the device under test (DUT) using the supplied adapter cable.

Figure 160: Pulse card connections using the SMA to dual SSMC adapter cable



A. White SMA cable (2 meters / 6.5 feet, 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 the previous figure are supplied by the user.

4225-RPM connections

CAUTION

Turn off the system and disconnect the power cord before connecting or disconnecting the RPM to or from the PMU. Failure to do so may result in RPM or PMU damage, possibly voiding the warranty.

RPM connection to the PMU

NOTE

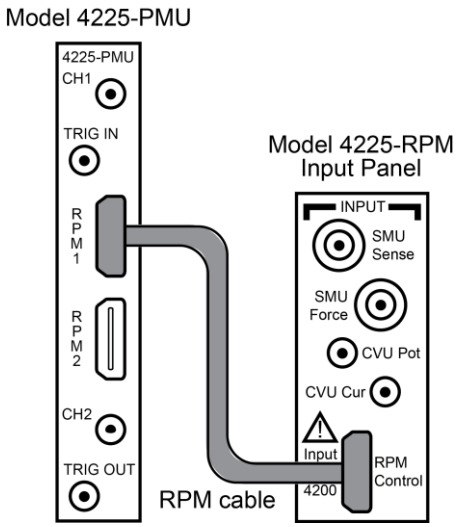
The RPM is matched to a PMU card and channel. Make sure to connect the RPM only to that PMU card and channel.

With system power off, use the supplied RPM cable to connect a 4225-RPM to the matching RPM channel of the 4225-PMU (see next figure).

CAUTION

With an RPM installed, never make connections directly to any of the SMA connectors (CH1 and CH2) on the PMU. This may result in damage to the PMU or DUT, or may produce corrupt data.

Figure 161: PMU connection to the RPM



NOTE

After connecting or removing an RPM, always perform the Update the RPM configuration (on page 7-12) procedure to ensure that KCon accurately represents the present 4200A-SCS hardware configuration.

RPM connections to DUT

CAUTION

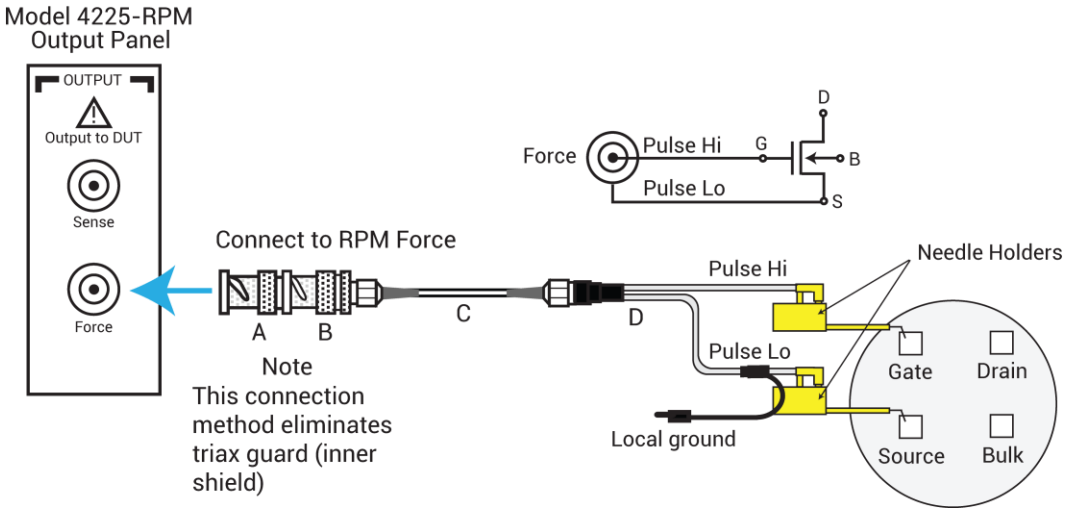
With an RPM installed, NEVER make connections directly to any of the SMA connectors (CH1 and CH2) on the PMU module. This may result in damage to the PMU or DUT or may produce corrupt data.

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. The next figure shows local sense connections using the supplied adapter cable and adapters. For the two-terminal test shown in the next figure, the local ground is left unconnected. Test circuit low is connected to the shield of the Force connector through the cables.

Figure 162: 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 20.32 cm (8 in.) supplied with the RPM)

D. SMA-to-SSMC Adapter Cable (4200-PRB-C, supplied with the PMU)

The needle holders shown in the previous two figures 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 next figure).

Remote sensing

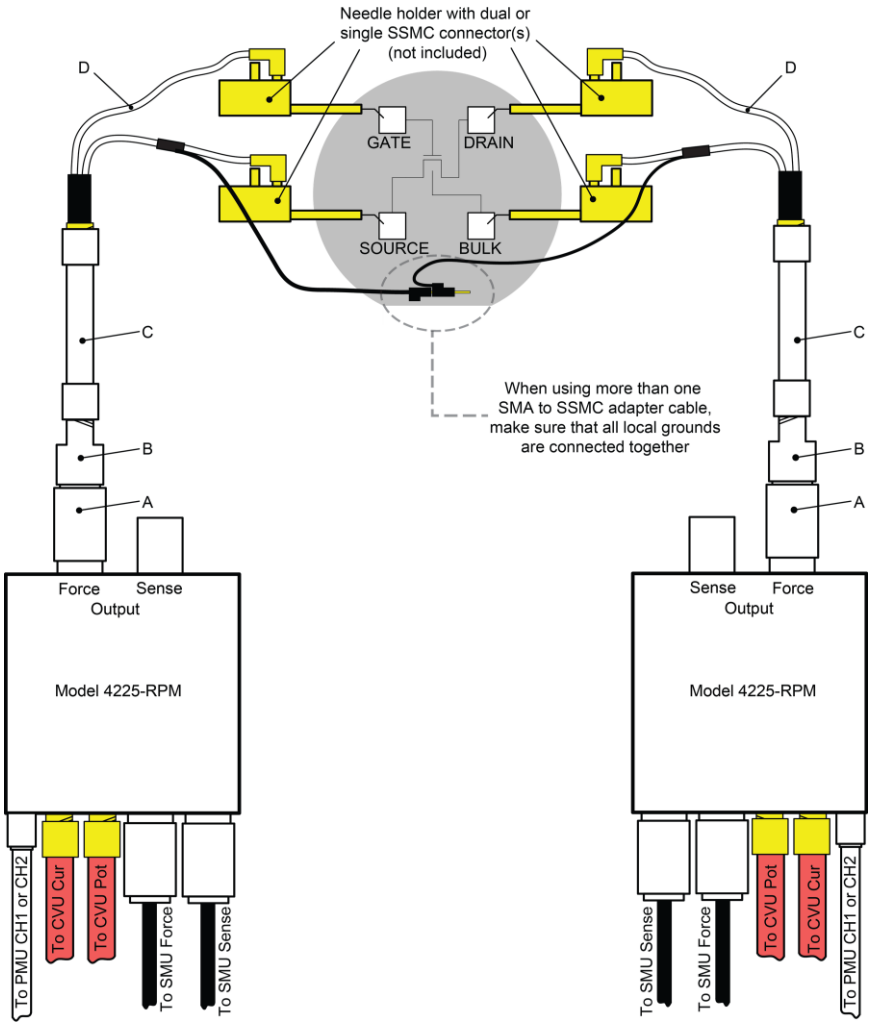
Optional prober cable kits are available from Keithley Instruments. These kits provide remote sense connections to a DUT:

- 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 163: 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)

Configure the PGU, PMU, and RPM using tests

To configure and control the PGU, a user test module (UTM) is needed. You can also use UTMs to configure and control the PMU and RPM preamplifier. For UTM programming, see LPT functions for PGUs and PMUs (on page 14-98).

You can use interactive test modules (ITMs) to configure and control a PMU and RPM preamplifier. Refer to Configure a test (on page 6-15) for information on configuring an ITM for the PMU.

An ITM automatically controls the RPM based on the type of test (pulse, CV, SMU). RPM switching can only be controlled using the rpm_config (on page 14-139) LPT function in a UTM. Refer to Using the RPM as a switch (on page 5-11) 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 in Update the RPM configuration (on page 7-12).

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 in the PMU Advanced Test Settings dialog box. An example is shown in the next figure.

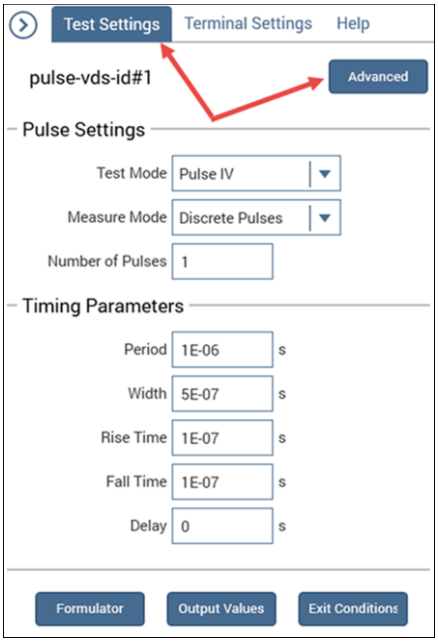
Figure 164: Pulse Timing Preview example

The preview displays representations of the waveforms for the different PMU channels in the test. The illustrated pulse waveforms are defined by the Pulse Timing 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 5-45).

To open the PMU Advanced Test Settings dialog box, select **Test Settings**, then **Advanced** (see next figure).

Figure 165: Timing button (PDU Definition tab)

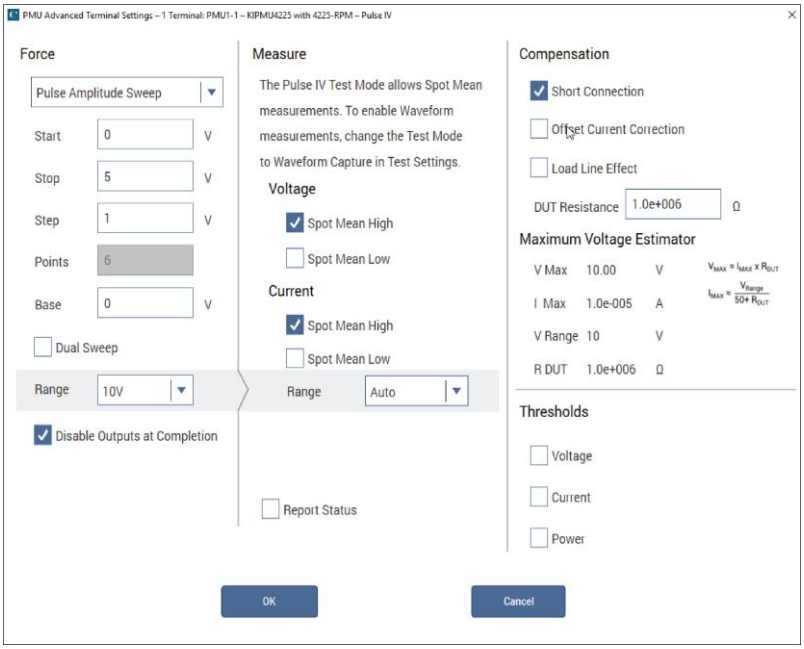


Review the following examples to understand the Pulse Timing Preview feature.

PMU amplitude sweep example (one-channel)

This figure illustrates the configuration for a pulse amplitude sweep using a single PMU channel.

Figure 166: One-channel PMU amplitude sweep



The next figure illustrates the configuration of a six-step pulse amplitude sweep. In this sweep, the PMU Advanced Settings dialog box (see previous figure) defines 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 reflect the PMU Advanced Settings for the PMU channel in the test. The cursor (the pair of vertical black lines on the Entire Test graph) defines the content of the upper graph (the upper graph is labeled “Expanded View”). Note that the test from the previous figure defines the six sweep points that are shown in the bottom graph of the next figure as six pulse periods. In this example, the pulse periods have voltage amplitude increasing from 0 V to 5 V (this is defined in previous figure).

The other graph, labeled “Expanded View,” provides in-depth information on the section designated by the cursor lines (on the graph labeled “Entire Test”). The graphed points reflect the timing parameters (Pulse Timing Preview) and settings from the PMU Advanced Settings dialog box for each PMU channel in the test, but only for the area specified by the cursor.

Figure 167: Six-point pulse amplitude sweep

Select **Preview Test** to animate the waveform. The cursor moves from left to right on the bottom graph while displaying the point waveform of each sweep within the cursor on the top graph.

NOTE

Selecting **Preview Test** 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.

Operation modes and voltages for next figure

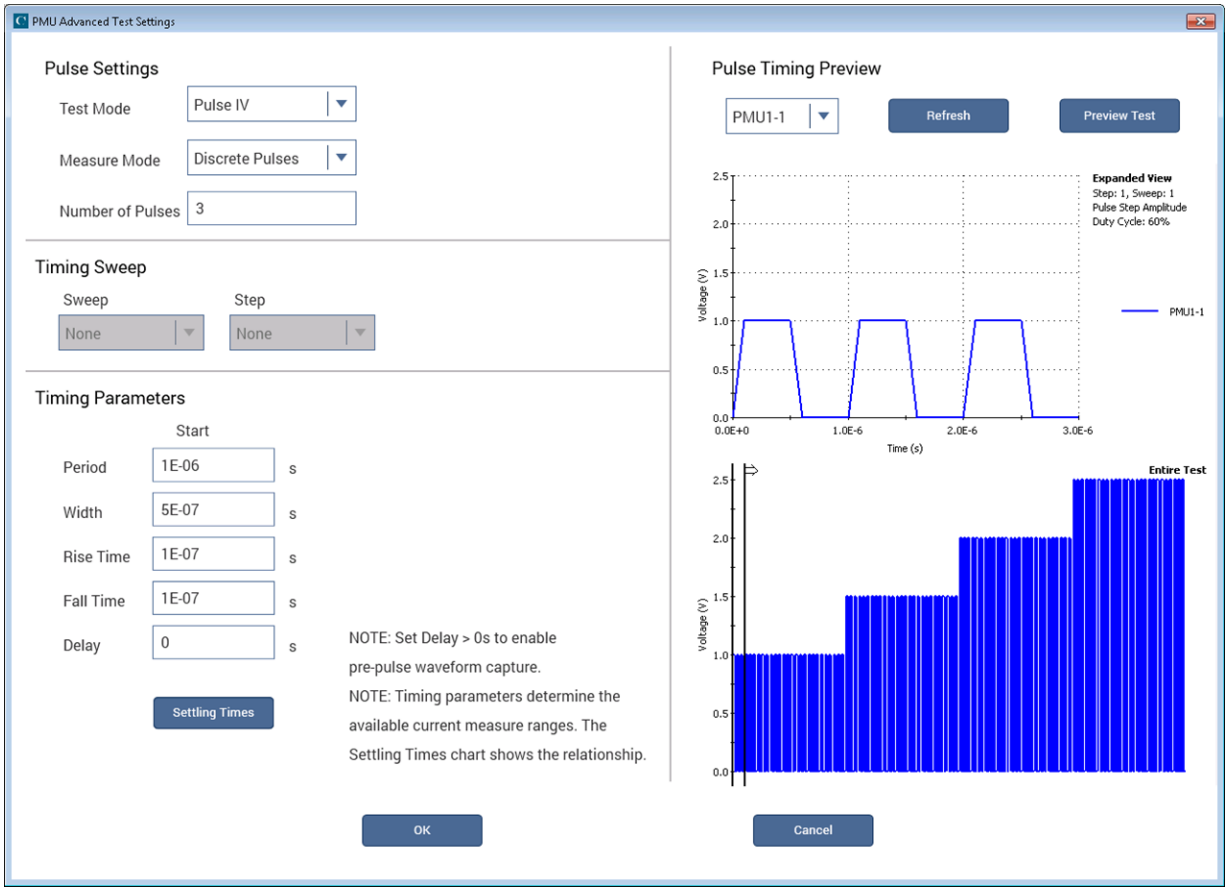
	PMU1-1 (blue waveform)	**PMU1-2 (light blue waveform)**
Operation modes	Pulse Amplitude Step	Pulse Amplitude Sweep
Start Voltage	1	0
Stop Voltage	2.5	4
Step Voltage	0.5	0.5
Number of steps or sweep points	4	9

To preview the test, select the Test Settings tab, then select Advanced. You can preview a single channel or all channels in the test. The graph of the entire test (the lower graph of next figure) shows four steps of nine points each. The sweep and step count displays the position of the cursor in the overall test. This example has the number of pulses set to one.

Figure 168: PMU Timing Preview, two channels

Changing the number of pulses to three (instead of one) and only displaying one channel (PMU1-1, instead of Show All), changes the preview of the waveform (see next figure). Each point in the step now uses three periods, so there are three pulses shown in the Expanded View. In the Expanded View, the 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 previous figure).

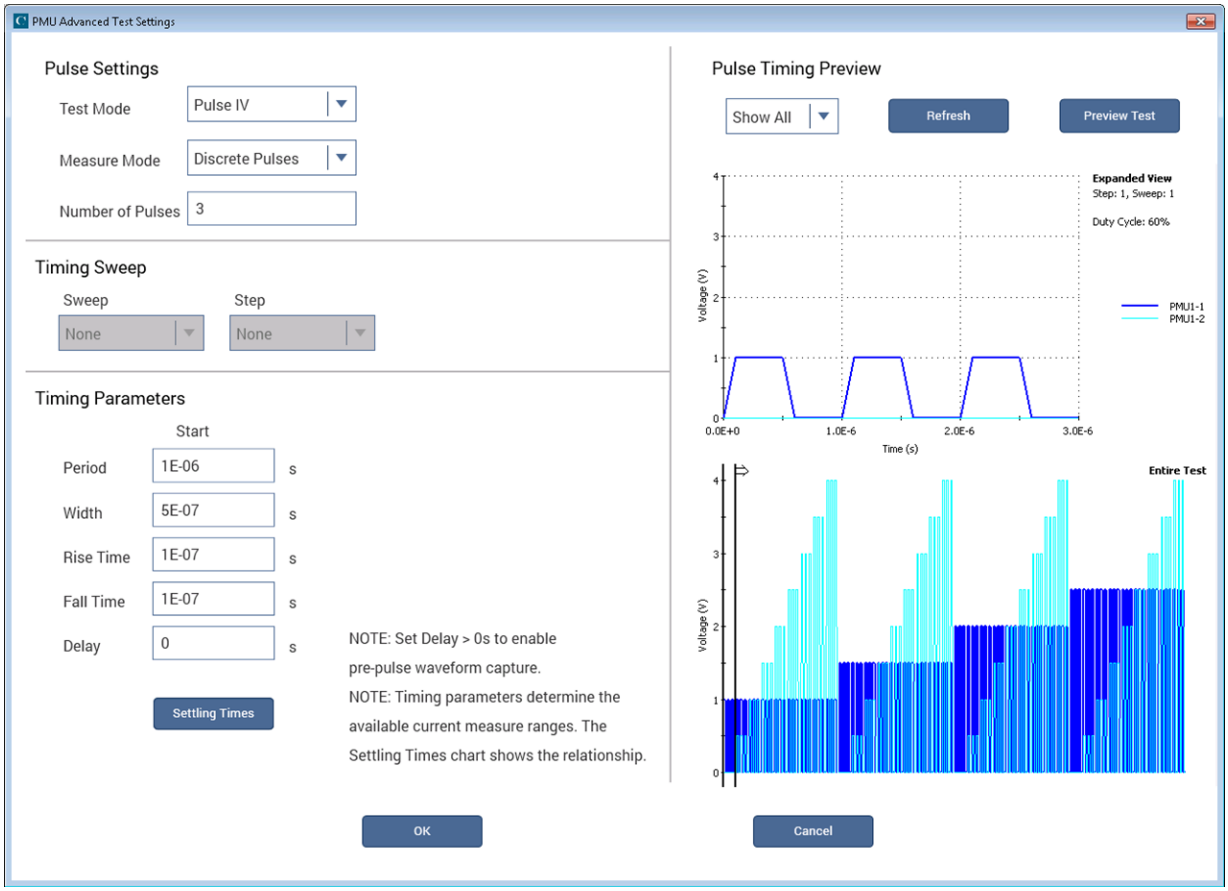
Figure 169: Two-channel test (stepping and sweeping) with three points (only PMU1-1 displayed)



Configuring the preview to all channels displays both the sweeper and stepper for the three-pulse test (see next figure).

The Number of Pulses parameter determines the number of pulses that are output and measured for each attempted point in the sweep or step. The PMU amplitude sweep and step example (two-channel) test figure shows PMU1-1 (the stepper) with the number of pulses changed to three. Notice that each sweep point of the displayed waveform in the figure has three pulses (number of pulses set to three). The light blue waveform in the figure 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. The next figure 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 170: 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. The next figure shows a four-channel test, reflecting the operation mode and voltages given in the next table. When the channel count is higher, using the zoom feature provides a more detailed view of the individual channel.

Expanded View zoom

On the Expanded View graph, drag the cursor to define the area to magnify (the cursor changes to a magnifying glass). 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 the graph or select Refresh. Multiple levels of zoom are supported, with a double-click returning each level. Note that the Refresh option restores normal magnification (shows the Entire Test waveform).

Operation mode and voltages for the four channels

	PMU1-1	PMU1-2	PMU2-1	PMU2-2
Operation mode	**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 the previous table for the key test parameters used in the following figure.

Figure 171: Four-channel sweep and step (2 pulse trains)

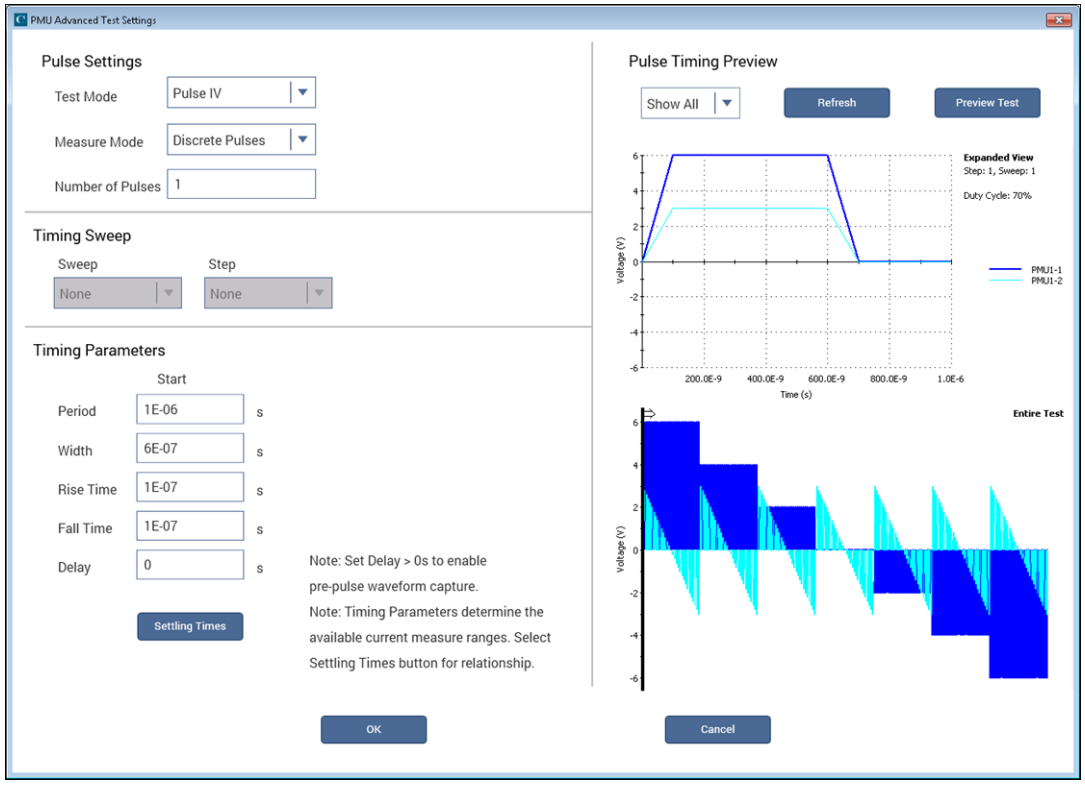
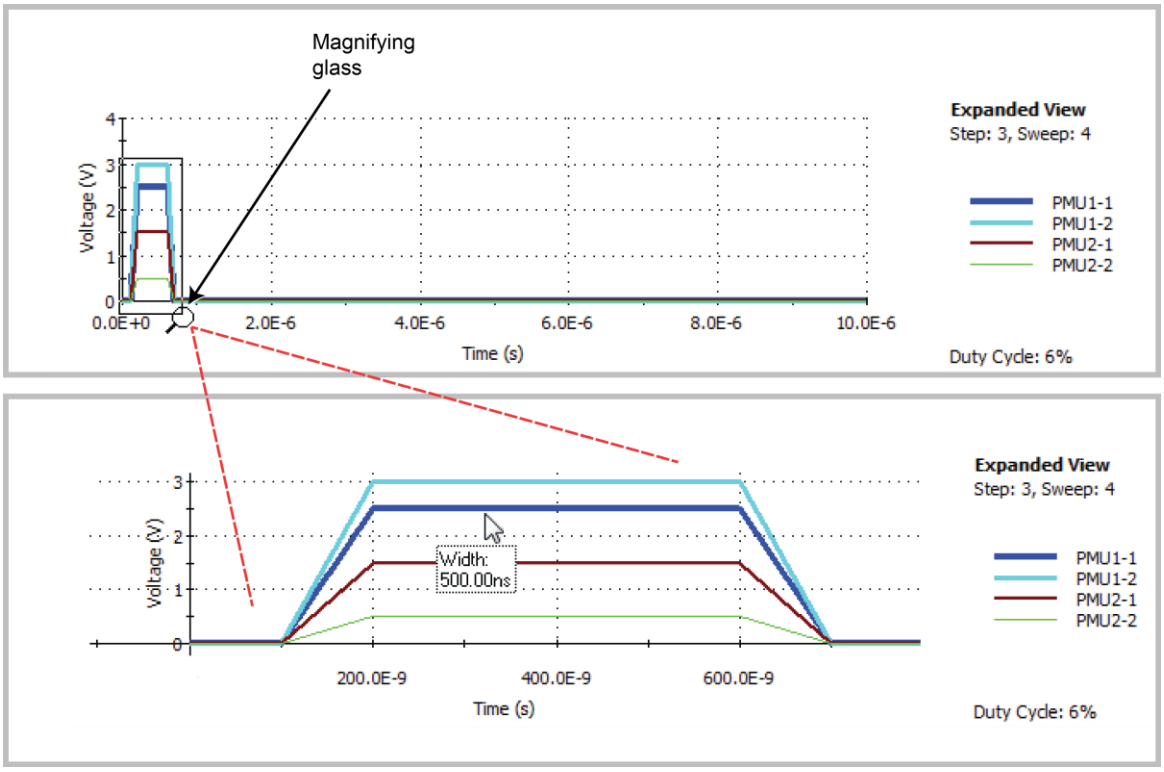


Figure 172: Zooming (Expanded View graph)



Scrolling the magnified area

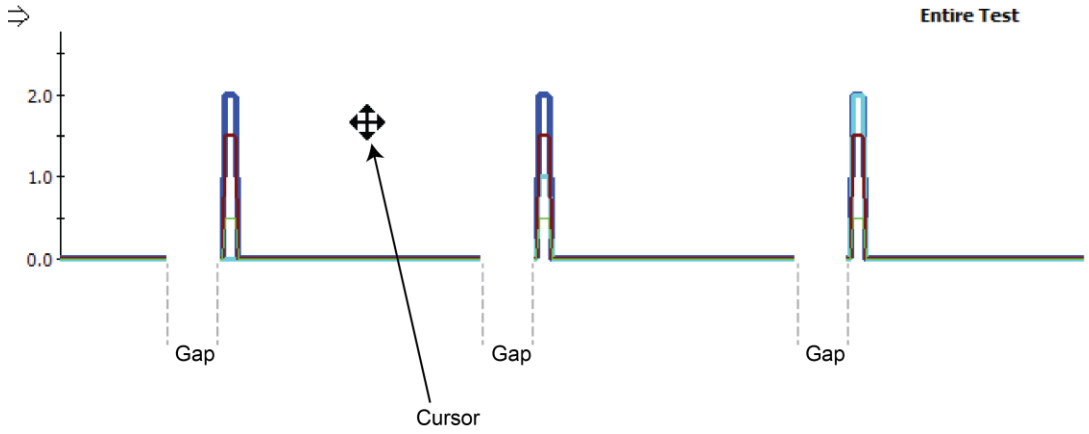
To move the viewable area of the graph after you zoom in on a graph, hold down Shift while dragging the mouse.

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 the next figure for a view of moving the lower graph using the mouse pointer. Note the gaps between the pulse waveforms shown in the Expanded View zoom (on page 5-35) figure; gaps also exist in the next figure. 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 5-45).

While holding down the Shift key on the 4200A-SCS keyboard and left mouse button, move or scroll a magnified graph by moving the mouse.

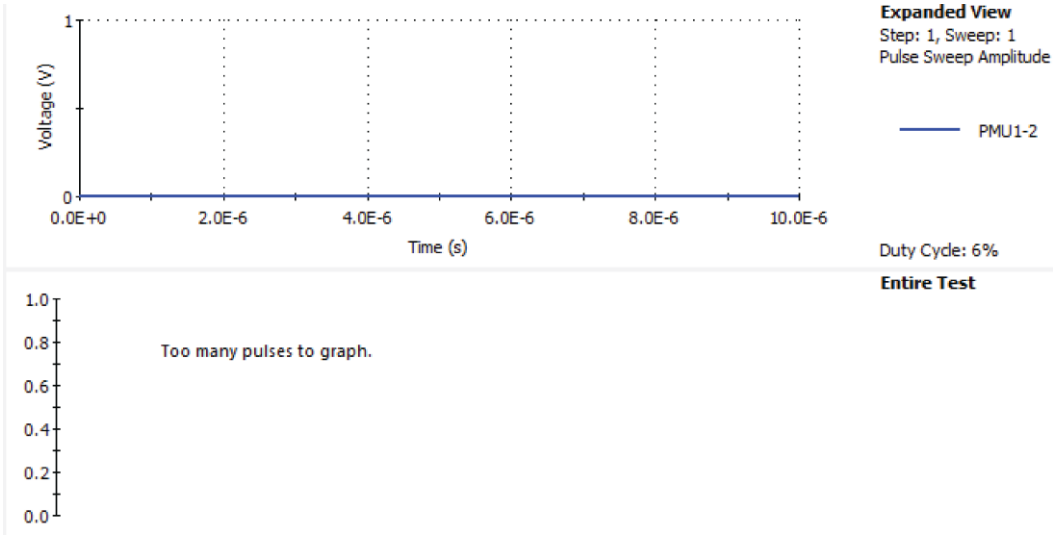
Figure 173: Scroll (or move) a magnified entire test graph



Errors

If either graph is not updating as expected, select Refresh 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 number of pulses. The next figure shows the output “Too many pulses to graph” for the lower graph. Note that this does not mean that the test will not run. If you select OK on the PMU Advanced Test Settings dialog box and it does not generate any warning or error messages, the test is valid and runs (even though the entire test waveform cannot be displayed).

Figure 174: Preview error (Too many pulses to graph)



PMU connection compensation

You can correct errors caused by connections and cable length between the 4225-PMU and the device under test (DUT) by using connection compensation. When connection compensation is enabled, the default or measured compensation values are factored into each DUT measurement.

Connection compensation includes short and offset current compensation options.

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 done from the Clarius software. The custom values provide optimum compensation. Custom connection compensation data is generated for offset current and short conditions. The custom connection compensation values can be enabled or disabled from a test in Clarius.

If connection compensation is disabled, the compensation values will not be applied to the measurements.

NOTE

For optimum performance, you should do connection compensation any time the connection setup is changed or disturbed. Changes in temperature or humidity do not affect connection compensation.

Short compensation

NOTE

For UTMs, the default connection compensation values for short can only be enabled using the pulse_conncomp (on page 14-105) function.

You can perform short compensation to remove measurement errors due to stray resistance in your test configuration. When you run short connection compensation, the following status messages are generated:

- `Starting PMU Cable Compensation...`
- `R = % Ohms`
- `PMU Cable Compensation complete.`

`%` = value (V and I measured, Ohms calculated).

Offset current compensation

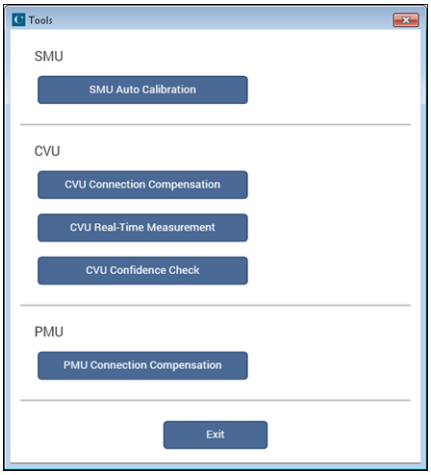
Error currents can be introduced into your pulsed measurements setup by PMUs. PMU offset compensation reduces error currents by subtracting measurements taken at 0 V from all subsequent readings.

Perform connection compensation

To compensate for connections:

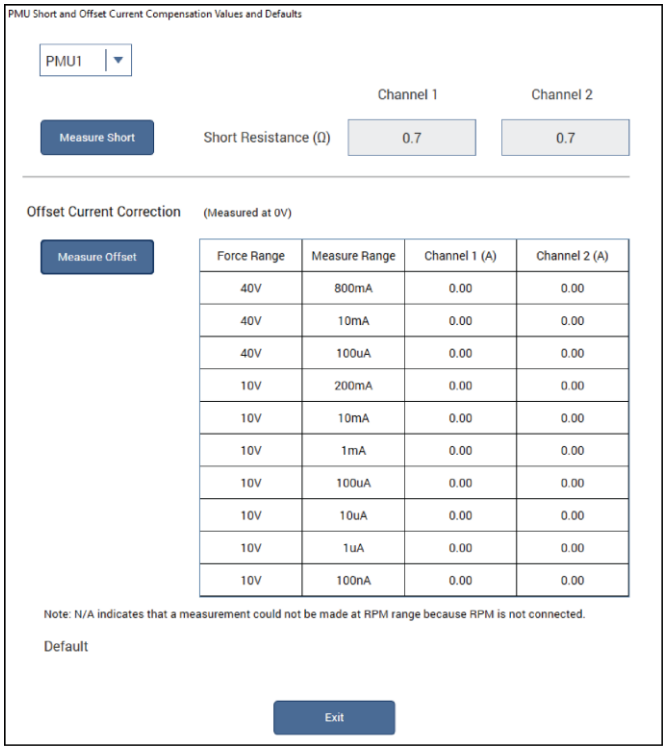
1. In Clarius, select **Tools**. The Clarius Tools dialog box opens.

Figure 175: Clarius Tools dialog box



2. Select **PMU Connection Compensation**. The Short and Offset Current Connection Compensation Values and Defaults dialog box opens.
3. From the drop-down arrow, select the PMU that you want to compensate.

Figure 176: PMU Connection Compensation dialog box



4. To perform short connection compensation, select **Measure Short,** then follow the on-screen instructions or replace the DUT in the test fixture with a short.
5. To perform offset current compensation, select **Measure Offset**, then follow the on-screen instructions.

Compensation results are displayed when compensation is complete. If an error occurred, it is displayed in the Clarius Messages area. The compensation data is displayed in the Short and Offset Current Connection Compensation Values and Defaults dialog box.

If your test setup used both PMU channels (example shown in Using the RPM as a switch (on page 5-11) figure), you will have new custom data for both channels.

Enabling connection compensation

NOTE

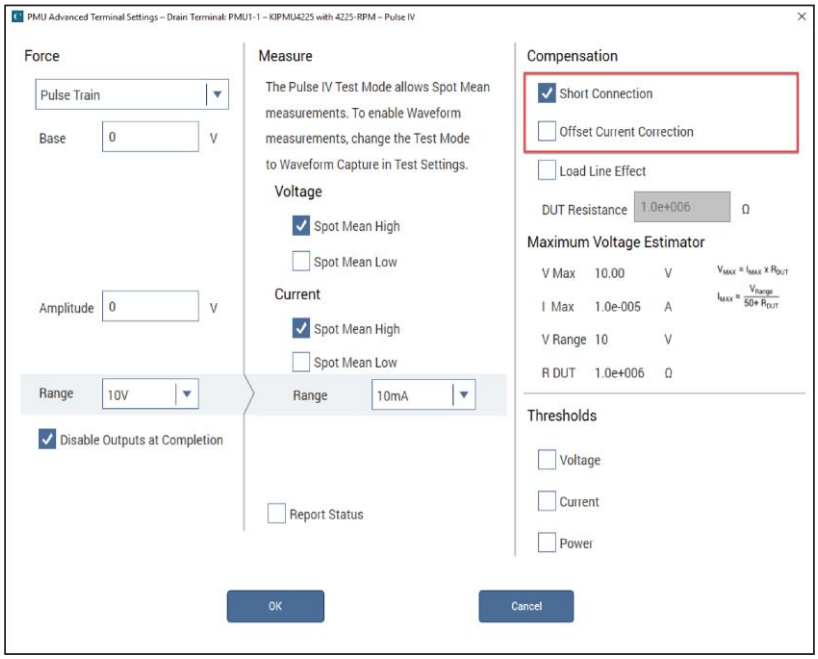
This procedure is for ITMs. For UTMs, you need to enable connection compensation data using the pulse_conncomp (on page 14-105) or setmode (on page 14-145) functions.

To apply the connection compensation data to DUT measurements, you must enable connection compensation for the test.

To enable connection compensation:

1. Select the test.
2. Select **Configure**.
3. Select the terminal to be compensated.
4. In the right pane, select **Terminal Settings**.
5. Select **Advanced**. The PMU Advanced Terminal Settings dialog box is displayed.
6. Select either **Short Connection** or **Offset Current Correction.** See the next figure.

Figure 177: Enabling connection compensation



7. Select **OK**.
8. To disable connection compensation, clear either the **Short Connection** or the **Offset Current Correction** check box, then select **OK**. When disabled, connection compensation values are not applied to DUT measurements.

Load-line effect compensation (LLEC) for the PMU

NOTE

Load-line effect compensation (LLEC) is only performed for standard pulse I-V testing using PMU measure ranges. It is not performed when using 4225-RPM measure ranges (≤10 µA). The active RPM circuitry provides its own analog LLEC (assuming there is a short cable from the RPM to the DUT).

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 (on page 5-80) 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 outputs 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 sources 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

The methods to compensate for load-line effect include:

- Use the built-in load-line effect compensation (LLEC) in the PMU for the standard 2-level pulse mode. Ideally (when LLEC is enabled), the PMU adjusts its output levels such that the programmed output voltage appears at the DUT. For ITMs, see Load-line effect compensation (on page 5-44). For UTMs, use the pulse_meas_sm (on page 14-119) function to control LLEC.
- Manual adjustment of 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.

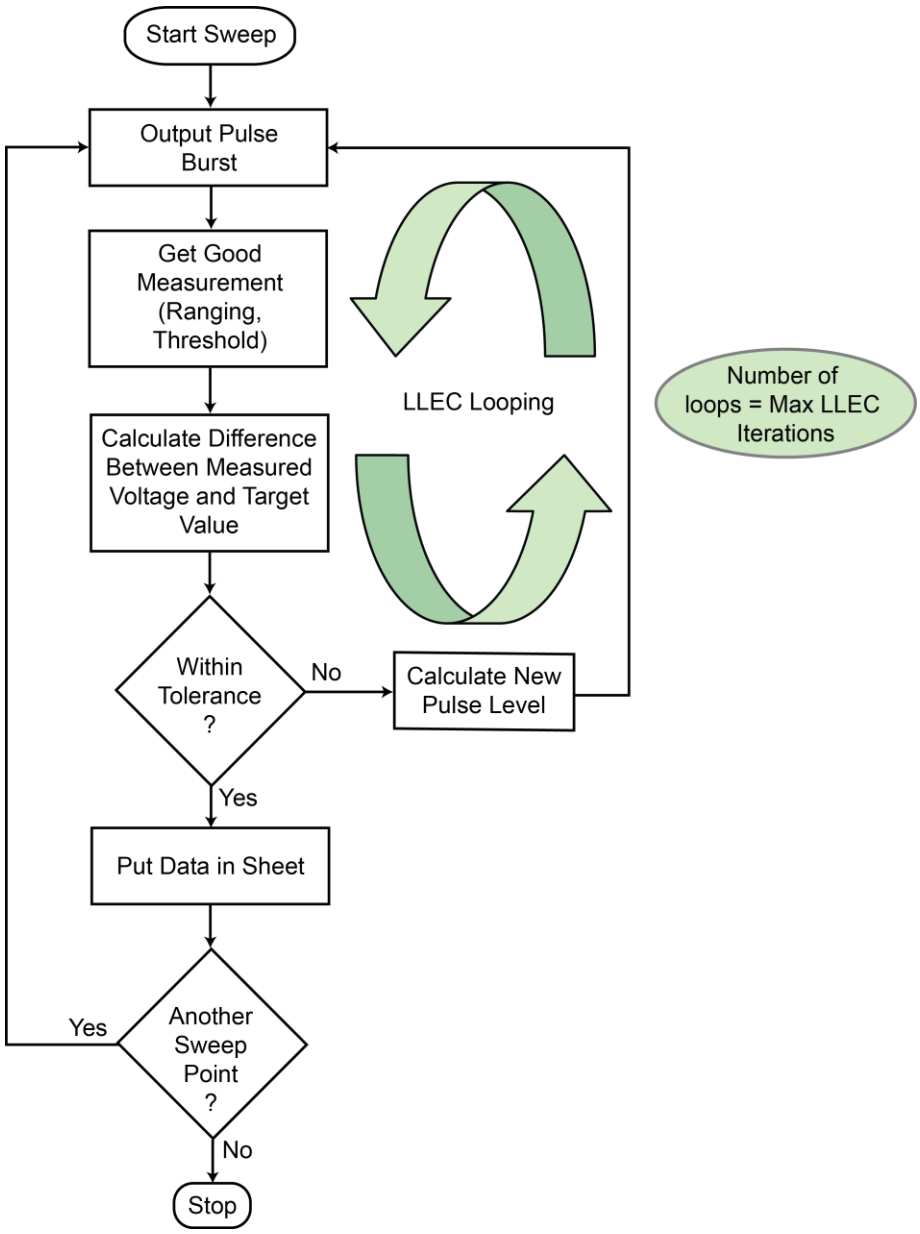
If you are not using LLEC, you can manually set the output impedance to match the impedance of the DUT. For example, if the impedance of the DUT is 1 kΩ, set the output impedance to 1 kΩ. Maximum power transfer is achieved when the DUT impedance matches the output impedance setting. In a real test environment, 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 Disable LLEC and set the output impedance (on page 5-50, on page 5-51)) or from a UTM (see pulse_load (on page 14-164) function).

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 the next figure. 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 for ITMs or the pulse_meas_timing (on page 14-121) 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 178: 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 the previous figure. 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 I-V 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 the previous figure 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 PMU - all terminal parameters (on page 6-95).

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 (on page 14-209) 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.

Coping with the load-line effect

There are several ways of working with this effect. The simplest one is to program the DUT load into the pulse card channel using the pulse_load (on page 14-164) command, or setting the Pulse Load value in the KPulse (on page 11-1) virtual front panel. 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 correct levels. This works well for high impedance devices or device terminals (R_{DUT} = 1 kΩ), 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 test voltage needs to be measured, with the pulse levels adjusted until the correct voltage is reached. Record each pulse level required to reach the required V_{DUT} levels.

NOTE

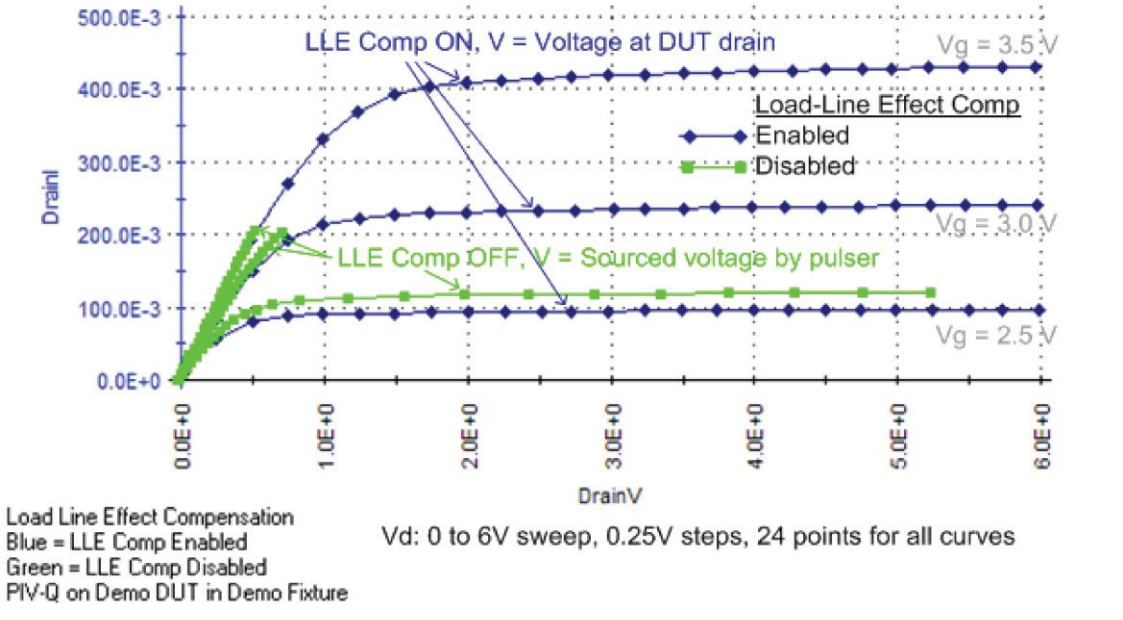
The 4225-PMU has built-in load-line effect compensation. For details, see Load-line effect compensation (LLEC) for the PMU (on page 5-44).

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. The next figure 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 179: Load-line effect on FET family of curves



In the previous figure, 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.

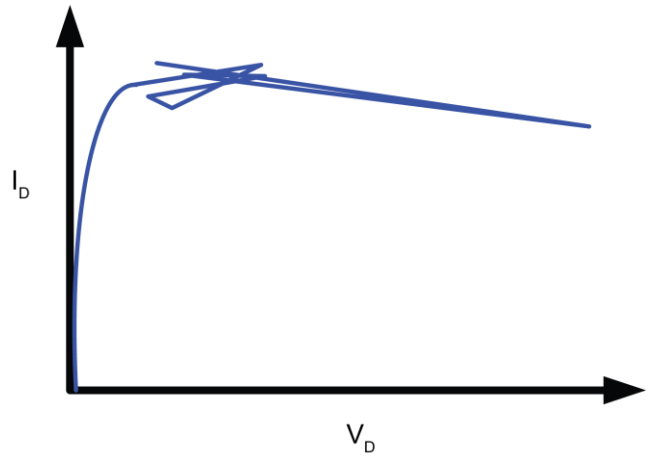
The same sweep is used to generate the green curves (LLEC off). The best green curve is the one at the bottom (bias V_g = 2.5 V). However, load-line effect prevents the PMU from sourcing 6 V to the DUT and the 24 pulse-measure points are not evenly spaced. Modifying the sweep to 0 to 6.5 V will ensure 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 (V_g = 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 (on page 5-50, on page 5-51). For UTMs, see the pulse_meas_sm (on page 14-119) and pulse_meas_wfm (on page 14-123) functions.

Figure 180: Curve showing poor LLEC compensation



LPT functions used to configure LLEC

The LPT functions used to configure LLEC for the PMU are:

- pulse_load (on page 14-164): 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 (on page 14-209): 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 (on page 14-119) and pulse_meas_wfm (on page 14-123): Use these functions to enable or disable LLEC.

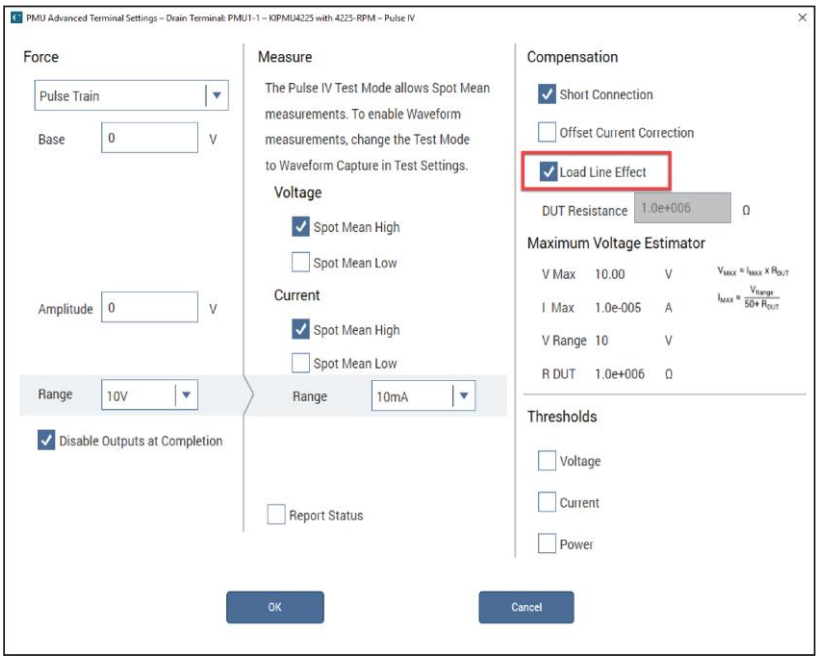
Enable LLEC

This option is available for ITMs.

To enable LLEC:

1. Select the pulse test.
2. Select **Configure**.
3. In the right pane, select **Terminal Settings**.
4. Select **Advanced**.
5. Select **Load Line Effect**, as shown in the following figure.
6. Select **OK**.

Figure 181: Enabling LLEC



Disable LLEC and set the output impedance

These options are available for ITMs.

With LLEC disabled, you can input the known resistance of the DUT. The resistance value is then used in a compensation process to output the voltage.

The Max Voltage Estimator is a tool that you can use to 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 maximum voltage 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 LLEC setting does not change the maximum output voltage or current of the PMU. The V Max and I Max values are the maximum output of the PMU and are valid if LLEC is enabled or disabled. The DUT Resistance setting also does not affet the maximum output voltage. 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. Refer to DUT resistance determines pulse voltage across DUT (on page 5-80) for detail on calculating DUT resistance.

To disable LLEC and set the output impedance:

1. To disable LLEC, clear **Load Line Effect**.
2. In the DUT Resistance box, enter the resistance value.
3. Review the Max Voltage Estimator values.
4. Click **OK**.

Pulse parameter definitions

For more information regarding the individual pulse parameters, refer to PMU - all parameters (on page 6-95).

PMU minimum settling times versus current measure range

The PMU and RPM current measure ranges require time to reach a settled value. The amount of settling time that is required increases for the lower current measure ranges.

You can set the pulse timing parameters (pulse width, rise, and fall) 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 are used, along with the minimum timing in the chart to determine which current measure ranges are available.

In ITMs, when you modify the PMU timing settings in the Advanced Test Settings, if you make a pulse timing change that affects the measure range of a PMU channel, a message noting the unavailable ranges is displayed. 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 window turns red and lists the unavailable ranges. Access the PMU force measure options by clicking the FORCE MEASURE button.

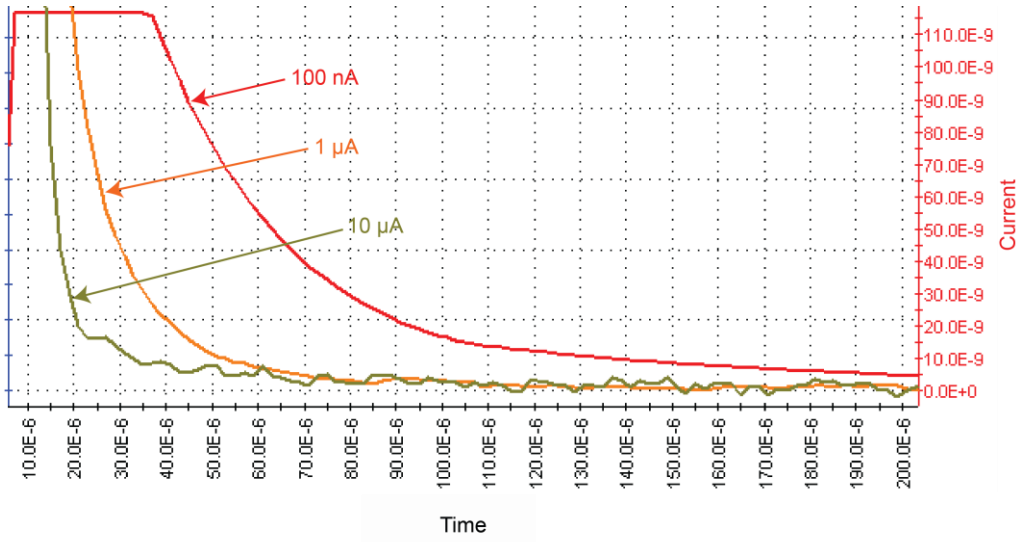
The Typical Minimum Timing Recommendations dialog box shows the recommended minimum pulse widths and transition (rise and fall) times for each current measurement range. These times are the minimum time necessary for the PMU, or PMU with RPM, to reach a settled value (into an open). Additional time must 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.

You can override the settled measurement requirement for the PMU or PMU+RPM measurements. In the Clarius My Settings options, select PMU: Allow unsettled measurements (on page 6-376).

The next figure contains a graph of the current signal settling time of three current measure ranges of the 4225-RPM. The graph shows the settling time of a 1 V amplitude pulse in to an open (high impedance) with a 300 ms pulse width and a 1 ms transition time. Note that this figure does not show the voltage signal.

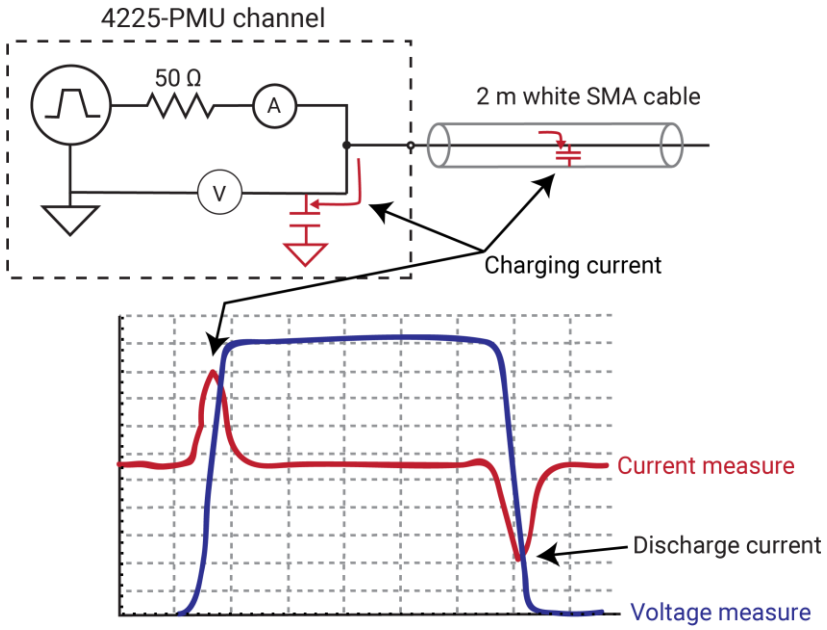
Figure 182: 4225-RPM current signal settling time



PMU capacitive charging/discharging effects

During pulse transitions, the measured current charges and discharges the capacitance in the system (see next figure, red waveform). This system capacitance consists of the cable capacitance, PMU (with RPM, if connected) capacitance, and device capacitance. The next figure 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 183: Capacitive charging and discharging



The setup used to generate these waveforms is shown in the previous figure, 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 the previous figure 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

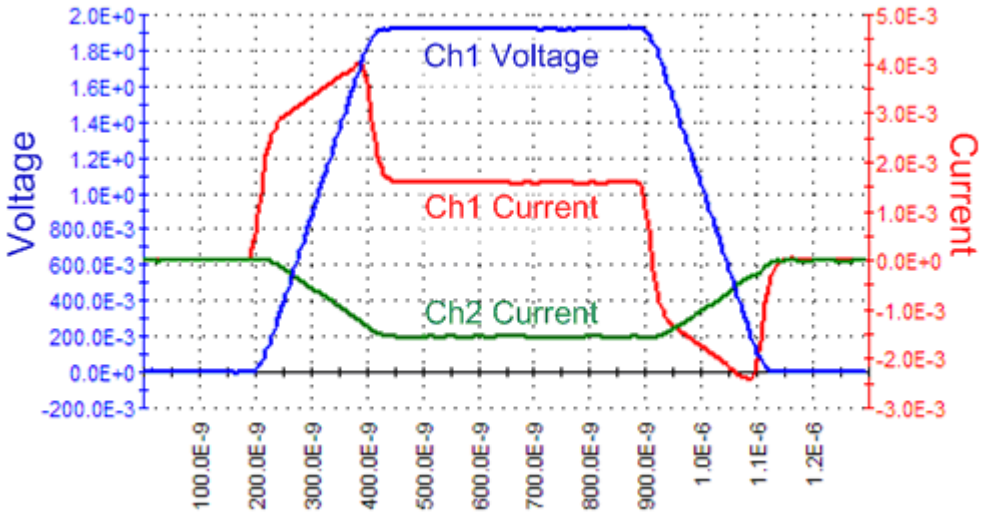
The equation shows that this effect is a function of the capacitance, as well as the dV/dt. Therefore, minimizing the capacitance will reduce this measurement artifact. The cabling is typically the largest contributor to the system capacitance. Slowing down the pulse transitions will also reduce the 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.

The next two figures show the waveforms and setup for a pulse test on a resistor DUT and illustrates a configuration to eliminate this artifact. The next figure 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.

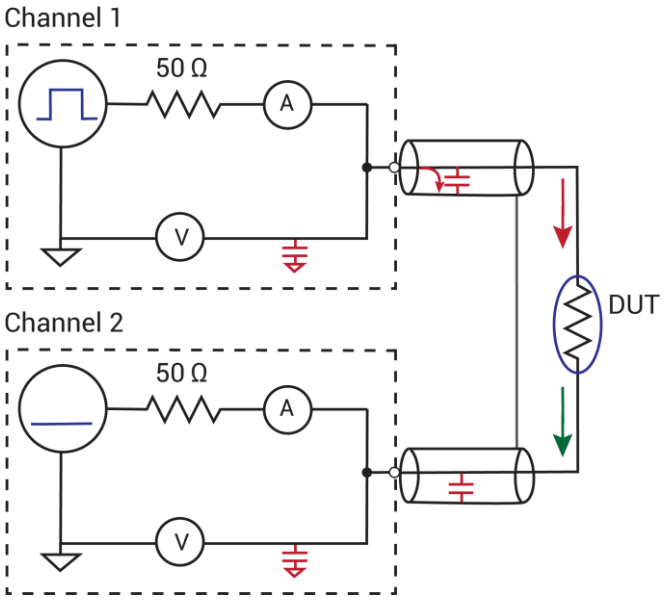
Figure 184: Low-side measurement waveforms

Figure 184: Low-side measurement waveforms



The previous figure 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.

Figure 185: Setup for low-side measurement



In the previous figure, 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 and 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.

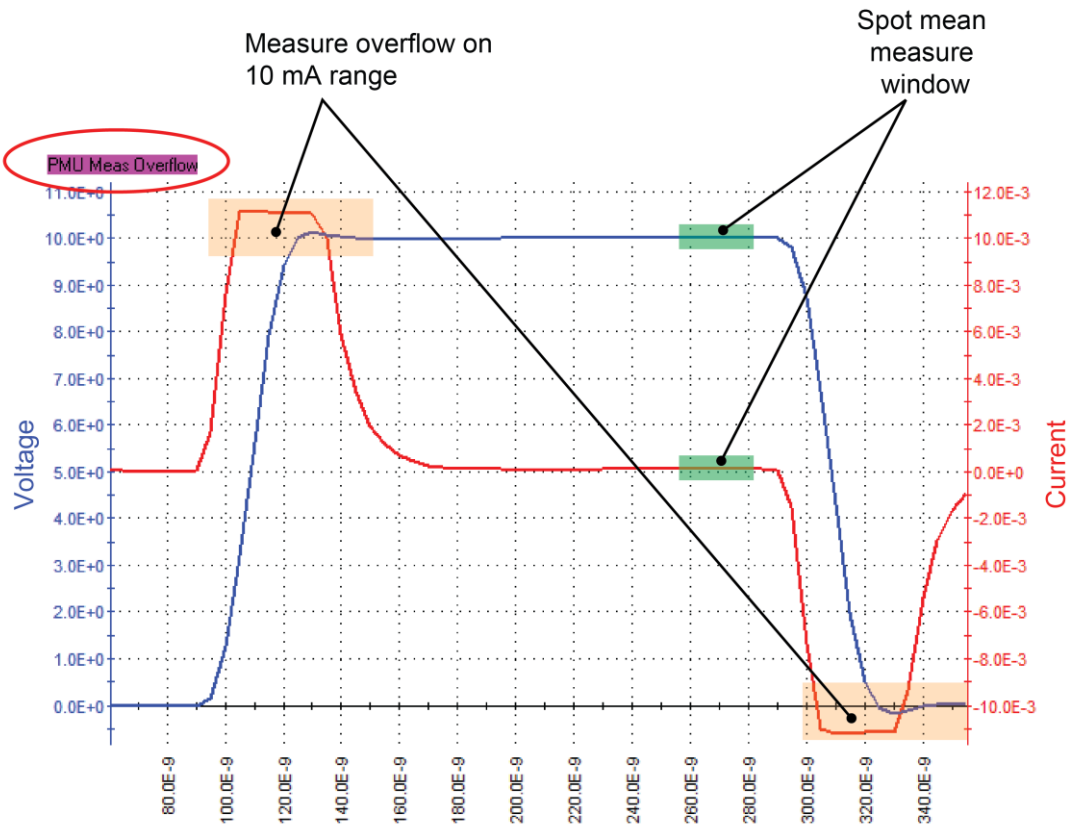
To minimize the measurement time, the signal at the device under test (DUT) must get to the desired voltage level and settle as quickly as possible. A key aspect of this goal is handling the capacitive charging effects during the pulse transitions (see PMU capacitive charging/discharging effects (on page 5-53)).

Since the interconnect and the DUT 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. The next figure shows an example of measurement overflow on the 10 mA range of the RPM, pulsing into a 1 MW 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 the next figure shows 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 (on page 5-53) for information on the cause of the capacitive charging effect and how to work around it.

Figure 186: PMU Waveform Capture (measurement overflow on the 10 mA current range)



4220-PGU and 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 5-80)), the pulse instrument cards also have a limit for the number of large amplitude pulse transitions within a period of time. The 4200A-SCS system enforces limits on the quantity and amplitude of waveforms that the 4220-PGU and 4225-PMU may generate. A test exceeding these internal limits generates error code `-830` (see pulse_init (on page 14-163) for more information). To fix this problem, increase pulse period, decrease voltage amplitude, or both.

4200A-SCS power supply limitations

In some system configurations, the 4200A-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 selected test has too many channels enabled.

Clarius 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 is displayed and the test will not run.

The next table 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 4210-SMU High Power SMU, 4225-PMU Ultra-Fast Pulse Measure Unit, and 4220-PGU High Voltage Pulse Generator draw the majority of the power in the 4200A-SCS chassis.

There are two parts to the total power supply draw. The first part is the power required for the instruments while the 4200A-SCS 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 preamplifiers (4200-PA), and 4210-CVU modules are not included in the equations, as their power draw is not significant.

4200A-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
10 V PGU or PMU channel*	NA	8.4
40 V PGU or PMU channel*	NA	54.6
Medium Power SMU (4200-SMU)	NS	NS
4210-CVU	NS	NS
NS = Not Significant		
NA = Not Available		
* There are two channels for each PGU and PMU instrument card.		

Equations to calculate power:

PowerIDLE = [(29.4 * nPGU) + (50.4 * nPMU) + (4.2 * nRPM)]

PowerTEST = [(45 * nHPSMU) + (8.4 * nC10) + (54.6 * nC40)]

PowerTOTAL = PowerIDLE + PowerTEST

PowerTOTAL = 500 maximum. If PowerTOTAL is less than or equal to 500, the test proceeds.

Where:

- nPGU = number of 4220-PGU cards in the 4200A-SCS chassis (idle power draw)
- nPMU = number of 4225-PMU cards in the 4200A-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 10 V PGU or PMU channels in the test
- nC40 = number of 40 V PGU or PMU channels in the test

Example 1: 4200A-SCS with two 4210-SMUs, four 4225-PMUs, and eight 4225-RPMs. The test uses all eight PMU+RPM channels set to the 10 V range (no SMUs in test).

PowerIDLE = [(29.4 * nPGU) + (50.4 * nPMU) + (4.2 * nRPM)] = [(29.4 * 0) + (50.4 * 4) + (4.2 * 8)] = 201.6 + 33.6 = 235.2

PowerTEST = [(45 * nHPSMU) + (8.4 * nC10) + (54.6 * nC40)] = [(45 * 0) + (8.4 * 8) + (54.6 * 0)] = 67.2

PowerTOTAL = PowerIDLE + PowerTEST = 235.2 + 67.2 = 302.4

This test has PowerTOTAL ≤500, so this test will proceed.

Example 2: 4200A-SCS with two 4210-SMUs, four 4225-PMUs, and eight 4225-RPMs. The test uses five PMU+RPM channels set to the 40 V range (no SMUs in test).

PowerIDLE = [(29.4 * nPGU) + (50.4 * nPMU) + (4.2 * nRPM)] = [(29.4 * 0) + (50.4 * 4) + (4.2 * 8)] = 201.6 + 33.6 = 235.2

PowerTEST = [(45 * nHPSMU) + (8.4 * nC10) + (54.6 * nC40)] = [(45 * 0) + (8.4 * 0) + (54.6 * 5)] = 273

PowerTOTAL = PowerIDLE + PowerTEST = 235.2 + 273 = 508.2

This test has PowerTOTAL >500, so this test will not proceed. Reduce the number of 40 V channels from five.

The next table shows the 4200A-SCS power requirements for valid combinations for the 4225-PMU, 4225-RPM, and high-power SMU.

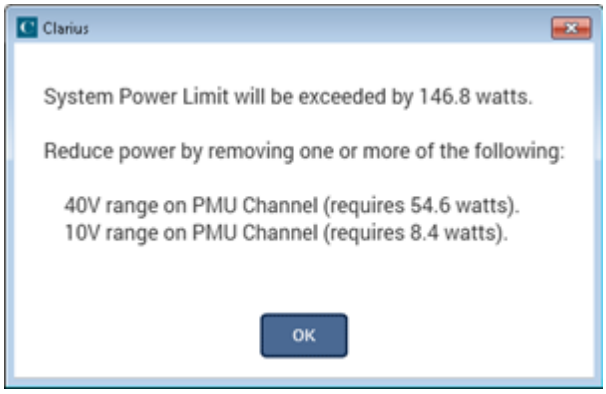
4200A-SCS power requirements for valid combinations of internal system instruments

Idle power		**Power used in test**					
nPMU	nRPM	nC10	nC40	nHPSMU	PowerIDLE	PowerTEST	PowerTOTAL
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
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 the next figure when configuring PMU.

Figure 187: ITM maximum power exceeded message



For UTMs, the message appears in the Clarius Messages pane.

Basic troubleshooting procedure

If the test 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, good pulse I-V results require a reasonable pulse shape.

Step 1. Verify prober connections from the PMU or RPM to the DUT

1. Use cabling and connections optimized for high frequency (>150 MHz).
2. Connect the low side of the device under test (DUT) to the shield of the coaxial cable (see Shield connections (on page 5-14)), or connect the low side to another PMU or RPM channel (see Connections to prober or test fixture bulkhead connectors (on page 5-21) or Pulse I-V (Average pulses) measurement example (on page 6-132)).
3. Connect the shields (refer to the figure in Local sensing (on page 5-24) that shows four-terminal local sense connections).
4. If you are not using the supplied cabling, minimize the loop area 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. Refer to Shield connections (on page 5-14).
5. Minimize the cable length. See Cable length (on page 5-15).

Step 2. Verify the pulse shape

To check that the pulse shape provides a flat, settled portion near the end of the pulse top:

1. Configure the test for Waveform Capture.
2. Ensure that voltage, current, and time waveforms are measured. See Waveform capture measurement configuration (on page 6-100).
3. Enable the status for each PMU channel in the test. See PMU - all terminal parameters (on page 6-95).
4. Select 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, viewing the waveform shape at two or three different voltage levels may be necessary.
5. Configure the ranges to match the pulse I-V test.
6. Configure the connection and load-line effect compensation (LLEC). To match the pulse I-V test conditions, enabling LLEC allows the PMU to compensate for lower voltage levels at the test device when current is flowing. See Controlling LLEC from an ITM (on page 5-50, on page 5-51).
7. 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 **Graph Settings** and select **Define Graph** to open the Graph Definition dialog box. By default, the voltage waveforms are blue and use the left (Y1) axis; the current waveforms are red and use the right (Y2) axis.
8. Save the project.
9. 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. The peaks during the transitions are expected; see PMU capacitive charging/discharging effects (on page 5-53). The current waveform may show settling, but should not have significant ringing. An example of a good waveform shape is shown in PMU and RPM measure ranges are not source ranges (on page 5-57).

Due to interconnect and DUT settling time requirements, low current measurements (< ~10 mA) may require a wider pulse than the recommended minimum timing values. This is especially important for <1 mA level current measurements using the 4225-RPM. See PMU minimum settling times versus current measure range (on page 5-52)

A waveform graph may show a Measurement Overflow condition.

This overflow during the pulse transitions does not affect the spot mean measurements, because the spot mean is taken during the settled portion of the pulse top. 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. Refer to Test Mode (PMU) (on page 6-130), PMU and RPM measure ranges are not source ranges (on page 5-57), and Measure Mode (on page 6-131) for more information.

Step 3. Is the pulse level correct for each channel?

1. If the pulse level is not correct for each channel, enable load-line effect compensation (LLEC). 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. See Controlling LLEC from an ITM (on page 5-50, on page 5-51).
2. Run the test again and compare results. If the results match the test settings, 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 5-44).

Refer to Step 4. Is the pulse I-V curve suspect? (on page 5-64) for examples with LLEC disabled and enabled.

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.

In this procedure, you set test parameters to provide boundaries for the test envelope. When load-line effect compensation (LLEC) is disabled, the source voltage must be bounded.

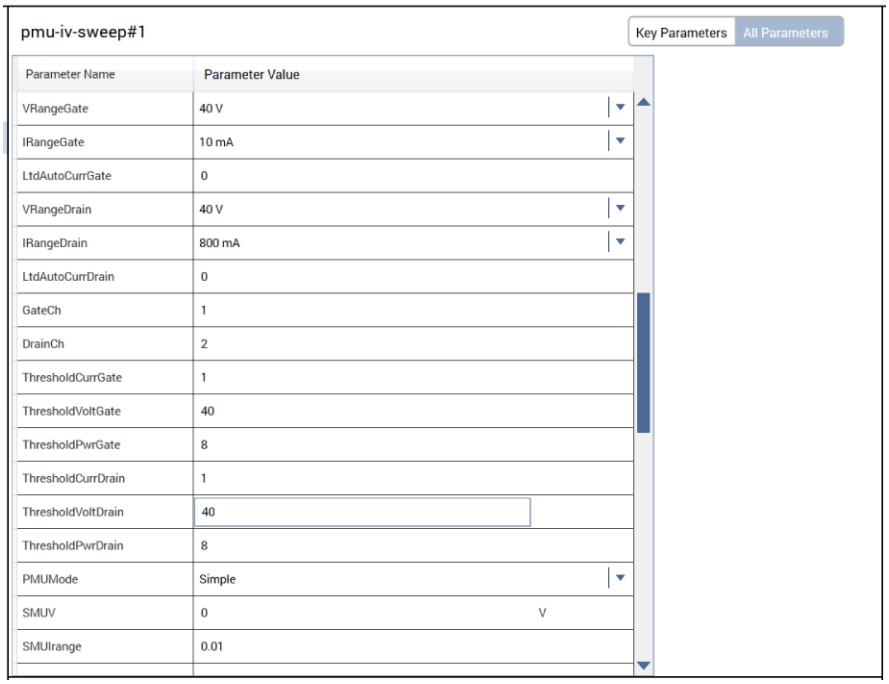
NOTE

LLEC may not respond properly for a high-gain transistor (for example, a compound semiconductor-based amplifier or power transistor).

If the pulse I-V curve is suspect:

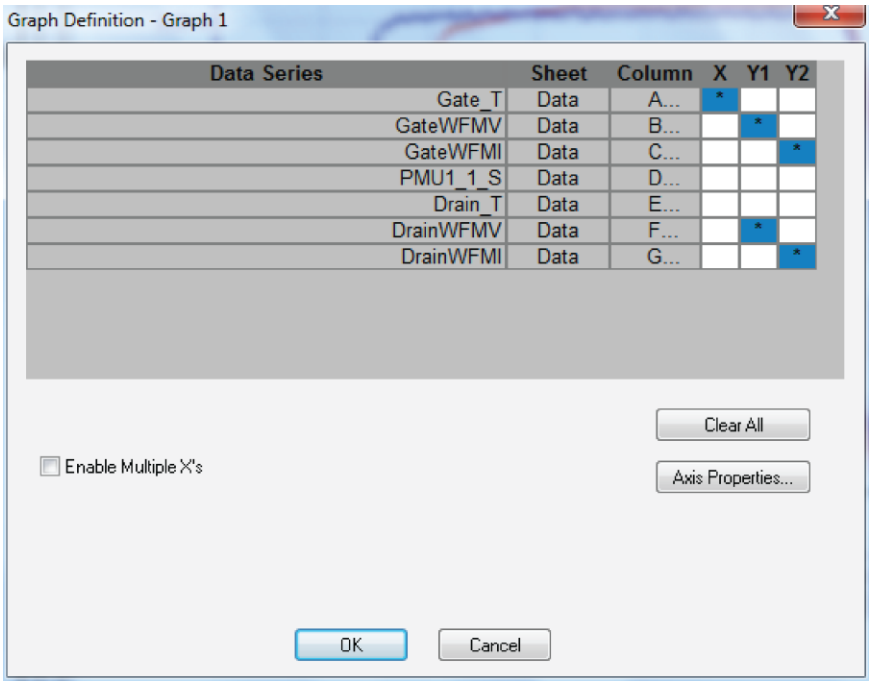
1. Disable the load line for the gate and drain in the Key Parameters. For an ITM, see Load-line effect compensation (on page 5-44).
2. Choose the maximum voltage for the selected source range. For example, many high power transistors require fairly high voltages and currents, so the PMU 40 V source range is common.
3. Set the thresholds in the All Parameters pane (see next figure and PMU - all terminal parameters (on page 6-95) for more detail). Enter the maximum voltage for the DUT. For a transistor, set the `ThresholdVoltDrain` voltage for the maximum voltage for the drain.

Figure 188: PMU IV sweep All Parameters thresholds



4. 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 the "Vd-Id family of curves with LLEC disabled" figure. 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 189: Four-channel waveform test graph definition (displaying V and I for each channel)



The next figure shows a family of curves from a fairly high powered device with the load-line effect. The following figure 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. In this case, the PMU is at its limit and cannot source any more voltage or current to this particular resistance. See the *4200A-SCS Parameter Analyzer Datasheet* for more information on the PMU maximum source power versus device resistance. You can access the datasheet from the *Learning Center*.

Figure 190: Vd-Id family of curves, showing load-line effect (LLEC disabled)

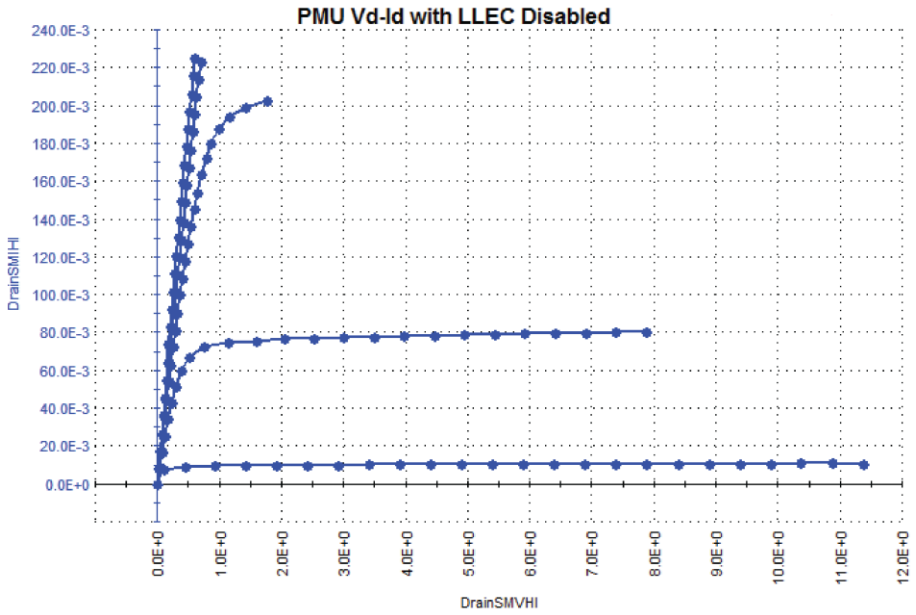


Figure 191: Vd-Id family of curves with LLEC enabled (every curve finishes at Vd = 12 V)

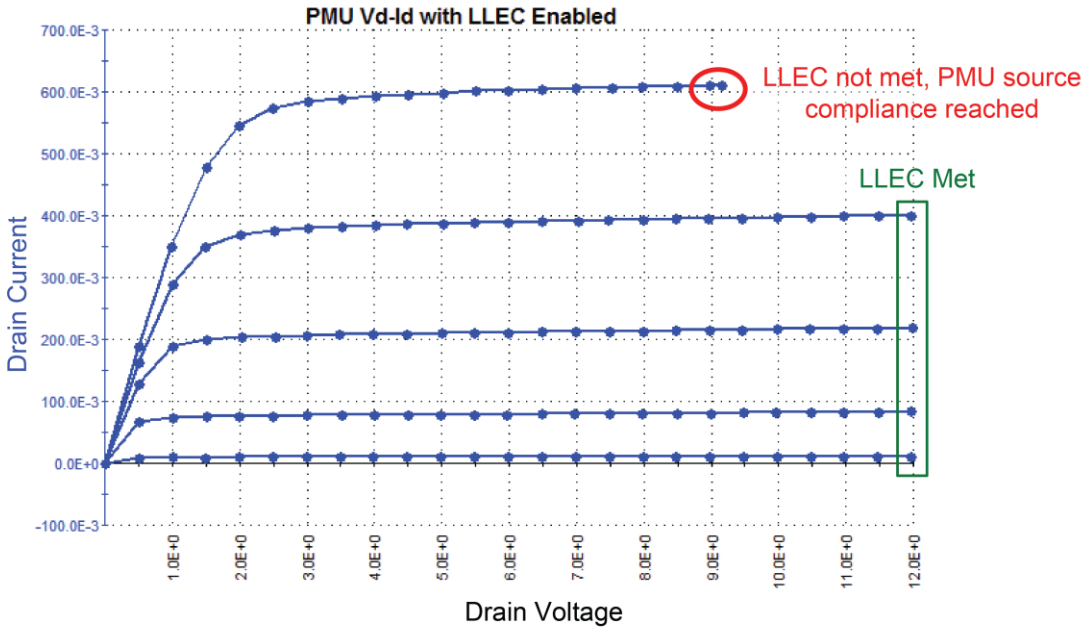
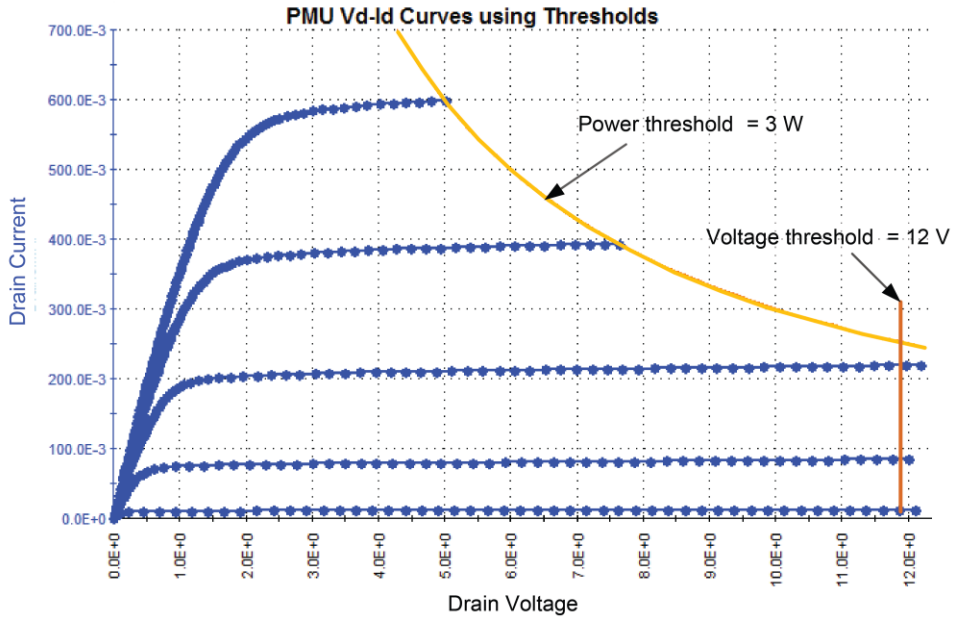


Figure 192: Vd-Id family of curves with LLEC disabled (thresholds_voltage = 12 V_power = 3 W)

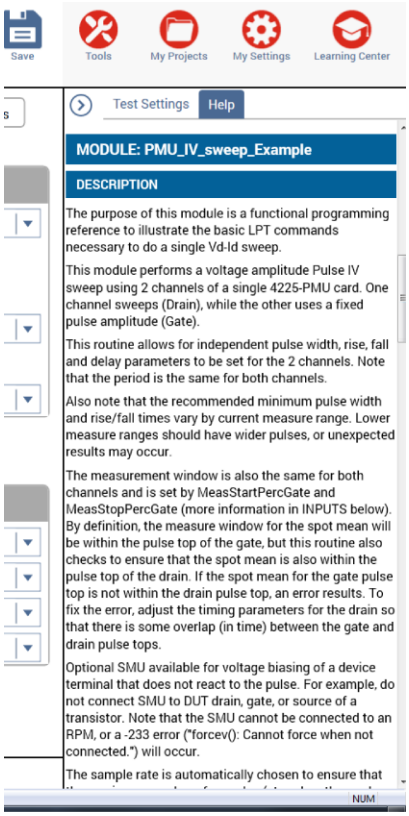


For user test modules (UTMs), although the process described above applies, there are a few differences. Although ITMs and UTMs can perform similar tests, UTMs are based on user modules. UTMs have a much wider range of test behaviors and possibilities than ITMs. This document provides a framework for 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 minimal 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 next figure shows the user module description for `PMU_IV_sweep_Example`. 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 (you will need to scroll down the Help pane).

Figure 193: PMU IV sweep Help pane description

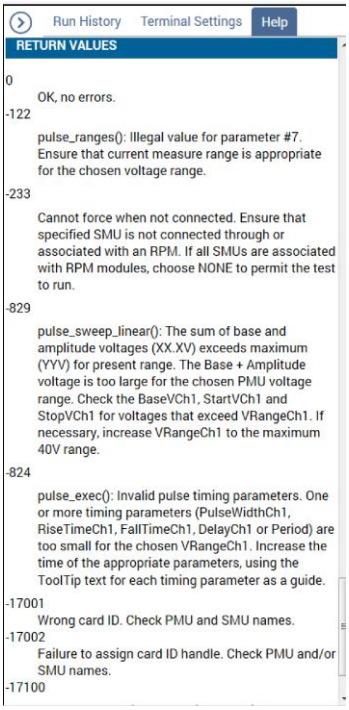


This user module allows for an optional SMU to DC bias a transistor bulk while performing a pulse I-V sweep with a 4225-PMU and optional 4225-RPM. However, due to system restrictions, this SMU must not be connected to the test device through the 4225-RPM (an error results).

This example test is configured to use SMU1. SMU1 is connected through RPM1 of the system.

1. Set the SMU_ID from NONE to SMU1.
2. Select **Analyze** and view the Messages area at the bottom of the screen for error messages. For example, if you get a message indicating "forcev(): Cannot force when not connected," you can check the Return Values in the Help pane for more troubleshooting information (see next figure). Note that value `-233` indicates, "Ensure that the specified SMU is not connected through or associated with an RPM. If all SMUs are associated with RPM modules, choose NONE to permit the test to run."

Figure 194: PMU IV sweep Return Values



NOTE

It may also be necessary to research further using additional error codes. Three-digit negative error codes are most likely LPT-command error codes. Refer to LPT Library Status and Error codes (on page 14-238). 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.

Pulse source-measure concepts

Ultra-fast I-V sourcing and measurement have become increasingly important capabilities for many technologies, including compound semiconductors, medium-power devices, nonvolatile memory, microelectromechanical systems (MEMs), nanodevices, solar cells, and CMOS devices. Using pulsed I-V signals to characterize devices rather than DC signals makes it possible to study or reduce the effects of self-heating (joule heating) or to minimize current drift or degradation in measurements due to trapped charge. Transient I-V measurements allow scientists and engineers to capture ultra high-speed current or voltage waveforms in the time domain or to study dynamic test circuits. Pulsed sourcing can be used to stress test a device using an AC signal during reliability cycling or in a multi-level waveform mode to program or erase memory devices. The 4225-PMU Ultra-Fast I-V Module for the 4200A-SCS supports many of these high speed sourcing and measurement applications.

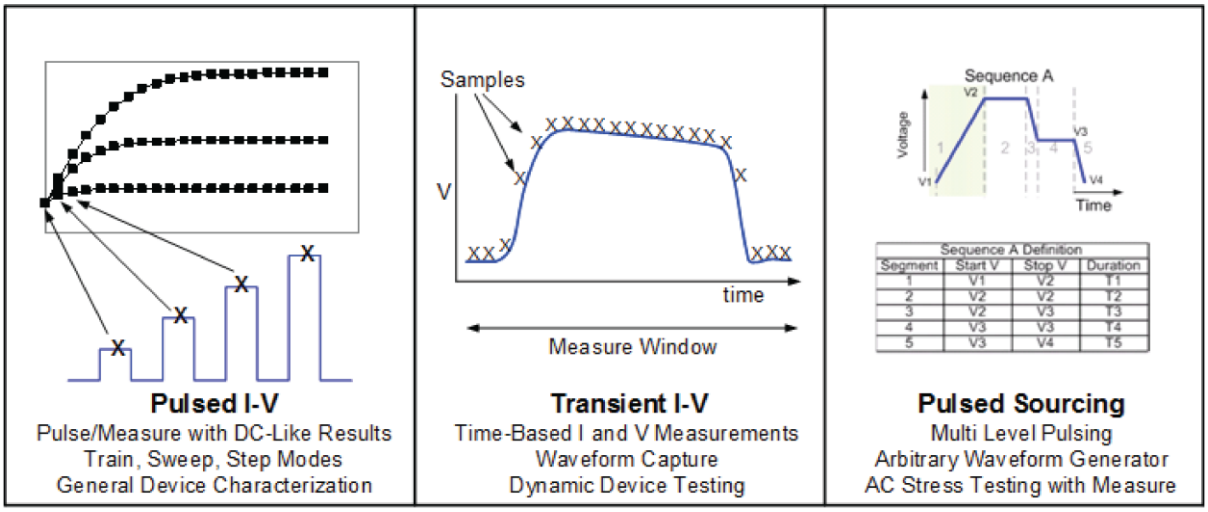
Ultra-fast I-V tests

You can use the 4225-PMU to do these types of ultra-fast I-V tests:

1. Pulsed I-V
2. Transient I-V
3. Pulsed sourcing

The modes are illustrated in the next figure.

Figure 195: 4200A Ultra-fast I-V tests



"Pulsed I-V" refers to any test with a pulsed source and a corresponding high speed, timed-based measurement that provides DC-like results. The current, plus voltage measurement, is an average of readings taken in a predefined measurement window on the pulse. This average of readings is called the "spot mean." You can define the parameters of the pulse, including the pulse width, duty cycle, rise and fall times, amplitude, and so on.

"Transient I-V," or waveform capture, is a time-based current, including voltage measurement, that is typically the capture of a pulsed waveform. A transient test is typically a single pulse waveform that is used to study time-varying parameters, such as the drain current degradation versus time due to charge trapping or self-heating. Transient I-V measurements can be made to test a dynamic test circuit or can be used as a diagnostic tool for choosing the appropriate pulse settings in the pulsed I-V mode.

"Pulsed sourcing" can involve outputting user-defined two-level pulses, outputting multi-level pulses using the built-in Segment Arb® function, or outputting an arbitrarily defined waveform using the arbitrary waveform generator in the KPulse software, which is included with the 4200A-SCS. The Segment Arb feature allows you to create waveforms from segments defined with separate voltages and time durations. In addition to its pulse generator capabilities, the 4225-PMU can measure AC or DC voltage and current, thereby reducing the need for additional measurement hardware and more complicated programming. The 4220-PGU is a two-channel pulse generator with output capabilities identical to the capabilities of the 4225-PMU, but without its measurement functions.

Segment Arb waveform

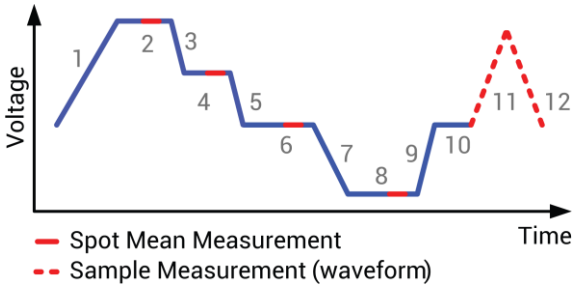
Each channel of a pulse card can be configured to output its own unique Segment Arb® waveform. A Segment Arb waveform is composed of user-defined line segments, up to 2048 for the 4220-PGU and 4225-PMU. Each segment can have a unique time interval, start value, stop value, output trigger level (TTL high or low) and output relay state (open or closed).

NOTE

If both channels of a pulse card are being used, the segment trigger levels for Channel 1 will be seen at the TRIGGER OUT connector. The trigger levels for Channel 2 are ignored.

A Segment Arb waveform consists of line segments illustrated in the figure below. The blue line represents the voltage waveform; the red represents the measure windows. This figure contains a 12-segment waveform.

Figure 196: Example of waveform measured using seg_arb_sequence



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.

Note that Segment Arb provides the capability to include measurement (for the 4225-PMU) and sequences (4225-PMU and 4220-PGU).

Pulse Mode: Sequence, Segment Arb

LPT Functions: seg_arb_sequence (on page 14-140) and seg_arb_waveform (on page 14-144)

These more advanced functions can be used by the 4220-PGUs and 4225-PMUs to define a Segment Arb waveform.

NOTE

If both channels of a pulse card are being used, the segment trigger levels for CHANNEL 1 will be seen at the TRIGGER OUT connector. The trigger levels for CHANNEL 2 are ignored.

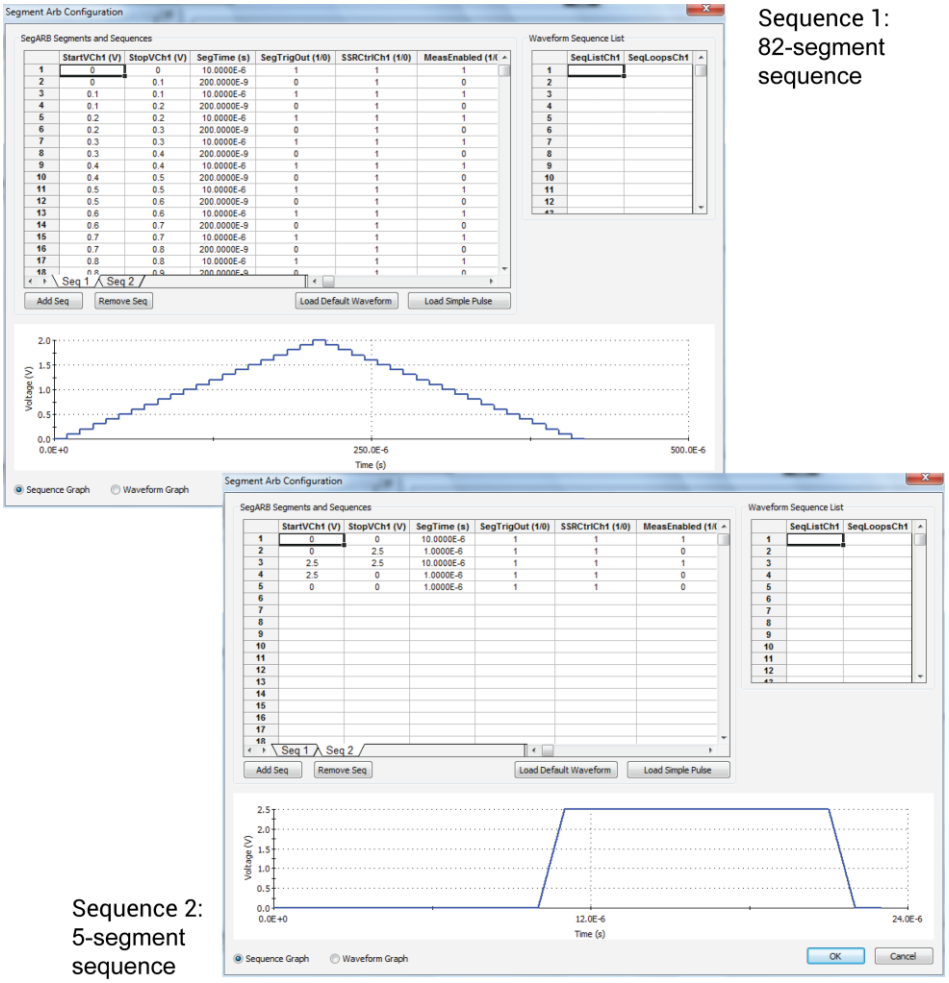
High-endurance output relay (HEOR): Each output channel of a pulse card has a high-speed, solid-state output relay. When this relay is closed, the waveform segment is output. When opened, the channel output is electrically isolated (floating) from the DUT. In the figure in Full arb waveform (on page 5-78), the output relay is opened during segment seven. This puts the output in a floating condition. The minimum time for a segment with a HEOR transition (open-to-close or close-to-open) is 25 µs for the 4220-PGU and 4225-PMU.

NOTE

Because of resources necessary to generate the Segment Arb waveforms, an additional 10 ns interval is added to the end of the last segment of a Segment Arb waveform. During this interval, the output voltage, HEOR, and trigger output values remain the same as the final value reached in the last segment.

In the following figure, sequence 1 is an 82-segment sequence; additional rows are not shown in the figure. Sequence 2 is a 5-segment sequence. There are additional columns to the right for per-segment measurement configuration.

Figure 197: Two sequence definitions

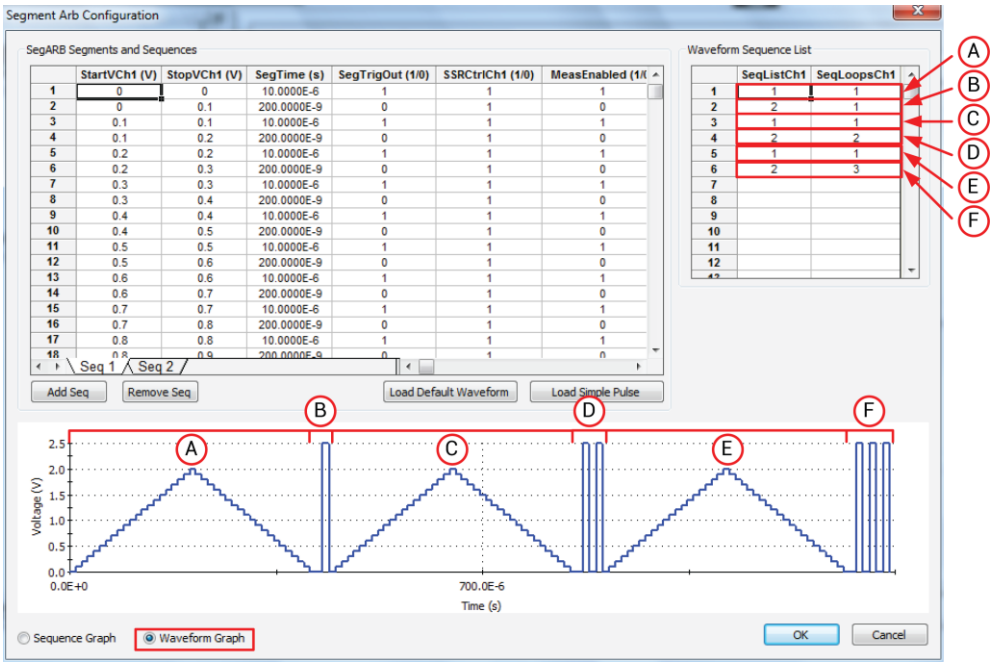


Sequence 1: 82-segment sequence

Sequence 2: 5-segment sequence

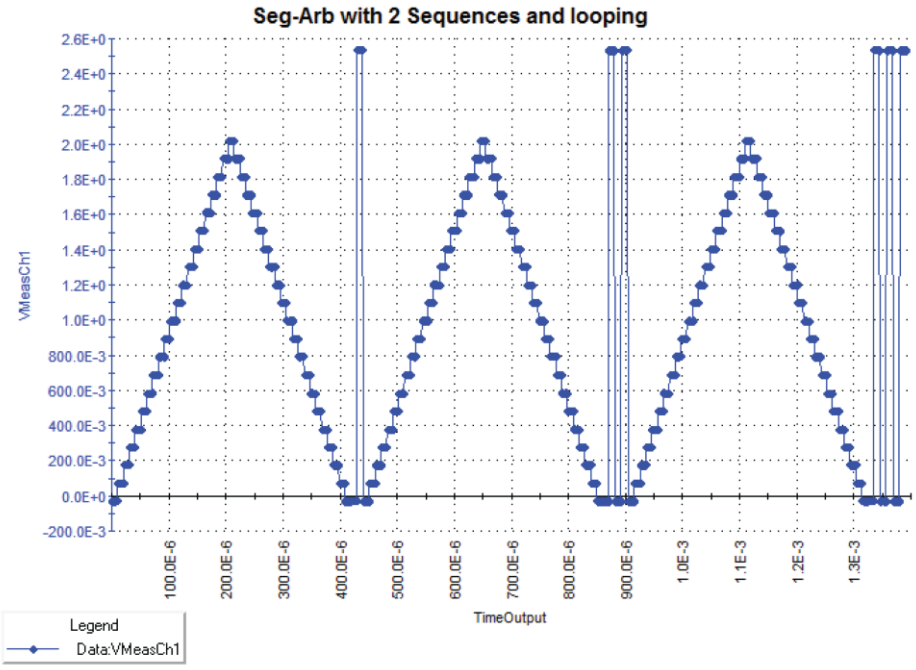
A definition showing two sequences with looping is illustrated in the following figure.

Figure 198: Definition showing two sequences with looping



The graph of the measurement of a two-sequence Segment Arb with looping, from UTM `pmu-segarb-complete` in the `pmu-dut-examples` project, is shown in the following figure.

Figure 199: Graph of the waveform (two-sequence with looping) measurement



Pulse modes: Source, Segment Arb

LPT function: seg_arb_define (on page 14-183)

This function is used to define a Segment Arb waveform. This function includes parameters to specify the number of segments (`nSegments`) and arrays for start (`startvals`), stop (`stopvals`), and time values (`timevals`). It also includes arrays for trigger levels (`triggervals`) and output relay states (`outputRelayVals`).

Full arb waveform

You can configure each channel of the pulse generator to generate its own unique full arb waveform. A full arb waveform is made up of user-defined points (up to 262,144).

Each waveform point can have its own unique voltage value. A time interval is set to control the time spent at each point in the waveform. The following figure shows an example of a user-defined full arb waveform. The waveform is made up of 80 voltage points, with the time interval between each point set to 10 ns.

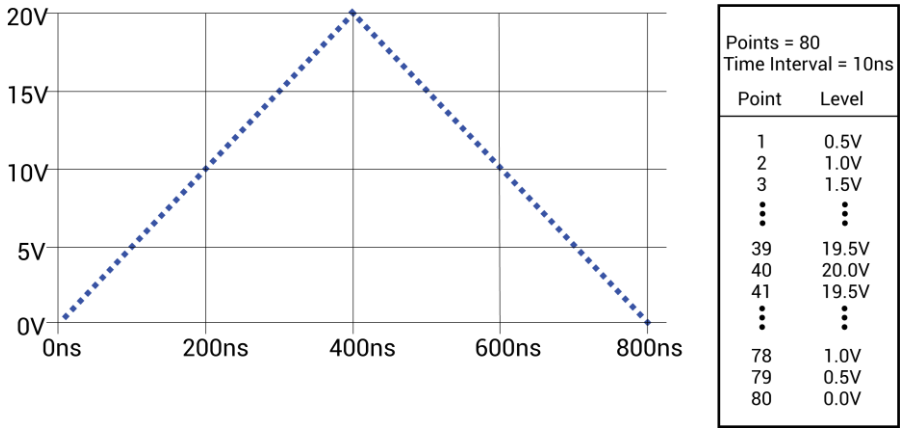
Use the `arb_array` function to define a full arb waveform. This function includes parameters to specify the number of waveform points, the time interval, an array of voltage levels, and a file name. For more information, refer to arb_array (on page 14-148).

The `arb_file` function is used to load the defined full arb waveform into the pulse generator. For more information, refer to arb_file (on page 14-149).

The following voltage level array is required for the full-arb waveform shown in this graph.

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

Figure 200: Full arb waveform example



KPulse full arb waveforms

The Keithley Pulse tool (KPulse) is a virtual front panel software application used to control the optional pulse generator cards. KPulse can be used to create, save and output full arb waveforms, and provides a collection of basic full arb waveform types such as sine, square, triangle, noise, Gaussian, and calculation. After configuring one of the basic waveform types, you can save it as a .kaf file.

Once a full arb waveform is saved as a .kaf file, it can later be imported back into KPulse. The waveform can also be loaded into the pulse generator card using the `arb_file` function. For more information, refer to KPulse full arb waveforms (on page 5-79).

Pulse waveforms for nonvolatile memory testing

The pulse card has several attributes that support nonvolatile memory (NVM) testing.

To perform the multi-level pulse waveforms for the typical program/erase waveform, each pulse card channel uses the Segment Arb mode. For more information about Segment Arb waveforms, refer to Segment Arb waveforms (on page 11-7). The ability to disconnect, or float, a particular device pin in the Segment Arb waveform requires an inline solid state relay.

The pulse card output channels each have 50 Ω output impedance for most ranges. When current flows through the pulse channel, there is a voltage drop across this 50 Ω resistor internal to the pulse card. This dictates that the voltage at the output may be different from what is expected based on the resistance of the DUT. This effect is called the load-line effect. For more information on the load-line effect and how to compensate for it, refer to Load-line effect compensation (LLEC) for the PMU (on page 5-44).

The DUT resistance determines pulse voltage across DUT:

- The gate of a flash or NVM device has high impedance.
- The voltage at the gate is double of the programmed voltage.
- The voltage at the drain is a function of the resistance of the drain-source, as mentioned above.

Adjusting the pulse level to match the expected drain voltage is performed iteratively with an oscilloscope to measure V_D during the pulse.

The methods used to define the multi-level waveforms used in flash memory testing include using the subsite stress/measure looping feature of Clarius. Use the KPulse application to define each unique voltage waveform. Refer to KPulse (on page 11-1) for details.

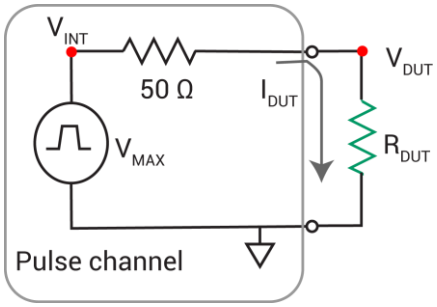
DUT resistance determines pulse voltage across DUT

Each output channel of the 4220-PGU pulse card is effectively a voltage source with a series 50 Ω resistance. Since the 4220-PGU pulse card 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 201: Schematic of pulse channel and connected DUT



As shown in the figure, the voltage across the DUT, V_{DUT}, is directly related to the resistance R_{DUT}. The discussion here is simplified, including resistive impedances only.

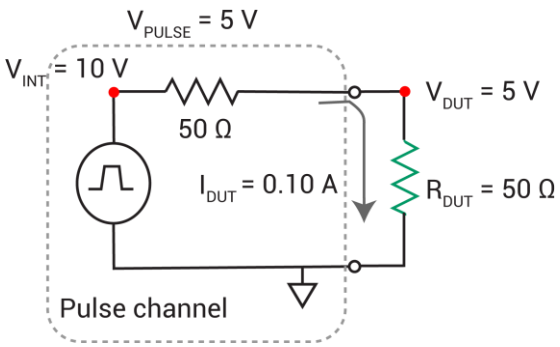
The 50 Ω output resistance of the pulse channel and the DUT resistance effectively make a voltage divider. Ohm's law is the only concept required to determine I_{DUT} and V_{DUT} in this simple non-dynamic approach. Calculate the current, and then use the current to calculate the voltage across the DUT.

Example 1: Ideal situation (50 Ω DUT)

The ideal situation (50 Ω DUT) is shown in the figure below.

R_{DUT}	50 Ω
Pulse V_{high}	5 V
Pulse V_{low}	0 V
Pulse load	50 Ω

Figure 202: 5 V pulse into a 50 Ω DUT load



Example 1, showing a 5 V pulse into a 50 Ω DUT load.

V = I * R (Ohm's Law)

Calculate the current, I_{DUT}:

$I_{DUT} = V_{TOTAL}/R_{TOTAL} = V_{INT}/ (50\ \Omega + 50\ \Omega) = 10\ V / 100\ \Omega = 0.1\ A$

Note that the internal voltage, V_{INT}, is 2 times the requested 5 V, so that V_{DUT} = 5 V. The 2 times 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}:

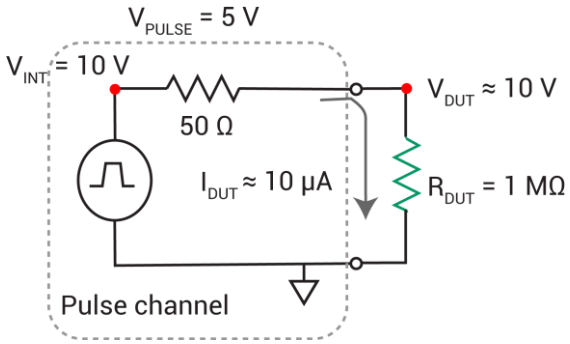
$V_{DUT} = I_{DUT} * R_{DUT} = 0.1\ A * 50\ \Omega = 5\ V.$

Example 2: High resistance DUT

A high resistance DUT is shown in the following figure.

R_{DUT}	1 MΩ
Pulse V_{high}	5 V
Pulse V_{low}	0 V
Pulse load	50 Ω

Figure 203: 5 V pulse into a 1 MΩ DUT load



This example shows a 5 V pulse into a 1 MΩ DUT load.

Calculate the current, I_{DUT}:

$I_{DUT} = V_{TOTAL}/R_{TOTAL} = V_{INT}/ (50\ \Omega + 1\ M\Omega) = 10\ V / 1.00005E+6\ \Omega = 9.9995E-6\ A \approx 10\ \mu A$.

Calculate V_{DUT}:

$V_{DUT} = I_{DUT} * R_{DUT} = 10\ \mu A * 1E6\ \Omega \approx 10\ V$.

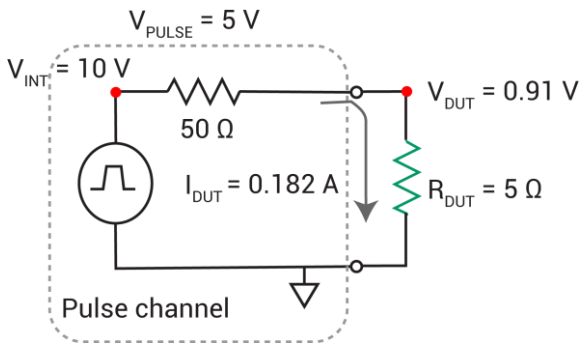
$V_{DUT} \approx 10$ V because the pulse load = 50 Ω, but R_{DUT} = 1 MΩ. The pulse generator calculates the output voltage based on the pulse load, and uses a 2 times multiplier to determine V_{INT} = 10 V. Since the pulse generator cannot measure the output voltage or DUT load, this example results with a V_{DUT} that 2 times the programmed level.

Example 3: Low resistance DUT

A low resistance DUT is shown in the following figure.

R_{DUT}	5 Ω
Pulse V_{high}	5 V
Pulse V_{low}	0 V
Pulse load	50 Ω

Figure 204: 5 V pulse into a 5 Ω DUT load



This figure shows a 5 V pulse into a 5 Ω DUT load.

Calculate the current, I_{DUT}:

$I_{DUT} = V_{TOTAL}/R_{TOTAL} = V_{INT}/ (50\ \Omega + \ 5\ \Omega) = 10\ V / 55\ \Omega = 0.182\ A$

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 1 V 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

$R_{DUT} = 5\ \Omega$

$I_{MAX} = V_{MAX} / \ \ / R_{TOTAL} = V_{MAX} / (R_{pulse\ generator} + R_{DUT})$

$I_{MAXDUT} = 10\ V / (50\ \Omega + 5\ \Omega) = 10\ V / 55\ \Omega = 0.182\ A$

$V_{MAXDUT} = I_{MAX} * R_{DUT} = 0.182\ A * 5\ \Omega = 0.91\ V$

These I_{MAX} and V_{MAX} values match the values in the Pulse Card High Voltage (5 V) Range table for the 5 Ω row (see next table). Note that these calculations do not incorporate any cabling or interconnect losses that may range from 1 Ω up to 10 to 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 Ω

$I_{MAX} = V_{MAX} /$ / $R_{TOTAL} = V_{MAX} / (R_{pulse\ generator} + R_{DUT})$

I_{MAXDUT} = 40 V / (50 Ω + 250 Ω) = 40 V / 300 Ω = 0.133 A

$V_{MAXDUT} = I_{MAX} * R_{DUT}$ = 0.133 A * 250 Ω = 33.3 V

These I_{MAX} and V_{MAX} values match the values in the Pulse Card High Voltage (20 V) Range table for the 250 Ω row. These calculations do not incorporate any cabling or interconnect losses that may range from 1 Ω up to 10 to 20 Ω, depending on the interconnect method and DUT type (packaged or on-wafer).

The tables 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}.

Because V_{DUT} is a function of the load applied to the pulse card channel, this effect is sometimes called the load-line effect. There are several ways of working with this effect. The simplest one is to program the DUT load into the pulse card channel using the pulse_load (on page 14-164) command, or setting the Pulse Load value in the KPulse (on page 11-1) virtual front panel. 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 correct levels. For details, see Coping with the load-line effect (on page 5-47).

Test device 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
* Approximate value; does not account for interconnect losses.		

Available I and V for the high voltage (20 V) range of the Keithley pulse card **Maximum I and V versus resistance**		
Test device 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
*Approximate value; does not account for interconnect losses.		

Triggering

Triggering is used for the following operations:

- Basic triggering (on page 5-86): Configure the Keithley pulse card for the trigger mode (Continuous, Burst or Trig Burst) and use a software trigger to start pulse output.
- Pulse-measure synchronization (on page 5-87): Synchronize the pulse-measure operation of a pulse generator card.

For each pulse in a test, a trigger pulse is sent to the trigger output. If you are using Segment Arb waveforms, you can define if a trigger out is sent for each segment.

The 4220-PMU and 4225-PGU are trigger out only (no trigger in). These pulse cards use the internal trigger bus for output synchronization when using any commands that require the `pulse_exec()` command to execute.

If you need to define a trigger in using a source-only pulse waveform (2-level or Segment Arb), you can create a user test module (UTM) that uses the command set described in Pulse source only (PG2) commands (on page 14-8). This allows an external trigger (LPT command `pulse_trig_source`) to output a 2-level pulse (LPT commands `pulse_vhigh` and `pulse_vlow`) or seg-arb (LPT command `seg_arb_define`). This allows for external triggering (a trigger input into the card) that causes a burst of pulses to be output.

NOTE

Triggering transition time must be <100 ns. Trigger output Level and Trigger In Level must be TTL.

NOTE

Details on the trigger functions discussed in the following paragraphs are provided in the LPT Library Function Reference (on page 14-1).

Basic triggering

The 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

To use a software trigger to start pulse output, the software trigger source must first be selected. The pulse_trig_source (on page 14-176) function is used to select the software trigger source.

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: 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 (on page 5-88)).

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 (on page 14-156) 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

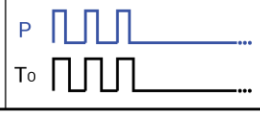
Synchronize the pulse-measure operation of a pulse generator card.

Pulse generator card output trigger

When output trigger is 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 figure below 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 trigger burst mode, output triggers stop when the pulse output stops.

Figure 205: Pulse generator card output trigger

Trigger Mode	Standard Pulse	Full Arb Pulse	Segment Arb Pulse*
Continuous			
Burst			
Trig Burst			

P = Pulse output
T_0 = Trigger output

*Segment Arb has user defined trigger output (0 or 1) for each segment.

The pulse_trig_output (on page 14-174) 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 (on page 14-175) function to set the polarity of output trigger.

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 source-measure connections

NOTE

TRIGGER OUT of a pulse generator card can 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 (on page 5-86).

The cables used for SMU connections are supplied with the SMUs and preamplifiers.

For information on fundamental connection schemes for test configurations using pulse generator cards, refer to pulse generator connections (on page 5-90).

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 4200A-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 42 V being applied to the device under test or fixture.

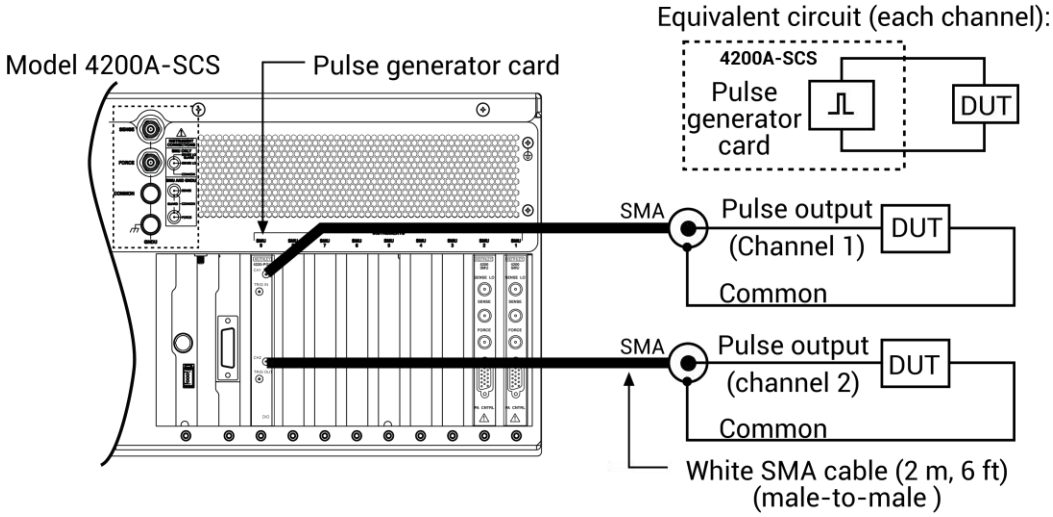
Pulse generator connections

The following figure shows a system that uses basic 2-channel pulse generator connections to DUTs.

NOTE

Use the supplied torque wrench to tighten SMA connections to 8 in. lb.

Figure 206: Basic pulse generator connections



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 (on page 5-91) and the pulse_meas_wfm (on page 14-123) function is used to configure Waveform measurements (on page 5-94).

The Model 4225-PMU makes the following types of pulse measurements:

- Spot mean discrete
- Spot mean average
- Waveform discrete
- 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 (on page 5-91) 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 (on page 5-94) 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 (on page 5-96) for details on pulse measurement timing.

Spot mean measurements

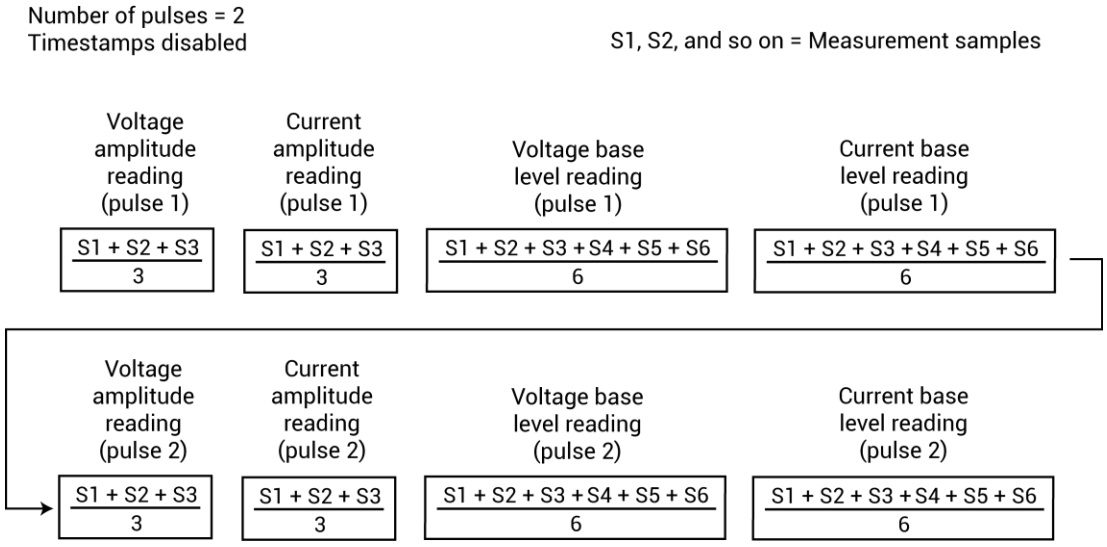
Spot mean measurements sample a portion of the amplitude and a portion of the base level (see Spot mean measurement timing (on page 5-96)). The portions to be sampled are specified as a percentage. Note that you can return an individual spot mean for each pulse (see Spot mean discrete readings (on page 5-92)), or have a single spot mean for all *NumPulses* (see Spot mean average readings (on page 5-93)).

If you enable both amplitude and base spot means, each pulse has two voltage measurements and two current measurements (see pulse_fetch (on page 14-110)).

Spot mean discrete readings

The averaged voltage and current readings for every sampled pulse period are returned in a single data set. The figure below 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.

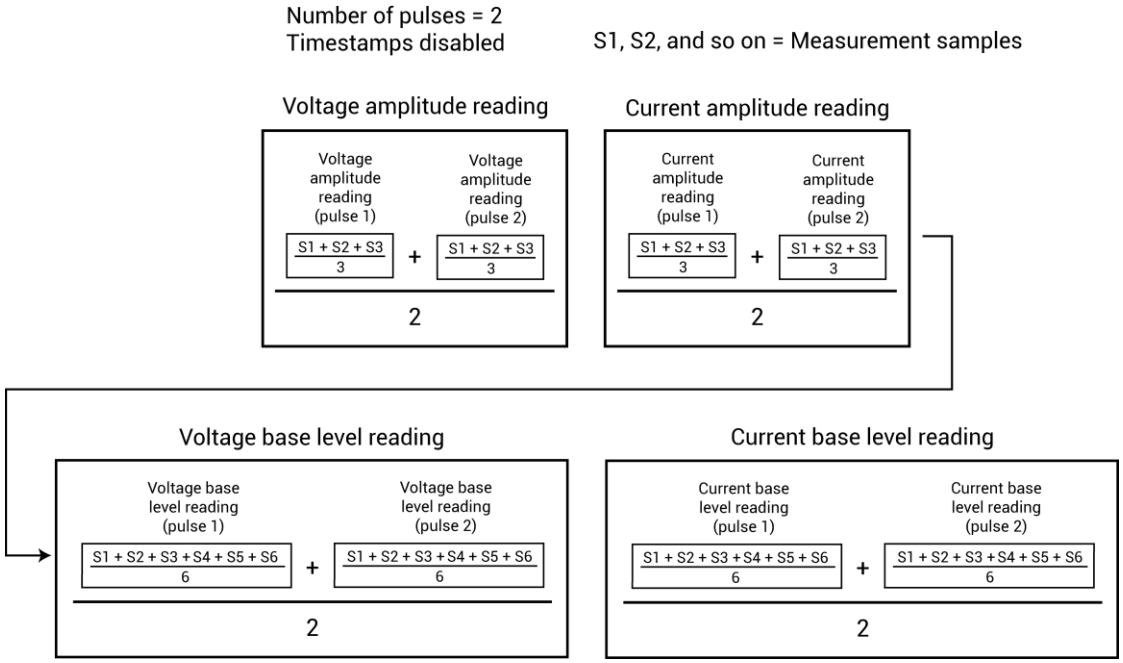
Figure 207: Returned data set for spot mean discrete readings



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 the figure in Spot mean discrete readings (on page 5-92), 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 below. With all voltage and current readings enabled, four readings are returned for the burst. When time stamps are enabled, a timestamp is included in the data set after each mean-of-the-means reading.

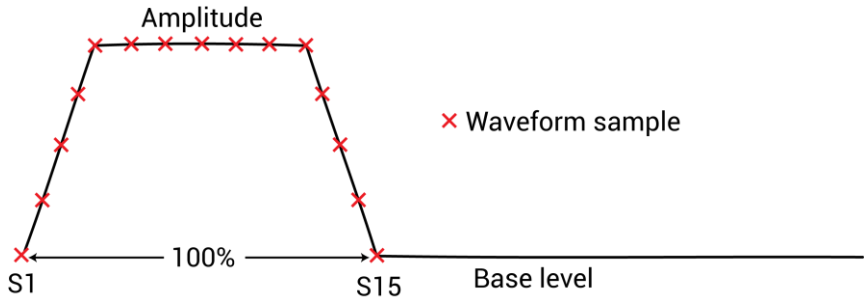
Figure 208: Returned data set for spot mean average readings



Waveform measurements

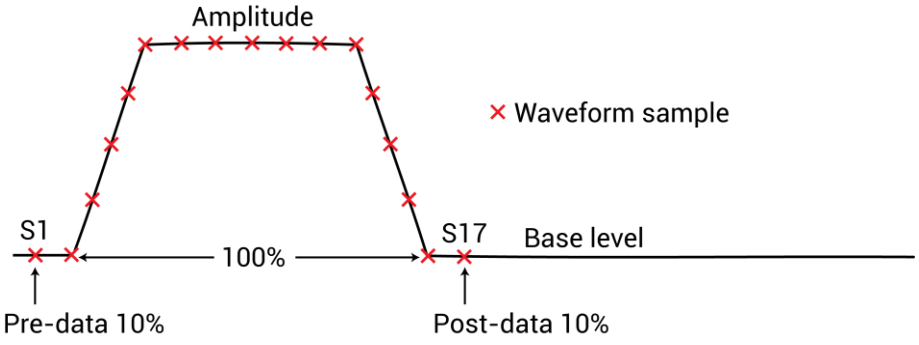
Waveform measurement readings sample the entire pulse. Sampling is performed on the rise time, top width, and fall time portions of the pulse. In the figure below, 15 samples are performed on the pulse waveform. Enabled voltage and current readings and time stamps are returned for every sample taken on the pulse. The `NumPulses` (number of pulses) parameter is used to specify the number of pulses to be output and sampled.

Figure 209: 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 below). See Waveform measurement timing (on page 5-97) for details on measurement timing for pre-data and post-data.

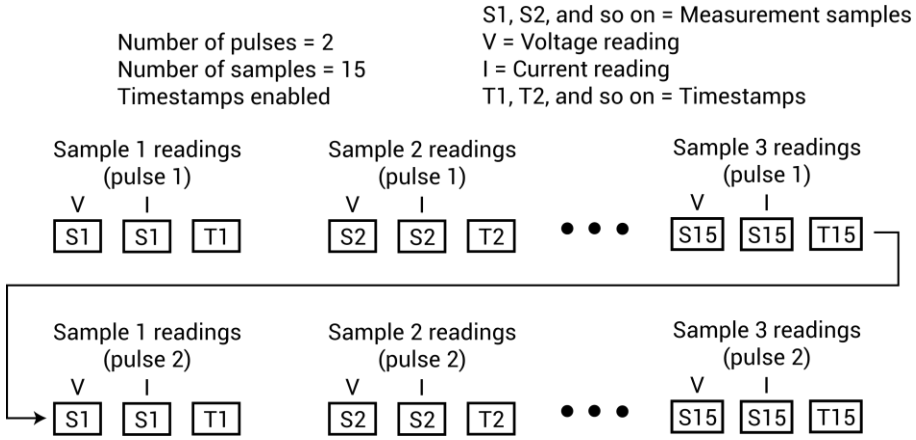
Figure 210: Waveform measurements with pre-data and post-data



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. The following figure shows how waveform discrete readings are returned as a data set for two pulse periods with voltage, current, and timestamp readings enabled.

Figure 211: Returned data set for waveform discrete readings

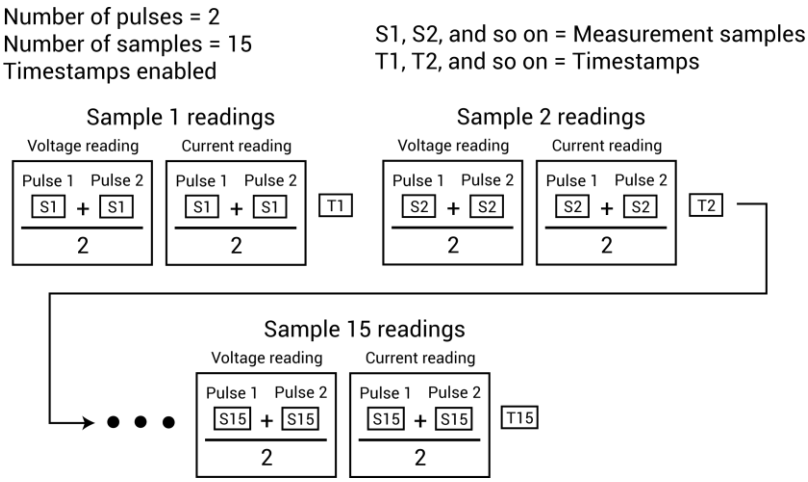


Waveform average readings

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 the figure in Waveform discrete readings (on page 5-95), the V and I readings for sample 1 - pulse 1 would be averaged with the V and I readings for sample 1 - pulse 2.

The figure below shows how waveform average readings are returned as a data set for two pulse periods with voltage, current, and time stamp readings enabled.

Figure 212: Returned data set for waveform average readings



Measurement timing

The pulse_meas_timing (on page 14-119) function is used to configure pulse measurement timing.

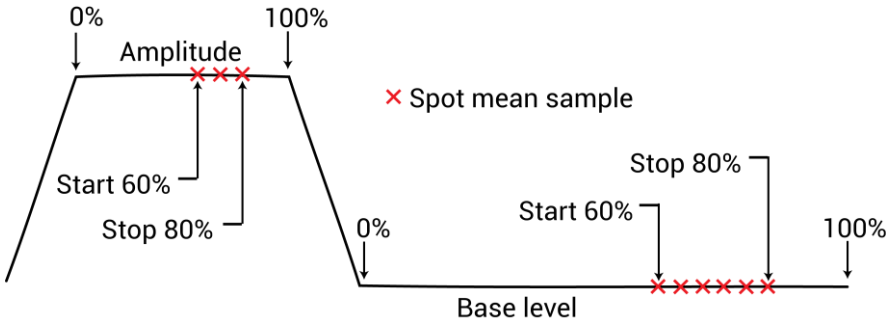
Spot mean measurement timing

The spot mean measurement type samples a portion of the amplitude and a portion of the base level. The 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 specifies the number of pulses to be output and sampled.

The figure below shows that three measured samples are taken on the amplitude and six samples are taken on the base level. The start and stop percentage values indicate the portions of the pulse that are sampled. 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% points. The start and stop points for amplitude and base-level sampling are expressed as a percentage between 0% and 100%. The default is 75% for the start of spot mean and 90% for end of spot mean. These percentages are applied to the total time of the pulse top. In this example, sampling for the amplitude and base levels starts at 60% (0.6) and ends at 80% (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 (on page 14-129) function to set the sampling rate for pulse measurements.

Figure 213: Spot mean measurements example



Waveform measurement timing

The waveform measurement type samples the pulse as shown in Waveform measurements (on page 5-94). Sampling is performed on the entire duration (100%) 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. Waveform measurements (on page 5-94) shows an example where 10% (0.1) pre-data and 10% (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 (on page 14-129) 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% as 0.5 in the function.

Example: 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)

Example: 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)

Example: 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%)
- *StopPercent* = 0.8 (80%)
- *NumPulses* = 5 (output one pulse)

Section 6

Clarius

In this section:

Get started with Clarius

Clarius is the primary application of Clarius+ and is the primary user interface for the 4200A-SCS. Clarius is a versatile tool that helps you characterize individual parametric test devices or automate testing of an entire semiconductor wafer. It allows you to create, execute, and evaluate tests and complex test sequences without programming.

Clarius helps you set up characterization of an individual parametric test device or automated testing of an entire semiconductor wafer.

Key features:

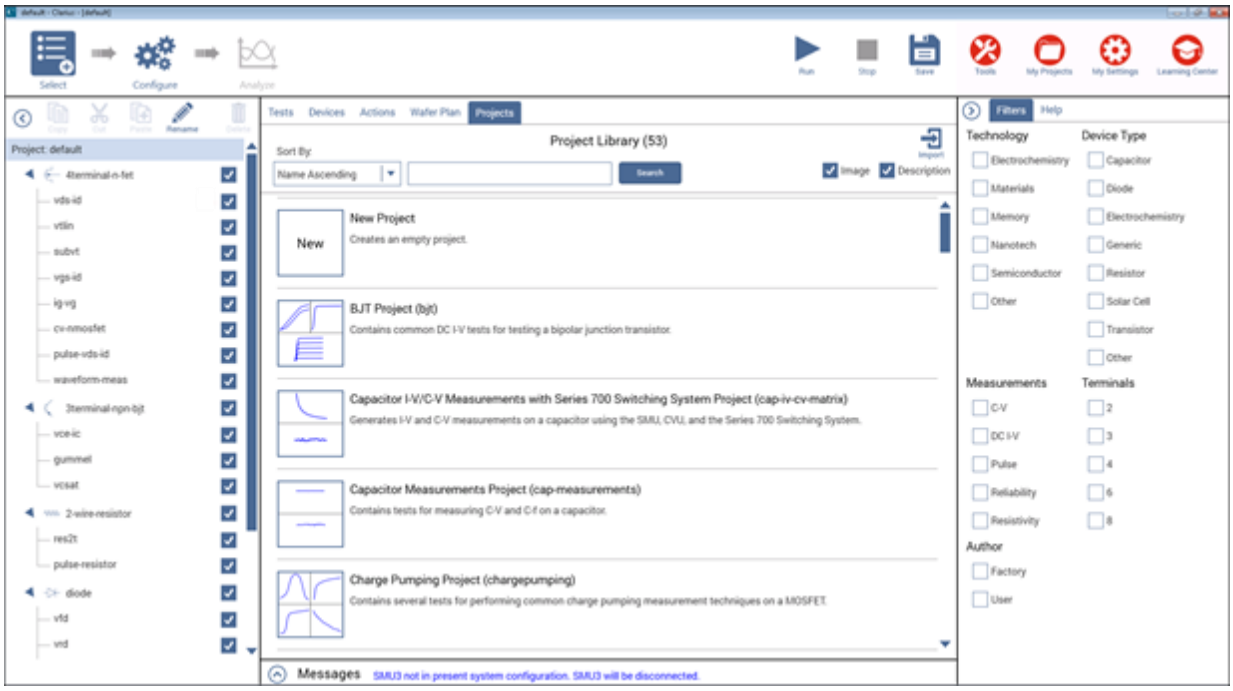
- Ready-to-use, modifiable application tests, projects, and devices that reduce test development time
- Built-in measurement videos from world-wide Application Engineers
- Multiple measurement functions
- Data display, analysis and arithmetic functions

Clarius interface

The Clarius interface allows you to:

- Build and edit project and execution sequences.
- Configure tests.
- Execute tests and actions, such as switch matrix connections and prober movements, including:
  - A single test for one device (such as a transistor, diode, resistor, capacitor).
  - A test sequence for one device.
  - Test sequences for multiple devices. For example, test all the devices contacted by a prober at a location on a semiconductor wafer.
  - The test sequences of an entire project, which may include multiple prober touchdowns for a single semiconductor die
- View test and analysis results.
- Analyze test results using built-in parameter extraction tools.

Figure 214: Clarius interface



Touchscreen basics

You can operate the 4200A-SCS using the touchscreen. You can use your fingers, clean room gloves, or any stylus manufactured for capacitive touchscreens.

To select and move on the screen:

- To scroll, swipe up or down on the screen.
- To select an item, touch it on the screen.
- To double-click an item, touch it twice.
- To right-click an item, touch and hold, then release to see the options.

To enter information, you can use the on-screen keyboard. Swipe from the left side of the display to open the keyboard.

The touchscreen uses standard Microsoft® Windows® touch actions. For additional information on the actions, refer to the Microsoft help information, available from the on-screen keyboard window menu option **Tool > Help Topics**.

You can also adjust the touch settings using the Pen and Touch options in the Windows Control Panel.

Choose the project phase

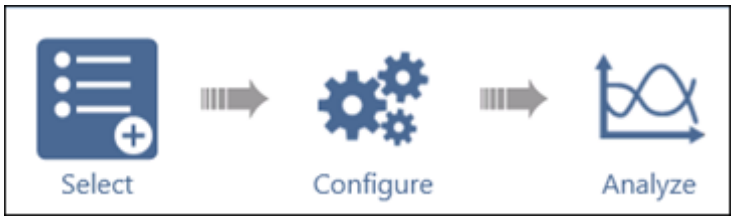
The options on the left side of the top pane of Clarius determine which phase of the project you are working on and allow you to select options to support your tests.

Select displays the libraries, which you can use to add existing projects, tests, devices, actions, and wafer plans to your project. You can also create your own tests, actions, and projects.

Configure displays the parameters for the item you selected in the project tree. For example, if you selected a test, the parameters for each terminal of the test and the entire test are available.

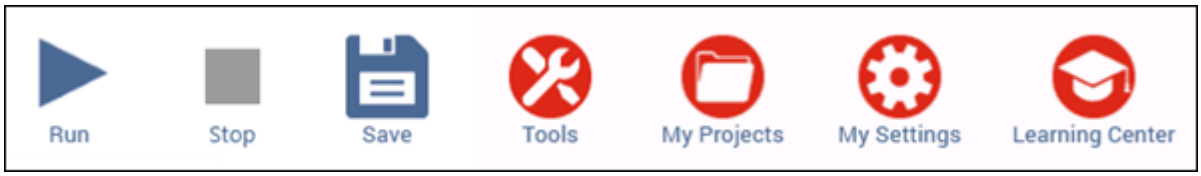
Analyze displays the results of the tes. You can also access analysis tools to explore and export your data.

Figure 215: Select, Configure, and Analyze



Run tests and set up your workspace

Figure 216: Clarius run test and workspace options



The options on the right side of the top pane of Clarius include options that allow you to run tests, configure instruments, manage projects, set up your workspace, and learn about the 4200A-SCS.

Run runs the highlighted item. You can run an individual test by highlighting only that test. You can run all the tests for a device, subsite, site, or project by highlighting the device, subsite, site, or project. Only items that are checked and below the selected item in the hierarchy are run.

Stop stops all running items.

Save saves the project configuration.

Tools provides module-specific tools. For source-measure units (SMUs), you can run autocalibration. For capacitance-voltage units (CVUs), you can set up connection compensation, do real-time measurements, and perform a confidence check. For pulse measure units (PMUs) and pulse generator units (PGUs), you can set up connection compensation.

Use **My Projects** to create, import, export, and manage your projects. Projects are automatically stored in My Projects when you create them in the project tree.

Use **My Settings** to customize Clarius to better meet your needs. You can change environment settings, run settings, graph defaults, and GPIB abort settings.

Use the **Learning Center** to access complete 4200A-SCS documentation, including online help, videos, instructions, application notes, white papers, and other materials to help you use your 4200A-SCS.

Organize items in the project tree

The project tree on the left side of the Clarius window displays the items in your project, including devices, tests, actions, and sites. The project tree for the default project is shown in the figure below.

The settings for the item you select in the project tree are displayed when you select Configure from the top bar. The test data for the item is displayed when you select Analyze.

When you run a test, the item that is highlighted runs. If the item is a project, site, subsite, or device, all checked items in the hierarchy below the highlighted item run.

Figure 217: Project tree for the default project

Select items from the libraries

When you choose Select, the center pane displays libraries of tests, devices, actions, wafer plans, or projects that you can add to the project tree. These libraries are templates that you can copy from to create your own tests, devices, actions, wafer plans, and projects. When you copy an item from the library to the project tree, the item in the projct tree is a copy. The item in the library is not affected by any changes you make to the copy.

The **Test Library** contains predefined tests. The predefined tests contain detailed definitions that tell Clarius how to characterize a device, including associated data analyses and parameter extractions. Clarius comes with a library of tests for commonly used devices, including transistors, diodes, resistors, and capacitors. You can also create your own tests.

The **Device Library** contains the devices that need to be characterized, such as transistors, diodes, resistors, or capacitors. Each test must be in the project tree under a device. The devices available in the library include the standard set of devices that come installed on the 4200A-SCS and any custom-name devices that you have submitted; refer to Submitting devices to a library (on page 6-372).

The **Action Library** contains items that support the tests and help control the project. Actions can generate dialog boxes to prompt test operator action, control prober movements, and manage switching. You can also create your own actions from user libraries.

The **Wafer Plan Library** contains sites and subsites. A site is used if you are testing a repeating pattern of dies and test structures on a wafer. Every wafer location that a prober can move to and contact at any one time is a subsite. There are typically multiple subsites for each site. Subsites typically correspond to a single test structure or other combination of devices that are tested as a group.

The **Project Library** contains predefined projects. Projects include the devices, tests, actions, sites, and subsites organized for testing a single device, group of devices, or wafer. You can also create your own empty project.

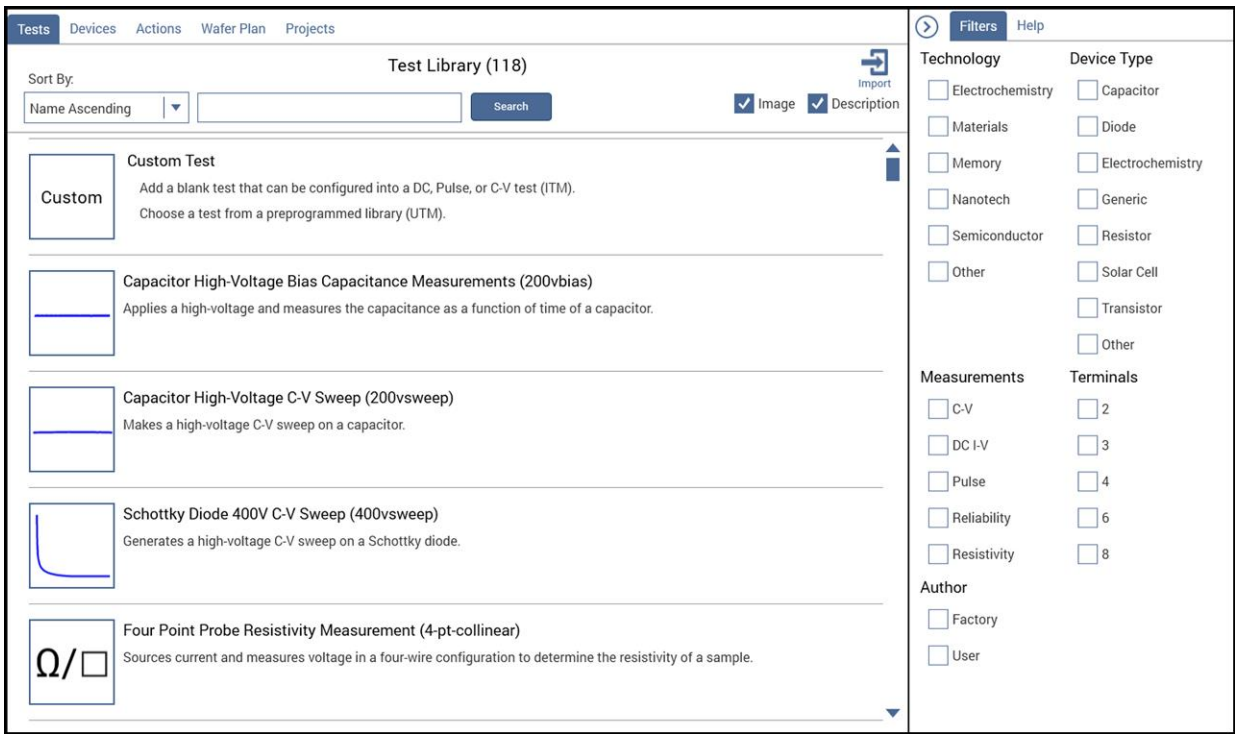
For most of the libraries, the right pane displays filters that you can use to reduce the list of library items to the items you need. You can also use the Search option at the top of the library to type a search term to reduce the number of items.

You can sort the libraries by name or title.

The **Image** and **Description** options at the top of the library allow you to turn the images and descriptions that are shown in the library on or off. Turning them off displays more items.

Import allows you to import items into the 4200A-SCS.

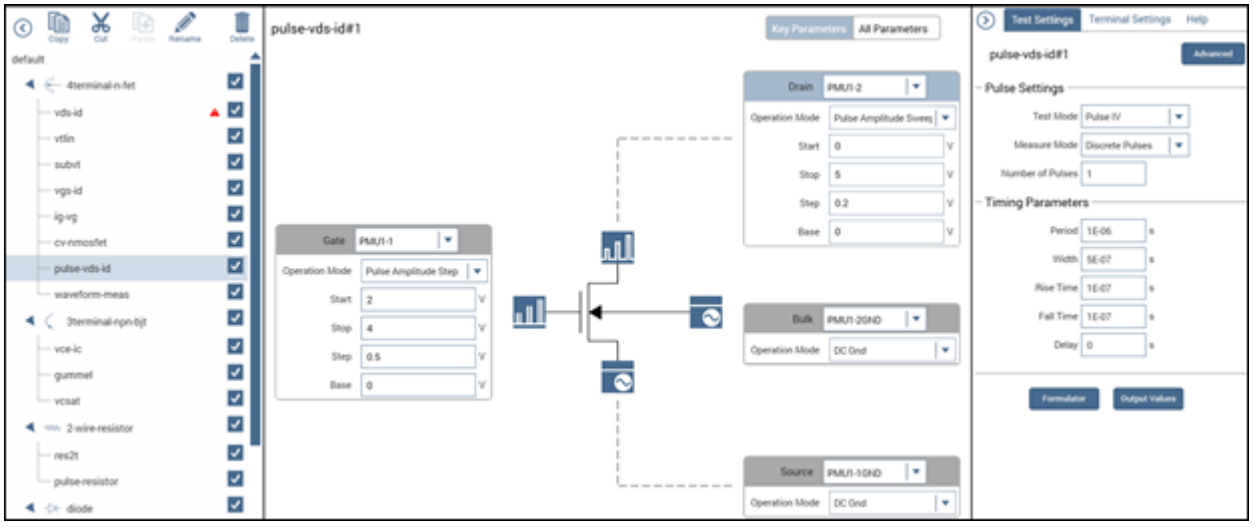
Figure 218: Test library



Configure the project

Select **Configure** for an item in the project tree to display the settings for that item. Depending on the item, settings are available in the center and right panes. Help for the selected item is also available in the Help pane.

Figure 219: Configure pane for the pulse-vds-id test



Analyze data

When you run tests, you can display and analyze test results and test definitions in the Analyze pane. The Analyze pane displays data in a spreadsheet and as a graph.

The **View** options change the view from both spreadsheet and graph to only the spreadsheet or only the graph. You can use **Save Data** to save the graph to a `png` or `bmp` image file or save the data into an `xls` spreadsheet file.

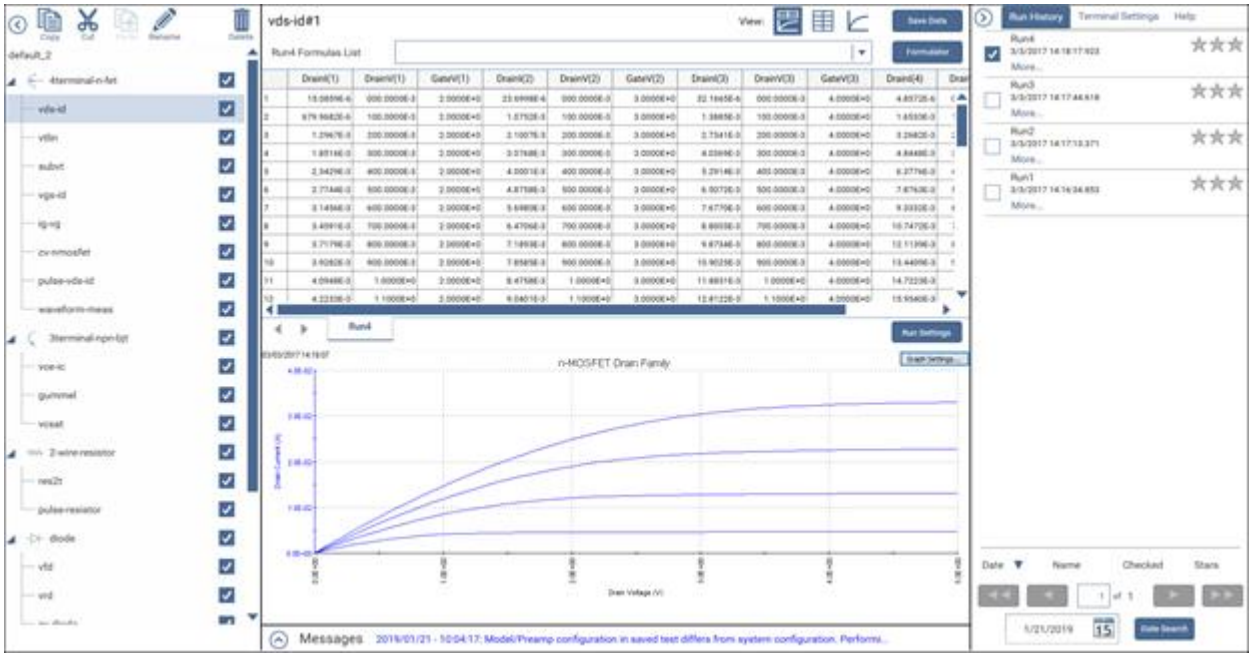
If the Formulator was used to calculate data for this test, the **Run Formulas List** displays the calculations. You can select **Formulator** to open the Formulator and edit the formulas or create new ones.

The **Graph Settings** allow you to change the display of the data on the graph.

The **Run History** pane on the right displays the time and name of each test run. When you select a run from Run History, the sheet and graph in the center pane change to show the data from that test run.

The **Terminal Settings** pane, also on the right, displays the settings for the presently selected test.

Figure 220: Analyze pane for the pulse-vds-id test



Messages

Messages regarding the test and execution are displayed at the bottom of the Clarius window. To expand the Messages window to view more detail, select the up arrow to the left of the Messages heading.

You can right-click a message to copy it or to select and copy all messages to the clipboard.

You can also right-click and select **Clear All** to remove the existing messages.

Help pane

The Help pane displays information that is related to the library item or project tree item that is selected.

If you have the Select pane open, the help describes the item that is selected in the Library.

If you have the Configure or Analyze pane open, the help describes the item that is selected in the project tree.

Additional Clarius+ applications

Two of the Clarius+ applications support Clarius:

- The Keithley User Library Tool (KULT) (on page 8-1) allows you to create libraries of test modules using the C programming language. These test modules are executed by Clarius.
- The Keithley Configuration Utility (KCon) (on page 7-1) manages the configuration and interconnections between all of the test system components that are controlled by Clarius.

Another Clarius+ software tool, the Keithley External Control Interface (KXCI) (on page 10-1), allows the 4200A-SCS to be controlled remotely by an external GPIB controller.

NOTE

You cannot run KXCI and Clarius simultaneously.

The Keithley Pulse tool (KPulse) controls the optional pulse cards. A pulse card is a dual-channel pulse card that is integrated inside the 4200A-SCS mainframe.

NOTE

Although KPulse can be launched at the same time as Clarius, KPulse and Clarius cannot communicate with hardware simultaneously.

Set up a simple project

To start testing, you can start with a new project, or use an existing project. A project consists of items such as devices and tests.

The order of operations of a test is determined by the order and selection of items in the project tree.

The following topics describe how to set up a run a simple project, using an existing project from the Project Library.

Select project components

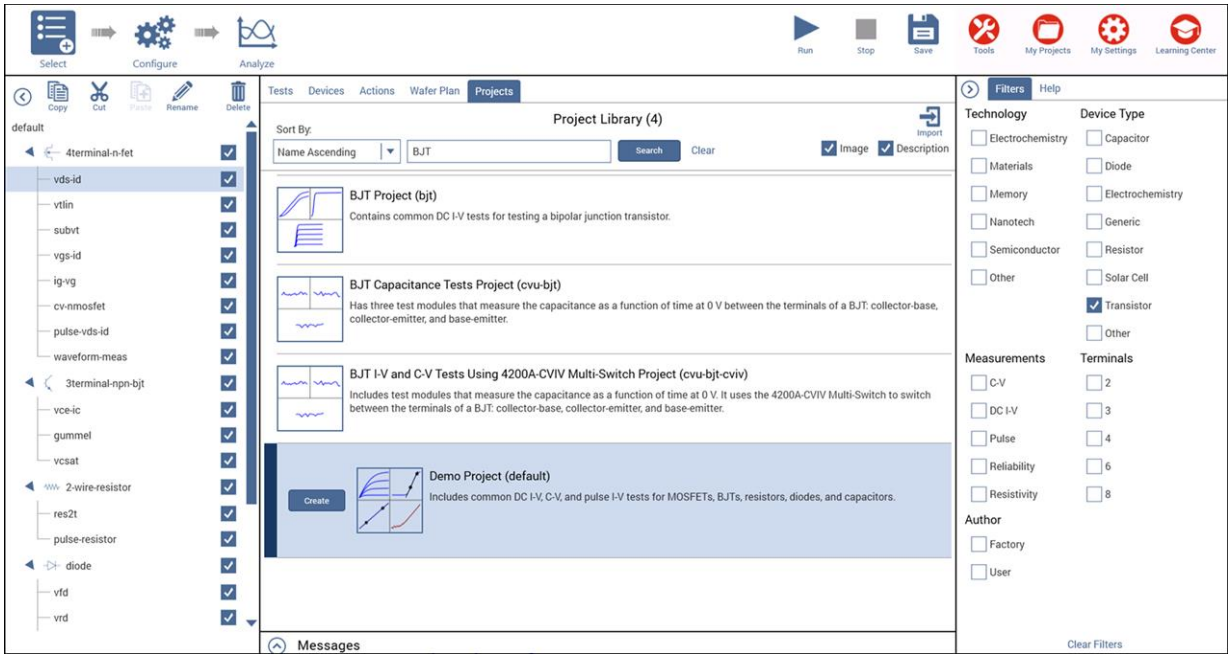
Use the Select pane to add items to the project tree. When Select is active, the center pane contains libraries for tests, devices, actions, wafer plans, and projects. You can use filters and search options to help you find the items you need for your test.

To clear filters, select **Clear Filters** at the bottom of the Filters pane. To clear the search, select **Clear** next to the Search button.

For example, if you want to test bipolar junction transistors (BJTs):

1. Select **Save** to save your existing project.
2. Choose **Select**.
3. Select **Projects**.
4. In the Filters pane, select **Transistor**.
5. In the Search box, type **BJT** and select **Search**. The Library displays projects that are intended for BJT transistor testing.
6. Select **Create** for the project you want to open. The project replaces the previous project in the project tree.

Figure 221: Filter and search for the bjt project



Add a device and test to the project

You can add additional items to a project. Once a project is added from the library to the project tree, it is copied from the project in the library, so you can make changes without affecting the original project. The project in the project tree is is stored in My Projects.

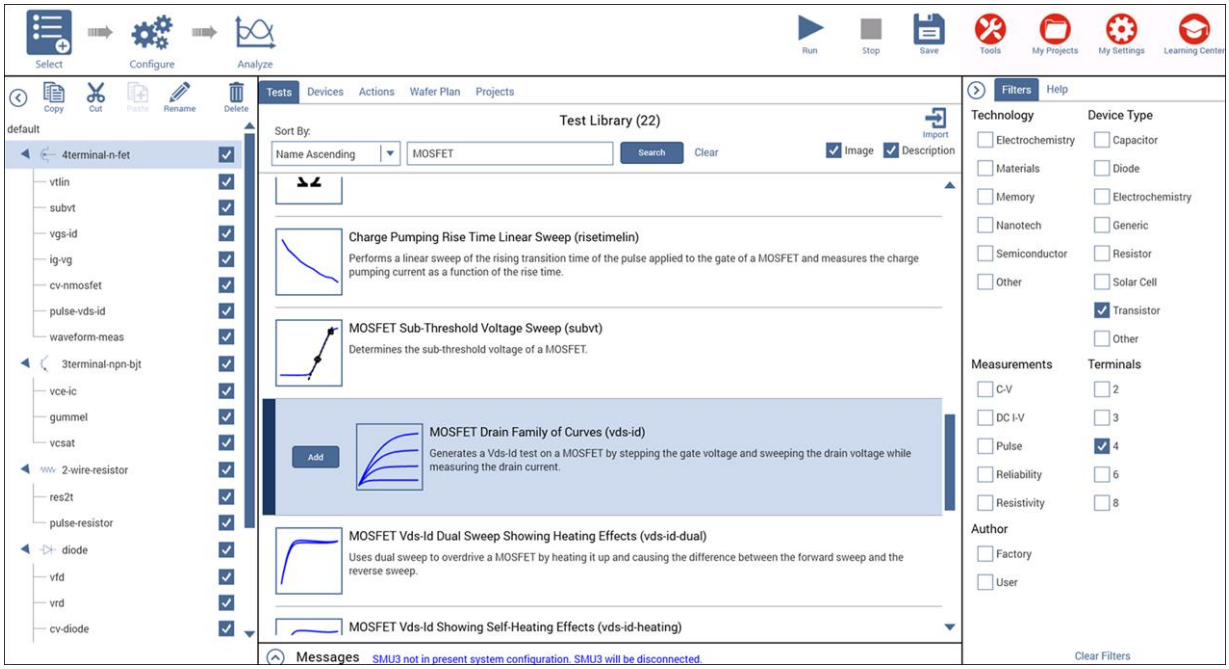
This example shows you how to add a predefined test to the project. Predefined tests are configured with commonly used parameter settings and a set of typical data. Once they are in a project, you can change the parameters as needed. They can be an efficient way for you to add a test to your project.

You can use the basic procedure described here to find any items in the library.

To add a four-terminal MOSFET device and test to the project:

1. Change the library to **Tests**.
2. In the Filters pane, select **Transistor** and **4** Terminals.
3. In the search box, type **MOSFET** and select **Search**.
4. Scroll to the **MOSFET Drain Family of Curves (vds-id)** test.
5. Select **Add**. The selected test and the device is added to the project tree under the previous item that was highlighted.
6. To move the device and test, drag the device to a new location.
7. Select **Save**.

Figure 222: Add a MOSFET test and device to the project



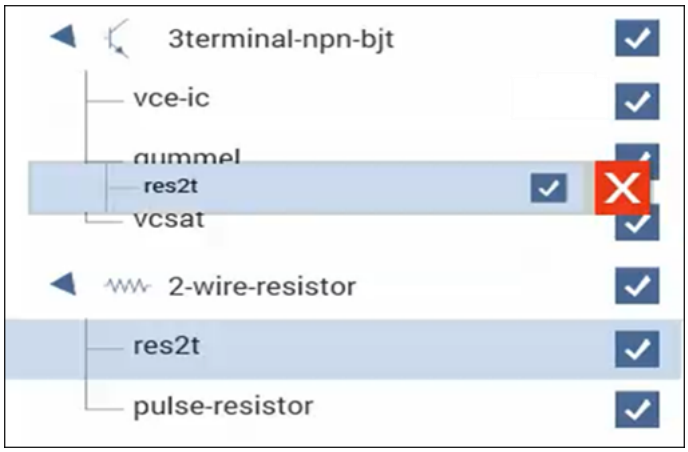
NOTE

If the device for a test is not in the project tree, Clarius adds the appropriate device when you add a test to the project tree. You can also add the device and test separately.

Rearrange items in the project tree

To rearrange items in the project tree, drag the items to the new location. If the item cannot be placed in the selected location, a red X is displayed. In the example below, a resistor test cannot be placed under a BJT device.

Figure 223: Object not allowed at this location in the project tree



For actions, if they are at the bottom of the project tree, you can promote or demote them to move them in the tree structure. For example, if the action is under a device, you might want to move it to be at the project level. To promote or demote an action, right-click the action and select **Promote Action** or **Demote Action**.

Delete objects in the project tree

CAUTION

If you delete an object, other items may also be deleted. For example, if you delete a subsite, all device and tests in the subsite are also deleted. If you delete a device, all tests in the device are deleted.

To delete an object:

1. In the project tree, select the item you want to delete.
2. Select the object.
3. Select **Delete** at the top of the project tree. A confirmation message is displayed.
4. Click **OK**.

Configure a simple test

Use the Configure pane to set up your test. For interactive test modules (ITMs), the Configure pane displays a schematic of the test device. The schematic is connected to an object that shows the operation mode and the type of instrument that is connected to the terminal.

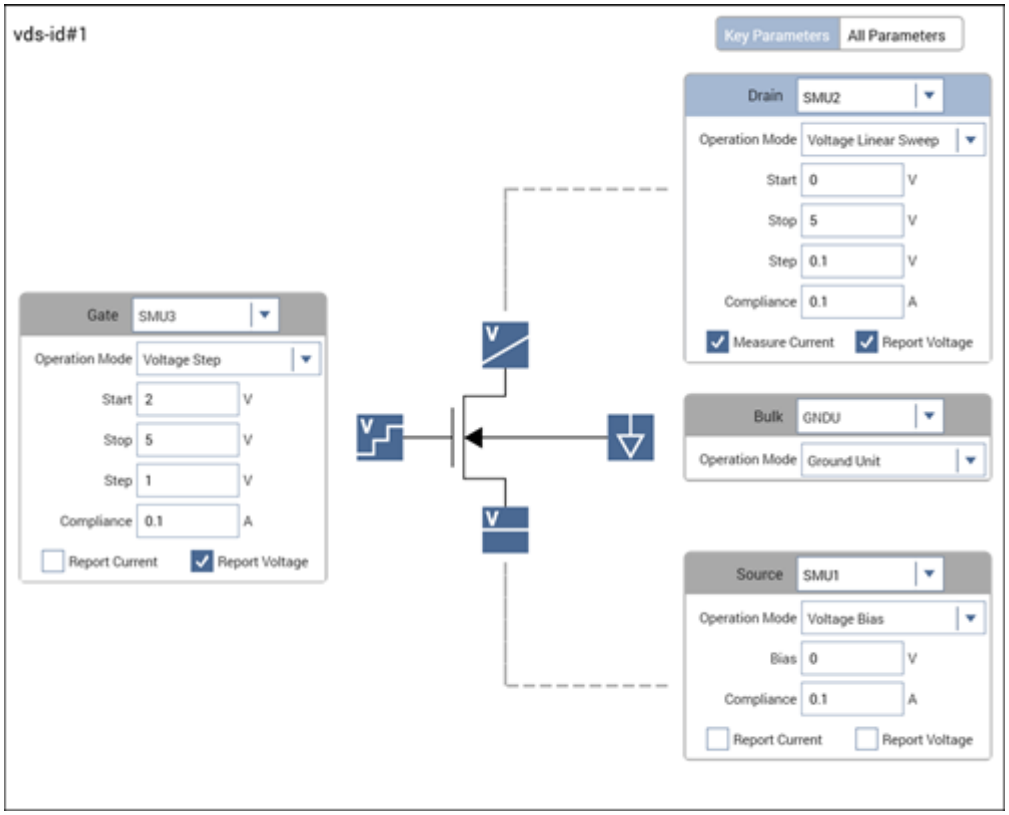
CAUTION

The software connections must accurately reflect the physical hardware connections when the test is executed. Incorrect terminal configurations can result in anomalous test results and device damage.

The key parameters for each terminal are displayed near the terminal. The key parameters include:

- The type of terminal, such as gate, drain, source, or collector.
- The instrument that is attached to the terminal. You assign the instrument, ground unit, or open circuit that is physically connected to the terminal during the test.
- The operation mode and basic settings for that mode. For example, the start and stop values are displayed if a sweep operation mode is selected.

Figure 224: Configure pane



NOTE

For user test modules (UTMs), the display depends on the settings of the user module that the UTM is based on.

Set the key parameters

The Key Parameters are the most commonly used parameters.

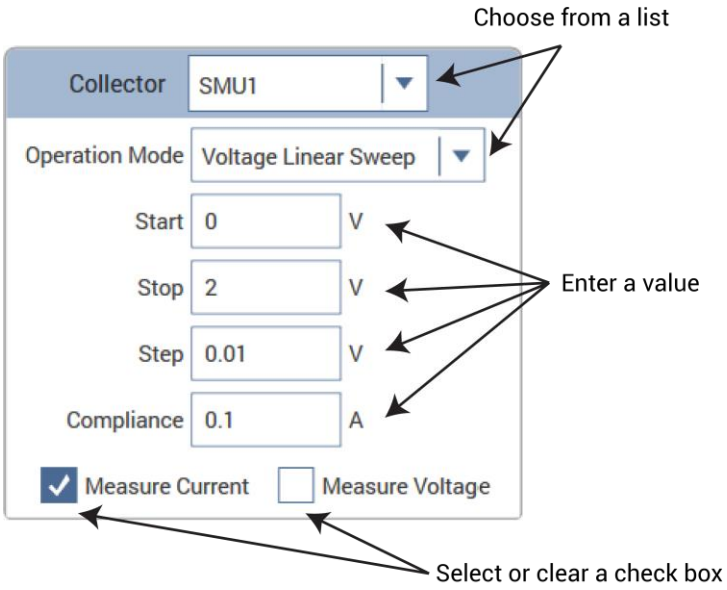
The parameters that are available depend on the instrument that is selected. For descriptions of parameters, refer to:

- SMU - all parameters (on page 6-48)
- CVU - all parameters (on page 6-77)
- PMU - all parameters (on page 6-95)

To set the Key Parameters:

1. Select the field that you want to change.
2. If there is a:
   - Down arrow to the right of the field: Select a value from the list.
   - Field: Type the value. Error messages are displayed if you type an out-of-range value.
   - Check box: Select or clear the check box to enable or disable an option.

Figure 225: Clarius selection options



3. Select **Save**.

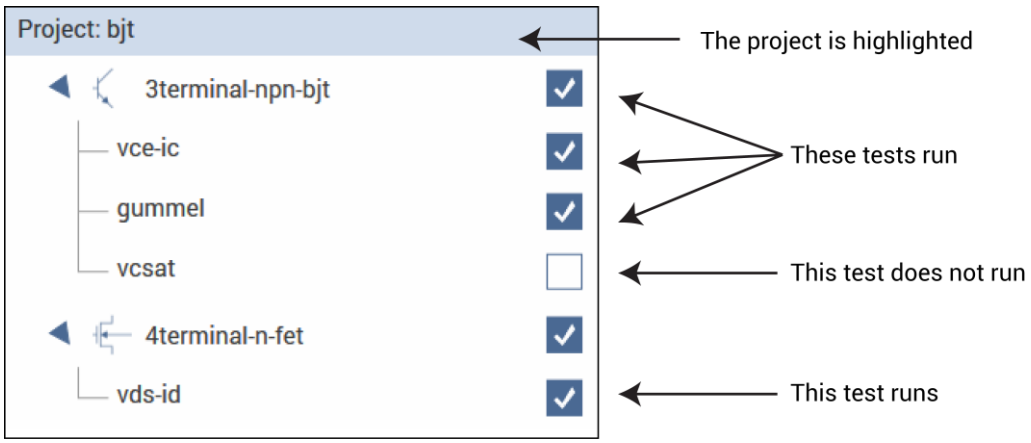
Run a simple test

When you select Run, tests and actions at a lower level than the highlighted item in the project tree are executed if they are selected, from top to bottom in the project tree. If you want to run an entire project, make sure the project name is highlighted. Running a project saves the configuration settings and the existing run history of the project.

In the following example, when you select Run, the following occurs:

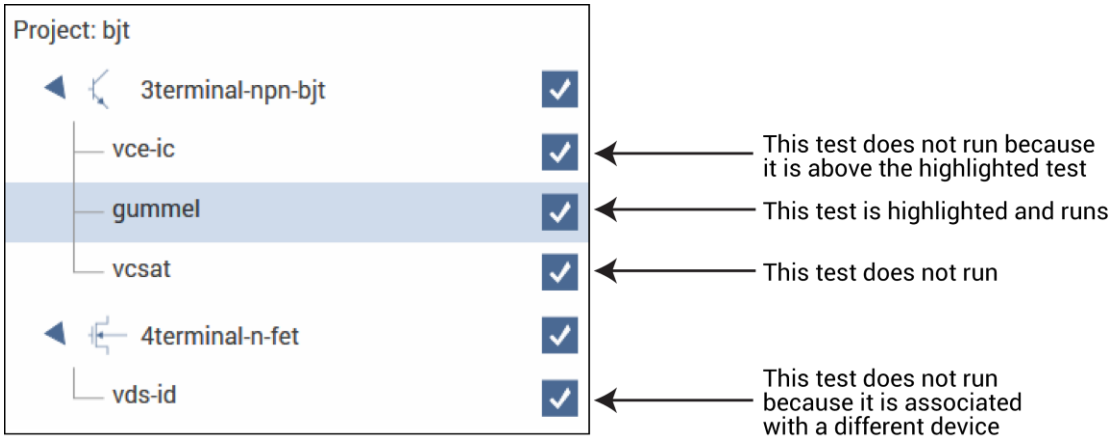
- The `vce-ic` test runs.
- The `gummel` test runs.
- The `vcsat` test is skipped.
- The `vds-id` test runs.

Figure 226: Run a test at the project level



In the following example, only the `gummel` test runs. Even though the other tests are selected, they are not below the `gummel` test in the hierarchy.

Figure 227: Run specific tests



To run a test in Clarius:

1. In the project tree, select that tests and actions that you want to run or execute.
2. Highlight the item where you want the test to start. For example, if you want to run the entire project, select the project.
3. Select **Run**.
4. Select **Analyze** to view the results.

NOTE

To abort a test, select **Stop**. All test and action execution stops immediately.

Working with My Projects

My Projects allows you to work with the projects you have created. Any projects that users on the 4200A-SCS (regardless of user account) have added to the project directory defined in My Settings are available through My Projects.

You can use My Projects to create new projects, import and export projects, and to duplicate, delete, edit, search for, and open projects.

NOTE

To change the project directory, see My Projects Directory (on page 6-377).

Open a project

Your projects are automatically saved in My Projects when they are added to the project tree. This procedure describes how to retrieve a project.

To open a project:

1. Choose **My Projects**.
2. Type the project name in the Search box.
3. Select **Search**.
4. Select the project.
5. Select **Open Project**.
6. Select the project. If the project in the project tree has unsaved changes, you are prompted to save the changes.

NOTE

You can clear the Image and Description options to hide the project images and descriptions and display more projects in the list.

Edit project information

You can edit information that is displayed in the Projects Library for a specific project. You can also edit the filters and keywords that are used.

To edit project information:

1. In Clarius, select **My Projects**.
2. Select the project to be edited.
3. Select **Edit**. The Project Information Editor opens.
4. In the Basic tab, complete the information as needed. Refer to the table below for the options.
5. Select the **Filters** tab. These options set the filters that will cause this item to appear in the library when you select the right-pane filters.
6. Select the filters that help a user find this item in the library.
7. Select the **Keywords** tab. These options determine what you can type in the library Search field to locate this item. You can use the Sort By options at the bottom of the lists to change the order of the entries in the Information Editor. It does not affect the order in the library.
8. Drag a keyword from the left to the right to add a keyword.
9. To remove a keyword, select the keyword and select **Delete**. This does not remove the keyword from the Global Keywords list.
10. To add a keyword, select **New** and type the keyword.
11. Select **OK** to save the changes.

Options in the Information Editor	
Preview	Displays the changes you make as they will appear in the library.
Name	Type the new name. This is the name that is used in the library and the project tree.
Title	Type the title. This is used in the library.
Description	Type a brief description of the item. This is displayed in the library.
Author	Type information that identifies who created this item. This is available only through the Library Information Editor.
Help	Only editable when adding an object from the project tree to the library. This is the help file that is displayed in the right pane when Configure or Analyze is selected. It is also displayed when the item is selected in the library.
Include Help	Only displayed when you add an item from the project tree to the library. Determines if the existing help is included with the new item. If you want to include the help that was associated with the original object, select **Include Help**. Clear **Include Help** to keep the help from displaying (the Help pane will be blank). You cannot change the help link; you can only include or hide it.
Library Image	The image that is displayed in the library. Click the image to select a different image. Images should be 400x400 pixels in `png` format. Larger images display, but anything larger than 400x400 is cut off in the library display. To re-use an image from an older project, you may need to save the existing `bmp` image to `png` format. You can use a tool such as Microsoft® Paint to convert the image. To leave the image area blank, select **Clear**.

Create a new project from My Projects

You can create a new project from the My Projects dialog box. This is the same as creating a new empty project from the Project Library "New Project" option.

To create a new project:

1. Open **My Projects**.
2. Select **Create New**. A confirmation message is displayed.
3. Select **Yes**. The new project replaces the existing project and closes the My Projects dialog box.
4. In the project tree, select **Rename** to assign a new name to the project.
5. Press **Enter** to accept the new name.

Export a project

You can export a project. An exported project can be imported into another 4200A-SCS.

The export includes all Run History data for each test in the project.

To export a project:

1. In Clarius, select **My Projects**.
2. Select the project to be exported.
3. Select **Export**. The Export From My Projects dialog box opens.
4. Select the location for the exported file. You can right-click to select options to create a new folder, rename an existing folder, or delete a folder.
5. Select **Export**.

Import a project

You can import a project from another 4200A-SCS.

You can import either an exported project or a project directory.

Exported projects have the extension `kzp`. Refer to Export a project (on page 6-22) for instructions.

If you are importing a project directory, you import the `kpr` file from that directory. The import includes all files from the project directory, assuming that the project directory is valid.

NOTE

Make sure that the files that you are importing are not set to read-only or run-only.

To import a project:

1. In Clarius, select **My Projects**.
2. Select **Import**. The Import To My Projects dialog box opens.
3. Select the exported file.
4. Select **Import**. The Project Information Editor opens.
5. In the Basic tab, complete the information as needed. Refer to the table below for the options.
6. Select the **Filters** tab. These options set the filters that will cause this item to appear in the library when you select the right-pane filters.
7. Select the filters that help a user find this item in the library.
8. Select the **Keywords** tab. These options determine what you can type in the library Search field to locate this item. You can use the Sort By options at the bottom of the lists to change the order of the entries in the Information Editor. It does not affect the order in the library.
9. Drag a keyword from the left to the right to add a keyword.
10. To remove a keyword, select the keyword and select **Delete**. This does not remove the keyword from the Global Keywords list.
11. To add a keyword, select **New** and type the keyword.
12. Select **Add To My Projects** to add the new object to the library.
13. To open the new project, select the project and select **Open Project**.

Options in the Information Editor	
Preview	Displays the changes you make as they will appear in the library.
Name	Type the new name. This is the name that is used in the library and the project tree.
Title	Type the title. This is used in the library.
Description	Type a brief description of the item. This is displayed in the library.
Author	Type information that identifies who created this item. This is available only through the Library Information Editor.
Help	Only editable when adding an object from the project tree to the library. This is the help file that is displayed in the right pane when Configure or Analyze is selected. It is also displayed when the item is selected in the library.
Include Help	Only displayed when you add an item from the project tree to the library. Determines if the existing help is included with the new item. If you want to include the help that was associated with the original object, select **Include Help**. Clear **Include Help** to keep the help from displaying (the Help pane will be blank). You cannot change the help link; you can only include or hide it.
Library Image	The image that is displayed in the library. Click the image to select a different image. Images should be 400x400 pixels in `png` format. Larger images display, but anything larger than 400x400 is cut off in the library display. To re-use an image from an older project, you may need to save the existing `bmp` image to `png` format. You can use a tool such as Microsoft® Paint to convert the image. To leave the image area blank, select **Clear**.

Migrate projects from 4200-SCS systems

You can use information from older 4200-SCS systems in the 4200A-SCS.

When you bring in information from the 4200-SCS, be aware:

- Copy all the files in the project directory for the project. Make sure the files in the project are kept together when you copy the files. By default, projects are stored in the `C:\s4200\kiuser\Projects` directory.
- If you used Segment Arb waveform files from KPulse in your projects, you need to manually copy and paste the waveform files from the 4200 to the 4200A-SCS. Segment Arb waveform files have the extension `.ksf` and are normally stored in the folder `C:\s4200\kiuser\KPulse\SarbFiles`.
- If your project contains user modules or user libraries that were created in KULT, those user modules are not included when you copy the project directory. See Copy user libraries and modules from a 4200-SCS for instructions on how to import the user libraries and user modules.
- Make sure the files to be imported are not set to read-only or run-only.
- Initialization steps and termination steps will be converted to actions.

You cannot migrate from a 4200A-SCS to a 4200-SCS.

To migrate a project from a 4200-SCS system:

1. On the 4200-SCS system, copy the directory for the project you want to transfer.
2. On the 4200A-SCS, paste the project directory in `C:\s4200\kiuser\Projects`.
3. Open Clarius.
4. Select **My Projects**.
5. Verify that the project is available. If you sort by Last Accessed, the imported projects are displayed at the bottom of the project list.

Duplicate a project

You can make a duplicate of a project. The new project does not maintain any links to the old project. For example, test settings in one project are independent of test settings in the duplicate project.

To duplicate a project:

1. Save the project in the project tree.
2. Select **My Projects**.
3. Select the project you want to duplicate.
4. Select **Duplicate**. You are prompted to close the project that is presently in the project tree.
5. Select **Yes**.
6. Select the project name.
7. Select **Rename**.
8. Type the new name and press **Enter**.

Delete a project

CAUTION

Before deleting a project, ensure that you and others will not need it in the future.

When you delete a project, all files associated with the project in the `C:\s4200\kiuser\Projects` directory are also deleted. If the deleted project is open in the project tree, the project tree is cleared.

To delete a project:

1. Select **My Projects**.
2. Select the project.
3. Select **Delete**. A confirmation message is displayed.
4. Select **Yes**.

View My Projects

You can change the view of the My Projects window. To:

- Change the sort order: Select an option from the Sort By list.
- Search for a specific project: Type a keyword from the name or title of the project (descriptions are not searched) and select **Search** to display only the projects that match the keyword.
- Change the display of the projects: Clear the **Image** and **Description** boxes to display only the names and titles of the projects.

Test and terminal setting descriptions

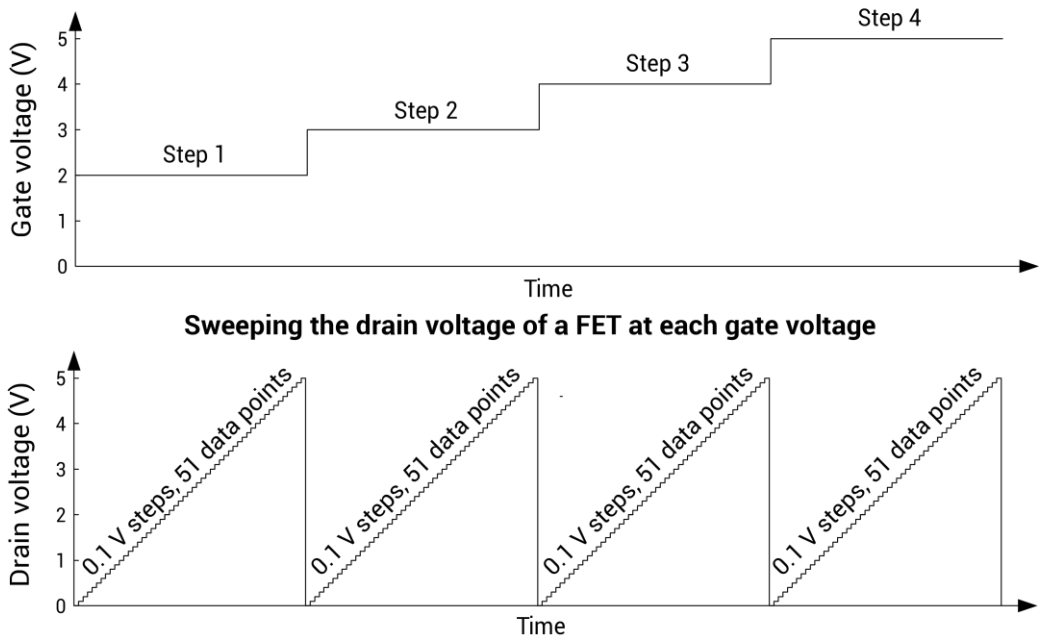
The following topics provide descriptions of the parameters you can set for each instrument and operation mode.

Operation Mode (SMU)

The following topics describe the operation modes that are available when a SMU is selected as the instrument.

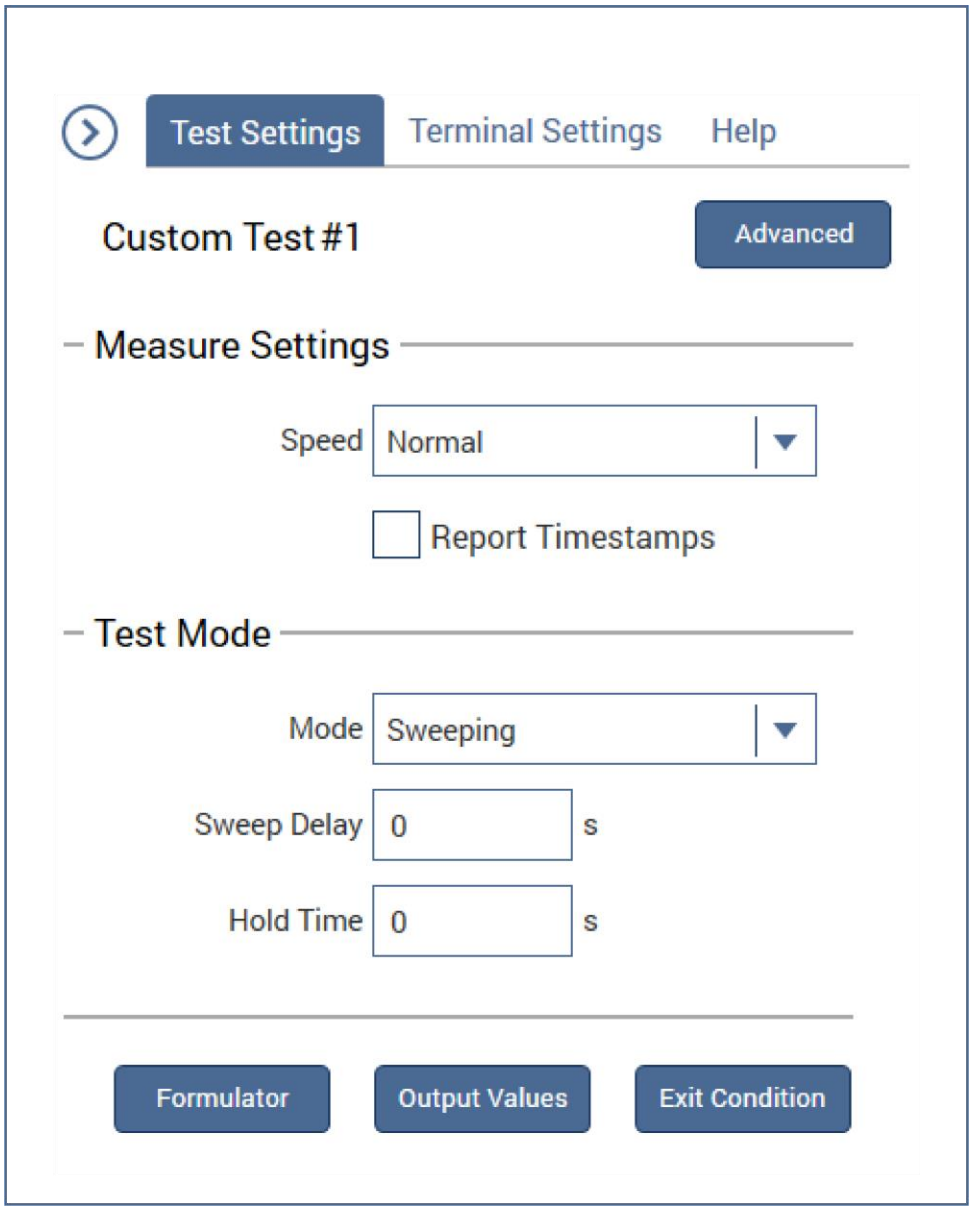
Some of the operation modes allow sweeping or stepping. The figure below illustrates the difference between steps and sweeps.

Figure 228: Stepping and sweeping example



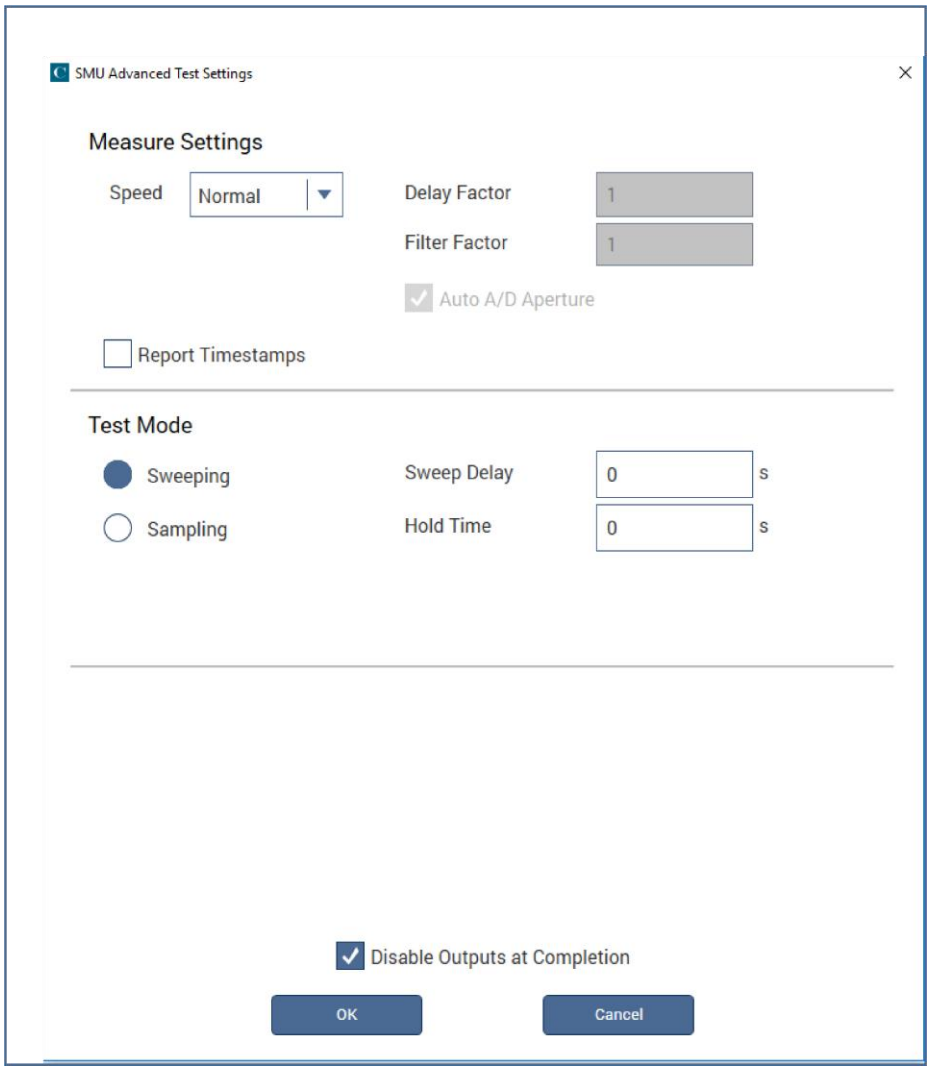
SMU Test Settings Pane

Figure 229: SMU Test Settings Pane



SMU Advanced Test Settings Pane

Figure 230: SMU Advanced Test Settings Pane



Open operation mode - SMU

Open operation mode maintains a zero-current state at the terminal, subject to the maximum voltage compliance of the connected SMU.

You cannot set any parameters for the Open operation mode.

Voltage Bias operation mode - SMU

The Voltage Bias operation mode maintains a selected constant-voltage state at the terminal, subject to a user-specified current compliance of the connected SMU.

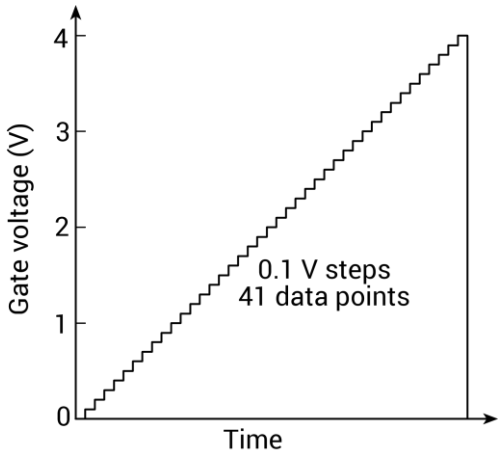
The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Bias (on page 6-48)	The bias is the amount of voltage to be forced.
Force Range (on page 6-52)	The SMU range that is used when forcing the voltage.
Compliance (on page 3-4)	A limit that stops the 4200A-SCS from sourcing a current that is more than that limit.
Power On Delay (on page 6-53)	The delay between when SMUs are powered on in a test sequence.
Pulse Mode (on page 6-54)	Pulse Mode allows you to apply voltage or current to a device for brief periods at widely spaced intervals. This avoids device overheating in some tests. Pulse Mode is only available if the source range and measure ranges are set to a fixed range or the Best Fixed range.
On Time (on page 6-56)	Available when Pulse Mode is selected. The amount of time that the pulse is on (5 ms to 20 s).
Off Time (on page 6-56)	Available when Pulse Mode is selected. The amount of time that the pulse is off (5 ms to 20 s).
Base Voltage (on page 6-57)	Available when Pulse Mode is selected. The voltage level that the instrument sources when the pulse output is off. The level that you can set depends on the present source range.
Overvoltage Protection (on page 6-57)	A limit that restricts the maximum voltage level that the instrument can source. It is in effect when either current or voltage is sourced.
Measure Current (on page 6-57)	When this box is selected, the instrument measures current. Current is recorded in the Analyze sheet and shown in the graph.
Current Column Name (on page 6-58)	The name of the current measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Current Range (on page 6-58)	The measure range determines the full-scale measurement span that is applied to the signal.
Low Range (on page 6-58)	Available when the Measure Range is set to Limited Auto. This sets the minimum range that the instrument uses.
Measure or Report Voltage (on page 6-58)	Determines if voltage values are recorded in the Analyze spreadsheet.
Report Value (Report Voltage or Measure Voltage) (on page 6-59)	Determines if programmed voltage values or actual measured voltage values are recorded in the Analyze spreadsheet.
Voltage Column Name (on page 6-59)	The name of the voltage measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Report Status (on page 6-60)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.

Voltage Linear Sweep operation mode - SMU

When you select the Voltage Linear Sweep operation mode, the test increments through a series of constant voltage steps. You define the start and stop voltages and the voltage size between each step. An example is shown in the next figure.

Figure 231: Example linear sweep



The voltage sweep generates parametric curve data that is recorded in the Analyze pane.

The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Start (on page 6-48)	The voltage source level at which the sweep starts.
Stop (on page 6-48)	The voltage source level at which the sweep stops.
Step (on page 6-49)	The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).
Points (on page 6-50)	The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters.
Dual Sweep (on page 6-51)	When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.
Force Range (Source Range) (on page 6-52)	The SMU range that is used when forcing the voltage.
Compliance (on page 3-4)	A limit that stops the 4200A-SCS from sourcing a current that is more than that limit.
Power On Delay (on page 6-53)	The delay between when SMUs are powered on in a test sequence.
Pulse Mode (on page 6-54)	Pulse Mode allows you to apply voltage or current to a device for brief periods at widely spaced intervals. This avoids device overheating in some tests. Pulse Mode is only available if the source range and measure ranges are set to a fixed range or the Best Fixed range.
On Time (on page 6-56)	Available when Pulse Mode is selected. The amount of time that the pulse is on (5 ms to 20 s).
Off Time (on page 6-56)	Available when Pulse Mode is selected. The amount of time that the pulse is off (5 ms to 20 s).
Base Voltage (on page 6-57)	Available when Pulse Mode is selected. The voltage level that the instrument sources when the pulse output is off. The level that you can set depends on the present source range.
Overvoltage Protection (on page 6-57)	A limit that restricts the maximum voltage level that the instrument can source. It is in effect when either current or voltage is sourced.
Measure Current (on page 6-57)	When this box is selected, the instrument measures current. Current is recorded in the Analyze sheet and shown in the graph.
Current Column Name (on page 6-58)	The name of the current measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Current Range (on page 6-58)	The measure range determines the full-scale measurement span that is applied to the signal.
Low Range (on page 6-58)	Available when the Measure Range is set to Limited Auto. This sets the minimum range that the instrument uses.
Measure or Report Voltage (on page 6-58)	Determines if voltage values are recorded in the Analyze spreadsheet.
Report Value (Report Voltage or Measure Voltage) (on page 6-59)	Determines if programmed voltage values or actual measured voltage values are recorded in the Analyze spreadsheet.

Parameter	Description
Voltage Column Name (on page 6-59)	The name of the voltage measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Report Status (on page 6-60)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.

Voltage Log Sweep operation mode - SMU

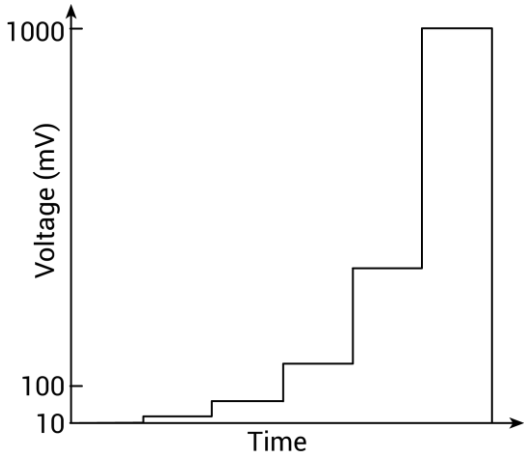
The Voltage Log Sweep operation mode allows you to sweep over a large range and plot the measurements on a logarithmic scale.

A linear sweep is typically unsatisfactory for such applications, because the first increment can miss several of the lower decades. For example, the first ~0.1 V step of a 101-point linear sweep from 0.001 V to 10 V misses the two decades between 0.001 V and 0.1 V.

By contrast, a log sweep varies the step size logarithmically over the specified range, so that all decades are characterized uniformly.

An example of a log sweep is shown in the following figure.

Figure 232: Example logarithmic sweep



The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

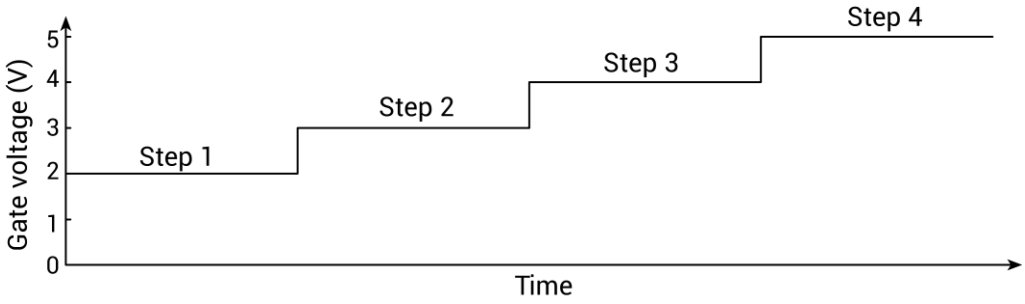
Parameter	Description
Start (on page 6-48)	The voltage source level at which the sweep starts.
Stop (on page 6-48)	The voltage source level at which the sweep stops.
Points (on page 6-50)	The number of data points that the sweep will generate. Clarius uses the start, stop, and data points to calculate the step size and forcing values.
Dual Sweep (on page 6-51)	When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.
Force Range (Source Range) (on page 6-52)	The SMU range that is used when forcing the voltage.
Compliance (on page 3-4)	A limit that stops the 4200A-SCS from sourcing a current that is more than that limit.
Power On Delay (on page 6-53)	The delay between when SMUs are powered on in a test sequence.
Pulse Mode (on page 6-54)	Pulse Mode allows you to apply voltage or current to a device for brief periods at widely spaced intervals. This avoids device overheating in some tests. Pulse Mode is only available if the source range and measure ranges are set to a fixed range or the Best Fixed range.
On Time (on page 6-56)	Available when Pulse Mode is selected. The amount of time that the pulse is on (5 ms to 20 s).
Off Time (on page 6-56)	Available when Pulse Mode is selected. The amount of time that the pulse is off (5 ms to 20 s).
Base Voltage (on page 6-57)	Available when Pulse Mode is selected. The voltage level that the instrument sources when the pulse output is off. The level that you can set depends on the present source range.
Overvoltage Protection (on page 6-57)	A limit that restricts the maximum voltage level that the instrument can source. It is in effect when either current or voltage is sourced.
Measure Current (on page 6-57)	When this box is selected, the instrument measures current. Current is recorded in the Analyze sheet and shown in the graph.
Current Column Name (on page 6-58)	The name of the current measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Current Range (on page 6-58)	The measure range determines the full-scale measurement span that is applied to the signal.
Low Range (on page 6-58)	Available when the Measure Range is set to Limited Auto. This sets the minimum range that the instrument uses.
Measure or Report Voltage (on page 6-58)	Determines if voltage values are recorded in the Analyze spreadsheet.
Report Value (Report Voltage or Measure Voltage) (on page 6-59)	Determines if programmed voltage values or actual measured voltage values are recorded in the Analyze spreadsheet.
Voltage Column Name (on page 6-59)	The name of the voltage measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Report Status (on page 6-60)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.

Voltage Step operation mode - SMU

The Voltage Step operation mode increments through evenly-spaced, constant voltage steps over a range that you specify. The time interval for each step is determined automatically by the time required to complete a sweep.

For each step, parametric curve data is generated. The data is recorded in the Analyze pane.

Figure 233: Stepping the gate voltage of a FET



The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

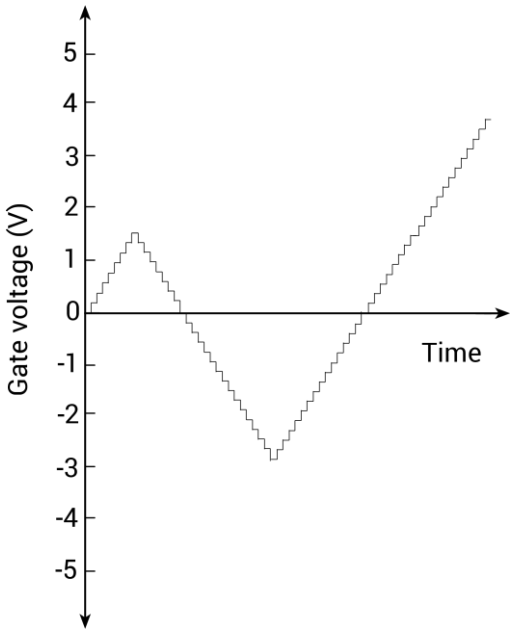
Parameter	Description
Start (on page 6-48)	The Start parameter is the voltage that is forced for the first step value.
Stop (on page 6-48)	The voltage that is forced for the last step value.
Step (on page 6-49)	The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).
Points (on page 6-50)	The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters.
Force Range (Source Range) (on page 6-52)	The SMU range that is used when forcing the voltage.
Compliance (on page 3-4)	A limit that stops the 4200A-SCS from sourcing a current that is more than that limit.
Power On Delay (on page 6-53)	The delay between when SMUs are powered on in a test sequence.
Overvoltage Protection (on page 6-57)	A limit that restricts the maximum voltage level that the instrument can source. It is in effect when either current or voltage is sourced.
Measure Current (on page 6-57)	When this box is selected, the instrument measures current. Current is recorded in the Analyze sheet and shown in the graph.
Current Column Name (on page 6-58)	The name of the current measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Current Range (on page 6-58)	The measure range determines the full-scale measurement span that is applied to the signal.
Low Range (on page 6-58)	Available when the Measure Range is set to Limited Auto. This sets the minimum range that the instrument uses.
Measure or Report Voltage (on page 6-58)	Determines if voltage values are recorded in the Analyze spreadsheet.
Report Value (Report Voltage or Measure Voltage) (on page 6-59)	Determines if programmed voltage values or actual measured voltage values are recorded in the Analyze spreadsheet.
Voltage Column Name (on page 6-59)	The name of the voltage measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Report Status (on page 6-60)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.

Voltage Segment Sweep operation mode - SMU

When you select the Voltage Segment Sweep operation mode, the test increments through a series of constant voltage steps. You can define the starting voltage and up to four stop voltage points and four step voltage points.

An example of a three-segment voltage sweep is shown in the following figure.

Figure 234: Example multi-segment voltage sweep



The parameters that are available for this mode are described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Segments (on page 6-48)	The number of source sweeps in Segment Sweep operation mode. You can define up to four segments with distinct Start, Stop, and Step points.
Start (on page 6-48)	The voltage at which a segment sweep starts.
Stop (on page 6-48)	The voltage at which a segment sweep stops.
Step (on page 6-49)	The voltage size of each step of a segment sweep.
Points (on page 6-50)	The number of data points that the sweep will generate. Clarius uses the start, stop, and data points to calculate the step size and forcing values.
Force Range (Source Range) (on page 6-52)	The SMU range that is used when forcing the voltage.
Compliance (on page 3-4)	A limit that stops the 4200A-SCS from sourcing a current that is more than that limit.
Power On Delay (on page 6-53)	The delay between when SMUs are powered on in a test sequence.
Overvoltage Protection (on page 6-57)	A limit that restricts the maximum voltage level that the instrument can source. It is in effect when either current or voltage is sourced.
Measure Current (on page 6-57)	When this box is selected, the instrument measures current. Current is recorded in the Analyze sheet and shown in the graph.
Current Column Name (on page 6-58)	The name of the current measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Current Range (on page 6-58)	The measure range determines the full-scale measurement span that is applied to the signal.
Low Range (on page 6-58)	Available when the Measure Range is set to Limited Auto. This sets the minimum range that the instrument uses.
Measure or Report Voltage (on page 6-58)	Determines if voltage values are recorded in the Analyze spreadsheet.
Report Value (Report Voltage or Measure Voltage) (on page 6-59)	Determines if programmed voltage values or actual measured voltage values are recorded in the Analyze spreadsheet.
Voltage Column Name (on page 6-59)	The name of the voltage measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Report Status (on page 6-60)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.

Current Bias operation mode - SMU

Current Bias operation mode maintains a selected constant-current state at the terminal, subject to the maximum voltage compliance of the connected SMU.

The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Bias (on page 6-48)	The bias is the amount of current to be forced.
Force Range (on page 6-52)	The SMU range that is used when forcing the current.
Compliance (on page 3-4)	A limit that stops the 4200A-SCS from sourcing a voltage that is more than that limit.
Power On Delay (on page 6-53)	The delay between when SMUs are powered on in a test sequence.
Pulse Mode (on page 6-54)	Pulse Mode allows you to apply voltage or current to a device for brief periods at widely spaced intervals. This avoids device overheating in some tests. Pulse Mode is only available if the source range and measure ranges are set to a fixed range or the Best Fixed range.
On Time (on page 6-56)	Available when Pulse Mode is selected. The amount of time that the pulse is on (5 ms to 20 s).
Off Time (on page 6-56)	Available when Pulse Mode is selected. The amount of time that the pulse is off (5 ms to 20 s).
Base Current (on page 6-57)	Available when Pulse Mode is selected. The current level that the instrument sources when the pulse output is off. The level that you can set depends on the present source range.
Overvoltage Protection (on page 6-57)	A limit that restricts the maximum voltage level that the instrument can source. It is in effect when either current or voltage is sourced.
Measure Current (on page 6-57)	When this box is selected, the instrument measures current. Current is recorded in the Analyze sheet and shown in the graph.
Report Value (Report Current or Measure Current) (on page 6-60)	The Report Value setting determines which current values are recorded in the Analyze spreadsheet.
Current Column Name (on page 6-58)	The name of the current measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Voltage (Report Voltage) (on page 6-58)	When this box is selected, the instrument measures voltage. Voltage is recorded in the Analyze sheet and shown in the graph.
Voltage Column Name (on page 6-59)	The name of the voltage measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Voltage Range (on page 6-59)	The measure range determines the full-scale measurement span that is applied to the signal.
Report Status (on page 6-60)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.

Current Linear Sweep operation mode - SMU

When you select the Current Linear Sweep operation mode, the test increments through a series of constant current steps. You define the start and stop currents and the current size between each step.

The current sweep generates parametric curve data that is recorded in the Analyze pane.

The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

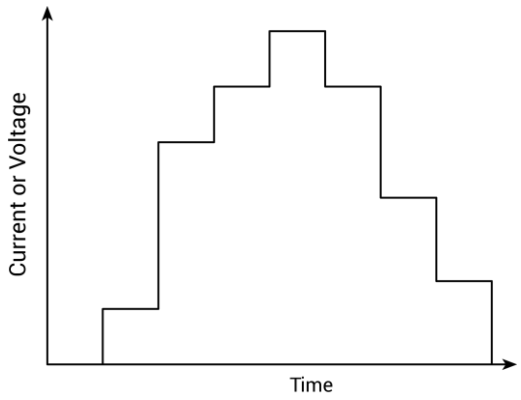
Parameter	Description
Start (on page 6-48)	The current source level at which the sweep starts.
Stop (on page 6-48)	The current source level at which the sweep stops.
Step	The current size of each step of the sweep. The current level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).
Points (on page 6-50)	The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters.
Dual Sweep (on page 6-51)	When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.
Force Range (Source Range) (on page 6-52)	The SMU range that is used when forcing the current.
Compliance (on page 3-4)	A limit that stops the 4200A-SCS from sourcing a voltage that is more than that limit.
Power On Delay (on page 6-53)	The delay between when SMUs are powered on in a test sequence.
Pulse Mode (on page 6-54)	Pulse Mode allows you to apply voltage or current to a device for brief periods at widely spaced intervals. This avoids device overheating in some tests. Pulse Mode is only available if the source range and measure ranges are set to a fixed range or the Best Fixed range.
On Time (on page 6-56)	Available when Pulse Mode is selected. The amount of time that the pulse is on (5 ms to 20 s).
Off Time (on page 6-56)	Available when Pulse Mode is selected. The amount of time that the pulse is off (5 ms to 20 s).
Base Current (on page 6-57)	Available when Pulse Mode is selected. The current level that the instrument sources when the pulse output is off. The level that you can set depends on the present source range.
Overvoltage Protection (on page 6-57)	A limit that restricts the maximum voltage level that the instrument can source. It is in effect when either current or voltage is sourced.
Measure Current (on page 6-57)	When this box is selected, the instrument measures current. Current is recorded in the Analyze sheet and shown in the graph.
Report Value (Report Current or Measure Current) (on page 6-60)	The Report Value setting determines which current values are recorded in the Analyze spreadsheet.
Current Column Name (on page 6-58)	The name of the current measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.

Parameter	Description
Voltage (Report Voltage) (on page 6-58)	When this box is selected, the instrument measures voltage. Voltage is recorded in the Analyze sheet and shown in the graph.
Voltage Column Name (on page 6-59)	The name of the voltage measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Voltage Range (on page 6-59)	The measure range determines the full-scale measurement span that is applied to the signal.
Report Status (on page 6-60)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.

Current List Sweep operation mode - SMU

The Current List Sweep operation mode allows you to customize the current values for each step of the sweep. 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. You can also use list sweeps to make pulsed measurements to avoid overheating of sensitive devices. The following figure illustrates a possible list sweep.

Figure 235: Example list sweep



The current sweep generates parametric curve data that is recorded in the Analyze pane.

The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

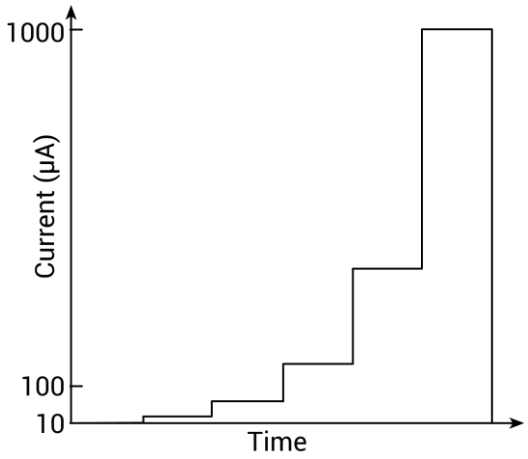
Parameter	Description
List Values (on page 6-50)	Select **Enter Values** to open a dialog box in which you can enter the current level for each step of the sweep in the rows. You can enter any valid instrument current.
Points (on page 6-50)	The number of sweep points that were defined in the List Values list. This number is automatically generated.
Dual Sweep (on page 6-51)	When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.
Force Range (Source Range) (on page 6-52)	The SMU range that is used when forcing the current.
Compliance (on page 3-4)	A limit that stops the 4200A-SCS from sourcing a voltage that is more than that limit.
Power On Delay (on page 6-53)	The delay between when SMUs are powered on in a test sequence.
Pulse Mode (on page 6-54)	Pulse Mode allows you to apply voltage or current to a device for brief periods at widely spaced intervals. This avoids device overheating in some tests. Pulse Mode is only available if the source range and measure ranges are set to a fixed range or the Best Fixed range.
On Time (on page 6-56)	Available when Pulse Mode is selected. The amount of time that the pulse is on (5 ms to 20 s).
Off Time (on page 6-56)	Available when Pulse Mode is selected. The amount of time that the pulse is off (5 ms to 20 s).
Base Current (on page 6-57)	Available when Pulse Mode is selected. The current level that the instrument sources when the pulse output is off. The level that you can set depends on the present source range.
Overvoltage Protection (on page 6-57)	A limit that restricts the maximum voltage level that the instrument can source. It is in effect when either current or voltage is sourced.
Measure Current (on page 6-57)	When this box is selected, the instrument measures current. Current is recorded in the Analyze sheet and shown in the graph.
Report Value (Report Current or Measure Current) (on page 6-60)	The Report Value setting determines which current values are recorded in the Analyze spreadsheet.
Current Column Name (on page 6-58)	The name of the current measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Voltage (Report Voltage) (on page 6-58)	When this box is selected, the instrument measures voltage. Voltage is recorded in the Analyze sheet and shown in the graph.
Voltage Column Name (on page 6-59)	The name of the voltage measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Voltage Range (on page 6-59)	The measure range determines the full-scale measurement span that is applied to the signal.
Report Status (on page 6-60)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.

Current Log Sweep operation mode - SMU

The Current Log Sweep operation mode allows you to sweep over a large range and plot the measurements on a logarithmic scale.

An example of a log sweep is shown in the following figure.

Figure 236: Example log sweep



The current log sweep generates parametric curve data that is recorded in the Analyze pane.

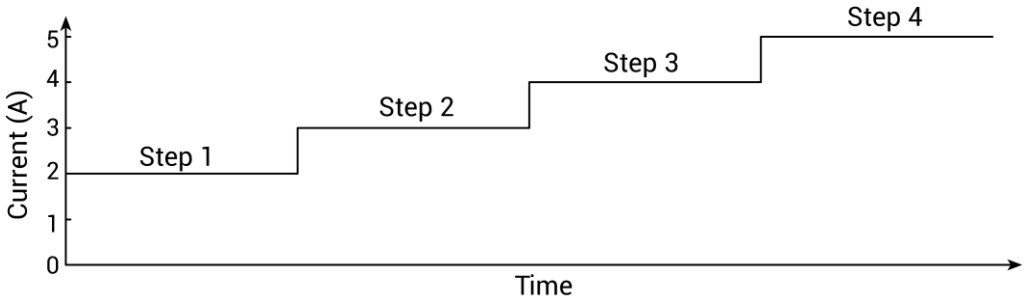
The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Start (on page 6-48)	The current source level at which the sweep starts.
Stop (on page 6-48)	The current source level at which the sweep stops.
Points	The number of data points that the sweep will generate. Clarius uses the start, stop, and data points to calculate the step size and forcing values.
Dual Sweep (on page 6-51)	When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.
Force Range (Source Range) (on page 6-52)	The SMU range that is used when forcing the current.
Compliance (on page 3-4)	A limit that stops the 4200A-SCS from sourcing a voltage that is more than that limit.
Power On Delay (on page 6-53)	The delay between when SMUs are powered on in a test sequence.
Pulse Mode (on page 6-54)	Pulse Mode allows you to apply voltage or current to a device for brief periods at widely spaced intervals. This avoids device overheating in some tests. Pulse Mode is only available if the source range and measure ranges are set to a fixed range or the Best Fixed range.
On Time (on page 6-56)	Available when Pulse Mode is selected. The amount of time that the pulse is on (5 ms to 20 s).
Off Time (on page 6-56)	Available when Pulse Mode is selected. The amount of time that the pulse is off (5 ms to 20 s).
Base Current (on page 6-57)	Available when Pulse Mode is selected. The current level that the instrument sources when the pulse output is off. The level that you can set depends on the present source range.
Overvoltage Protection (on page 6-57)	A limit that restricts the maximum voltage level that the instrument can source. It is in effect when either current or voltage is sourced.
Measure Current (on page 6-57)	When this box is selected, the instrument measures current. Current is recorded in the Analyze sheet and shown in the graph.
Report Value (Report Current or Measure Current) (on page 6-60)	The Report Value setting determines which current values are recorded in the Analyze spreadsheet.
Current Column Name (on page 6-58)	The name of the current measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Voltage (Report Voltage) (on page 6-58)	When this box is selected, the instrument measures voltage. Voltage is recorded in the Analyze sheet and shown in the graph.
Voltage Column Name (on page 6-59)	The name of the voltage measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Voltage Range (on page 6-59)	The measure range determines the full-scale measurement span that is applied to the signal.
Report Status (on page 6-60)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.

Current Step operation mode - SMU

The Current Step operation mode increments through evenly-spaced, constant current steps over a range that you specify. The time interval for each step is determined automatically by the time required to complete a sweep.

Figure 237: Stepping the diode current



For each step, parametric curve data is generated that is recorded in the Analyze pane.

The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

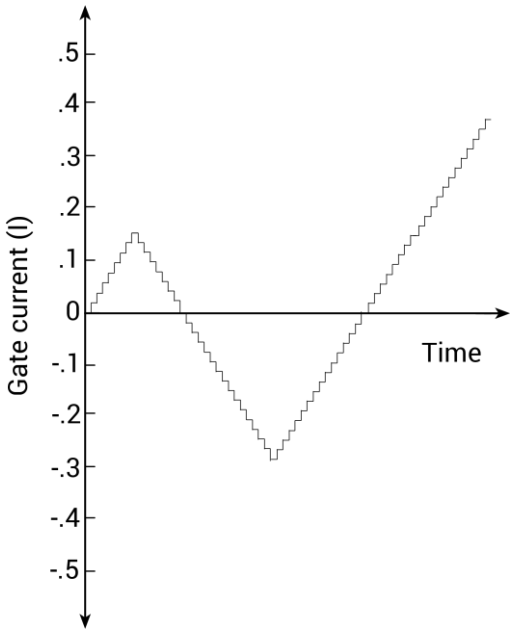
Parameter	Description
Start (on page 6-48)	The current that is forced for the first step value.
Stop (on page 6-48)	The current that is forced for the last step value.
Step (on page 6-49)	The current size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).
Points (on page 6-50)	The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters.
Force Range (Source Range) (on page 6-52)	The SMU range that is used when forcing the current.
Compliance (on page 3-4)	A limit that stops the 4200A-SCS from sourcing a voltage that is more than that limit.
Power On Delay (on page 6-53)	The delay between when SMUs are powered on in a test sequence.
Overvoltage Protection (on page 6-57)	A limit that restricts the maximum voltage level that the instrument can source. It is in effect when either current or voltage is sourced.
Measure Current (on page 6-57)	When this box is selected, the instrument measures current. Current is recorded in the Analyze sheet and shown in the graph.
Report Value (Report Current or Measure Current) (on page 6-60)	The Report Value setting determines which current values are recorded in the Analyze spreadsheet.
Current Column Name (on page 6-58)	The name of the current measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Voltage (Report Voltage) (on page 6-58)	When this box is selected, the instrument measures voltage. Voltage is recorded in the Analyze sheet and shown in the graph.
Voltage Column Name (on page 6-59)	The name of the voltage measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Voltage Range (on page 6-59)	The measure range determines the full-scale measurement span that is applied to the signal.
Report Status (on page 6-60)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.

Current Segment Sweep operation mode - SMU

When you select the Current Segment Sweep operation mode, the test increments through a series of constant current steps. You can define the starting current and up to four stop current points and four step current points.

An example of a three-segment current sweep is shown in the following figure.

Figure 238: Example multi-segment current sweep



The parameters that are available for this mode are described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Segments (on page 6-48)	The number of source sweeps in Segment Sweep operation mode. You can define up to four segments with distinct Start, Stop, and Step points.
Start (on page 6-48)	The current at which a segment sweep starts.
Stop (on page 6-48)	The current at which a segment sweep stops.
Step	The current size of each step of a segment sweep.
Points (on page 6-50)	The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters.
Force Range (Source Range) (on page 6-52)	The SMU range that is used when forcing the current.
Compliance (on page 3-4)	A limit that stops the 4200A-SCS from sourcing a voltage that is more than that limit.
Power On Delay (on page 6-53)	The delay between when SMUs are powered on in a test sequence.
Overvoltage Protection (on page 6-57)	A limit that restricts the maximum voltage level that the instrument can source. It is in effect when either current or voltage is sourced.
Measure Current (on page 6-57)	When this box is selected, the instrument measures current. Current is recorded in the Analyze sheet and shown in the graph.
Report Value (Report Current or Measure Current) (on page 6-60)	The Report Value setting determines which current values are recorded in the Analyze spreadsheet.
Current Column Name (on page 6-58)	The name of the current measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Voltage (Report Voltage) (on page 6-58)	When this box is selected, the instrument measures voltage. Voltage is recorded in the Analyze sheet and shown in the graph.
Voltage Column Name (on page 6-59)	The name of the voltage measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.
Voltage Range (on page 6-59)	The measure range determines the full-scale measurement span that is applied to the signal.
Report Status (on page 6-60)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.

Common operation mode - SMU

Common operation mode maintains a zero-voltage state at the terminal, subject to the maximum current compliance of the connected SMU. The 4200-SMU has a compliance level of 105 mA. The 4210-SMU has a compliance level of 1.05 A.

You cannot set any parameters for the Common operation mode.

SMU - all terminal parameters

When you select **All Parameters**, the Configure pane displays all available parameters for the test that is selected in the project tree.

The descriptions of each operation mode contains a list and brief description of the parameters that are available when that mode is selected. Full parameter descriptions are provided in the following topics.

Bias

The bias is the amount of voltage or current to be forced.

Segments

The number of source sweeps in Segment Sweep operation mode. You can define up to four segments with distinct Start, Stop, and Step points.

Start (sweep)

The current or voltage source level at which the sweep starts. For a log sweep, the start value cannot be 0.

Stop (sweep)

The voltage or current source level at which the sweep stops. For a log sweep, the stop value cannot be 0 or the opposite polarity of the start value.

Start (step)

The Start parameter is the current or voltage that is forced for the first step value.

Stop (step)

The current or voltage that is forced for the last step value.

Step (step)

Specifies the current or voltage increments of the steps.

Clarius 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.

Use a step value that does not result in a fractional number of data points. If the data point is fractional, the step value is forced to a value that results in a whole number of data points. To calculate the data points:

$$data\ points = \frac{(stop - start)}{step} + 1$$

The data points are rounded to the nearest value.

For example, if Start = 0 V, Stop = 5 V, and Step = 0.6 V:

$$data\ points = \frac{(5 - 0)}{0.6} + 1$$

In this case, the Step value is forced to 0.625 V, which results in a data point value of 9.333, which is rounded to 9. The instrument forces nine voltages at 0 V, 0.625 V, 1.25 V, 1.875 V, 2.5 V, 3.125 V, 3.75 V, 4.375 V, and 5 V.

If Start = 0 A, Stop = 0.005 A, and Step = 0.0015 A:

$$data\ points = \frac{(0.005 - 0)}{0.0015} + 1$$

This results in a data point value of 4.333, which is rounded to 4. The instrument forces four values at 0 A, 0.001666 A, 0.0033326 A, and 0.0049992 A.

Step (voltage sweep)

The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).

Clarius 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.

Use a step value that does not result in a fractional number of data points. If the data point is fractional, the step value is forced to a value that results in a whole number of data points. To calculate the data points:

$$data\ points = \frac{(stop - start)}{step} + 1$$

The data points are rounded to the nearest value.

For example, if Start = 0 V, Stop = 5 V, and Step = 0.6 V:

$$data\ points = \frac{(5 - 0)}{0.6} + 1$$

In this case, the Step value is forced to 0.625 V, which results in a data point value of 9.333, which is rounded to 9. The instrument forces nine voltages at 0 V, 0.625 V, 1.25 V, 1.875 V, 2.5 V, 3.125 V, 3.75 V, 4.375 V, and 5 V.

Points

The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters, using the equation:

$$data\ points = \frac{(stop - start)}{step} + 1$$

List Values

Select **Enter Values** to open a dialog box in which you can type the current or voltage level for each step of the sweep in the rows. You can type any valid instrument current or voltage.

You can select a value or multiple values in the list and copy, cut, or delete them. Use Paste to add values that you copied or cut to a new location in the list.

Note that you cannot have blank rows in between values.

You can use the Ctrl key plus mouse selections to pick selected rows, then use the buttons to copy, cut, or delete those rows. Note that when you paste the rows, any skipped rows are ignored (there will be no blank rows).

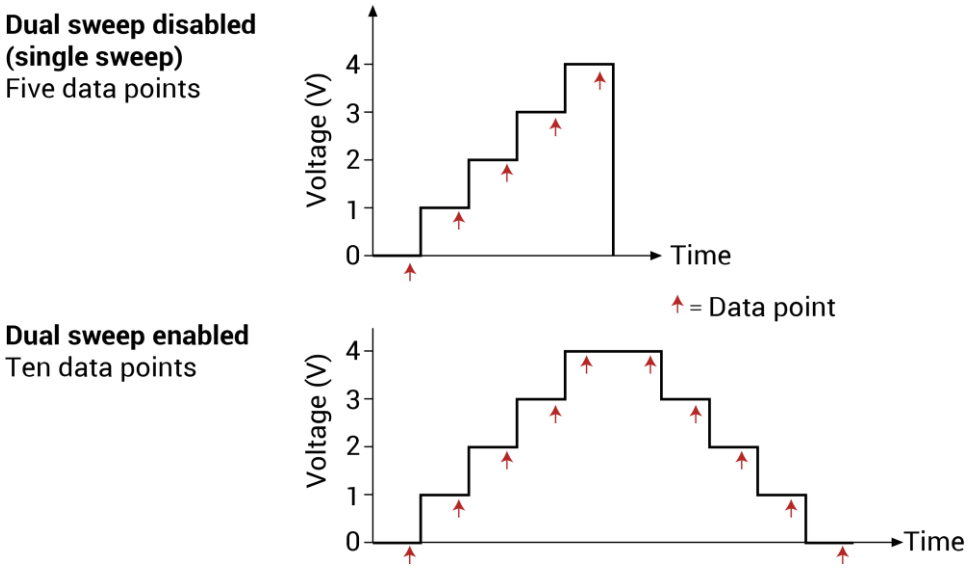
You can use the Shift key plus mouse selections to pick a range of rows. When you paste the rows, any blank rows are ignored.

Dual Sweep

When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.

The following figure compares a single sweep to a dual sweep.

Figure 239: Single and dual sweep examples (linear voltage sweep; 0 V to 4 V in 1 V steps)



Points (list sweep)

The number of sweep points that were defined in the List Values list. This number is automatically generated.

Force Range (Source Range)

The SMU range that is used when forcing the voltage or current.

You can select:

- **Best Fixed**: The instrument selects a single fixed source range that will accommodate all the source levels in the test.
- **Auto:** The instrument selects the most sensitive source range for each source level in the test. This option provides the best resolution and control when sweeping or stepping several decades. However, the range changes can reduce speed.
- **Specific range**: Select one of the listed ranges. The source remains on the range that is set. If you are sweeping and a sweep point exceeds the source range capability, the source outputs the maximum level for that range. This range must be equal to or greater than the largest value in the sweep.

Compliance limits

You can set a limit that stops a SMU from sourcing a current or voltage that is more than that limit. This limit is called compliance and helps prevent damage to the device under test (DUT). The SMU will not exceed the maximum limit set for compliance. When a SMU is acting as a current source, the voltage is clamped at the compliance value; conversely, the current is clamped at the compliance value when the SMU is acting as a voltage source.

When a SMU reaches compliance, it continues to make measurements. However, the measurement stays at the value it was at when compliance occurred. For example, if you are sourcing voltage and the compliance is set to 100 mA, it continues to measure 100 mA after compliance is reached. The voltage, however, is not at the programmed value.

You can stop the test if the source reaches compliance. Refer to Compliance exit condition options (on page 6-117) for detail.

If you set a specific measurement range, compliance can also be restricted by the range. Compliance must be more than 11% of the measurement range. If not, an event is generated and the compliance setting is automatically changed to the maximum compliance value for the selected range. For example, if compliance is set to 1 V and the measurement range is 200 mV, output voltage will clamp at 210 mV. If you attempt to change compliance to a value that is not appropriate for the selected range, compliance is not changed and a warning is generated. You must change the range before you can select the new compliance value. If you set the measurement range to be automatically selected, the measurement range does not affect compliance.

The lowest allowable compliance is based on the load and the source value. For example, if you are sourcing 1 V to a 1 kΩ resistor, the lowest allowable current compliance is 1 mA (1 V/1 kΩ = 1 mA). Setting a compliance lower than 1 mA limits the source.

For another example, assume the following conditions:

- Current compliance: 10 mA
- Voltage sourced by the instrument: 10 V
- DUT resistance: 10 Ω

With a source voltage of 10 V and a DUT resistance of 10 Ω, the current through the DUT should be 10 V / 10 Ω = 1 A. However, because compliance is set to 10 mA, the current cannot exceed 10 mA, and the voltage across the resistance is limited to 100 mV. In effect, the 10 V voltage source is transformed into a 10 mA current source.

In steady-state conditions, the set compliance value restricts the instrument output unless there are fast transient load conditions.

When measurement autorange is disabled, the maximum and minimum compliance values cannot be set below the minimum value. When autorange is enabled, the programmed compliance value cannot be set below 10 nA when sourcing voltage, or below 20 mV when sourcing current.

Power On Delay

When a test is run, the SMUs power on in a specific sequence. You can change the amount of time between when the SMUs power on in the sequence.

The first SMU in the sequence always powers on immediately. Subsequent SMUs power on in sequence after the delay.

For example, assume there are three SMUs in a test and the power-on sequence is SMU1, SMU2, and SMU3. The power-on delay for SMU1 is set to 50 ms and the delay for SMU2 is set to 100 ms. When the test is started, the power-on sequence for the SMUs is:

1. SMU1 powers on.
2. 50 ms delay.
3. SMU2 powers on.
4. 100 ms delay.
5. SMU3 powers on.

You can set the delay from 0 to 1.0 s.

For information on changing the sequence, refer to SMU Power On Sequence (on page 6-115).

Pulse Mode

Pulse Mode allows you to apply voltage or current to a device for brief periods at widely spaced intervals. This avoids device overheating in some tests. Pulse Mode is only available if the source range and measure ranges are set to a fixed range or the Best Fixed range. Select this option to enable Pulse Mode.

When Pulse Mode is selected, you can set the pulse on and off times and the pulse base voltage or current. The pulse output goes to the specified voltage or current level when the pulse is on. When the pulse is off, the pulse output returns to the specified base voltage or base current level.

The pulse on and off times determine the pulse period and pulse width as follows:

Pulse period = On Time + Off Time + cumulative measure time (if set to measure)

Pulse width = On Time

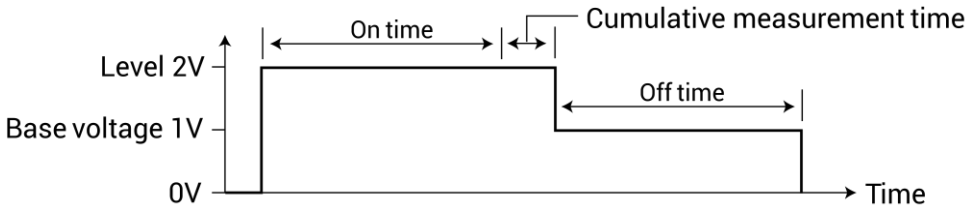
Pulse on and off times can be set from 5 ms to 20 s. The base voltage (or current) that can be set is dependent the present source range.

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 take precedence so that the SMUs operate synchronously and run at the same speed.

An example single-sweep pulse output for the Voltage Bias operation mode is shown in the following figure. The pulse output goes to the specified pulse level during the pulse on time. If the instrument is set to measure, the measurement occurs after the on time expires and before the transition to the off time. 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.

Figure 240: Pulse Mode example: Voltage bias: 2 V level, 1 V base



When a measurement is made, it effectively increases the on time by the amount of time required to make the measurement.

To minimize this extra time:

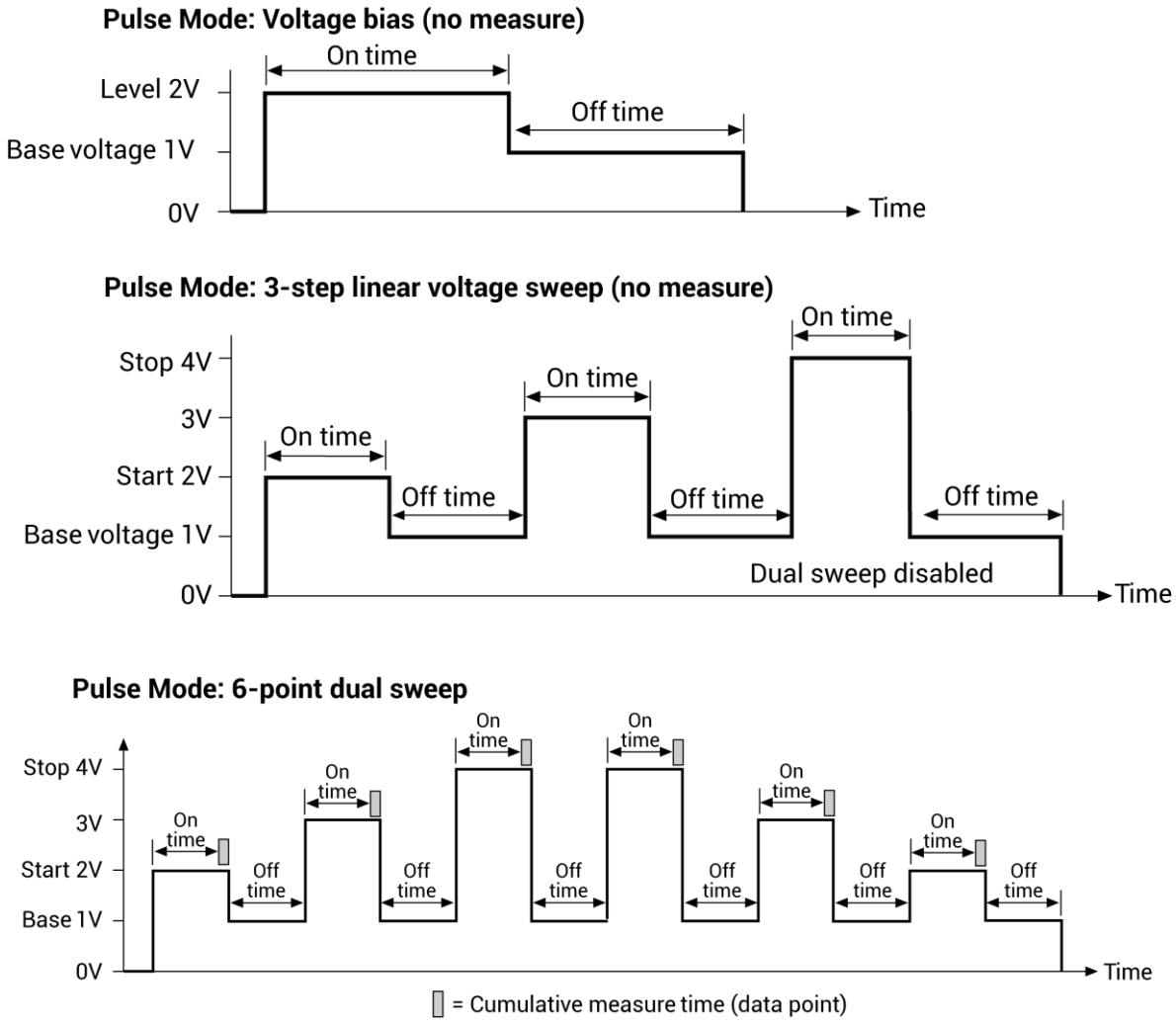
1. Select **Test Settings**.
2. Select **Advanced**.
3. For the Speed, select **Custom**.
4. Set the Delay Factor to **0**.
5. Set the Filter Factor to **0**.
6. Set the A/D aperture time to **0.01**.

This is the fastest measurement timing, but it reduces accuracy.

For a sweep operation mode, pulse output goes to the sweep step levels during the pulse on times. During the off times, pulse output goes to the specified Base Voltage (or Base Current) level. If set to measure, the measurement will occur after each on time period expires and before the pulse transitions to the off time level.

The figure below shows some Pulse Mode examples.

Figure 241: Pulse Mode examples



On Time

Available when Pulse Mode is selected. The amount of time that the pulse is on (5 ms to 20 s).

Off Time

Available when Pulse Mode is selected. The amount of time that the pulse is off (5 ms to 20 s).

Base Voltage

Available when Pulse Mode is selected. The voltage level that the instrument sources when the pulse output is off. The level that you can set depends on the present source range.

Base Current

Available when Pulse Mode is selected. The current level that the instrument sources when the pulse output is off. The level that you can set depends on the present source range.

Overvoltage protection

Overvoltage protection restricts the maximum voltage level that the instrument can source. It is in effect when either current or voltage is sourced.

This protection is in effect for both positive and negative output voltages.

When this attribute is used in a test sequence, it should be set before turning the source on.

⚠ WARNING

Even with the overvoltage protection set to the lowest value, never touch anything connected to the terminals of the 4200A-SCS when the instrument is on. Always assume that a hazardous voltage (greater than 30 V_{RMS}) is present when the instrument is on. To prevent damage to the device under test or external circuitry, do not set the voltage source to levels that exceed the value that is set for overvoltage protection.

⚠ WARNING

Asserting the interlock allows the SMU and preamplifier terminals to become hazardous, exposing the user to possible electrical shock that could result in personal injury or death. SMU and preamplifier 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 V, Category O.

Measure Current (Current)

When this box is selected, the instrument measures current. Current is recorded in the Analyze sheet and shown in the graph.

Current Column Name

The name of the current measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.

If you do not define a name, Clarius assigns a name. The assigned name is a combination of the terminal label and I for current. For example, the Base terminal is assigned the name `BaseI`.

This cannot be left blank. If it is left blank, Clarius uses the previous value.

The name is updated when the test is run. The results from earlier tests that are available through Run History are not changed to the new name.

Low Range

Available when the Measure Range is set to Limited Auto. This sets the minimum range that the instrument uses.

Current Range

The measure range determines the full-scale measurement span that is applied to the signal. Therefore, it affects both the accuracy of the measurements and the maximum signal that can be measured.

The current range options are:

- **Auto:** The instrument automatically optimizes the measurement range as the test progresses. This option provides the best resolution when the measurements span several decades. However, time delays can occur with range changes that can limit the measurement speed.
- **Limited Auto:** A compromise between Auto and a fixed range option. It allows you to specify the minimum range that the SMU uses when it automatically optimizes the current measurements. This option reduces test time when you do not need maximum resolution at minimum currents.
- **Best Fixed:** The instrument automatically selects a single measurement range based on the current or voltage compliance value.
- **Specific ranges:** You can select a fixed measurement range.

Voltage (Report Voltage)

When this box is selected, the instrument measures voltage. Voltage is recorded in the Analyze sheet and shown in the graph.

Voltage Range

The measure range determines the full-scale measurement span that is applied to the signal. Therefore, it affects both the accuracy of the measurements and the maximum signal that can be measured.

The measurement range options are:

- **Auto:** The instrument automatically optimizes the measurement range as the test progresses. This option provides the best resolution when the measurements span several decades. However, range-change time delays limit the measurement speed.
- **Best Fixed:** The instrument automatically selects a single measurement range based on the current or voltage compliance value.
- **Specific ranges:** You can select a fixed measurement range from a list.

Voltage Column Name

The name of the voltage measurement. This is the name that Clarius displays in the Analyze spreadsheet column for that measurement.

If you do not define a name, Clarius assigns a name. The assigned name is a combination of the terminal label and V for voltage. For example, the Drain terminal is assigned the name `DrainV`.

This cannot be left blank. If it is left blank, Clarius uses the previous value.

The name is updated when the test is run. The results from previous tests that are available through Run History are not changed to the new name.

Report Value (Report Voltage or Measure Voltage)

If Voltage is selected, Report Value determines which voltage values are recorded in the Analyze spreadsheet. You can select:

- **Programmed:** Requested voltage values are recorded. For example, if you specified a voltage of 2.5 V, the reported value is 2.5 V, even if the measured value is 2.4997 V.
- **Measured:** Recorded voltage values are actual measured values. For example, if you specified a voltage of 2.5 V, the actual measured value, such as 2.4997 V, is recorded. The measured mode increases the measurement time because it requires an additional analog-to-digital (A/D) conversion.

Report Value (Measure Current or Report Current)

The Report Value setting determines which current values are recorded in the Analyze spreadsheet. You can select:

- **Programmed:** Requested current values are recorded. For example, if you specified a current of 10 mA, the reported value is 10 mA, even if the measured value is 9.9982 mA.
- **Measured:** Recorded voltage values are actual measured values. For example, if you specified a current of 10 mA, the actual measured value, such as 9.9982 mA, is recorded. The Measured mode increases the measurement time because it requires an additional analog-to-digital (A/D) conversion.

Report Status (SMU)

When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information. An example of the status information is displayed in the following figure.

Figure 242: SMU Report Status column in the Analyze pane

	A	B	C	D
1	Time	AV	SMU1_S	SMU2_S
2	174.6265E-3	173.5735E-3	00240251	009000B0
3	215.1949E-3	180.1555E-3	00240251	009000B0
4	256.2046E-3	186.7446E-3		009000B0
5	297.2239E-3	193.3215E-3		009000B0
6	338.2374E-3	199.8583E-3		009000B0
7	379.2452E-3	206.4212E-3		009000B0

Status:
Source Current
Measure Voltage
100nA Current Range
20V Voltage Range
Interlock open

Step (voltage sweep)

The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).

Clarius 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.

Use a step value that does not result in a fractional number of data points. If the data point is fractional, the step value is forced to a value that results in a whole number of data points. To calculate the data points:

$$data\ points = \frac{(stop - start)}{step} + 1$$

The data points are rounded to the nearest value.

For example, if Start = 0 V, Stop = 5 V, and Step = 0.6 V:

$$data\ points = \frac{(5 - 0)}{0.6} + 1$$

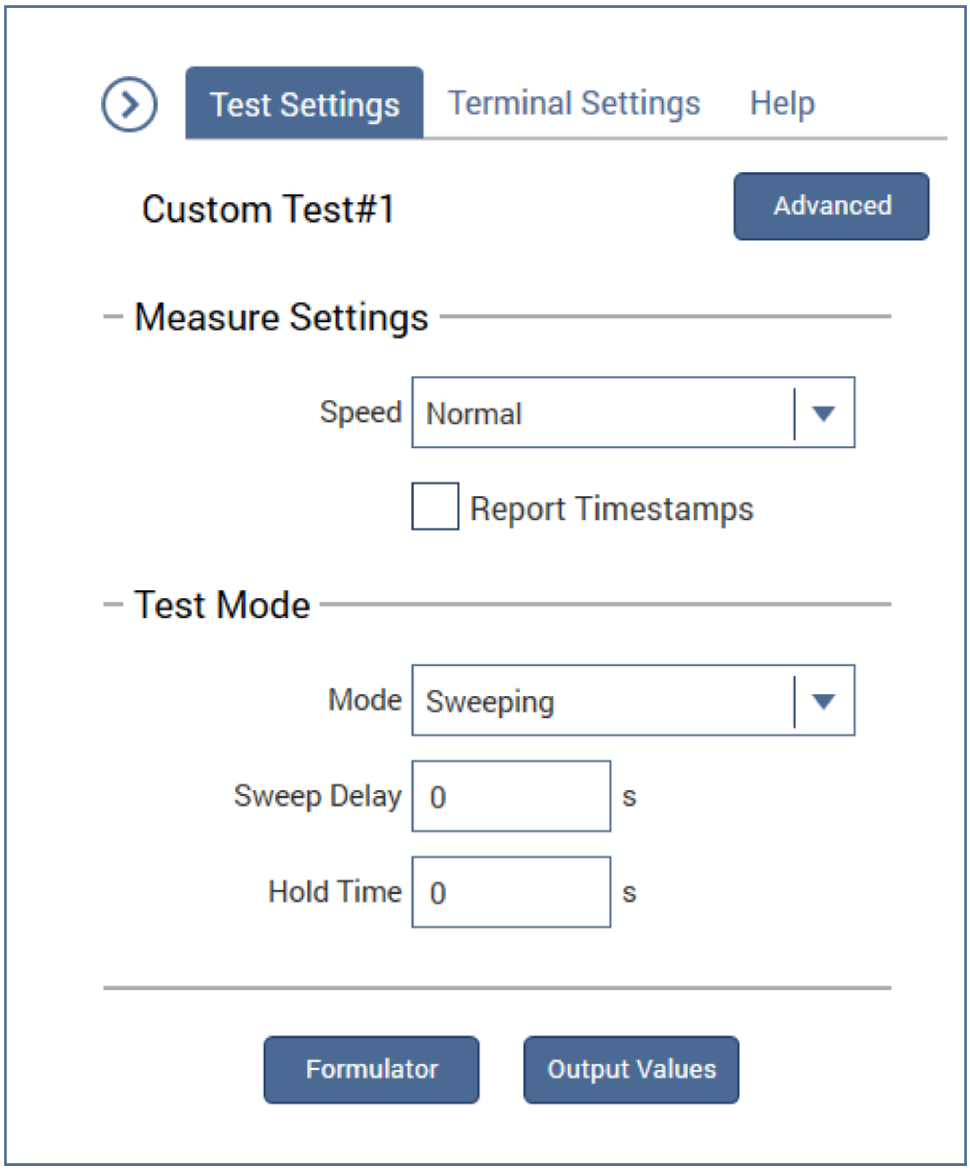
In this case, the Step value is forced to 0.625 V, which results in a data point value of 9.333, which is rounded to 9. The instrument forces nine voltages at 0 V, 0.625 V, 1.25 V, 1.875 V, 2.5 V, 3.125 V, 3.75 V, 4.375 V, and 5 V.

Operation Mode (CVU)

The following topics describe the operation modes that are available when a CVU is selected as the instrument.

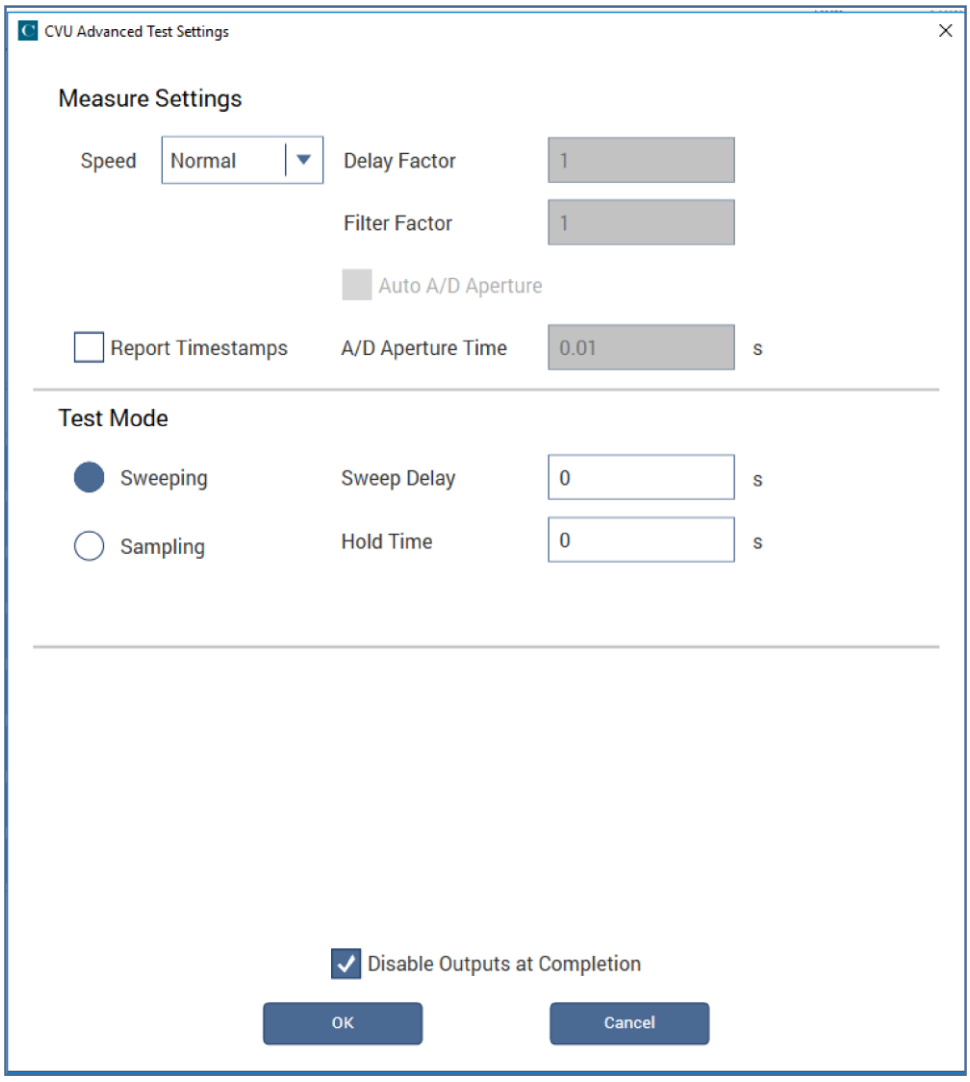
CVU Test Settings Pane

Figure 243: 4200A_CVU_TestSettingsPane.png



CVU Advanced Test Settings Pane

Figure 244: CVU Advanced Test Settings Pane



Voltage Bias operation mode - CVU

The Voltage Bias operation mode maintains a selected constant-voltage state at the terminal.

The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

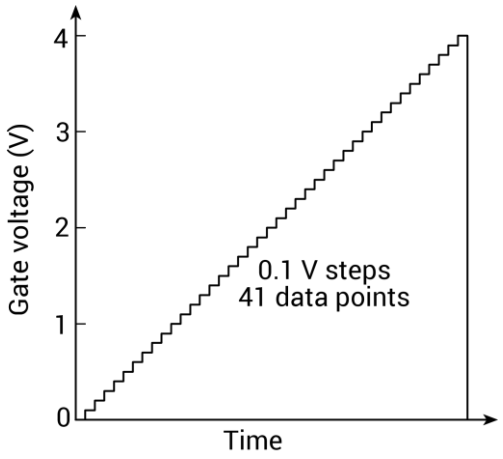
Parameter	Description
Presoak (on page 6-77)	Type a presoak voltage of –30 V to +30 V.
DC Bias (on page 6-80)	The DC bias range accepts a value between –30 V to 30 V.
Frequency (on page 6-80)	Select the frequency from the list of values.
AC Drive Signal (on page 6-80)	The AC drive signal accepts a value between 10 mV and 100 mV.
Parameters (on page 6-80)	Select the type of parameters. When using any of the tests supplied by Keithley Instruments, leave the measurement option set to `Cp-Gp`.
Param1 Column Name (on page 6-81)	The name that is used for parameter 1 in the Analyze sheet.
Param2 Column Name (on page 6-81)	The name that is used for parameter 2 in the Analyze sheet.
Report Test Conditions (on page 6-81)	Select this option to display the DC bias and drive frequency values on the Analyze sheet.
DCV Column Name (on page 6-81)	Available if you select Report Test Conditions. Determines the name of the column that contains the DC bias information in the Analyze sheet where this data is reported. You can change the name of the column by typing a new value.
Freq Column Name (on page 6-82)	Available if you select Report Test Conditions. Determines the name of the column that contains the frequency information in the Analyze sheet where this data is reported. You can change the name of the column by typing a new value.
Report Status (on page 6-82)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.
Compensation Open (on page 6-82)	Use the CVU connection compensation value that was generated for open connection compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.
Compensation Short (on page 6-82)	Use the CVU connection compensation value that was generated for short connection compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.
Compensation Load (on page 6-83)	Use the CVU connection compensation value that was generated for load compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.
Compensation Cable Length (on page 6-83)	The cable length that was used to generate connection compensation data. Make sure this cable length is the same as the Cable Length setting in the Tools > CVU Connection Compensation dialog box. Refer to Connection compensation (on page 4-14) for additional information.
AC Source V (on page 6-83)	Selects the terminal to use to source AC drive voltage.
AC Measure I Range (on page 6-83)	The measure range determines the full-scale measurement span that is applied to the signal.

Parameter	Description
DC Source V (on page 6-83)	Selects the terminal to use to source DC drive voltage.
DC Offset (on page 6-83)	Allows you offset the voltage by up to ±30 V on one terminal. For example, this allows you to output voltage from 0 V to 60 V instead of –30 V to +30 V.
Capacitance Range Estimator (on page 6-84)	The Capacitance Range Estimator automatically calculates the maximum capacitance value based on the parameter settings in the Advanced settings dialog box.

Voltage Linear Sweep operation mode - CVU

When you select the Voltage Linear Sweep operation mode, the test increments through a series of constant voltage steps. You define the start and stop voltages and the voltage size between each step. An example is shown in the next figure.

Figure 245: Example linear sweep



The voltage sweep generates parametric curve data that is recorded in the Analyze pane.

The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

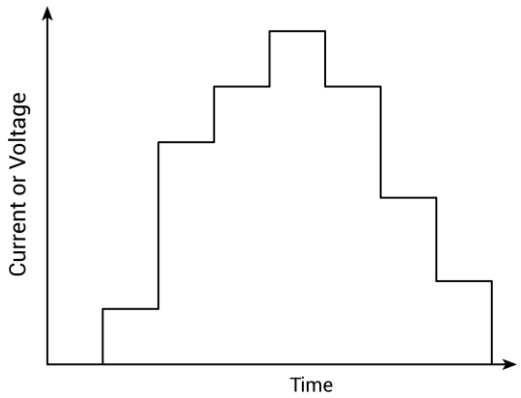
Parameter	Description
Presoak (on page 6-77)	Type a presoak voltage of –30 V to +30 V.
Start (on page 6-77)	The voltage source level at which the sweep starts.
Stop	The voltage source level at which the sweep stops.
Step (on page 6-49)	The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).
Points (on page 6-50)	The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters
Dual Sweep (on page 6-51)	When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.
Frequency (on page 6-80)	Select the frequency from the list of values.
AC Drive Signal (on page 6-80)	The AC drive signal accepts a value between 10 mV and 100 mV.
Parameters (on page 6-80)	Select the type of parameters. When using any of the tests supplied by Keithley Instruments, leave the measurement option set to `Cp-Gp`.
Param1 Column Name (on page 6-81)	The name that is used for parameter 1 in the Analyze sheet.
Param2 Column Name (on page 6-81)	The name that is used for parameter 2 in the Analyze sheet.
Report Test Conditions (on page 6-81)	Select this option to display the DC bias and drive frequency values on the Analyze sheet.
DCV Column Name (on page 6-81)	Available if you select Report Test Conditions. Determines the name of the column that contains the DC bias information in the Analyze sheet where this data is reported. You can change the name of the column by typing a new value.
Freq Column Name (on page 6-82)	Available if you select Report Test Conditions. Determines the name of the column that contains the frequency information in the Analyze sheet where this data is reported. You can change the name of the column by typing a new value.
Report Status (on page 6-82)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.
Compensation Open (on page 6-82)	Use the CVU connection compensation value that was generated for open connection compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.
Compensation Short (on page 6-82)	Use the CVU connection compensation value that was generated for short connection compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.

Parameter	Description
Compensation Load (on page 6-83)	Use the CVU connection compensation value that was generated for load compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.
Compensation Cable Length (on page 6-83)	The cable length that was used to generate connection compensation data. Make sure this cable length is the same as the Cable Length setting in the Tools > CVU Connection Compensation dialog box. Refer to Connection compensation (on page 4-14) for additional information.
AC Source V (on page 6-83)	Selects the terminal to use to source AC drive voltage.
AC Measure I Range (on page 6-83)	The measure range determines the full-scale measurement span that is applied to the signal.
DC Source V (on page 6-83)	Selects the terminal to use to source DC drive voltage.
DC Offset (on page 6-83)	Allows you offset the voltage by up to ±30 V on one terminal. For example, this allows you to output voltage from 0 V to 60 V instead of –30 V to +30 V.
Capacitance Range Estimator (on page 6-84)	The Capacitance Range Estimator automatically calculates the maximum capacitance value based on the parameter settings in the Advanced settings dialog box.

Voltage List Sweep operation mode - CVU

The Voltage List Sweep operation mode allows you to customize the voltage values for each step of the sweep. List Sweeps allow you to make measurements only at selected forced voltages. For example, they allow you to skip unimportant measurement points or to synthesize a custom sweep that is based on a special mathematical equation. You can also use list sweeps to make pulsed measurements to avoid overheating of sensitive devices. The following figure illustrates a possible list sweep.

Figure 246: Example list sweep



The voltage list sweep generates parametric curve data that is recorded in the Analyze pane.

The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Presoak (on page 6-77)	Type a presoak voltage of –30 V to +30 V.
List Values (on page 6-50)	Select Enter Values to open a dialog box in which you can enter the voltage level for each step of the sweep in the rows. You can enter any valid instrument voltage.
Points (on page 6-51)	The number of sweep points that were defined in the List Values list. This number is automatically generated.
Frequency (on page 6-80)	Select the frequency from the list of values.
AC Drive Signal (on page 6-80)	The AC drive signal accepts a value between 10 mV and 100 mV.
Parameters (on page 6-80)	Select the type of parameters. When using any of the tests supplied by Keithley Instruments, leave the measurement option set to `Cp-Gp`.
Param1 Column Name (on page 6-81)	The name that is used for parameter 1 in the Analyze sheet.
Param2 Column Name (on page 6-81)	The name that is used for parameter 2 in the Analyze sheet.
Report Test Conditions (on page 6-81)	Select this option to display the DC bias and drive frequency values on the Analyze sheet.
DCV Column Name (on page 6-81)	Available if you select Report Test Conditions. Determines the name of the column that contains the DC bias information in the Analyze sheet where this data is reported. You can change the name of the column by typing a new value.
Freq Column Name (on page 6-82)	Available if you select Report Test Conditions. Determines the name of the column that contains the frequency information in the Analyze sheet where this data is reported. You can change the name of the column by typing a new value.
Report Status (on page 6-82)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.
Compensation Open (on page 6-82)	Use the CVU connection compensation value that was generated for open connection compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.
Compensation Short (on page 6-82)	Use the CVU connection compensation value that was generated for short connection compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.
Compensation Load (on page 6-83)	Use the CVU connection compensation value that was generated for load compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.
Compensation Cable Length (on page 6-83)	The cable length that was used to generate connection compensation data. Make sure this cable length is the same as the Cable Length setting in the Tools > CVU Connection Compensation dialog box. Refer to Connection compensation (on page 4-14) for additional information.
AC Source V (on page 6-83)	Selects the terminal to use to source AC drive voltage.
AC Measure I Range (on page 6-83)	The measure range determines the full-scale measurement span that is applied to the signal.

Parameter	Description
DC Source V (on page 6-83)	Selects the terminal to use to source DC drive voltage.
DC Offset (on page 6-83)	Allows you offset the voltage by up to ±30 V on one terminal. For example, this allows you to output voltage from 0 V to 60 V instead of –30 V to +30 V.
Capacitance Range Estimator (on page 6-84)	The Capacitance Range Estimator automatically calculates the maximum capacitance value based on the parameter settings in the Advanced settings dialog box.

Frequency Sweep - DC Bias operation mode - CVU

When performing a frequency sweep, the 4210-CVU steps through all the frequency points from start to stop. For example, if the start frequency is 800 kHz and the stop frequency is 3 MHz, the CVU steps through the frequency points 800 kHz, 900 kHz, 1 MHz, 2 MHz, and 3 MHz.

The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

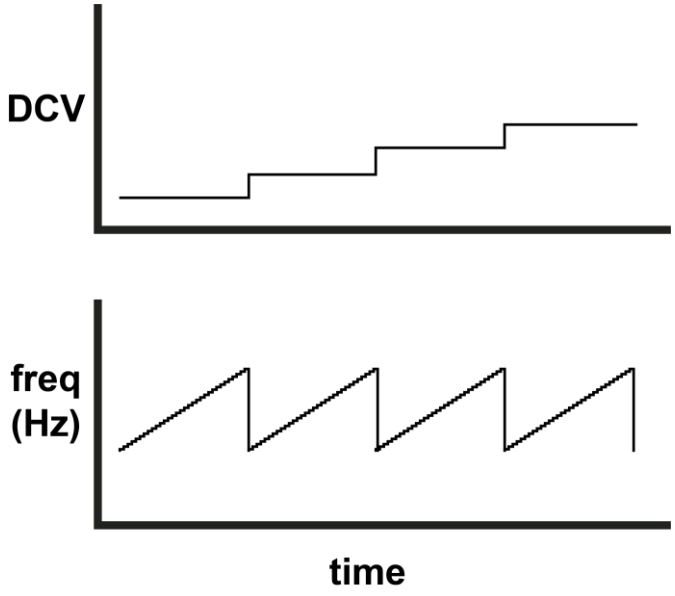
Parameter	Description
Presoak (on page 6-77)	Type a presoak voltage of –30 V to +30 V.
DC Bias (on page 6-80)	The DC bias range accepts a value between –30 V to 30 V.
Start Frequency (on page 6-78)	The frequency at which the sweep starts.
Stop Frequency (on page 6-79)	The frequency at which the sweep stops.
Frequency Points (on page 6-79)	The number of frequency points to sweep. This is calculated from the start and stop frequency values.
AC Drive Signal (on page 6-80)	The AC drive signal accepts a value between 10 mV and 100 mV.
Parameters (on page 6-80)	Select the type of parameters. When using any of the tests supplied by Keithley Instruments, leave the measurement option set to `Cp-Gp`.
Param1 Column Name (on page 6-81)	The name that is used for parameter 1 in the Analyze sheet.
Param2 Column Name (on page 6-81)	The name that is used for parameter 2 in the Analyze sheet.
Report Test Conditions (on page 6-81)	Select this option to display the DC bias and drive frequency values on the Analyze sheet.
DCV Column Name (on page 6-81)	Available if you select Report Test Conditions. Determines the name of the column that contains the DC bias information in the Analyze sheet where this data is reported. You can change the name of the column by typing a new value.
Freq Column Name (on page 6-82)	Available if you select Report Test Conditions. Determines the name of the column that contains the frequency information in the Analyze sheet where this data is reported. You can change the name of the column by typing a new value.
Report Status (on page 6-82)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.
Compensation Open (on page 6-82)	Use the CVU connection compensation value that was generated for open connection compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.
Compensation Short (on page 6-82)	Use the CVU connection compensation value that was generated for short connection compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.
Compensation Load (on page 6-83)	Use the CVU connection compensation value that was generated for load compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.

Parameter	Description
Compensation Cable Length (on page 6-83)	The cable length that was used to generate connection compensation data. Make sure this cable length is the same as the Cable Length setting in the Tools > CVU Connection Compensation dialog box. Refer to Connection compensation (on page 4-14) for additional information.
AC Source V (on page 6-83)	Selects the terminal to use to source AC drive voltage.
AC Measure I Range (on page 6-83)	The measure range determines the full-scale measurement span that is applied to the signal.
DC Source V (on page 6-83)	Selects the terminal to use to source DC drive voltage.
DC Offset (on page 6-83)	Allows you offset the voltage by up to ±30 V on one terminal. For example, this allows you to output voltage from 0 V to 60 V instead of –30 V to +30 V.
Capacitance Range Estimator (on page 6-84)	The Capacitance Range Estimator automatically calculates the maximum capacitance value based on the parameter settings in the Advanced settings dialog box.

Frequency Sweep - DC Step operation mode - CVU

When the 4210-CVU does a frequency sweep with DC Step operation mode selected, the 4210-CVU sweeps through all the frequency points from start to stop for each DC step.

Figure 247: new



For example, if the:

- Start frequency is 800 kHz
- Stop frequency is 3 MHz
- DC start value is -1
- Stop value is 1
- Step value is 0.5

The CVU sweeps through the frequency points 800 kHz, 900 kHz, 1 MHz, 2 MHz, and 3 MHz for DC biases -1 V, -0.5 V, 0 V, 0.5 V, and 1 V.

When this test is run, the following sequence occurs:

1. The DC source goes to the Presoak voltage.
2. After the hold time (set in the Test Settings), DC bias goes to the Start voltage.
3. After the delays, the 4210-CVU makes a measurement for the Start Frequency point. The AC signal is applied before the start of the measurement.
4. The 4210-CVU sweeps through the frequency points until it reaches the Stop Frequency point.
5. DC bias goes to the next voltage as defined by the Step value.
6. Steps 4 and 5 are repeated until the DC bias reaches the Stop value.

The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

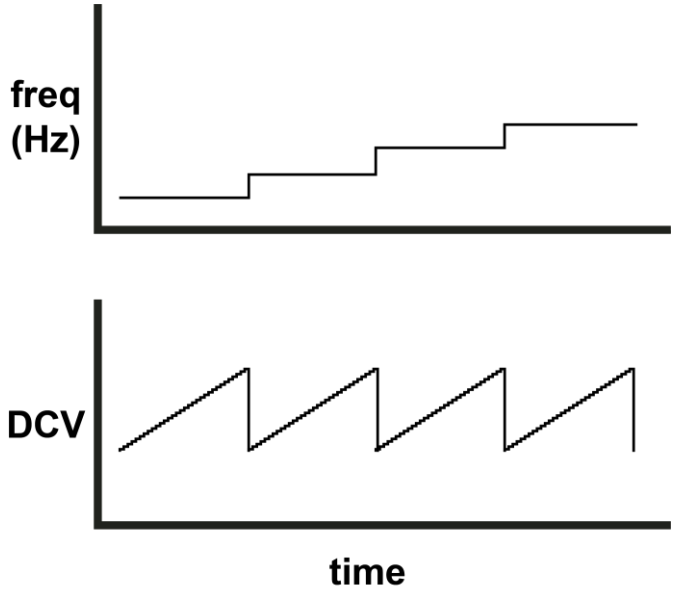
Parameter	Description
Presoak (on page 6-77)	Type a presoak voltage of –30 V to +30 V.
Start (on page 6-77)	The voltage source level at which the step starts.
Stop	The voltage source level at which the step stops.
Step (on page 6-49)	The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).
Points (on page 6-50)	The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters.
Start Frequency (on page 6-78)	The frequency at which the sweep starts.
Stop Frequency (on page 6-79)	The frequency at which the sweep stops.
Frequency Points (on page 6-79)	The number of frequency points to sweep. This is calculated from the start and stop frequency values.
AC Drive Signal (on page 6-80)	The AC drive signal accepts a value between 10 mV and 100 mV.
Parameters (on page 6-80)	Select the type of parameters. When using any of the tests supplied by Keithley Instruments, leave the measurement option set to `Cp-Gp`.
Param1 Column Name (on page 6-81)	The name that is used for parameter 1 in the Analyze sheet.
Param2 Column Name (on page 6-81)	The name that is used for parameter 2 in the Analyze sheet.
Report Test Conditions (on page 6-81)	Select this option to display the DC bias and drive frequency values on the Analyze sheet.
DCV Column Name (on page 6-81)	Available if you select Report Test Conditions. Determines the name of the column that contains the DC bias information in the Analyze sheet where this data is reported. You can change the name of the column by typing a new value.
Freq Column Name (on page 6-82)	Available if you select Report Test Conditions. Determines the name of the column that contains the frequency information in the Analyze sheet where this data is reported. You can change the name of the column by typing a new value.
Report Status (on page 6-82)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.
Compensation Open (on page 6-82)	Use the CVU connection compensation value that was generated for open connection compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.
Compensation Short (on page 6-82)	Use the CVU connection compensation value that was generated for short connection compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.
Compensation Load (on page 6-83)	Use the CVU connection compensation value that was generated for load compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.

Parameter	Description
Compensation Cable Length (on page 6-83)	The cable length that was used to generate connection compensation data. Make sure this cable length is the same as the Cable Length setting in the Tools > CVU Connection Compensation dialog box. Refer to Connection compensation (on page 4-14) for additional information.
AC Source V (on page 6-83)	Selects the terminal to use to source AC drive voltage.
AC Measure I Range (on page 6-83)	The measure range determines the full-scale measurement span that is applied to the signal.
DC Source V (on page 6-83)	Selects the terminal to use to source DC drive voltage.
DC Offset (on page 6-83)	Allows you offset the voltage by up to ±30 V on one terminal. For example, this allows you to output voltage from 0 V to 60 V instead of –30 V to +30 V.
Capacitance Range Estimator (on page 6-84)	The Capacitance Range Estimator automatically calculates the maximum capacitance value based on the parameter settings in the Advanced settings dialog box.

DC Sweep - Frequency Step operation mode - CVU

When the 4210-CVU performs a DC voltage sweep with Frequency Step operation mode selected, the 4210-CVU sweeps through all DC voltage points from start to stop for each frequency step.

Figure 248: Frequency Sweep - DC Step operation mode



For example, if the:

- Start frequency is 800 kHz
- Stop frequency is 3 MHz
- DC start value is -1
- Stop value is 1
- Step value is 0.5

The CVU sweeps through the voltage points -1 V, -0.5 V, 0 V, 0.5 V, and 1 V for frequency points 800 kHz, 900 kHz, 1 MHz, 2 MHz, and 3 MHz.

When this test is run, the following sequence occurs:

1. The DC source goes to the Presoak voltage.
2. After the hold time (set in the Test Settings), DC source goes to the Start voltage.
3. After the delays, the 4210-CVU makes a measurement for the Start voltage point. The AC signal is applied before the start of the measurement.
4. The 4210-CVU sweeps through the DC bias points until it reaches the Stop voltage point.
5. The frequency steps to the next available frequency. See Test signal (on page 4-4) for the available frequencies.
6. Steps 4 and 5 are repeated until the frequency reaches the Stop value.

The parameters that are available for this mode are briefly described in the following table. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Presoak (on page 6-77)	Type a presoak voltage of –30 V to +30 V.
Start (on page 6-77)	The voltage source level at which the step starts.
Stop	The voltage source level at which the step stops.
Step (on page 6-49)	The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).
Points (on page 6-50)	The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters.
Start Frequency (on page 6-78)	The frequency at which the sweep starts.
Stop Frequency (on page 6-79)	The frequency at which the sweep stops.
Frequency Points (on page 6-79)	The number of frequency points to sweep. This is calculated from the start and stop frequency values.
AC Drive Signal (on page 6-80)	The AC drive signal accepts a value between 10 mV and 100 mV.
Parameters (on page 6-80)	Select the type of parameters. When using any of the tests supplied by Keithley Instruments, leave the measurement option set to `Cp-Gp`.
Param1 Column Name (on page 6-81)	The name that is used for parameter 1 in the Analyze sheet.
Param2 Column Name (on page 6-81)	The name that is used for parameter 2 in the Analyze sheet.
Report Test Conditions (on page 6-81)	Select this option to display the DC bias and drive frequency values on the Analyze sheet.
DCV Column Name (on page 6-81)	Available if you select Report Test Conditions. Determines the name of the column that contains the DC bias information in the Analyze sheet where this data is reported. You can change the name of the column by typing a new value.
Freq Column Name (on page 6-82)	Available if you select Report Test Conditions. Determines the name of the column that contains the frequency information in the Analyze sheet where this data is reported. You can change the name of the column by typing a new value.
Report Status (on page 6-82)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information.
Compensation Open (on page 6-82)	Use the CVU connection compensation value that was generated for open connection compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.
Compensation Short (on page 6-82)	Use the CVU connection compensation value that was generated for short connection compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.
Compensation Load (on page 6-83)	Use the CVU connection compensation value that was generated for load compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.

Parameter	Description
Compensation Cable Length (on page 6-83)	The cable length that was used to generate connection compensation data. Make sure this cable length is the same as the Cable Length setting in the Tools > CVU Connection Compensation dialog box. Refer to Connection compensation (on page 4-14) for additional information.
AC Source V (on page 6-83)	Selects the terminal to use to source AC drive voltage.
AC Measure I Range (on page 6-83)	The measure range determines the full-scale measurement span that is applied to the signal.
DC Source V (on page 6-83)	Selects the terminal to use to source DC drive voltage.
DC Offset (on page 6-83)	Allows you offset the voltage by up to ±30 V on one terminal. For example, this allows you to output voltage from 0 V to 60 V instead of –30 V to +30 V.
Capacitance Range Estimator (on page 6-84)	The Capacitance Range Estimator automatically calculates the maximum capacitance value based on the parameter settings in the Advanced settings dialog box.

DC Gnd operation mode - CVU

Select the DC Gnd operation mode for the CVU.

CVU - all terminal parameters

When you select **All Parameters**, the Configure pane displays all available parameters for the test that is selected in the project tree.

Parameter descriptions are provided in the following sections.

Presoak

Type a presoak voltage of –30 V to +30 V.

Start

The voltage source level at which the sweep starts.

Stop (PMU Amplitude Sweep)

The voltage source level at which the sweep stops.

Step (voltage sweep)

The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).

Clarius 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.

Use a step value that does not result in a fractional number of data points. If the data point is fractional, the step value is forced to a value that results in a whole number of data points. To calculate the data points:

$$data\ points = \frac{(stop - start)}{step} + 1$$

The data points are rounded to the nearest value.

For example, if Start = 0 V, Stop = 5 V, and Step = 0.6 V:

$$data\ points = \frac{(5 - 0)}{0.6} + 1$$

In this case, the Step value is forced to 0.625 V, which results in a data point value of 9.333, which is rounded to 9. The instrument forces nine voltages at 0 V, 0.625 V, 1.25 V, 1.875 V, 2.5 V, 3.125 V, 3.75 V, 4.375 V, and 5 V.

List Values

Select **Enter Values** to open a dialog box in which you can type the current or voltage level for each step of the sweep in the rows. You can type any valid instrument current or voltage.

You can select a value or multiple values in the list and copy, cut, or delete them. Use Paste to add values that you copied or cut to a new location in the list.

Note that you cannot have blank rows in between values.

You can use the Ctrl key plus mouse selections to pick selected rows, then use the buttons to copy, cut, or delete those rows. Note that when you paste the rows, any skipped rows are ignored (there will be no blank rows).

You can use the Shift key plus mouse selections to pick a range of rows. When you paste the rows, any blank rows are ignored.

Start Frequency

The frequency at which the sweep starts.

Stop Frequency

The frequency at which the sweep stops.

Frequency Points

The number of frequency points to sweep. This is calculated from the start and stop frequency values.

Points

The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters, using the equation:

$$data\ points = \frac{(stop - start)}{step} + 1$$

Points (list sweep)

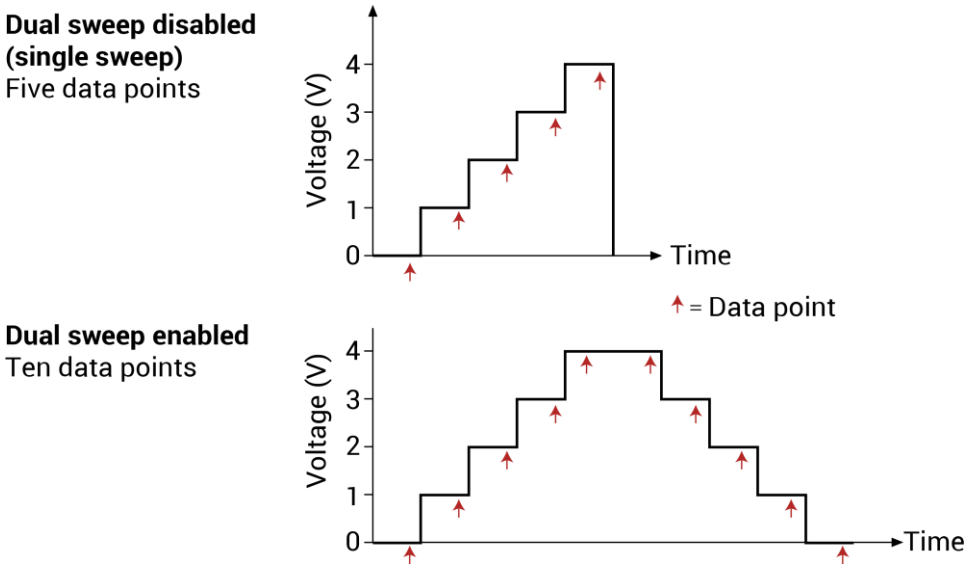
The number of sweep points that were defined in the List Values list. This number is automatically generated.

Dual Sweep

When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.

The following figure compares a single sweep to a dual sweep.

Figure 249: Single and dual sweep examples (linear voltage sweep; 0 V to 4 V in 1 V steps)



DC Bias

The DC bias range accepts a value between −30 V to 30 V.

Frequency

Select the frequency from the list.

For sweeps, the AC drive conditions include frequency (Hz) and voltage (mV_{RMS}). You can set frequency to the following values:

- 1 kHz through 10 kHz in 1 kHz steps
- 10 kHz to 90 kHz in 10 kHz steps
- 100 kHz to 900 kHz in 100 kHz steps
- 1 MHz to 10 MHz in 1 MHz steps

AC Drive Signal

The AC drive signal accepts a value between 10 mV and 100 mV.

Parameters

CAUTION

If you change the Parameters, it will change the Column names, which can cause Formulator functions to be erased. When using any of the tests or libraries supplied by Keithley Instruments, leave the measurement option set to Cp-Gp.

You can select the following measurement options:

- **Z, Theta:** Impedance and phase angle (degrees)
- **R+jX:** Resistance and reactance
- **Cp-Gp:** Parallel capacitance and conductance
- **Cs-Rs:** Series capacitance and resistance
- **Cp-D:** Parallel capacitance and dissipation factor
- **Cs-D:** Series capacitance and dissipation factor

Param1 Column Name

The name that is used for parameter 1 in the Analyze sheet.

If you do not define a name, Clarius assigns a name. The assigned name is a combination of the parameter name and the terminal label.

This cannot be left blank. If it is left blank, Clarius uses the previous value.

If you change the name, it is updated when the test is run. The results from earlier tests that are available through Run History are not changed to the new name.

Param2 Column Name

The name that is used for parameter 2 in the Analyze sheet.

If you do not define a name, Clarius assigns a name. The assigned name is a combination of the parameter name and the terminal label.

This cannot be left blank. If it is left blank, Clarius uses the previous value.

If you change the name, it is updated when the test is run. The results from earlier tests that are available through Run History are not changed to the new name.

Report Test Conditions

If you select the Report Test Conditions option, the DC bias voltage and drive frequency values that were used for the test are displayed on the Analyze sheet.

When this option is selected, column name fields are available. These are the names of the columns in the Analyze sheet where this data is reported. You can change the names of the columns by entering new values for DCV Column Name and Freq Column Name.

DCV Column Name

Available if you select Report Test Conditions. Determines the name of the column that contains the DC bias information in the Analyze sheet where this data is reported. You can change the name of the column by typing a new value.

This cannot be left blank. If it is left blank, Clarius uses the previous value.

If you change the name, it is updated when the test is run. The results from earlier tests that are available through Run History are not changed to the new name.

Freq Column Name

Available if you select Report Test Conditions. Determines the name of the column that contains the frequency information in the Analyze sheet where this data is reported. You can change the name of the column by typing a new value.

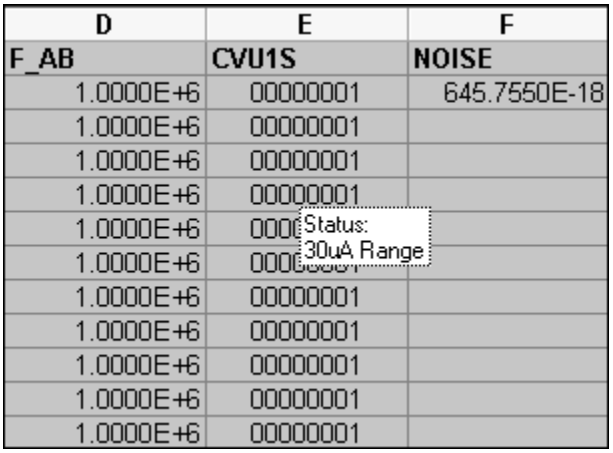
This cannot be left blank. If it is left blank, Clarius uses the previous value.

If you change the name, it is updated when the test is run. The results from earlier tests that are available through Run History are not changed to the new name.

Report Status (CVU)

When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information. An example of the status information is displayed in the following figure.

Figure 250: Report Status column for CVU in Analyze sheet



Open

Use the CVU connection compensation value that was generated for open connection compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.

Short

Use the CVU connection compensation value that was generated for short connection compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.

Load

Use the CVU connection compensation value that was generated for load compensation. Refer to Connection compensation (on page 4-14) for information on generating and using compensation values.

Cable Length

The cable length that was used to generate connection compensation data. Make sure this cable length is the same as the Cable Length setting in the Tools > CVU Connection Compensation dialog box. Refer to Connection compensation (on page 4-14) for additional information.

AC Source V

Selects the terminal to use to source AC drive voltage.

AC Measure I Range

The measure range determines the full-scale measurement span that is applied to the signal. Therefore, it affects both the accuracy of the measurements and the maximum signal that can be measured.

The current range options are:

- **Auto:** The instrument automatically optimizes the measurement range as the test progresses. This option provides the best resolution when the measurements span several decades. However, time delays can occur with range changes that can limit the measurement speed.
- **Specific ranges:** You can select a fixed measurement range.

DC Source V

Selects the terminal to use to source DC drive voltage.

DC Offset

Allows you offset the voltage by up to ±30 V on one terminal. For example, this allows you to output voltage from 0 V to 60 V instead of –30 V to +30 V.

Capacitance Range Estimator

The measurement accuracy of the 4210-CVU is specified up to 100 nF. However, the CVU can make much higher capacitance measurements. The maximum capacitance is based on the test frequency, AC drive signal, and range.

The Capacitance Range Estimator automatically calculates the maximum capacitance value based on the parameter settings. It shows the maximum capacitance values in the mF ranges. Note that capacitance values above 100 nF are not tested or calibrated.

In general, to measure high capacitances, use the lowest test frequency (1 kHz), the smallest AC drive voltage (10 mV_{RMS}), and the maximum current range (1 mA or autorange).

To use the Capacitance Range Estimator:

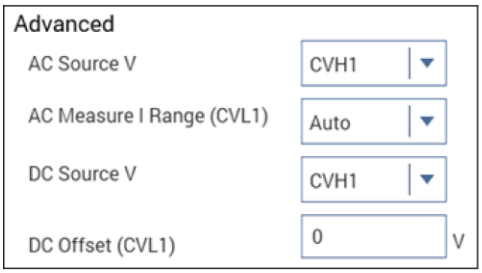
1. Select **Configure**.
2. In the Key Parameters pane, click the CVH1 terminal.
3. Select the **Terminal Settings** pane.
4. Select **Advanced**.
5. Change the parameters as needed to view the maximum capacitor value for those settings.

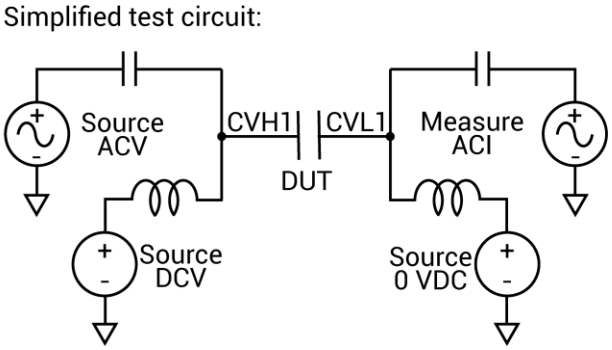
CVU Terminal Settings Advanced settings and circuits

You can apply AC drive voltage and DC bias voltage to either the CVH1 terminal or the CVL1 terminal. To change the options, select the **Terminals Settings** pane, then select **Advanced.**

By default, AC source voltage is applied to the CVH1 terminal and the current measurement is made at the CVL1 terminal. Also by default, the DC source voltage (bias) is applied to the CVL1 terminal. The settings and test circuits are shown in the following figures.

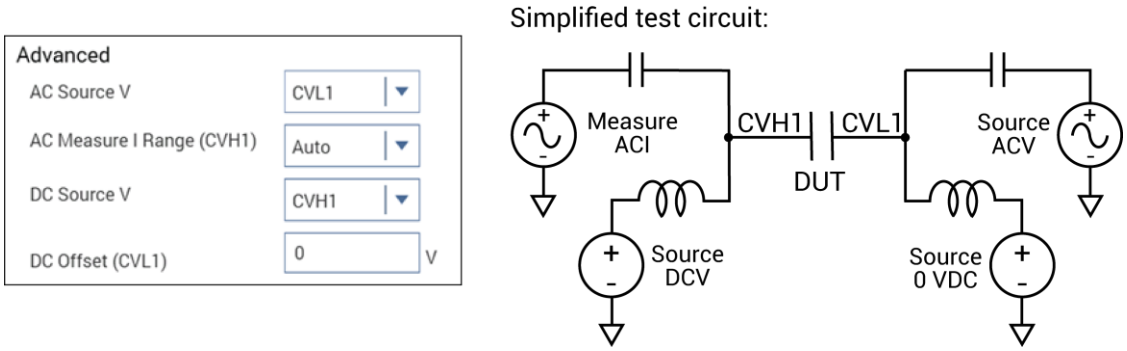
Figure 251: AC source and DC source applied to CVH1





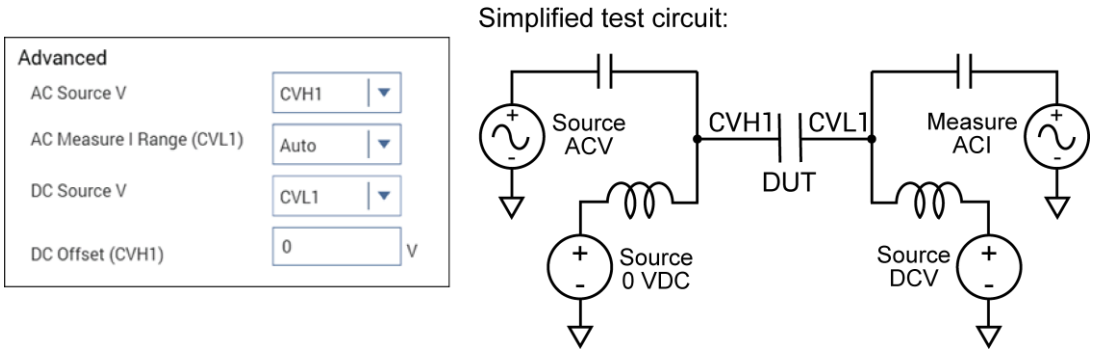
You can change the configuration to source AC voltage to CVL1 and DC source voltage to CVH1 while measuring AC current at CVH1, as shown in the following figure.

Figure 252: AC source applied to CVL1 and DC source applied to CVH1



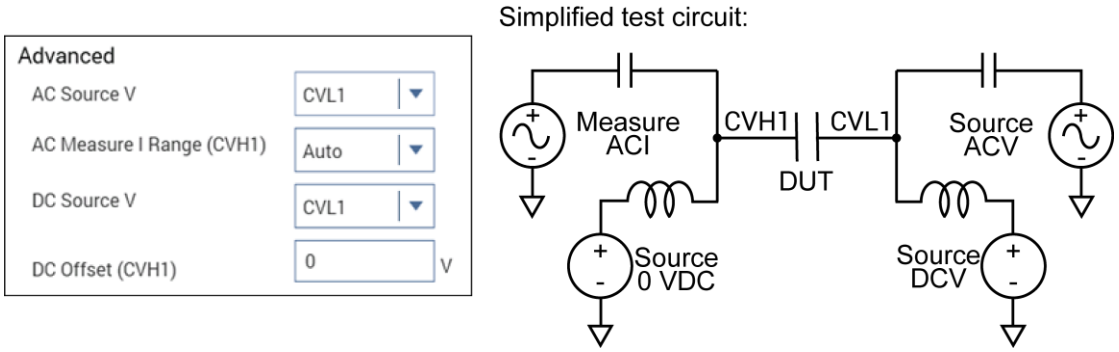
Another configuration sources AC drive voltage to the CVH1 terminal and sources DC bias voltage to the CVL1 terminal. AC current is measured at CVL1.

Figure 253: ACV source applied to CVH1 and DC bias applied to CVL1



The following settings source AC drive voltage and DC bias voltage to the CVL1 terminal. AC current is measured at CVH1.

Figure 254: ACV source and DCV bias applied to CVL1



Operation Mode (PMU)

The operation mode determines what type of test is run on the terminal. Selecting the appropriate mode simplifies configuration options.

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.

A Pulse Train outputs one or more pulses of the same magnitude and base level.

When there are two or more PMUs in the test system, the Sweep Master (on page 6-139) option is used to select one of them as the master PMU.

Figure 255: Pulse operation modes

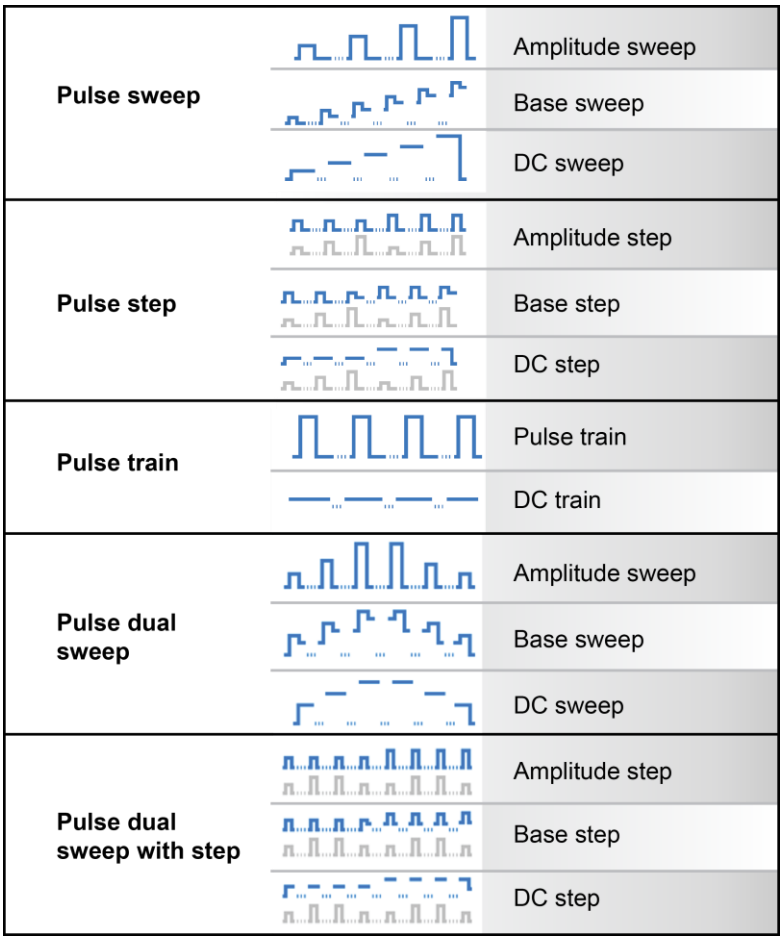
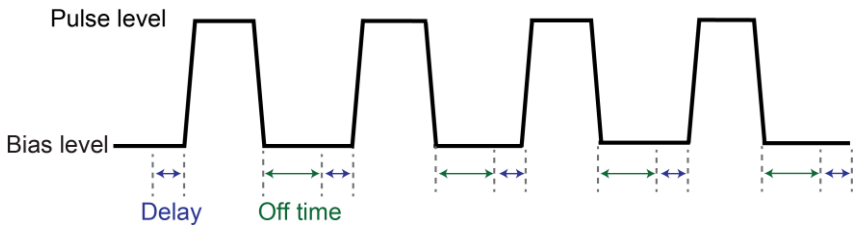


Figure 256: Pulse train



The PMU operation modes are described in the following sections.

Pulse Amplitude Sweep operation mode - PMU

The pulse amplitude sweep generates data that is recorded in the Analyze pane.

The parameters that are specific to this mode are briefly described in the following table. Select the links to access additional information. Additional parameters are described in Parameters common to PMU operation modes (on page 6-94). The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Start (on page 6-48)	The voltage source level at which the sweep starts.
Stop (on page 6-77)	The voltage source level at which the sweep stops.
Step (on page 6-49)	The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).
Points (on page 6-50)	The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters.
Base (on page 6-96)	The voltage offset (from 0 V) that is the reference for the pulse amplitude.
Dual Sweep (on page 6-96)	When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.
Force Range (on page 6-97)	The range that is used when sourcing. Select one of the listed ranges. The source remains on the range that is set. If you are sweeping and a sweep point exceeds the source range capability, the source outputs the maximum level for that range. This range must be equal to or greater than the largest value in the sweep.

Pulse Base Sweep operation mode - PMU

The pulse base sweep generates data that is recorded in the Analyze pane.

The parameters that are specific to this mode are briefly described in the following table. Select the links to access additional information. Additional parameters are described in Parameters common to PMU operation modes (on page 6-94). The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Start (on page 6-48)	The voltage source level at which the sweep starts.
Stop (on page 6-77)	The voltage source level at which the sweep stops.
Step (on page 6-49)	The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).
Points (on page 6-50)	The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters.
Amplitude (on page 6-96)	The amplitude voltage.
Dual Sweep (on page 6-96)	When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.
Force Range (on page 6-97)	The range that is used when sourcing. Select one of the listed ranges. The source remains on the range that is set. If you are sweeping and a sweep point exceeds the source range capability, the source outputs the maximum level for that range. This range must be equal to or greater than the largest value in the sweep.

DC Sweep operation mode - PMU

The DC sweep generates data that is recorded in the Analyze pane.

The parameters that are specific to this mode are briefly described in the following table. Select the links to access additional information. Additional parameters are described in Parameters common to PMU operation modes (on page 6-94). The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Start (on page 6-48)	The voltage source level at which the sweep starts.
Stop (on page 6-77)	The voltage source level at which the sweep stops.
Step (on page 6-49)	The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).
Points (on page 6-50)	The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters.
Dual Sweep (on page 6-96)	When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.
Force Range (on page 6-97)	The range that is used when sourcing. Select one of the listed ranges. The source remains on the range that is set. If you are sweeping and a sweep point exceeds the source range capability, the source outputs the maximum level for that range. This range must be equal to or greater than the largest value in the sweep.

Pulse Amplitude Step operation mode - PMU

Tests set to the pulse amplitude step operation mode generate data that is recorded in the Analyze pane.

The parameters that are specific to this mode are briefly described in the following table. Select the links to access additional information. Additional parameters are described in Parameters common to PMU operation modes (on page 6-94). The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Start (on page 6-48)	The voltage source level at which the sweep starts.
Stop (on page 6-77)	The voltage source level at which the sweep stops.
Step (on page 6-49)	The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).
Points (on page 6-50)	The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters.
Base (on page 6-96)	The voltage offset (from 0 V) that is the reference for the pulse amplitude.
Dual Sweep (on page 6-96)	When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.
Force Range (on page 6-97)	The range that is used when sourcing. Select one of the listed ranges. The source remains on the range that is set. If you are sweeping and a sweep point exceeds the source range capability, the source outputs the maximum level for that range. This range must be equal to or greater than the largest value in the sweep.

Pulse Base Step operation mode - PMU

The pulse base step operation mode generates data that is recorded in the Analyze pane.

The parameters that are specific to this mode are briefly described in the following table. Select the links to access additional information. Additional parameters are described in Parameters common to PMU operation modes (on page 6-94). The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Start (on page 6-48)	The voltage source level at which the sweep starts.
Stop (on page 6-77)	The voltage source level at which the sweep stops.
Step (on page 6-49)	The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).
Points (on page 6-50)	The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters.
Amplitude (on page 6-96)	The amplitude voltage.
Dual Sweep (on page 6-96)	When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.
Force Range (on page 6-97)	The range that is used when sourcing. Select one of the listed ranges. The source remains on the range that is set. If you are sweeping and a sweep point exceeds the source range capability, the source outputs the maximum level for that range. This range must be equal to or greater than the largest value in the sweep.

DC Step operation mode - PMU

The DC step operation mode generates data that is recorded in the Analyze pane.

The parameters that are specific to this mode are briefly described in the following table. Select the links to access additional information. Additional parameters are described in Parameters common to PMU operation modes (on page 6-94). The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Start (on page 6-48)	The voltage source level at which the sweep starts.
Stop (on page 6-77)	The voltage source level at which the sweep stops.
Step (on page 6-49)	The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).
Points (on page 6-50)	The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters.
Dual Sweep (on page 6-96)	When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.
Force Range (on page 6-97)	The range that is used when sourcing. Select one of the listed ranges. The source remains on the range that is set. If you are sweeping and a sweep point exceeds the source range capability, the source outputs the maximum level for that range. This range must be equal to or greater than the largest value in the sweep.

Pulse Train operation mode - PMU

Pulse Train operation mode outputs a number of identical pulses.

The pulse train operation mode generates data that is recorded in the Analyze pane.

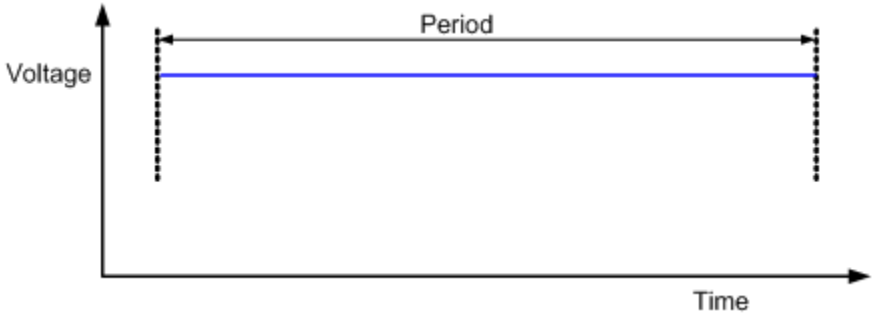
The parameters that are specific to this mode are briefly described in the following table. Select the links to access additional information. Additional parameters are described in Parameters common to PMU operation modes (on page 6-94). The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Base (on page 6-96)	The voltage offset (from 0 V) that is the reference for the pulse amplitude.
Amplitude (on page 6-96)	The amplitude voltage.
Force Range (on page 6-97)	The range that is used when sourcing. Select one of the listed ranges. The source remains on the range that is set. If you are sweeping and a sweep point exceeds the source range capability, the source outputs the maximum level for that range. This range must be equal to or greater than the largest value in the sweep.

DC Bias - PMU

When DC Bias operation mode is selected, the PMU outputs the DC base voltage. The next figure shows a representation of a DC voltage waveform.

Figure 257: DC Bias waveform



The DC Bias operation mode generates data that is recorded in the Analyze pane.

The parameters that are specific to this mode are briefly described in the following table. Select the links to access additional information. Additional parameters are described in Parameters common to PMU operation modes (on page 6-94). The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Base (on page 6-96)	The voltage offset (from 0 V) that is the reference for the pulse amplitude.
Force Range (on page 6-97)	The range that is used when sourcing. Select one of the listed ranges. The source remains on the range that is set. If you are sweeping and a sweep point exceeds the source range capability, the source outputs the maximum level for that range. This range must be equal to or greater than the largest value in the sweep.

Parameters common to PMU operation modes

The following parameters are common to all PMU operation modes except DC Gnd. Brief descriptions are provided here. Select the links to access additional information. The parameters are listed in the order in which they appear in the All Parameters pane.

Parameter	Description
Disable Outputs at Completion (on page 6-97)	When a test is complete, you can disable the outputs or leave them enabled.
Current Spot Mean High (on page 6-97)	Makes current measurements on the amplitude. Available when the Test Mode is set to Pulse I-V.
Current Spot Mean Low (on page 6-98)	Makes current measurements on the base level. Available when the Test Mode is set to Pulse IV.
Measure Current Range (on page 6-98)	The measure range determines the full-scale measurement span that is applied to the signal.
Low Range (on page 6-58)	Available when the Measure Range is set to Limited Auto. This sets the minimum range that the instrument uses.
Current Sample Waveform (on page 6-100)	Makes current measurements on the waveform. Available when the Test Mode is set to Waveform Capture.
Voltage Spot Mean High (on page 6-100)	Makes voltage measurements on the amplitude. Available when the Test Mode is set to Pulse IV.
Voltage Spot Mean Low (on page 6-100)	Makes voltage measurements on the base level. Available when the Test Mode is set to Pulse IV.
Voltage Sample Waveform (on page 6-100)	Makes voltage measurements on the waveform. Available when the Test Mode is set to Waveform Capture.
Report Timestamps (on page 6-100)	Available when the Test Mode is set to Waveform Capture. When Report Timestamps is enabled, each measurement includes a timestamp and the waveform is graphed (voltage/current versus time) in the Analyze graph.
Report Status (on page 6-101)	When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information. For PMU measurements, this information is also available as a 32-bit word. The status code bit map is listed in pulse_fetch (on page 14-110).
Compensation Short Connection (on page 6-105)	Use to enable or disable short connection compensation. Refer to PMU connection compensation (on page 5-40) for detail on setting up and using connection compensation.
Compensation Load Line Effect (on page 6-105)	Use to enable or disable load-line effect compensation (LLEC). Refer to Load-line effect compensation (LLEC) for the PMU (on page 5-44) for detail on setting up and using load-line effect compensation.
Compensation DUT Resistance (on page 6-105)	The DUT resistance value. Refer to DUT resistance determines pulse voltage across DUT (on page 5-80) for detail on calculating DUT resistance.
Max Voltage Estimator (on page 6-105)	Only available in the Advanced Terminal Settings dialog box. The Maximum Voltage Estimator is a tool that enables you to calculate the maximum voltage and current based on the selected voltage range and the DUT resistance value you type in. The estimator does not affect the pulse output.
Voltage Threshold (on page 6-106)	The voltage threshold allows you to set a voltage threshold. If the threshold is reached or exceeded, the present sweep is stopped. Testing continues with any subsequent sweeps.
Current Threshold (on page 6-105)	The current threshold allows you to set a threshold. If the threshold is reached or exceeded, the present sweep is stopped. Testing continues with any subsequent sweeps.

Parameter	Description
Power Threshold (on page 6-106)	The power threshold allows you to set a power threshold. If the threshold is reached or exceeded, the present sweep is stopped. Testing continues with any subsequent sweeps.

DC Gnd operation mode - PMU

The operation mode for the PMU GND terminal.

PMU - all terminal parameters

When you select **All Parameters**, the Configure pane displays all available parameters for the test that is selected in the project tree.

Parameter descriptions are provided in the following sections.

Start (PMU Amplitude Sweep)

The voltage source level at which the sweep starts.

Stop (PMU Amplitude Sweep)

The voltage source level at which the sweep stops.

Step (Pulse Amplitude Sweep)

The voltage size of each step of the sweep. The source level changes in equal steps of this size from the start level to the stop level. A measurement is made at each source step (including the start and stop levels).

Clarius 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.

Use a step value that does not result in a fractional number of data points. If the data point is fractional, the step value is forced to a value that results in a whole number of data points. To calculate the data points:

$$data\ points = \frac{(stop - start)}{step} + 1$$

The data points are rounded to the nearest value.

For example, if Start = 0 V, Stop = 5 V, and Step = 0.6 V:

$$data\ points = \frac{(5 - 0)}{0.6} + 1$$

In this case, the Step value is forced to 0.625 V, which results in a data point value of 9.333, which is rounded to 9. The instrument forces nine voltages at 0 V, 0.625 V, 1.25 V, 1.875 V, 2.5 V, 3.125 V, 3.75 V, 4.375 V, and 5 V.

Points

The number of data points that will be measured. This value is calculated by Clarius using the information entered for the Start, Stop, and Step parameters, using the equation:

$$data\ points = \frac{(stop - start)}{step} + 1$$

Base

The voltage offset (from 0 V) that is the reference for the pulse amplitude.

Amplitude

The amplitude voltage.

Dual Sweep (pulse)

When you select Dual Sweep, the instrument sweeps from start to stop, then from stop to start. When you clear Dual Sweep, the instrument sweeps from start to stop only.

The amplitude, base, and DC sweeps have an option for Dual Sweep. The figure below shows the difference between a single pulse sweep and a dual pulse sweep. This figure illustrates a linear voltage sweep; 0 V to 4 V in 1 V steps.

Figure 258: Single and dual pulse amplitude sweep examples

Force Range (PMU)

The range that is used when sourcing. Select one of the listed ranges. The source remains on the range that is set. If you are sweeping and a sweep point exceeds the source range capability, the source outputs the maximum level for that range. This range must be equal to or greater than the largest value in the sweep.

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.

Disable outputs at completion

When a test is complete, you can disable the outputs or leave them enabled.

To set output action on completion:

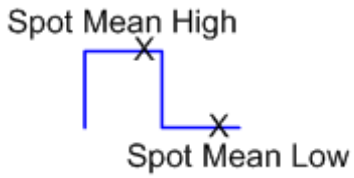
1. Select the test.
2. Select **Configure**.
3. In the right pane, select the **Test Settings** tab.
4. Select **Advanced**.
5. Select or clear **Disable Outputs at Completion**.

Current Spot Mean High

Makes current measurements on the amplitude. Available when the Test Mode is set to Pulse I-V.

The next figure shows an example of a spot mean measurement on pulse high (amplitude) and pulse low (base level). For a more detailed example, refer to Test Mode (PMU) (on page 6-130).

Figure 259: Spot mean measurements

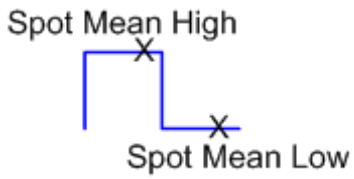


Current Spot Mean Low

Makes current measurements on the base level. Available when the Test Mode is set to Pulse IV.

The next figure shows an example of a spot mean measurement on pulse high (amplitude) and pulse low (base level). For a more detailed example, refer to Test Mode (PMU) (on page 6-130).

Figure 260: Spot mean measurements



Measure Current Range (PMU)

The measure range determines the full-scale measurement span that is applied to the signal. Therefore, it affects both the accuracy of the measurements and the maximum signal that can be measured.

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, some ranges will be unavailable.

As shown in the table below, 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.

Fixed current measurement ranges (PMU only)

10 V range	40 V range
10 mA	100 µA
200 mA	10 mA
–	800 mA

If you are using an RPM, additional current measurement ranges become available to the PMU. The next table 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.

Current measurement ranges for the PMU with RPM

10 V range	40 V range
100 nA	100 µA

10 V range	40 V range
1 µA	10 mA
10 µA	800 mA
100 µA	–
1 mA	–
10 mA	–
200 mA	–

The current range options are:

- **Auto:** The instrument automatically optimizes the measurement range as the test progresses. This option provides the best resolution when the measurements span several decades. However, time delays can occur with range changes that can limit the measurement speed.
- **Limited Auto:** A compromise between Auto and a fixed range option. It allows you to specify the minimum range that the PMU uses when it automatically optimizes the current measurements. This option reduces test time when you do not need maximum resolution at minimum currents. The available ranges are limited by the chosen pulse timing parameters. For additional information, see PMU minimum settling times versus current measure range.
- **Specific ranges:** You can select a fixed measurement range.

The available ranges are limited by the pulse timing parameters. For additional information, see PMU minimum settling times versus current measure range (on page 5-52).

Low Range

Available when the Measure Range is set to Limited Auto. This sets the minimum range that the instrument uses.

Current Sample Waveform

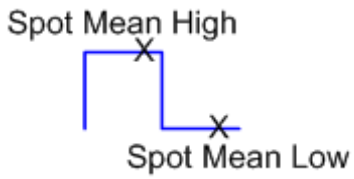
Makes current measurements on the waveform. Available when the Test Mode is set to Waveform Capture.

Voltage Spot Mean High

Makes voltage measurements on the amplitude. Available when the Test Mode is set to Pulse IV.

The next figure shows an example of a spot mean measurement on pulse high (amplitude) and pulse low (base level). For a more detailed example, refer to Test Mode (PMU) (on page 6-130).

Figure 261: Spot mean measurements

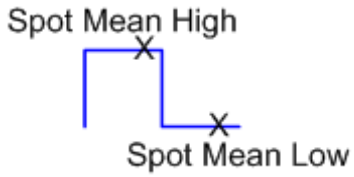


Voltage Spot Mean Low

Makes voltage measurements on the base level. Available when the Test Mode is set to Pulse IV.

The next figure shows an example of a spot mean measurement on pulse high (amplitude) and pulse low (base level). For a more detailed example, refer to Test Mode (PMU) (on page 6-130).

Figure 262: Spot mean measurements



Voltage Sample Waveform

Makes voltage measurements on the waveform. Available when the Test Mode is set to Waveform Capture.

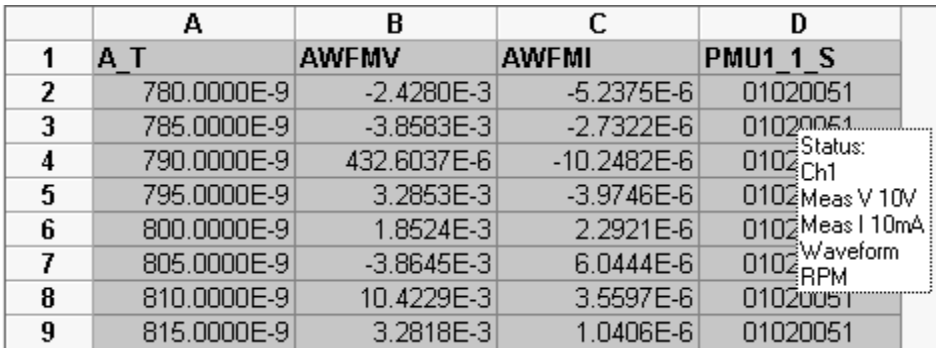
Report Timestamps

Available when the Test Mode is set to Waveform Capture. When Report Timestamps is enabled, each measurement includes a timestamp and the waveform is graphed (voltage/current versus time) in the Analyze graph.

Report Status (PMU)

When this option is selected, Clarius records measurement status information when the test executes. A column of the Analyze spreadsheet displays this information. Hover over a cell to review the information. An example of the status information is displayed in the following figure.

Figure 263: Report Status for PMU in the Analyze sheet



	A	B	C	D
1	A_T	AWFMV	AWFMI	PMU1_1_S
2	780.0000E-9	-2.4280E-3	-5.2375E-6	01020051
3	785.0000E-9	-3.8583E-3	-2.7322E-6	01020051
4	790.0000E-9	432.6037E-6	-10.2482E-6	0102
5	795.0000E-9	3.2853E-3	-3.9746E-6	0102
6	800.0000E-9	1.8524E-3	2.2921E-6	0102
7	805.0000E-9	-3.8645E-3	6.0444E-6	0102
8	810.0000E-9	10.4229E-3	3.5597E-6	01020051
9	815.0000E-9	3.2818E-3	1.0406E-6	01020051

For PMU measurements, this information is also available as a 32-bit word. The status code bit map is listed in pulse_fetch (on page 14-110).

PMU measurement status

ITMs can provide status information for the 4225-PMU measurements in the Analyze sheet Run tab. The column for the status codes is labeled PMU*x*_*y*_S, where *x* is the PMU instrument number (PMU1, PMU2, 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 load-line effect compensation (LLEC) status, and flags any faults (errors).

If a pulse measurement fault occurs, the data values in the entire row are displayed in a different color. A single measurement may have more than one condition. Hover over a cell to review the status information.

The data values identify the fault type as follows:

- Red = Source in compliance
- Magenta = Measurement overflow
- Orange = LLEC failed
- Blue = Current, voltage, or power threshold reached

Figure 264: Status tab showing faults

	BSMVHI	BSMIHI	ASMVHI	PMU1-1_S	CURRENT	RESISTANCE
12	-12.9475E-3	-115.7772E-9	10.9358E+0	42014075	115.7772E-9	94.4556E+6
12	-12.9475E-3	-115.7772E-9	10.9345E+0	42014075	115.7772E-9	94.4444E+6

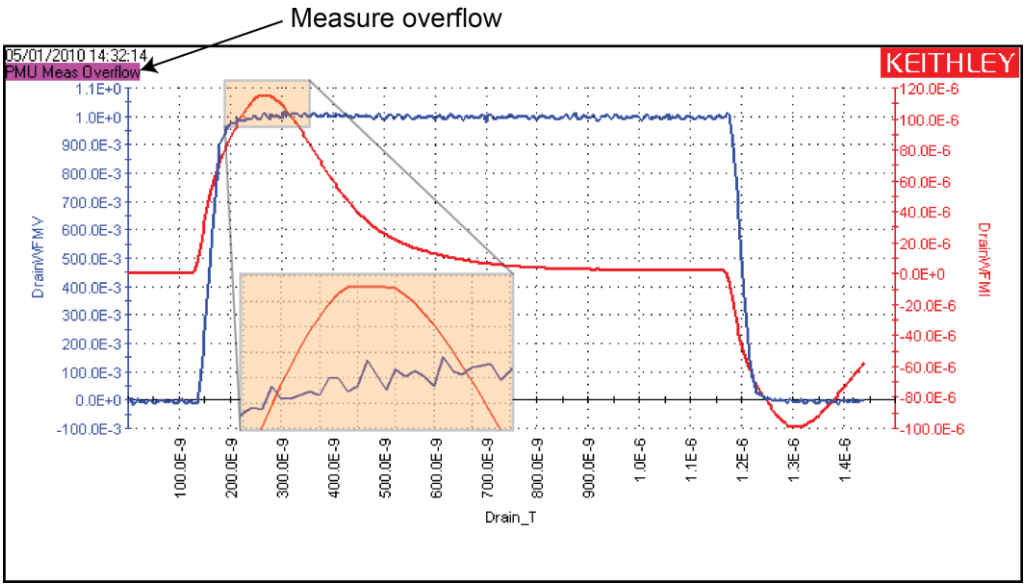
	GateSMIHI(1)	DrainSMVHI(2)	DrainSMIHI(2)	PMU1-2_S(2)	GateSMVHI(2)	GateSMIHI(2)
42	2.8212E-6	5.4766E+0	16.0399E-6	12010036	2.4419E+0	8.3808E-6
43	5.3936E-6	5.3484E+0	15.7569E-6	12010036	2.4473E+0	10.9531E-6
44	4.0659E-6	5.6272E+0	15.7512E	6	2.4256E+0	11.2850E-6
45	10.7872E-6	5.6272E+0	15.7512E	6	2.4310E+0	4.5638E-6
46	8.0489E-6	5.6272E+0	16.2302E	6	2.4365E+0	8.4638E-6
47	1.4936E-6	5.5826E+0	15.9216E	6	2.4365E+0	20.4127E-6
48	1.5765E-6	5.6272E+0	15.7213E	6	2.4419E+0	4.3978E-6

Status:
Ch2
Meas V 40V
Meas I 100uA
Spot Mean
(Bypass)PMU
LLEC Enabled
LLEC Not Met

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. The figure in PMU and RPM measure ranges are not source ranges (on page 5-57) shows a measurement overflow on the Graph. The next figure 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 5-61) for more information.

Figure 265: Sample measurement overflow



Status codes

Each measurement includes an 8-digit status code. Assign the following letters to the code and use the next table to determine status:

A B C D E F G H

In the figure in PMU measurement status (on page 6-102), 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

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, 40 V

Compensation Short Connection

Use to enable or disable short connection compensation. Refer to PMU connection compensation (on page 5-40) for detail on setting up and using connection compensation.

Load Line Effect Compensation

Use to enable or disable load-line effect compensation (LLEC). Refer to Load-line effect compensation (LLEC) for the PMU (on page 5-44) for detail on setting up and using load-line effect compensation.

DUT Resistance (R DUT)

The DUT resistance value. Refer to DUT resistance determines pulse voltage across DUT (on page 5-80) for detail on calculating DUT resistance.

Max Voltage Estimator

The Maximum Voltage Estimator is a tool that enables you to calculate the maximum voltage and current based on the selected voltage range and the DUT resistance value you type in. The estimator does not affect the pulse output.

For a more accurate maximum voltage 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. Poor pin-to-pad contacts could be in the range of 10 Ω to 15 Ω.

The V Max and I Max values are valid with LLEC enabled or disabled. It is the maximum output of the PMU. The LLEC does not change the maximum output voltage or current of the PMU.

Threshold Current

The current threshold allows you to set a threshold. If the threshold is reached or exceeded, the present sweep is stopped. Testing continues with any subsequent sweeps.

This threshold is 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 to which 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. You can specify more than one threshold for each channel.

Threshold Voltage

The voltage threshold allows you to set a voltage threshold. If the threshold is reached or exceeded, the present sweep is stopped. Testing continues with any subsequent sweeps.

This threshold is 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 to which 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. You can specify more than one threshold for each channel.

Threshold Power

The power threshold allows you to set a power threshold. If the threshold is reached or exceeded, the present sweep is stopped. Testing continues with any subsequent sweeps.

This threshold is 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 to which 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. You can specify more than one threshold for each channel.

Test settings

When Configure is selected, you can configure settings for the selected test.

NOTE

The following discusses the Test Settings pane for interactive test modules (ITMs). For tests that are based on user modules (UTMs), you use the options in the Test Settings pane to select the User Library and User Module for the test. Refer to Create a custom test (on page 6-143) for additional information on settings available for UTMs.

In the Test Settings pane, you can set items such as:

- Test speed.
- Timestamp reporting.
- Delays.
- Hold times.
- In-test and post-test data computations using the Formulator. Refer to The Formulator (on page 6-290) for detail on using the Formulator.
- Which readings to send to the subsite sheet using the Output Values.
- Compliance exit condition for SMUs.

NOTE

You can configure multiple measurement delays in the Test Settings pane. However, if you are using autoranging, note that these delays do not include autoranging delays, which can be substantial.

Access the test settings

To access the Test Settings pane:

1. In the project tree, select the test.
2. Select **Configure**.
3. Select **Test Settings** in the right pane.
4. Adjust settings as needed.
5. If needed, select Advanced to access additional settings.

The options that are available depend on the instrument that is selected. For descriptions of parameters, refer to:

- SMU Test Settings (on page 6-108)
- CVU Test Settings (on page 6-118)
- PMU Test Settings (on page 6-129)

SMU Test Settings

The settings that are available for SMU tests are briefly described in the following table. Select the links to access additional information.

Parameter	Description
Speed (on page 6-109)	Select a measurement speed that provides the best trade-off between speed, noise, and accuracy for your test.
Report Timestamps (on page 6-110)	When this option is selected, every measurement in the Analyze sheet includes a timestamp. When this option is cleared, the timestamp is not included.
Delay Factor (on page 6-112)	Custom Speed only. After applying a voltage or current, the instrument waits for a delay time before making a measurement. The delay time allows for source settling. The Delay Factor adjusts the delay time for the selected speed. It is fixed for Fast, Normal, and Quiet speeds. If you select Custom Speed, you can type a factor from 0 to 100.
Filter Factor (on page 6-113)	Custom Speed only. To reduce measurement noise, each 4200A-SCS applies filtering, which averages multiple readings to make one measurement. The Filter Factor is fixed for Fast, Normal, and Quiet speeds. If you select the Custom Speed mode, you can type a Filter Factor of 0 to 100.
Auto A/D Aperture (on page 6-113)	Custom Speed only. When Auto A/D Aperture is selected, the instrument picks the optimum A/D conversion time for most normal measurements.
A/D Aperture Time (on page 6-113)	Custom Speed only when Auto A/D Aperture is cleared. Specifies the A/D converter integration time that is used to measure a signal.
Sweeping Test Mode (on page 6-114)	Select this mode for tests in which the voltage or current varies with time.
Sweep Delay (on page 6-114)	Sweep mode only. If you are using a sweep operation mode and need extra settling time before each measurement, you can specify an additional delay with the Sweep Delay. The Sweep Delay is independent of the Delay Factor (on page 6-112). You can specify a Sweep Delay from 0 s to 999 s. The default Sweep Delay is 0 s.
Sampling Test Mode (on page 6-114)	Select this mode for tests in which the forced voltage and frequency or current are static, with measurements made at timed intervals.
Interval (on page 6-114)	Sampling mode only. The time between measurements (data points). Interval can be set from 0 s to 1000 s.
Number of Samples (on page 6-115)	Sampling mode only. Specifies the number of data points to be acquired. Set the number of samples to a value from 1 to 4096.
Hold Time (on page 6-115)	Allows for extra source settling before making the first measurement. The Hold Time is the same for all SMUs in the test system. Set a hold time between 0 and 999 s. The default Hold Time is 0 s.
SMU Power On Sequence (on page 6-115)	Allows you to adjust the power on sequence of the SMUs.
Disable Outputs at Completion (on page 6-115)	Allows you to choose whether to disable or enable the SMU outputs when a test completes.
Formulator (on page 6-290)	The Formulator allows you to make data calculations on test data and on the results of other Formulator calculations.
Output Values (on page 6-116)	Select values that are included in the Analyze sheet.
Exit Condition (on page 6-117)	Choose the action that is taken if a source goes into compliance.

Speed

The SMU is highly tuned to automatically take into account both settling time and noise issues. It provides measurement speeds that allow you to select trade-offs between speed and noise. You can select:

- **Fast:** Optimizes the SMU for speed at the expense of noise performance. It is a good choice for 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 SMU 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. With Custom, you can configure the A/D aperture 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.

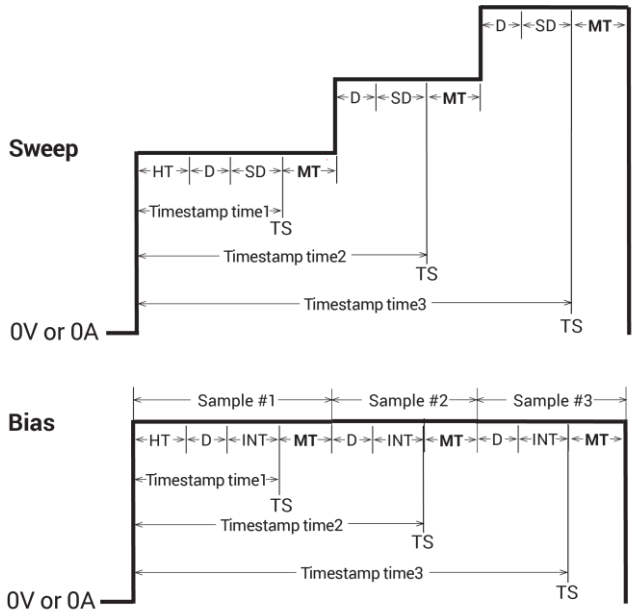
Report Timestamps

When this option is selected, every measurement in the Analyze sheet includes a timestamp. When this option is cleared, the timestamp is not included.

The timestamp records the elapsed time for each measurement. Each elapsed time value is placed in the Analyze sheet 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 Clarius requires only one reading for each measurement, Clarius records the timestamp at the beginning of this reading, as shown in the following figure.

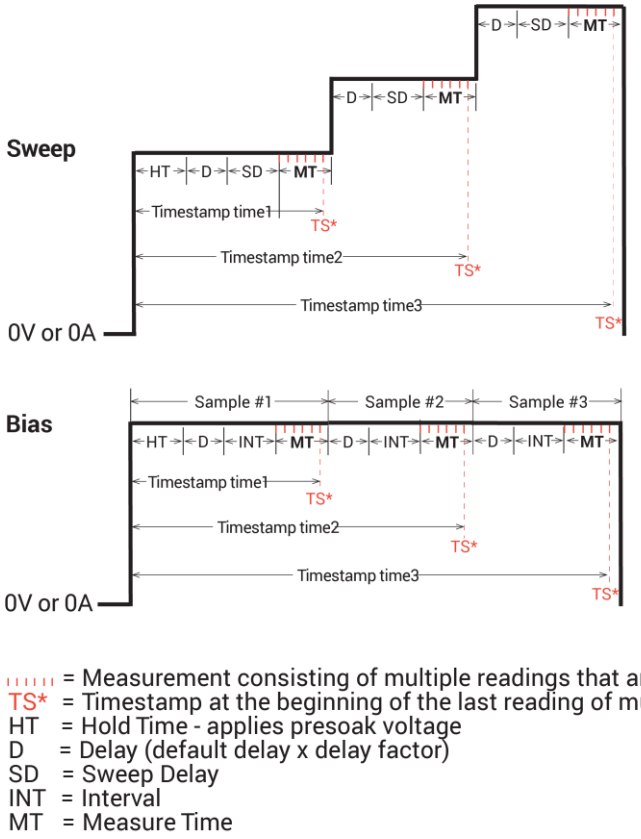
Figure 266: Timestamps when Clarius requires only one reading for a measurement



TS = Timestamp at the start of the reading when one reading is made
HT = Hold Time - applies presoak voltage
D = Delay (default delay x delay factor)
SD = Sweep Delay
INT = Interval
MT = Measure Time

If Clarius makes and averages multiple readings for a measurement, then Clarius records the timestamp at the last of these readings, as shown in the following figure.

Figure 267: Timestamps when Clarius requires multiple readings for a measurement



You can enable the timestamp for any of the measurement speeds.

Delay Factor

After applying a voltage or current, the instrument waits for a delay time before making a measurement. The delay time allows for source settling. The Delay Factor adjusts the delay time for the selected speed. It is fixed for Fast, Normal, and Quiet speeds. If you select Custom Speed, you can type a factor from 0 to 100.

The applied delay time is a multiple of the default delay time. The Delay Factor specifies this multiple. That is:

Applied delay time = (Default delay time) x (Delay Factor)

For example, if the default delay time is 1 ms and the Delay Factor is 0.7, the actual applied delay time is 0.7 ms (1 ms x 0.7).

The following table summarizes the Delay Factor settings.

Speed Mode	Delay Factor
Fast	0.7
Normal	1.0
Quiet	1.3
Custom	0 to 100

When typing a custom Delay Factor setting, consider the following:

- A Delay Factor of 1 allows for 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.

NOTE

If you set the Filter Factor and Delay Factor to `0`, the internal pre-programmed values are ignored.

In general, cables and matrices increase the settling time. You may need to experiment to find the ideal time. However, for a good quality switch, such as the Keithley Instruments 7174A Ultra-Low Current matrix, you should not need to increase the Delay Factor by more than two times.

Filter Factor

To reduce measurement noise, each 4200A-SCS applies filtering, which averages multiple readings to make one measurement. The Filter Factor is fixed for Fast, Normal, and Quiet speeds. If you select the Custom Speed mode, you can type a Filter Factor of 0 to 100.

The default Filter Factor settings are listed in the following table.

Speed Mode	Filter Factor
Fast	0.2
Normal	1
Quiet	3
Custom	0 to 100

When typing a custom Filter Factor, consider the following:

- A Filter Factor of 1 provides filtering that appropriate for most applications.
- Increase the Filter Factor to average multiple readings to reduce noise. As a rule of thumb, doubling the Filter Factor halves the measurement noise.
- Measurement time is increased by the square of the filter factor.
- A Filter Factor of 0 nullifies the internal filtering.

NOTE

When the Filter Factor and Delay Factor are set to zero, the internal pre-programmed values are ignored.

Auto A/D Aperture

When Auto A/D Aperture is selected, the instrument picks the optimum A/D conversion time for most normal measurements.

A/D Aperture Time

If Custom Speed is selected and Auto A/D Aperture is cleared, you can specify the A/D converter integration time that is used to measure a signal. Each measured reading 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 with increased noise. A long integration time results in a relatively low-noise reading at a relatively low speed.

The integration time setting is based on the number of power line cycles (NPLCs). For 60 Hz line power, 1.0 PLC = 16.67 ms (1/60). For 50 Hz line power, 1.0 PLC = 20 ms (1/50).

The applied A/D conversion time also depends on the Filter Factor setting:

- If the Filter Factor is not zero, the SMU 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 Aperture Time.
- If the Filter Factor setting is zero, the SMU applies a fixed A/D converter time that equals the specified A/D Aperture Time.

The Auto A/D Aperture and A/D Aperture Time settings for each Speed Mode are shown in the following table.

Speed Mode	A/D Aperture Time
Fast	Auto
Normal	Auto
Quiet	Auto
Custom	0.01 to 10 PLC

Test Mode

You can set a test to the sweeping or sampling test mode.

Sweeping test mode is used for tests in which the voltage, current, or frequency varies with time.

Sampling Mode allows you to measure voltages or currents as a function of time while forcing constant voltages or currents. The sampling test mode is used for tests in which the forced voltage and frequency are static, with measurements made at timed intervals. For example, you could use sampling mode to profile a capacitor charging voltage while forcing a constant current. Time is measured relative to when the instruments apply the forced voltage or current (that is, t = 0 at the step change from 0.0 V or 0.0 A to the applied voltage or current).

When Sampling Mode is selected, all device terminals are set to a static operation mode, such as Open or Voltage Bias.

Refer to Operation mode timing diagrams (on page 3-47) for additional detail.

Sweep Delay

If you are using a sweep operation mode and need extra settling time before each measurement, you can specify an additional delay with the Sweep Delay. The Sweep Delay is independent of the Delay Factor (on page 6-112). You can specify a Sweep Delay from 0 s to 999 s. The default Sweep Delay is 0 s.

Interval

Sampling mode only. The time between measurements (data points). Interval can be set from 0 s to 1000 s.

Number of Samples

Specifies the number of data points to be acquired. Set the number of samples to a value from 1 to 4096.

Hold Time

The starting voltages or currents of a sweep may be substantially larger than the voltage or current increments of the sweep. Accordingly, the source settling time required to reach the starting voltages or currents 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 at the beginning of each sweep. You can specify a Hold Time of 0 to 1000 s.

SMU Power On Sequence

When you run a test, the SMUs for the given test power on in a specific sequence. You can adjust this sequence.

To adjust the power-on sequence for the SMUs:

1. Select the test.
2. Select **Configure**.
3. In the right pane, select the **Test Settings** tab.
4. Select **Advanced**.
5. Under SMU Power On Sequence, use the **Move Up** and **Move Down** buttons to change the sequence.
6. Select **OK**.

Disable outputs at completion - SMU

⚠ WARNING

If "Disable Outputs at Completion" is cleared, the SMU outputs remain at their last programmed levels when the test is completed. To prevent electrical shock that could cause injury or death, never make or break connections to the 4200A-SCS while the output is on.

When a test is complete, you can disable the outputs or leave them enabled.

If you enable the outputs, the SMU starts the next test from its state at the end of the previous test. If the next test does not use the SMU, the SMU remains in that state.

To set the output action on completion:

1. Select the test.
2. Select **Configure**.
3. In the right pane, select the **Test Settings** tab.
4. Select **Advanced**.
5. Select or clear **Disable Outputs at Completion**.

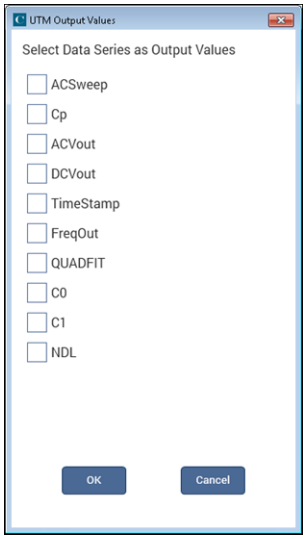
Output Values

If you are using subsites, each time a subsite is cycled, the measurements for the enabled Output Values are placed in the Analyze sheet for the subsite. For example, if the subsite is cycled five times, there will be five measured readings (Output Values) for the test. For details, see Subsite cycling (on page 6-231).

To select output values:

1. In the project tree, select the test.
2. Select **Configure**.
3. In the right pane, select **Test Settings**.
4. Select **Output Values**. The Output Values dialog box is displayed.
5. Select the data series that you want to send to the Analyze sheet.
6. Select **OK**.

Figure 268: Select Output Values



Compliance exit condition options

The compliance exit condition options are available from the Test Settings pane when you select **Exit Condition**.

When the source goes into compliance, you can choose the action that the system takes.

You can select one of the following exit condition options:

- **None**: The run continues. This is the default setting.
- **Test**: The 4200A-SCS exits the test that is presently being run. If there are additional tests, operation continues to the next test.
- **Device**: The 4200A-SCS exits the device that is presently being run. If there are additional devices, operation continues to the next device.
- **Subsite**: The 4200A-SCS exits the subsite presently being run. If there are additional subsites, operation continues to the next subsite.
- **Site**: The 4200A-SCS exits the site. If there are multiple sites, operation continues to the next site.
- **Project**: The 4200A-SCS exits the project. All testing stops.

Set the compliance exit condition

To set the exit condition when compliance occurs:

1. Select the test in the project tree.
2. Select **Configure**.
3. Select **Test Settings**.
4. Select **Exit Condition**.
5. Select the option for your test.
6. Select **OK**.

CVU Test Settings

The settings that are available for CVU tests are briefly described in the following table. Select the links to access additional information.

Parameter	Description
CVU speed settings (on page 6-119)	Select a measurement speed that provides the best trade-off between speed, noise, and accuracy for your test.
Report Timestamps (on page 6-124)	When this option is selected, every measurement in the Analyze sheet includes a timestamp. When this option is cleared, the timestamp is not included.
Delay Factor	Custom Speed only. After applying a voltage or current, the instrument waits for a delay time before making a measurement. The delay time allows for source settling. The Delay Factor adjusts the delay time for the selected speed. It is fixed for Fast, Normal, and Quiet speeds. If you select Custom Speed, you can type a factor from 0 to 100.
Filter Factor (on page 6-122)	Custom Speed only. To reduce measurement noise, each 4200A-SCS applies filtering, which averages multiple readings to make one measurement. The Filter Factor is fixed for Fast, Normal, and Quiet speeds. If you select the Custom Speed mode, you can type a Filter Factor of 0 to 707.
Auto A/D Aperture (on page 6-122)	Custom Speed only. When Auto A/D Aperture is selected, the instrument picks the optimum A/D conversion time for most normal measurements.
A/D Aperture Time (on page 6-122)	Custom Speed only if Auto A/D Aperture is cleared. Specifies the A/D converter integration time that is used to measure a signal.
Sweeping Test Mode (on page 6-125)	Select this mode for tests in which the voltage or frequency varies with time.
Sweep Delay (on page 6-126)	Sweep mode only. If you are using a sweep operation mode and need extra settling time before each measurement, you can specify an additional delay with the Sweep Delay. The Sweep Delay is independent of the Delay Factor (on page 6-112). You can specify a Sweep Delay from 0 s to 999 s. The default Sweep Delay is 0 s. See Choose hold and sweep delay times for CVUs (on page 6-127) for information on determining sweep delay times.
Sampling Test Mode (on page 6-114)	Select this mode for tests in which the forced voltage and frequency are static, with measurements made at timed intervals.
Interval (on page 6-126)	Sampling mode only. The time between measurements (data points). Interval can be set from 0 s to 1000 s.
Number of Samples (on page 6-126)	Sampling mode only. Specifies the number of data points to be acquired. Set the number of samples to a value from 1 to 4096.
Hold Time (on page 6-115)	Allows for extra source settling before making the first measurement. The Hold Time is the same for all SMUs or CVUs in the test system. Set a hold time between 0 and 999 s. The default Hold Time is 0.1 s.
Disable Outputs at Completion (on page 6-127)	Allows you to choose whether to disable or enable the CVU outputs when a test completes.
Formulator (on page 6-290)	The Formulator allows you to make data calculations on test data and on the results of other Formulator calculations.
Output Values (on page 6-127)	Select values that are included in the Analyze sheet.

CVU speed settings

To make successful C-V measurements, you must choose the appropriate speed settings. Adjustments to the speed settings can help prevent noisy measurements and allow you to acquire readings in equilibrium.

To adjust the speed settings, in Configure, select the **Test Settings** pane, then select **Advanced**.

The Speed settings allow you to select trade-offs between speed and noise. You can select:

- Fast: Optimizes the CVU for speed at the expense of noise performance. It is a good choice for measurements where noise and settling time are not concerns.
- Normal: The default and most commonly used setting. It provides a balanced combination of speed and low noise and is the best setting for most cases.
- Quiet: Optimizes the CVU 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: While Fast, Normal, or Quiet should be appropriate for most applications, Custom allows you to fine tune the timing parameters if needed.

Considerations when making Custom speed settings

You can optimize your measurements by making adjustments to the custom speed settings. With Custom, you can configure the A/D aperture 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

If you select Custom speed, you can fine-tune the speed using the Delay Factor, Filter Factor, Auto A/D Aperture, and A/D Aperture Time.

Each measured reading by a CVU is the result of one or more analog to digital (A/D) conversions. Reducing the aperture time for each A/D conversion results in a relatively fast measurement speed with increased noise. Increasing the A/D Aperture reduces noisy readings.

An overview of each adjustment and its effects are provided in the following table. Additional detail on each setting is provided after the table.

To:	
Reduce noise	■ Increase Filter Factor ■ Increase the A/D aperture time ■ Use Quiet mode
Increase measurement speed	■ Use a fixed measurement range ■ Reduce Filter Factor ■ Reduce A/D aperture time ■ Reduce the Delay Factor
Increase settling time (often needed for long cables or switch matrices)	■ Increase the Delay Factor ■ Use sweep delay and hold times to charge up the device to equilibrium; refer to Operation Mode (CVU) (on page 6-61) for more information on working with sweeps
Get more consistent time intervals between measurements	■ Used a fixed measurement range ■ Specify the A/D aperture time ■ Set Filter Factor and Delay Factor to zero

Delay Factor

The Delay Factor adjusts the delay time for the selected speed. In some cases, you may need an additional delay before making a measurement to allow for range changes and to accommodate the number of points, especially for DC bias. The Delay Factor is a multiple of the default delay time:

Applied delay time = (Default delay time) x (Delay Factor)

For example, if the default delay time is 1 ms and the Delay Factor is 0.7, the actual applied delay time is 0.7 ms (1 ms x 0.7).

The following table shows the default Delay Factor settings for Fast, Normal, and Quiet, and the range of values available when Custom is selected.

Speed Mode	Delay Factor
Fast	0.7
Normal	1.0
Quiet	1.3
Custom	0 to 100

When typing a custom Delay Factor setting, consider the following:

- A Delay Factor of 1 allows for 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 additional delay. However, instrument processing time may cause some small delay.

NOTE

If you set the Filter Factor and Delay Factor to `0`, the internal pre-programmed values are ignored.

In general, cables and matrices increase the settling time. You may need to experiment to find the ideal time. However, for a good quality switch, such as the Keithley Instruments 7174A Ultra-Low Current matrix, you should not need to increase the Delay Factor by more than two times.

Filter Factor

The Filter Factor adjusts how many readings are averaged before a measurement is recorded. In some cases, you may need adjust the default filtering.

The following table shows the default Filter Factor settings for Fast, Normal, and Quiet, and the range of values available when Custom is selected.

Speed Mode	Filter Factor
Fast	1
Normal	1
Quiet	1
Custom	707

When typing a custom Filter Factor, consider the following:

- A Filter Factor of 1 provides filtering that appropriate for most applications.
- Increase the Filter Factor to average multiple readings to reduce noise. As a rule of thumb, doubling the Filter Factor halves the measurement noise.
- Measurement time is increased by the square of the filter factor.

NOTE

When the Delay Factor is set to zero, the internal pre-programmed values are ignored.

Auto A/D Aperture

When Auto A/D Aperture is selected, the instrument picks the optimum A/D conversion time for most normal measurements.

A/D Aperture Time

If adjusting the Delay and Filter Factors does not provide enough noise reduction you can also adjust A/D converter aperture time that is used to measure a signal. Reducing the aperture time for each A/D conversion results in a relatively fast measurement speed with increased noise. Increasing hasurement time results in a relatively low-noise reading at a relatively low speed.

For 60 Hz line power, 1.0 PLC = 16.67 ms (1/60). For 50 Hz line power, 1.0 PLC = 20 ms (1/50).

The applied A/D conversion time also depends on the Filter Factor setting:

- If the Filter Factor is not zero, the CVU 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 Aperture Time.
- If the Filter Factor setting is zero, the CVU applies a fixed A/D converter time that equals the specified A/D Aperture Time.

The following table shows the default A/D Aperture Time settings for Fast, Normal, and Quiet, and the range of values available when Custom is selected.

Speed Mode	**A/D Aperture Time (seconds)**
Fast	.001
Normal	.01
Quiet	0.1
Custom	.0001 to 10

To adjust the aperture time, you must select Custom Speed and clear A/D Aperture.

The applied A/D conversion time also depends on the Filter Factor setting:

The CVU 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 Aperture Time.

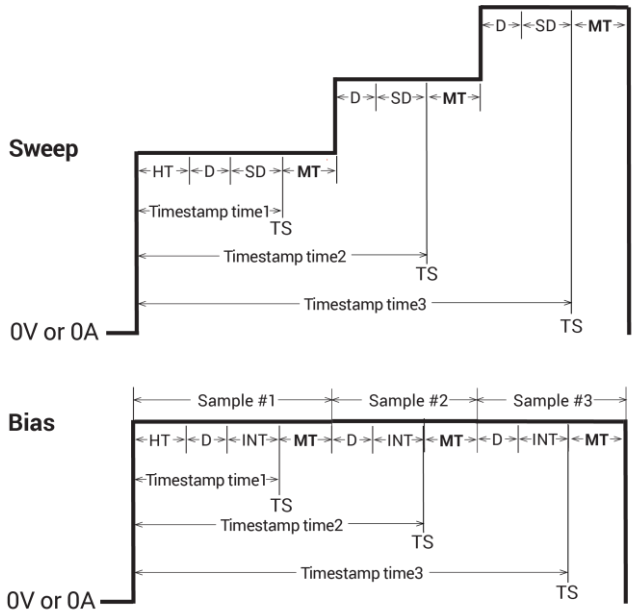
Report Timestamps

When this option is selected, every measurement in the Analyze sheet includes a timestamp. When this option is cleared, the timestamp is not included.

The timestamp records the elapsed time for each measurement. Each elapsed time value is placed in the Analyze sheet 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 Clarius requires only one reading for each measurement, Clarius records the timestamp at the beginning of this reading, as shown in the following figure.

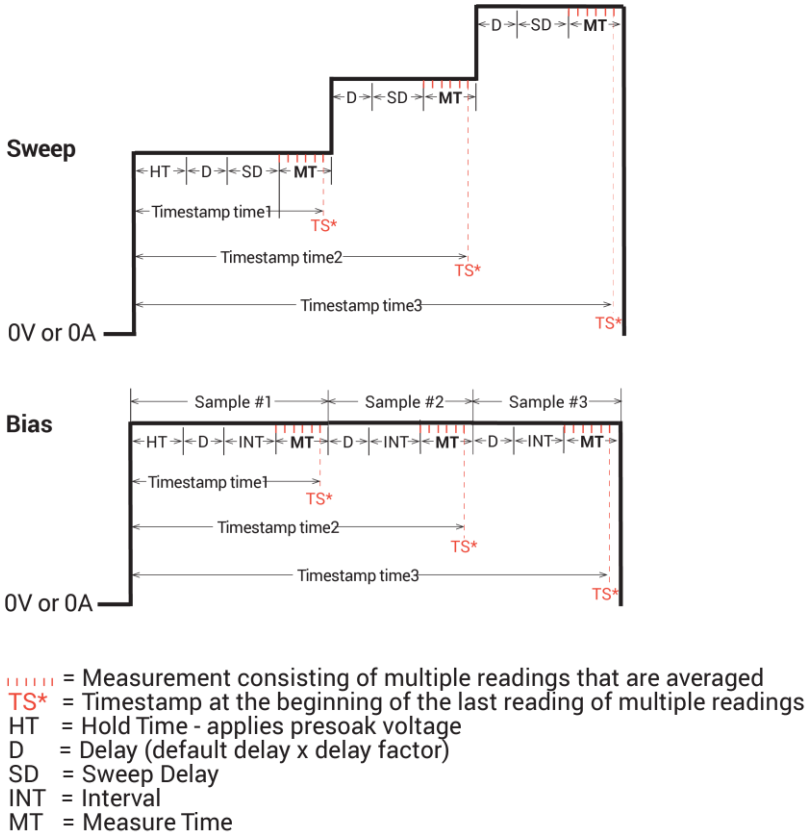
Figure 269: Timestamps when Clarius requires only one reading for a measurement



TS = Timestamp at the start of the reading when one reading is made
HT = Hold Time - applies presoak voltage
D = Delay (default delay x delay factor)
SD = Sweep Delay
INT = Interval
MT = Measure Time

If Clarius makes and averages multiple readings for a measurement, then Clarius records the timestamp at the last of these readings, as shown in the following figure.

Figure 270: Timestamps when Clarius requires multiple readings for a measurement



You can enable the timestamp for any of the measurement speeds.

Test Mode

You can set a test to the sweeping or sampling test mode.

Sweeping test mode is used for tests in which the voltage or frequency varies with time.

Sampling Mode allows you to measure capacitance or currents as a function of time while forcing constant voltages. The sampling test mode is used for tests in which the forced voltage and frequency are static, with measurements made at timed intervals. For example, you could use sampling mode to profile a capacitor charging voltage while forcing a constant current. Time is measured relative to when the instruments apply the forced voltage.

When Sampling Mode is selected, all device terminals are set to a static operation mode, such as Open or Voltage Bias.

Refer to Operation mode timing diagrams (on page 3-47) for additional detail.

Sweep Delay

If you are using a sweep operation mode and need extra settling time before each measurement, you can specify an additional delay with the Sweep Delay. The Sweep Delay is independent of the Delay Factor (on page 6-112). You can specify a Sweep Delay from 0 s to 999 s. The default Sweep Delay is 0 s.

Interval

Sampling mode only. The time between measurements (data points). Interval can be set from 0 s to 1000 s.

Number of Samples

Specifies the number of data points to be acquired. Set the number of samples to a value from 1 to 4096.

Hold Time

The starting voltages of a sweep may be substantially larger than the voltage increments of the sweep. Accordingly, the source settling time required to reach the starting voltages 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 at the beginning of each sweep. You can specify a Hold Time of 0 to 1000 s.

The presoak voltage set in the advanced terminal settings window will be applied for the specified hold time.

Choose hold and sweep delay times for CVUs

The condition of a device when all internal capacitances are fully charged after an applied step voltage is referred to as equilibrium.

If capacitance measurements are made before the device is in equilibrium, you may get inaccurate results.

To choose the delay times for a C-V sweep:

1. Set up conditions of:
   - Hold time: 0 s
   - Interval time: 0 s
   - Presoak: 0 V
2. Step an applied voltage using the sampling mode.
3. Plot the capacitance as a function of time.
4. Observe the settling time from the graph.
5. Use this time for the hold time for the initial applied voltage or for the sweep delay time applied at each step in the sweep. The sweep delay time may not need to be as long as the first step.

Disable outputs at completion - CVU

When a test is complete, you can disable the outputs or leave them enabled.

When Disable Outputs at Completion is selected, the output turns off (0 V) when the test is complete. If this option is not selected, the DC Bias voltage remains at the last bias voltage level at the end of the previous test.

To set output action on completion:

1. Select the test.
2. Select **Configure**.
3. In the right pane, select the **Test Settings** tab.
4. Select or clear **Disable Outputs at Completion**.

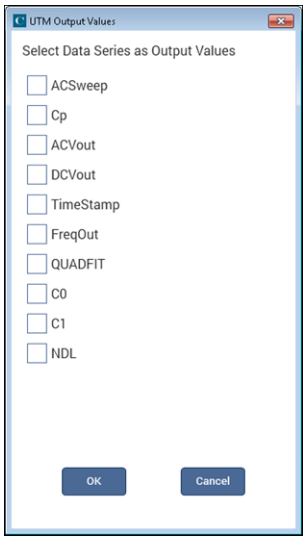
Output Values

If you are using subsites, each time a subsite is cycled, the measurements for the enabled Output Values are placed in the Analyze sheet for the subsite. For example, if the subsite is cycled five times, there will be five measured readings (Output Values) for the test. For details, see Subsite cycling (on page 6-231).

To select output values:

1. In the project tree, select the test.
2. Select **Configure**.
3. In the right pane, select **Test Settings**.
4. Select **Output Values**. The Output Values dialog box is displayed.
5. Select the data series that you want to send to the Analyze sheet.
6. Select **OK**.

Figure 271: Select Output Values



PMU Test Settings

The settings that are available for PMU tests are briefly described in the following table. Select the links to access additional information.

Parameter	Description
Test Mode (on page 6-130)	Select Pulse IV to make spot mean measurements (V or I) of the amplitude or base portions of one or more pulses. Select Waveform Capture to sample the entire waveform.
Measure Mode (on page 6-131)	Select Average Pulses to average the measured values of two or more pulses. Select Discrete Pulses to report the measured value of each pulse.
Number of Pulses (on page 6-134)	The number of pulses for the PMU channel to output and measure for each step in the sweep (1 to 10,000). If you enabled autorange, load-line effect compensation (LLEC), or thresholds, the number of pulses is output multiple times for each step in a sweep.
Timing Sweep (on page 6-135)	For Sweep and Step, select None, Period, Width, Rise Time, or Fall Time. This is only available if the Operation Mode is set to either pulse sweep or pulse train.
Period (on page 6-136)	The pulse period is the time interval between the start of the rising transition edge of consecutive output pulses.
Width (on page 6-136)	The pulse width is the interval between the rising-edge and falling-edge medians.
Rise Time (on page 6-136)	The rise transition time for the pulse output.
Fall Time (on page 6-137)	The fall transition time for the pulse output.
Pulse Delay (on page 6-137)	The pulse delay is the time interval between the start of the rising pulse edge of the trigger output pulse and the output pulse.
Settling Times (on page 6-138)	The settling time is the time that it takes for pulse levels to stabilize after the voltage or current is changed, such as during execution of a sweep.
Sweep Master (on page 6-139)	Designates which instrument is the master if there are multiple steps or sweeps in a test. Refer to Step or sweep multiple device terminals in the same test (on page 6-189) for detail.
Formulator (on page 6-290)	The Formulator allows you to make data calculations on test data and on the results of other Formulator calculations.
Output Values (on page 6-116)	Select values that are included in the Analyze sheet.

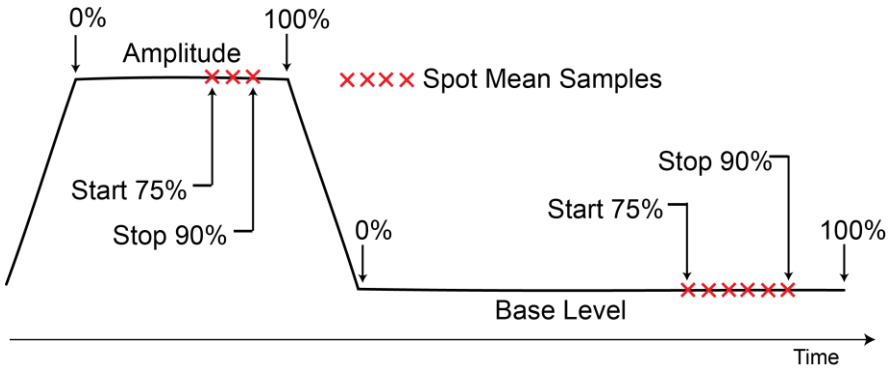
Test Mode (PMU)

The Test Mode selects the type of pulse test. You can select:

- **Pulse IV**: This test mode performs spot mean measurements (V or I) of the amplitude or base portions of one or more pulses. Typically, current versus voltage is graphed in the Analyze graph. The `pmu-iv-sweep` project provides an example of a pulse drain family of curves using the pulse test test mode.
- **Waveform Capture**: This test mode samples the entire waveform. The waveform is graphed (voltage and/or current versus time) in the Analyze graph. The `pmu-1ch-wfm` project provides an example of a waveform capture. Note that the Timing Sweep Step option (in the Advanced dialog box) is disabled when Waveform Capture is selected.

For pulse I-V, spot mean measurements are made on pulse amplitude and base level. You can measure voltage and current. The next figure shows the measure windows for spot mean measurements. The start measure and stop measure points for ITMs are fixed at 75 percent and 90 percent for both amplitude and base level.

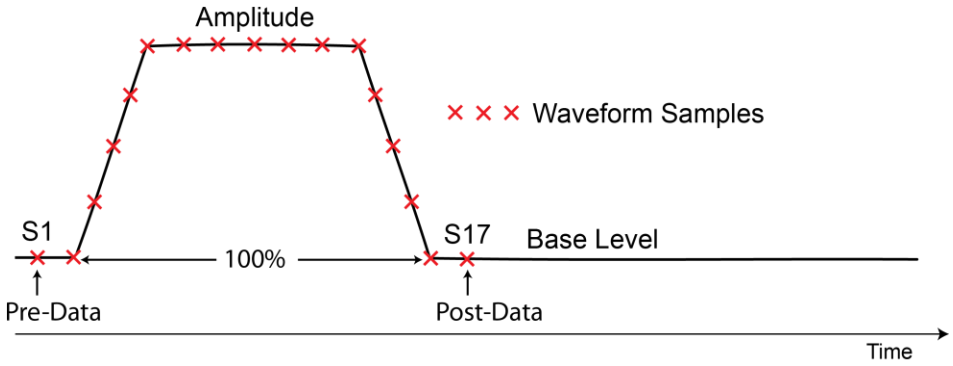
Figure 272: Pulse I-V (spot mean measurements)



The number of measurement samples that are made 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.

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, as shown in the next figure.

Figure 273: Waveform capture



For detail on spot mean measurements, refer to Spot mean measurements (on page 5-91). For user test module (UTM) programming, refer to the pulse_meas_sm (on page 14-119) function.

For detail on waveform measurements, refer to Waveform measurements (on page 5-94). For UTM programming, the pulse_meas_wfm (on page 14-123) function is used to configure waveform measurements.

For UTMs, the pulse_meas_timing (on page 14-121) function allows you to adjust the measure window for the pulse I-V test mode or the pre-and post-data settings for the waveform measurements. For ITMs, the measure window and pre-data and post-data settings are fixed.

Measure Mode

For Measure Mode, you can select Average Pulses or Discrete Pulses.

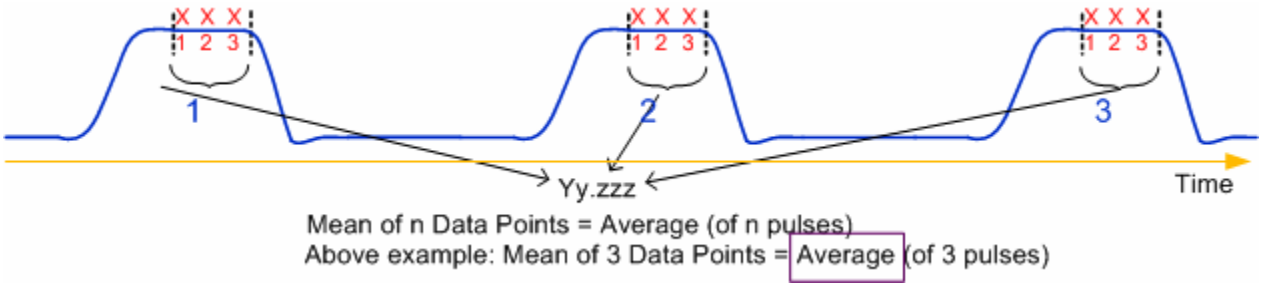
When Average Pulses is selected for Pulse IV, the measured values of two or more pulses are averaged. See the Pulse IV (Average pulses) measurement example (on page 6-132). For Waveform Capture, select Average Pulses to average each sample in the waveform with the same point in subsequent waveforms, for each pulse specified by the number of pulses.

When Discrete Pulses is selected for Pulse IV, the measured value of each pulse is acquired. Refer to the Pulse IV (Discrete pulses) measurement example (on page 6-133). For Waveform Capture, each sample for every waveform is recorded in the Analyze sheet. Refer to the Waveform Capture (Discrete Pulses) measurement example (on page 6-134).

Pulse I-V (Average pulses) measurement example

For the example shown in the next figure, 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 Analyze sheet.

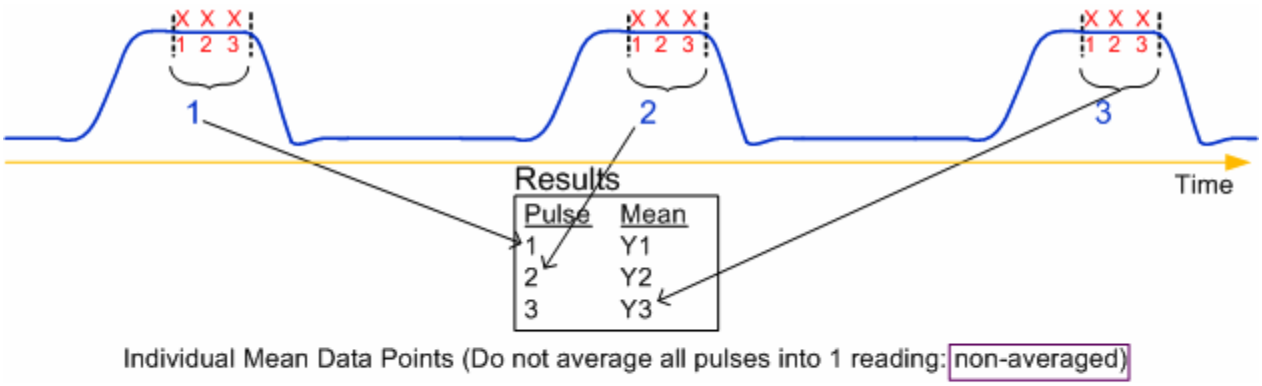
Figure 274: Pulse IV - Average pulses



Pulse I-V (Discrete pulses) measurement example

For the example shown in the next figure, 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 Analyze sheet for the 12 readings are also included in the next figure.

Figure 275: Pulse IV - Discrete pulses

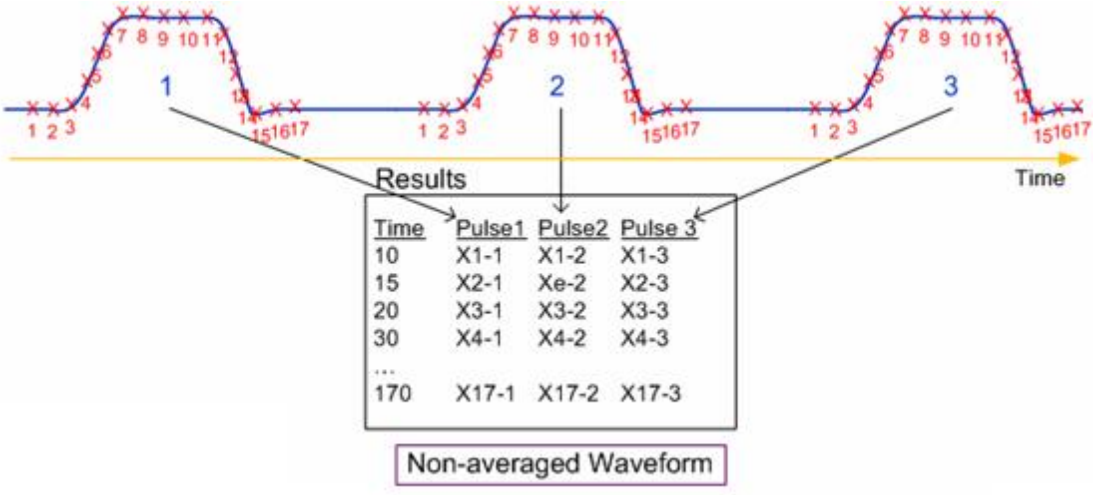


	A	B
1	ASMVHI	ASMIHI
2	1.9326E+0	1.9520E-3
3	1.9330E+0	1.9523E-3
4	1.9327E+0	1.9523E-3
5	2.9023E+0	2.9286E-3
6	2.9028E+0	2.9285E-3
7	2.9026E+0	2.9284E-3
8	3.8683E+0	3.9021E-3
9	3.8685E+0	3.9023E-3
10	3.8680E+0	3.9019E-3
11	4.8338E+0	4.8765E-3
12	4.8339E+0	4.8770E-3
13	4.8339E+0	4.8769E-3

Waveform Capture (Discrete Pulses) measurement example

For the example shown in the next figure, the samples of three pulses are captured. The 51 samples (17 samples x 3 pulses) are placed in the Analyze sheet.

Figure 276: Waveform capture - Discrete pulses



Number of Pulses

The number of pulses for the PMU channel to output and measure for each step in the sweep (1 to 10,000). If you enabled autorange, load-line effect compensation (LLEC), or thresholds, the number of pulses is output multiple times for each step in a sweep.

Timing Sweep

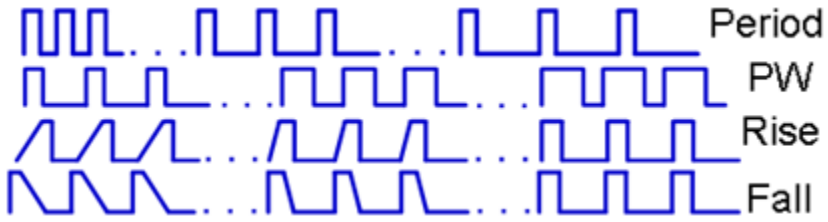
You can only step the timing parameters if the Operation Mode is set to pulse sweep or pulse train (not pulse step) and if Test Mode is set to Pulse IV.

You can only sweep the timing parameters if the Operation Mode is set to pulse step or pulse train (not pulse sweep).

The Timing Sweep Step option is disabled when Waveform Capture is selected.

The next figure shows parameter stepping and sweeping examples. In these examples, the Number of Pulses parameter is set to 3, so there are three pulses per period.

Figure 277: Parameter stepping sweeping examples



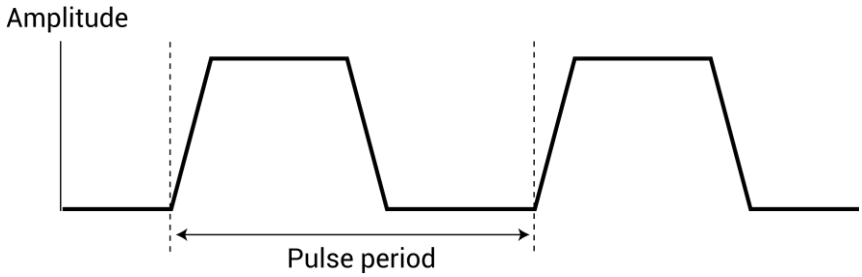
- **Period:** The pulse period steps or sweeps from a short period to a long period (or from a long period to a short period).
- **PW:** The pulse width (PW) steps or sweeps from a short pulse width to a long pulse width (or from a long width to a short width).
- **Rise:** The pulse rise time steps or 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 or sweeps from a long fall time to a short fall time (or from a long fall time to a short fall time).

In the figure above, the ellipsis (...) between each burst of pulses indicates additional time that the pulse channel is outputting 0 VDC (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 How LLEC adjusts pulse output to the target levels (on page 5-45).

Period

The pulse period is the time interval between the start of the rising transition edge of consecutive output pulses, as shown in the following figure. 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 to 10 percent.

Figure 278: Pulse period



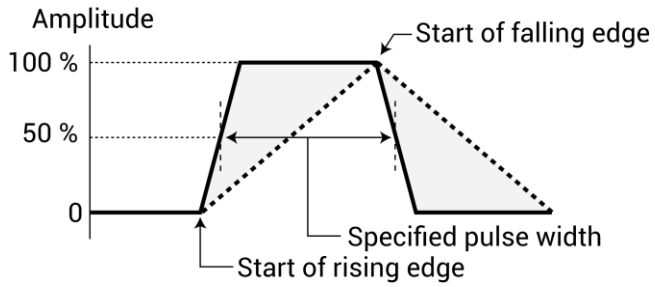
Width

The pulse width is the interval between the rising-edge and falling-edge medians. For a pulse with fast edges (a 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.

Rise and fall times are set independently. The pulse width is not affected by a change to the transition time. However, start points may shift with changes in transition time.

The dashed line pulse in the following figure 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, the pulse would not reach its programmed amplitude (100%).

Figure 279: Pulse width



Rise Time

The rise transition time for the pulse output.

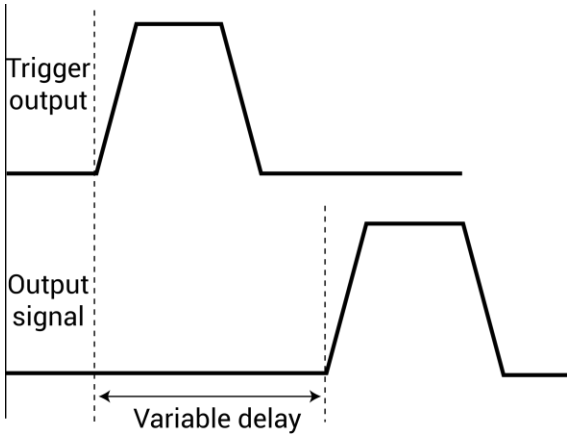
Fall Time

The fall transition time for the pulse output.

Pulse Delay

The pulse delay is the time interval between the start of the rising pulse edge of the trigger output pulse and the output pulse. You can set the rising or falling polarity of the trigger output pulse edge to the output pulse. The pulse delay has a variable delay with respect to the trigger output.

Figure 280: Pulse delay

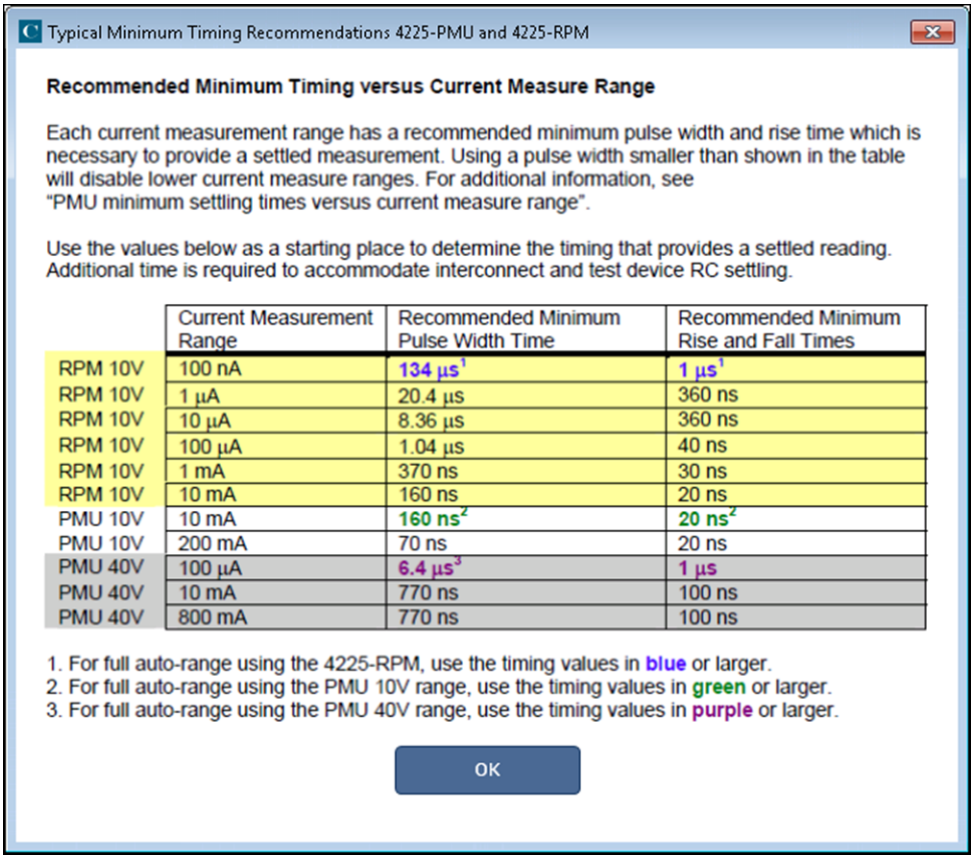


Settling Times

The settling time is the time that it takes for pulse levels to stabilize after the voltage or current is changed, such as during execution of a sweep.

The Typical Minimum Timing Recommendations dialog box displays typical minimum timing versus current measure range recommendations. These are the recommended minimums for each current measure range. Longer times may be required to accommodate DUT and interconnect settling.

Figure 281: Typical minimum timing recommendations



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

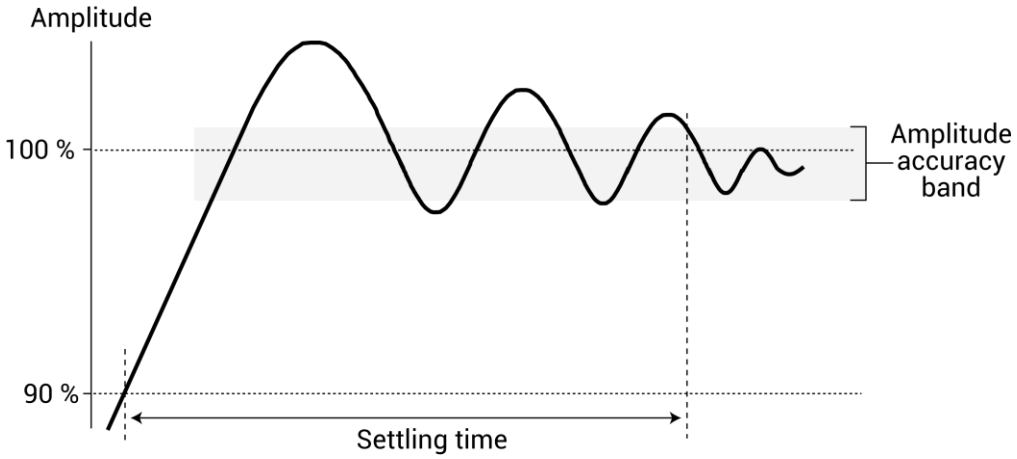
If pulse width or period timing parameters are too narrow for a chosen measure range, a message is displayed showing the change to the current measure range for each effected channel. To avoid these changes, follow the guidelines shown in the previous figure. See PMU minimum settling times versus current measure range (on page 5-52) for more information.

Settling time includes the following τ (tau) time constants:

- **τInstrument:** Varies mainly with current range.
- **τSystem:** Due to cables, switches, and probers.
- **τDUT (Device Under Test):** Due to the implicit characteristics of the DUT.
- **τDielectric Absorption:** An issue only on the low current ranges.

Settling time is measured from the 90 percent point on the rising edge to the point where the pulse level remains in the accuracy band, as shown in the following figure.

Figure 282: Settling time



Sweep Master

Designates which instrument is the master if there are multiple steps or sweeps in a test. Refer to Step or sweep multiple device terminals in the same test (on page 6-189) for detail.

Set up a complex project

Set up a simple project (on page 6-10) describes how to set up a project with devices and tests for those devices. However, if your system includes wafers, external equipment, or custom tests, you need to add additional items to your project tree to accommodate them.

You can include the following operations and objects in the project:

- Custom tests or actions.
- Prober movement between project sites and subsites. Refer to Set up a Probe Station (on page F-1) for detail.
- Using switch matrices to cycle electrical connections from the 4200A-SCS between the devices of a subsite. Refer to Using Switch Matrices (on page A-1) for detail.

This section describes how to add custom tests, actions, sites, and subsites to Clarius and to the project tree.

NOTE

For information on adding, duplicating, and importing projects, refer to Working with My Projects (on page 6-18).

Customize tests

There are two types of tests in Clarius:

- **Interactive Test Modules (ITM):** Predefined tests that you can select and configure through the Clarius interface. They are used exclusively for parametric testing. You can create blank ITMs in Clarius that you can customize.
- **User Test Module (UTM):** A test that is based on a user module. Once the user module is incorporated into a test or action in Clarius, you can select and configure it in the Clarius interface. In addition to controlling tests, UTMs can control internal instrumentation or external instrumentation that is connected through the GPIB bus or RS-232 port. They can also be used for other tasks in the project, such as displaying prompts for test operators.

User modules are created in Keithley User Library Tool (KULT). Clarius+ comes with many predefined user modules, organized into user libraries. Refer to User library descriptions (on page 6-289) for descriptions of the pre-built user libraries and modules.

You can also use KULT to create your own user modules or modify the source code for a module supplied by Keithley Instruments. Refer to Keithley User Library Tool (on page 8-1) for detail.

Both ITMs and UTMs share common data analysis functions, such as the Analyze spreadsheet and graph.

You can customize tests in the following ways:

- Start with a predefined test and modify it.
- Start with a blank ITM test and modify it.
- Start with a blank UTM test, define the user module, and modify it.

After modifying a test, you can save it to the test library as a predefined test that can be used in other projects.

Modify a predefined test

You can modify an existing test that you added using the steps in Add a device and test to the project (on page 6-12).

Settings that you make to a test that is in the project tree are stored with the project. If you need to return to the settings of the test that is in the library, you can add the test from the library again.

Create a custom ITM

You can create a custom interactive test module (ITM) in Clarius. You do not need to create any external files (such as user modules) to create a custom ITM.

When you create a blank ITM, the number of terminals in the new test are determined by the type of device the test is placed under.

To create an ITM custom test:

1. Choose **Select**.
2. Highlight a device in the project tree or add a device.
3. If you need to add a device, open the Devices tab and select a device.
4. Select the **Tests** tab.
5. In the Test Library, select **Custom Test**.
6. Select **Add a blank test that can be configured into a DC, Pulse, or CV test (ITM)**.
7. Drag **Custom Test** to the project tree. The test has a red triangle next to it to indicate that it is not configured.
8. Select **Rename**.
9. Type a name for the test and press **Enter**.
10. Select **Configure** to set up the test.
11. Select the instrument.
12. Refer to Test and terminal setting descriptions (on page 6-26) to configure the other options.
13. For each device terminal, ensure that the physical device connections match the device connections defined in Clarius. If necessary, shut down the instrumentation and correct the physical connections.

CAUTION

Physical device-terminal connections must accurately match virtual connections to avoid inaccurate test results and potential device damage.

Create a custom UTM

User test modules (UTMs) are created from user modules. Many user modules are provided with the 4200A-SCS in the user libraries. You can also create your own user modules. For information on creating your own user modules, refer to Keithley User Library Tool (KULT) (on page 8-1).

You can use one user module for multiple UTMs. Each instance of the user module is treated separately.

Data generated by a UTM is displayed in the Analyze sheet and graph.

NOTE

When you are building a project, it may be convenient to add all new UTMs first without immediately connecting them to user modules. This allows you to focus on project structure without being distracted with configuration details. To add a UTM without connecting it to a user modules, stop the following procedure after renaming the test.

To create a UTM:

1. Choose **Select**.
2. Select the **Tests** tab.
3. For the Custom Test, select **Choose a test from the pre-programmed library (UTM)**.
4. Drag **Custom Test** to the project tree. The test has a red triangle next to it to indicate that it is not configured.
5. Select **Rename**.
6. Type a name for the test.
7. Select **Configure**.
8. In the right pane, from the User Libraries list, select the user library that contains the user module that contains the test.
9. From the User Modules list, select the user module.
10. Enter the parameters in the Configure pane. Refer to the Help pane for descriptions of the options in the UTM.
11. You can use the Formulator to do calculations on the test results. See Formulator (on page 6-290) for additional information.
12. If needed, select **Output Values** to specify output values to export into the subsite data sheet.
13. You can edit the user interface (UI) of the UTM. Refer to Define the user interface for a user test module (on page 6-144) for instruction.

Define the user interface for a user test module (UTM)

After you create a user module, you can use it as a user test module (UTM) in a 4200A-SCS project.

When you create a user interface for a user test module (UTM), you can set up the Key Parameters Configure pane to:

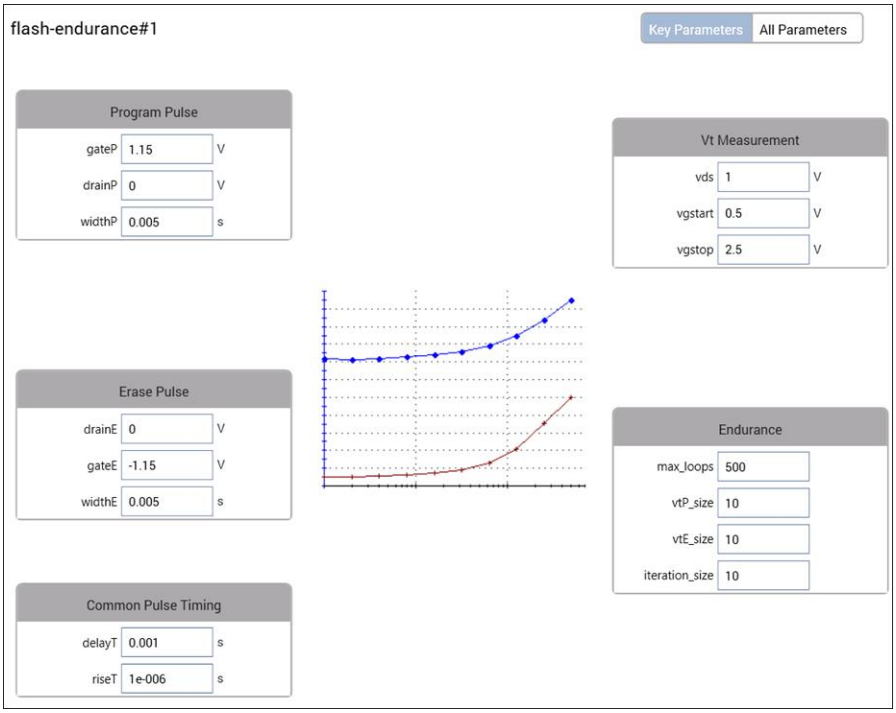
- Group parameters logically
- Add an image of the test to illustrate the overall test capability
- Add tooltips for each parameter

When creating the user interface definition for Key Parameters, display only the most important or the most commonly used parameters.

If you do not define the user interface, Clarius creates one automatically. The parameters are placed in groups around a default image of the device under test (DUT) graphic.

The following topics describe how to customize the Key Parameters user interface (UI) definition for a UTM. The following graphic is an example view of a UTM that is displayed in the Configure pane.

Figure 283: Key Parameters view of Configure pane



NOTE

If you change a UTM parameter name using KULT after the defining the user interface, make sure you update the UI 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 will not be displayed.

Allow access to the UTM UI editor

To use the UTM UI editor, you need to enable it in Clarius.

To enable the editor in Clarius:

1. Select **My Settings**.
2. Select **Environment Settings.**
3. Select **Allow access to UTM UI editor**.

NOTE

After making edits, you can clear “Allow access to UTM UI editor” to prevent accidental modifications to the UTM UI definitions.

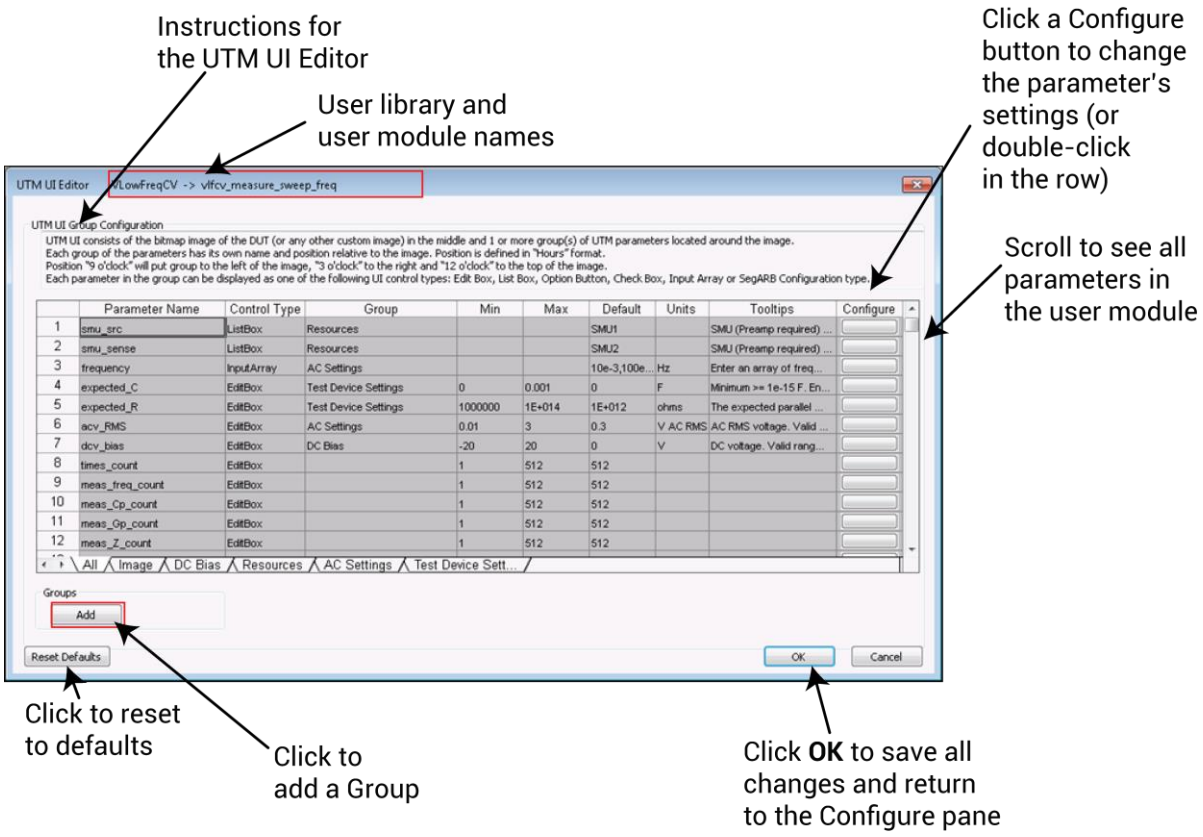
Open the UTM UI editor

To open the UTM UI editor:

1. Select a user test module (UTM).
2. Open the **Configure** pane.
3. Right-click in the **Configure** pane.
4. Select **Edit UTM UI**. The UTM UI Editor dialog box is displayed.

An example of the UTM UI editor dialog box is shown below.

Figure 284: UTM UI editor



Select groups

Groups organize parameters into related groups on the Clarius Configure pane. For example, in the example in Define the user interface for a user test module (on page 6-144), parameters are organized into the groups Program Pulse, Erase Pulse, Common Pulse Timing, Vt Measurement, and Endurance.

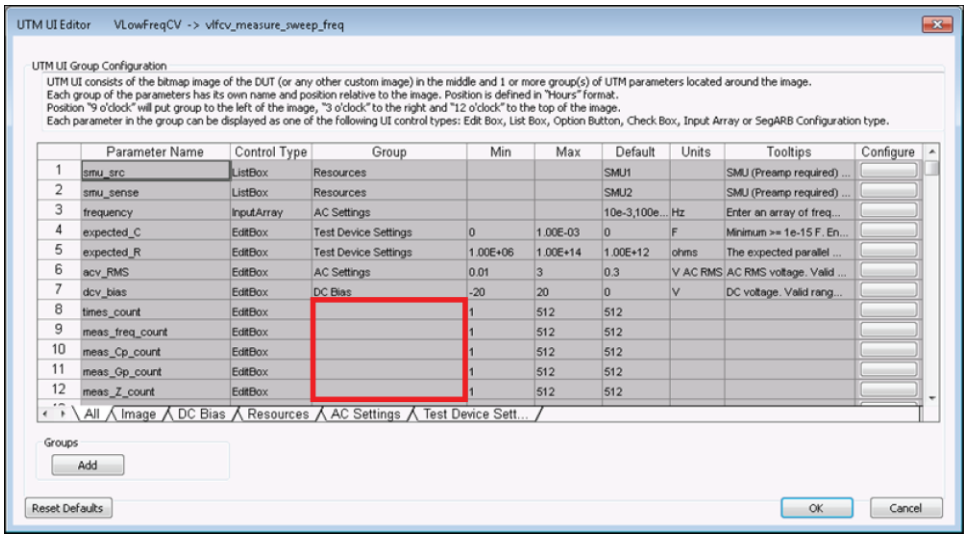
The maximum number of parameters in a group is ten.

When creating groups, keep the following items in mind:

- A UTM UI definition must have at least one group.
- Each group must have a unique name.
- Each group is shown in a tab in the UTM UI Editor.
- The All tab contains all parameters. If no group is displayed in the Group column, these parameters are not displayed in the Configure pane.
- Display only the important or commonly changed parameters; a user interface with fewer parameters is easier to understand and use than one with too many parameters.

You do not need to place all parameters in a group. Only place those parameters that you want to display to the user. For example, 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. The unassigned parameters are not shown in the Key Parameters pane but are available through the All Parameters pane.

Figure 285: Parameters not assigned a group (not displayed in Key Parameters)

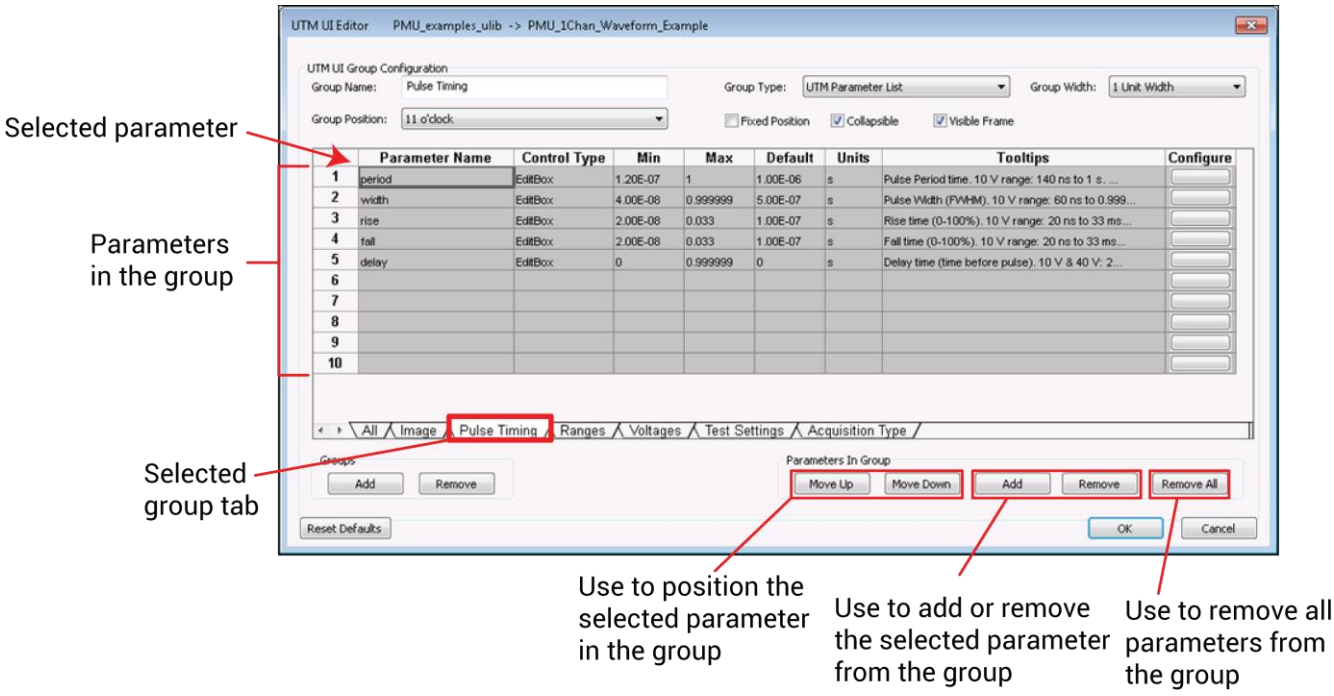


Add a group

To add a group:

1. Click **Add** (near the lower left corner of the UTM UI Editor). A new tab for the group is added.
2. Complete the **Group Name**. The group name entered in this view appears exactly as entered. Use standard characters a-z, A-Z, 0-9, and space.
3. Select the **Group Position**. This position is in relation to the UI image bitmap. Select a clock hour from the Group Position list. 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 4200A-SCS display, there is limited space at 12 o'clock (above the image) and 6 o'clock (below the image).
4. Select the **Group Type**.
5. Select **Fixed Position**, **Collapsible**, and **Visible Frame** as needed.
6. Add and configure the parameters. Refer to Edit the attributes for a test parameter (on page 6-151) for detail.
7. Click **OK**.

Figure 286: Example group view for pulse timing

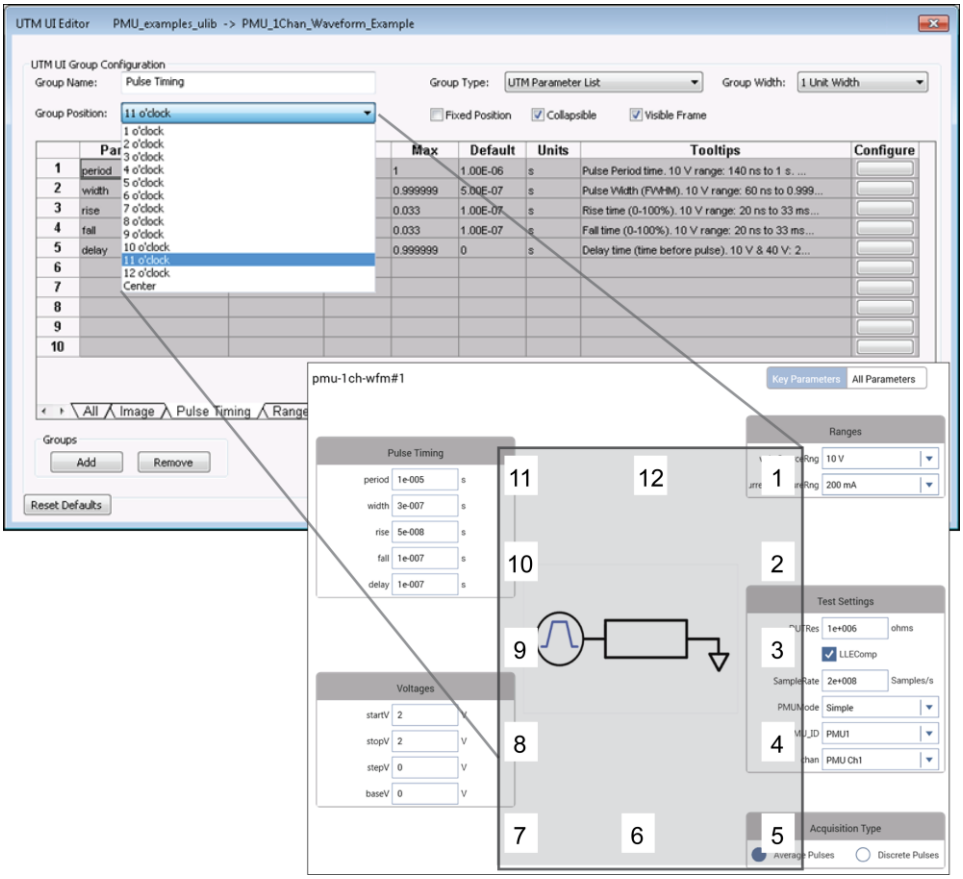


Modify a group

To modify a group:

1. Click the tab for the group. 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 to the left of the All tab).
2. You can change the following group-level items (see the example in Add a group (on page 6-148)):
   - **Group name**.
   - **Group position**.
   - **Parameter order**: Select a parameter row, and then click **Move Up** or **Move Down** to change the position of the parameter 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 remove all parameters from the group, click **Remove All**. To add a parameter, click **Add** and then select and configure the parameter.

Figure 287: UTM UI Editor group position example



Define an image for the user interface

You can add an image for the UTM. The image is displayed in the Configure pane when Key Parameters is selected. Images must be:

- In bitmap format (`.bmp`)
- 120x100 pixels to 480x400 pixels
- File size less than 500 kB

You can use full color bitmaps. Larger pixel size images render better on the 4200A-SCS screen.

Each UTM must have only one image. If nothing is defined, Clarius uses the image of the device that the UTM is structured under in the project tree.

The default Clarius images are stored in the source directory of the user library. For example, an image for a UTM in the `VLowFreqCV` user library is stored in:

```
C:\s4200\kiuser\usrlib\VLowFreqCV\src
```

To add an image:

1. In the UTM UI Editor, select the **Image** tab.
2. For the Group Position, select the location for the image. You can select Center or a clock location that orients around the center.
3. For the Group Type, select **UI Image**.
4. To keep the graphic at a fixed size, regardless of Clarius window scaling, select **Fixed Size Image**. When you use a fixed image size, you might need a smaller pixel size image for large numbers of test parameters or groups.
5. Double-click the row in the table and select an image.
6. Select **OK**.

NOTE

Some items in the UTM UI Editor can make UI level modifications; if you change any of these items, you will change them for the entire UTM UI. These items include UI Image and Fixed Size Image.

Edit the attributes for a test parameter

You can edit the display attributes of a test parameter. You can set the display attributes from the All tab or from a specific group tab.

To edit the attributes:

1. Double-click a row on the tab to open the UTM UI Parameter Configuration dialog box.
2. Select the **Control Type**. Refer to Control types (on page 6-154) for detail on the options.
3. For the **Displayed Group**, select the group for this parameter.
4. If the **Minimum Value** and **Maximum Value** fields are not dimmed, you can enter values (integer and double types allowed). If they are dimmed, they were automatically assigned based on the KULT user module. If you need to change these values, you must change them in KULT.
5. Set the **Default Value**. The existing value comes from KULT, but you can change it here as needed.
6. Set the **Displayed Units**. These are the units of measure for the value. Note that no conversions are made, so these must be the same units as the applicable command.
7. Enter the **Displayed Tooltips**. This is information that is displayed when the user hovers over a field with a mouse or long-holds using the touchscreen. Avoid using non-text characters, such as line feeds or carriage returns, in this field. Use standard characters (a-z, A-Z, 0-9, and space) and no symbols. Keep tooltips short, use simple present tense, use clear and consistent language, and check your spelling.

The figure below illustrates a sample UTM UI Parameter Configuration dialog box. The parameters that you can configure depend on which tab was used to enter the parameter configuration dialog box. If the configuration dialog box is accessed from the All tab, all the parameter names are available in the related menu; if accessed from a tab for a group, only the parameter names available in the group are available.

Figure 288: GUI Configuration for the voltsSourceRng parameter (ListBox)

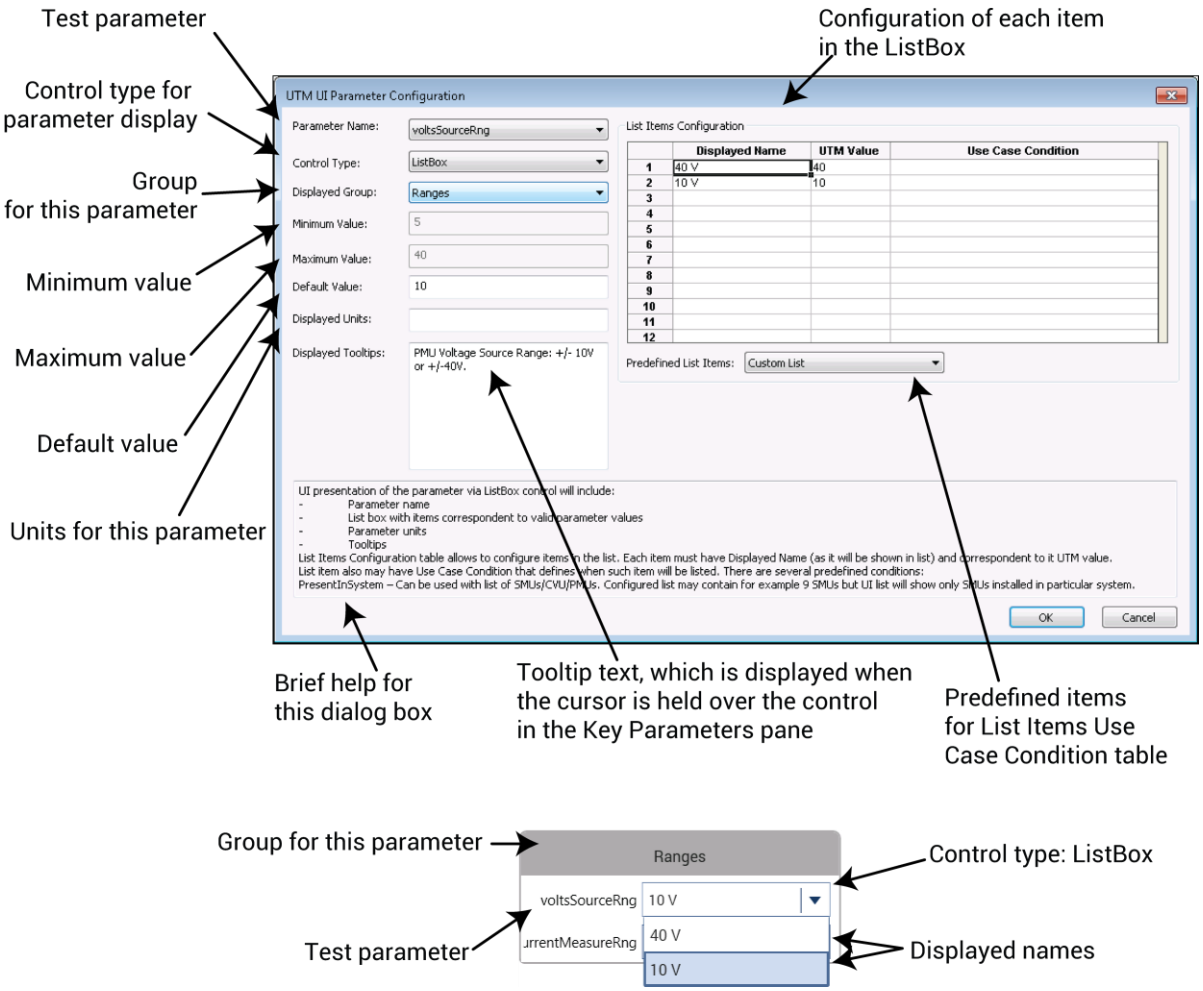
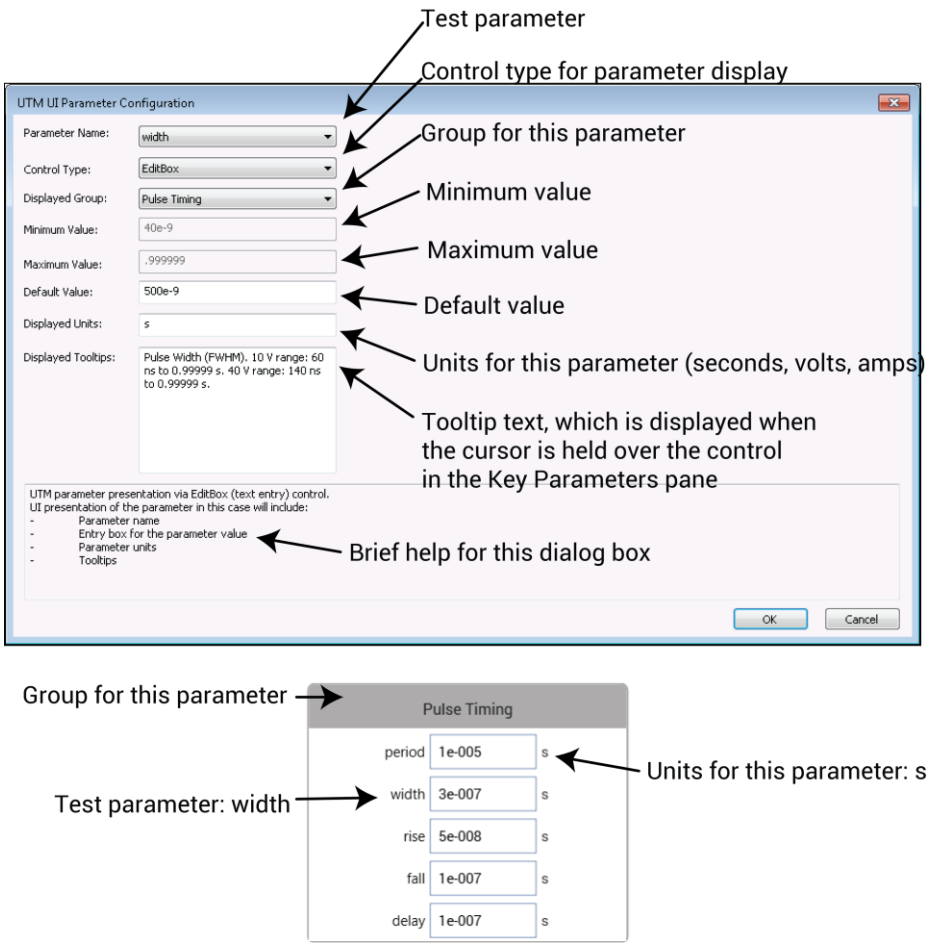


Figure 289: GUI Configuration for the width parameter (EditBox)



NOTE

The Minimum, Maximum, and Default Values are defined in the KULT user module. To change the Minimum and Maximum Values, you must use KULT. You can edit the Default Value in Clarius.

Control types

You can set the control type to one of six different types for entry and display of a parameter in the Key Parameters pane:

- EditBox
- ListBox
- CheckBox
- OptionBtn
- InputArray
- SegARBConfig

EditBox

The EditBox Control type is the simplest method to allow users to change a parameter value. You can use this control type for source values (such as voltage or current), pulse timing parameters, or any other parameter that has a wide range of continuous values. 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 UI views.

The EditBox Control type is shown in Edit the attributes for a test parameter (on page 6-151).

ListBox

Use a list box to specify a value that the user can select, such as a measure or source range.

A ListBox can hold a minimum of 2 to a maximum of 12 values.

If a ListBox Control type is chosen for a parameter, fill in at least one row of the List Items Configuration.

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, the Default Value field is populated. However, you can change it using this dialog box.

Be sure to enter appropriate Displayed Units, if applicable. Note that the Displayed Units field does not affect the test or parameters.

You can enter a tooltip to assist the user in understanding the parameter values. Enter text in the Displayed Tooltip field with a short informative phrase or sentence. When entering tooltips:

- Avoid using non-text characters, such as line feed or carriage return.
- Use standard characters (a-z, A-Z, 0-9, and space) and no symbols.
- Use simple present tense.
- Use clear and consistent language.
- Check your spelling.

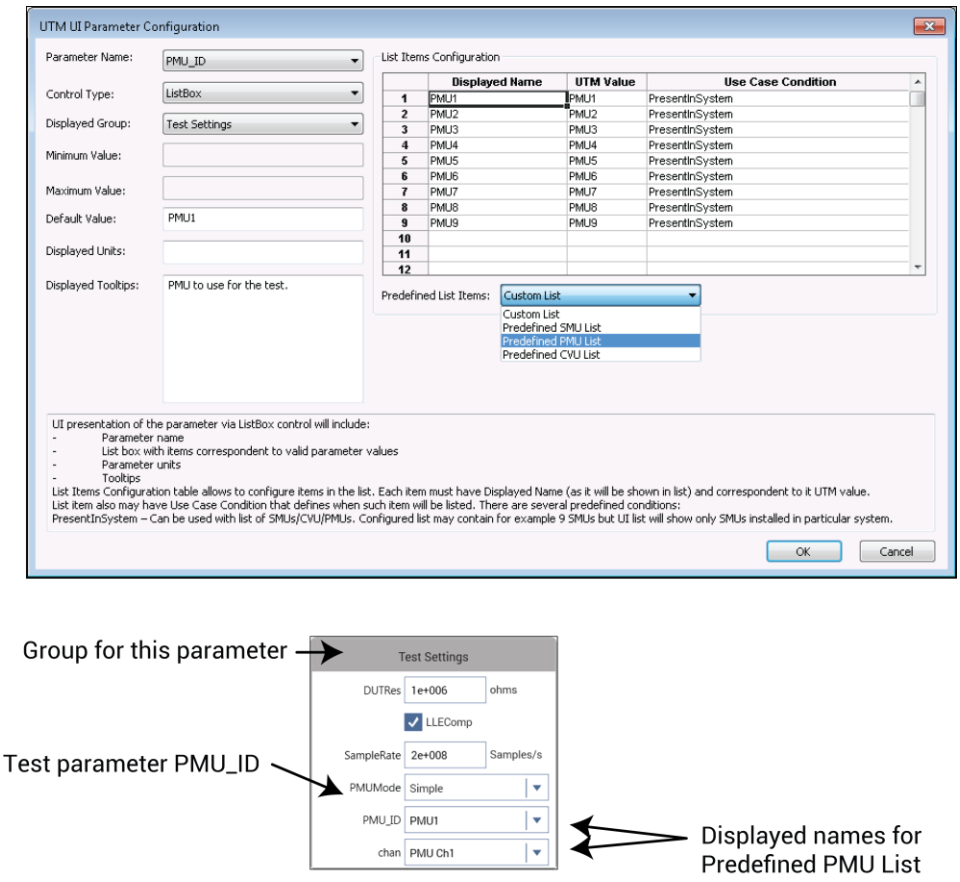
When finished, click **OK**.

You can set conditions that determine when a list box choice is displayed. If the Use Case Condition is true, the Displayed Name for the row appears in the ListBox.

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 UI view has information about the system configuration. You can use this to direct the user to select appropriate parameter values. For example, if a particular current range is only available when a 4225-RPM or SMU preamplifier is connected on the chosen channel, you can add logic to permit these use case conditions.

There are predefined lists you can select to automatically check to see if an option is in the system. For example, the figure below shows a list box that is set for the Predefined PMU List. The PMUs are only displayed if they are available in the system.

Figure 290: ListBox UTM UI Parameter Configuration for PMU_ID



This figure shows the `PMU_ID` list in a system with two 4225-PMUs. SMU lists, CVU lists, and a customizable list are also available as items in the Predefined List Items list.

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. Refer to ListBox “Use Case Condition” keywords and operators (on page 6-160) for information about these conditions.

The following figure shows the UTM UI Parameter Configuration dialog box for the parameter `currentMeasureRng`.

The Use Case Condition field conditionally displays the Displayed Name in the list box. The expression you enter in 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 the figure below, the Use Case Condition of the first line means, “If voltsSourceRng is equal to 40, then display 800 mA” in the list. This effectively allows the 800 mA range to display and be selected when the voltage range is set to 40 V.

Note the complexity of the `currentMeasureRdg` lower-current use case conditions. These use case conditions account for the three different sets of current measure ranges, which depend on the voltage range selected (10 V or 40 V) and the presence of a 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 below. ListBox “Use Case Condition” keywords and operators (on page 6-160) lists keywords and operators available for use in the Use Case Condition field.

Figure 291: ListBox UTM UI Parameter Configuration for currentMeasureRng

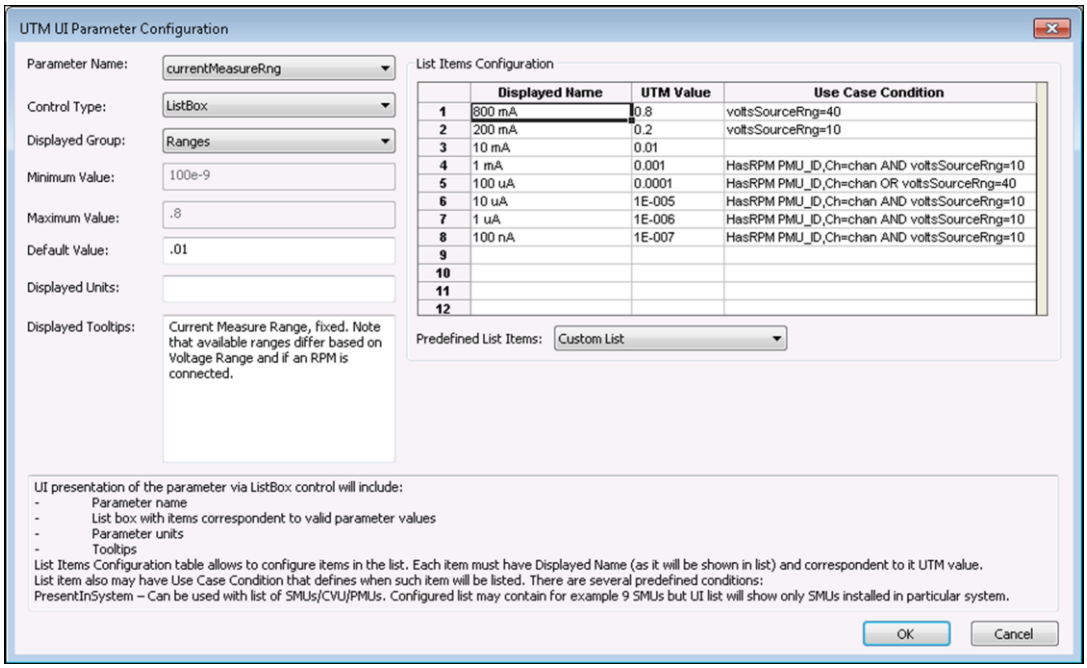
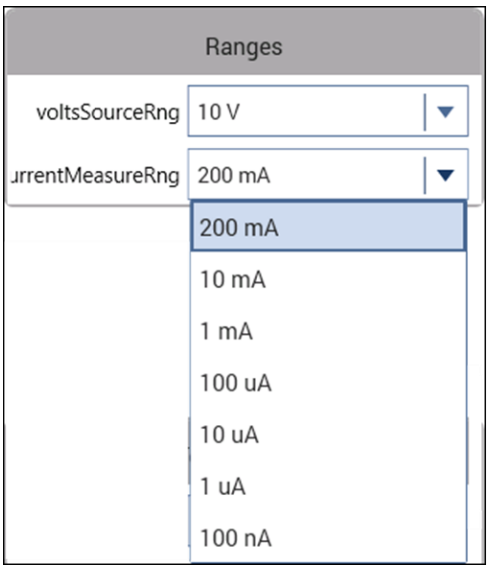


Figure 292: currentMeasureRng list box with a connected RPM



	Displayed Name	UTM Value	Use Case Condition	Comment

	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; the 40 V range has 800 mA, 10 mA, and 100 µA current measure ranges
2	200 mA	0.2	`voltsSourceRng=10`	Display this name when the chosen voltage range = 10 V
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. The 40 V range has 800 mA, 10 mA, and 100 µA current measure ranges
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 nA	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)

ListBox “Use Case Condition” keywords and operators

Keyword or operator	Explanation	Comment
`PresentInSystem`	Used for the 4200A-SCS instrument: 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 preamplifier 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

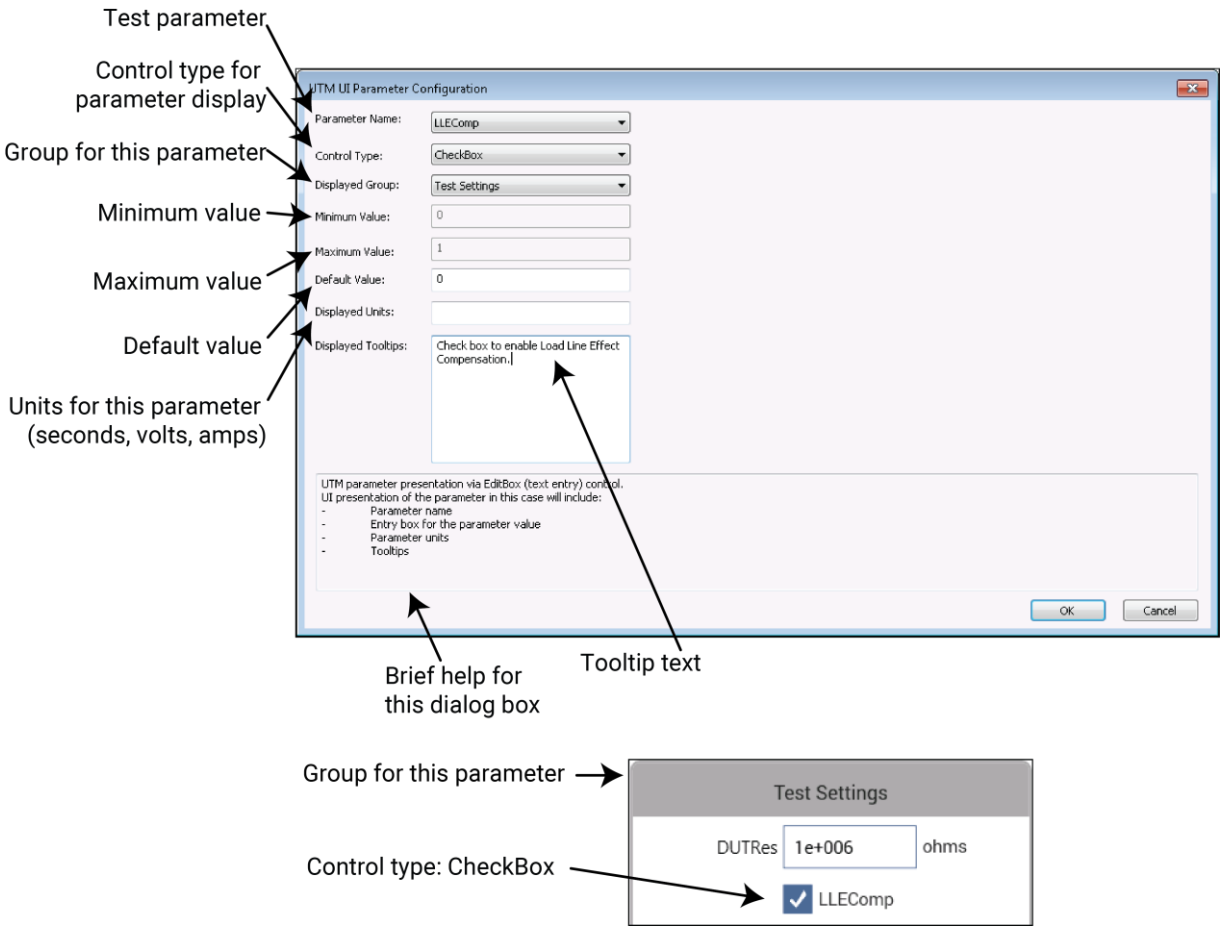
Use a check box control (CheckBox) for parameters that have two values or states. This control returns a zero (`0`) to indicate the box is not selected or a one (`1`) to indicate that it is selected.

For example, the LLEComp parameter from the `PMU_1Chan_Waveform_Example` user module has two states: Disabled and Enabled. These values refer to the state of the LLEC compensation and is used in the LPT command `pulse_meas_wfm`.

Provide or change the Default Value and supply explanatory tooltip text. The actual parameter name is used as a label, so make sure to provide a tooltip that helps explain the check box. The tooltips are available when users hover over the field or long press on the check box on the touchscreen.

If needed, you can also note the unit of measurement in the Displayed Units field for reference by the UTM UI programmer. When finished, click **OK** to exit this dialog box.

Figure 293: CheckBox UTM parameter UI configuration



NOTE

The Minimum, Maximum, and Default Values are defined in the KULT user module. To change the Minimum and Maximum Values, you must use KULT. You can edit the Default Value in Clarius.

OptionBtn control

Use the option button control (OptionBtn) when only one item of a group may be selected. This control is also useful for a parameter with a limited number of values (from 2 to 4).

NOTE

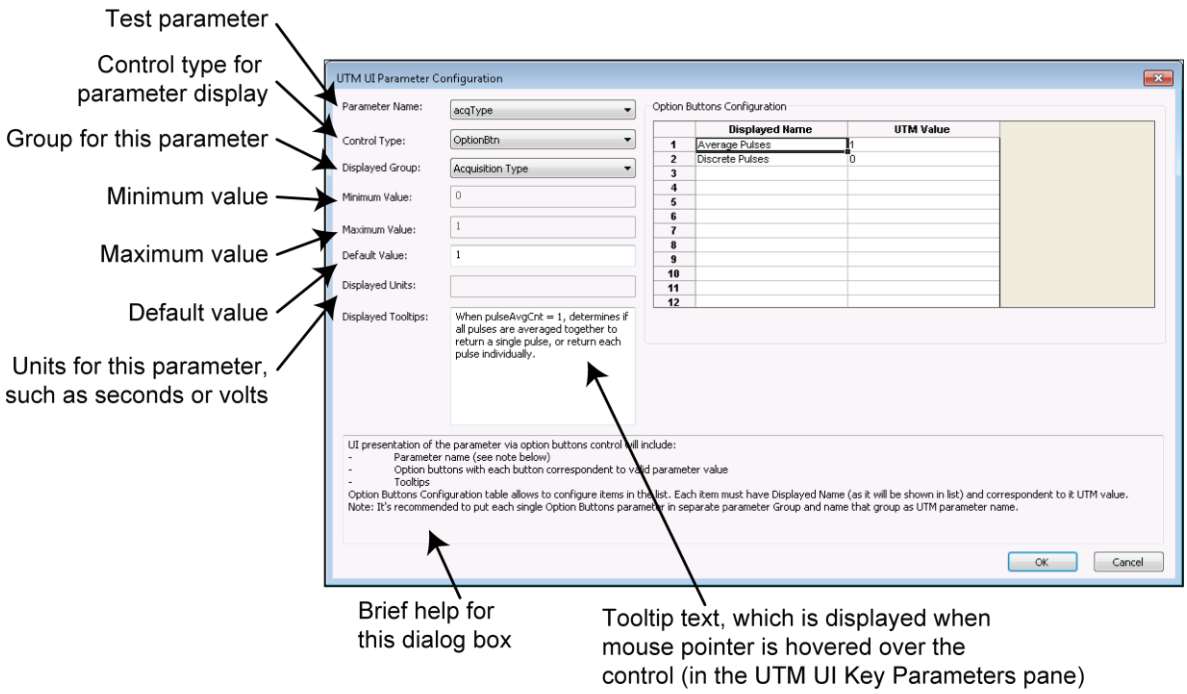
List boxes take up less space in the Key Parameters pane because only one state displays unless being selected; the option button control must show all choices.

See an example of this control in the lower right of the figures in Example of using the editor (on page 6-181) for the `AcqType` parameter (the Acquisition Type group is an option button control).

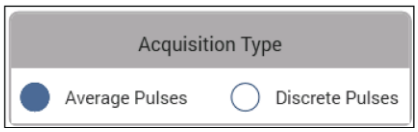
Using an option button permits different values to be returned for each choice in the same way as a list box; a check box control only returns a zero (0) or a one (1).

In the example shown here, these values correspond to the two measurement modes for the spot mean measurement LPT command pulse_meas_wfm (on page 14-123). The two measurement modes are Discrete and Average. This option determines whether measurements from multiple pulses (set by the *`pulseAvgCnt`* parameter) are averaged together into a single value (average) or as each pulse with its own measurement (discrete). Refer to Waveform measurements (on page 5-94) for additional information on this measurement mode.

Figure 294: OptionBtn UTM UI Parameter Configuration dialog box



Group as shown in Key Parameters pane:



NOTE

The Minimum, Maximum, and Default Values are defined in the KULT user module. To change the Minimum and Maximum Values, you must use KULT. You can edit the Default Value in Clarius.

InputArray control

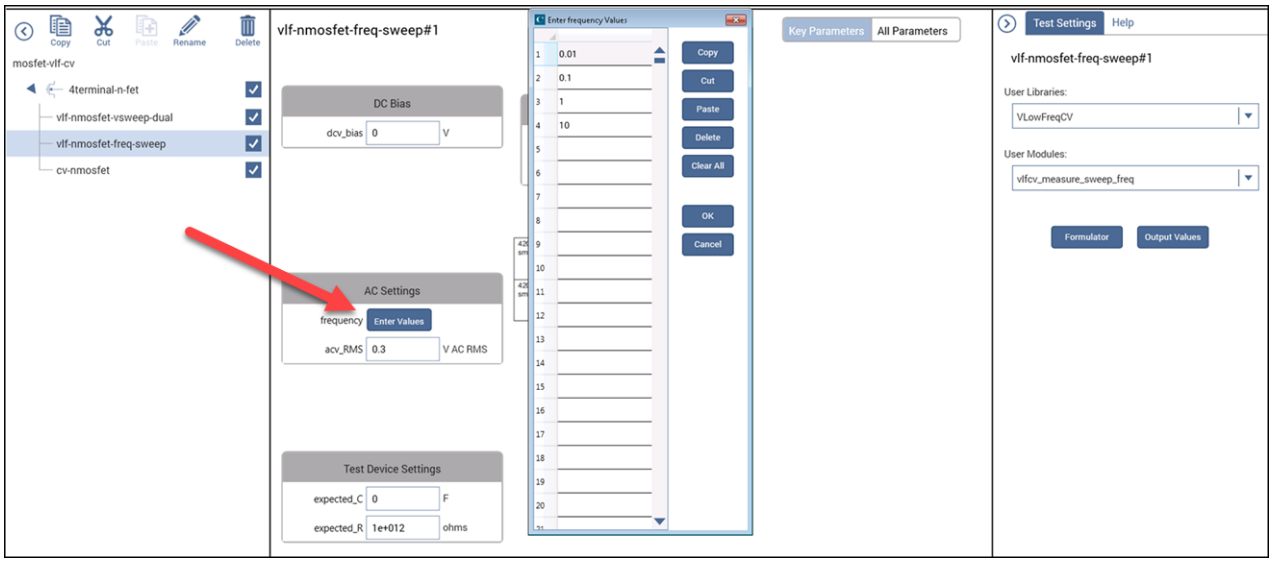
Use the input array control (InputArray) for array input types in user modules. Configure the parameter variable types using the Parameters tab in KULT (see Parameters tab area (on page 8-5)).

For this control type, the example uses the `vlfcv_measure_sweep_freq` user module, which is in the `VLowFreqCV` user library. When placed in the UTM UI, the user can select the Enter Values button and enter array values, as shown in the figure below.

NOTE

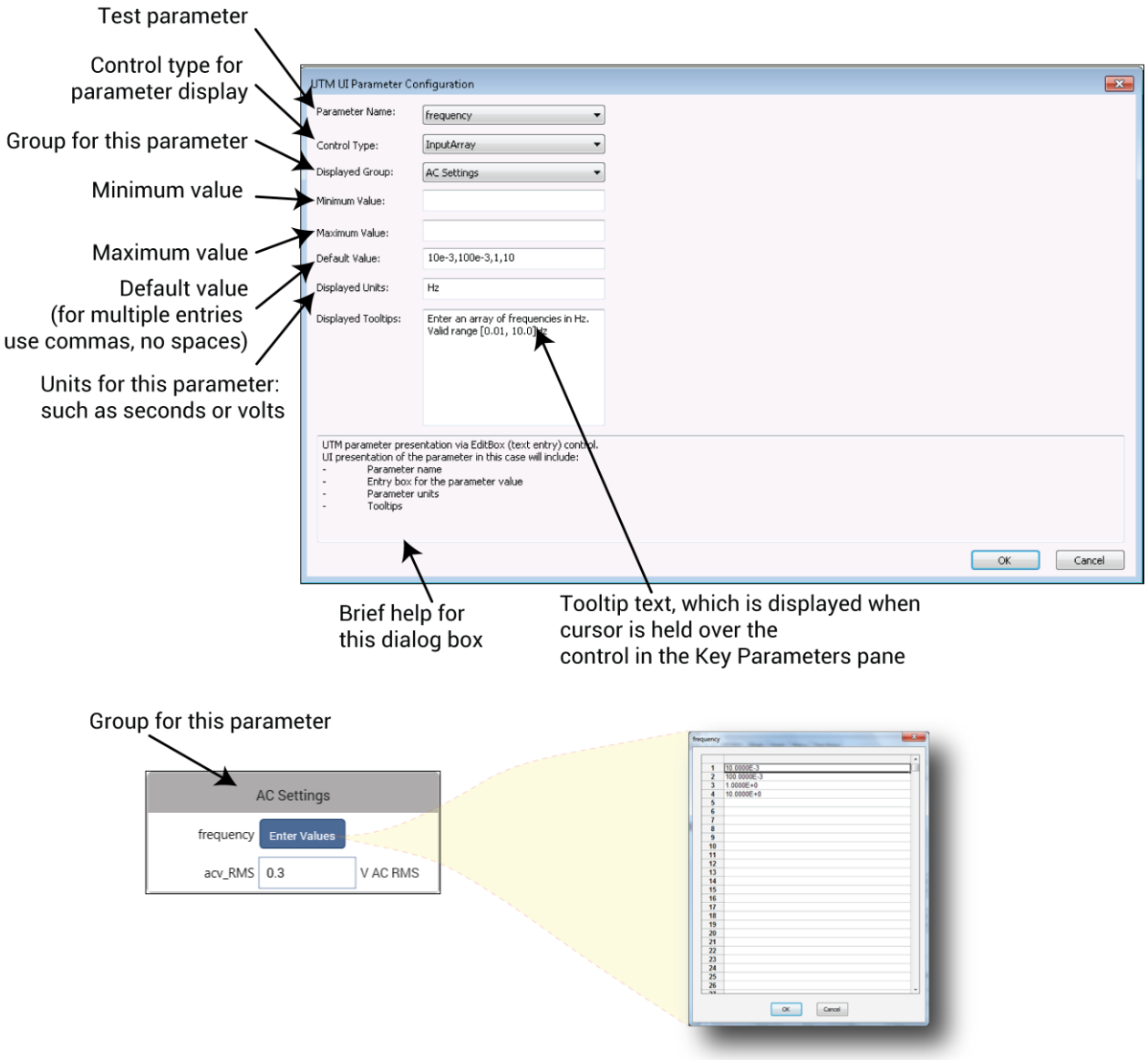
Make sure the values the user enters in the array tables are contiguous (no blank rows in the middle of filled cells). An error results if one (or more) blank row is in the table before the end of the data.

Figure 295: UTM Key Parameters InputArray control



The configuration of an input array control, shown below, 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 UI 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 296: InputArray UTM parameter UI configuration

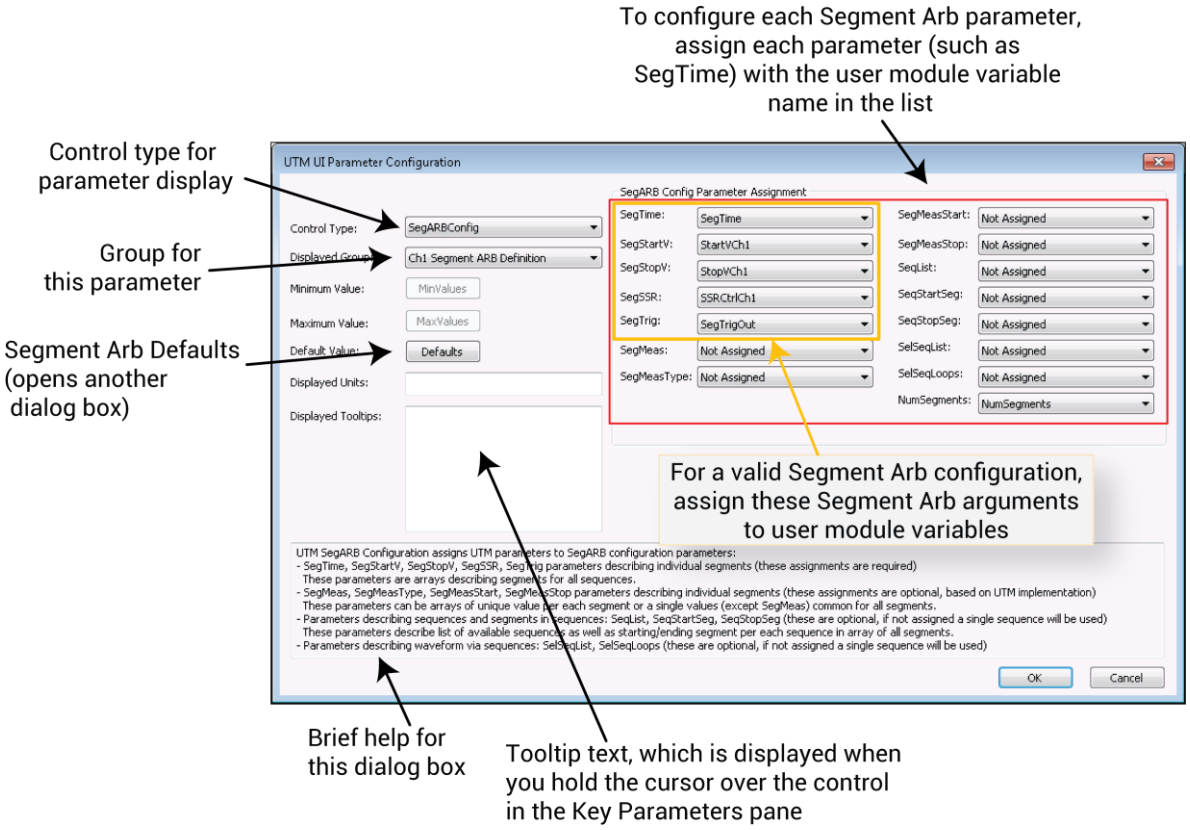


SegARBConfig

The `SegARBConfig` is the most complex control type available for the UTM UI. The Segment Arb® mode 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 14-140) and seg_arb_waveform (on page 14-144). There are two dialog boxes that configure the Segment Arb UI Key Parameters:

- The UTM UI Parameter Configuration dialog box, shown in the figure below. This dialog box is the primary dialog box that you use to configure the `SegARBConfig` control.
- The Segment Arb Defaults Configuration dialog box, shown in the figure "Segment Arb Defaults Configuration" shown in Starting with default waveforms (on page 6-175).

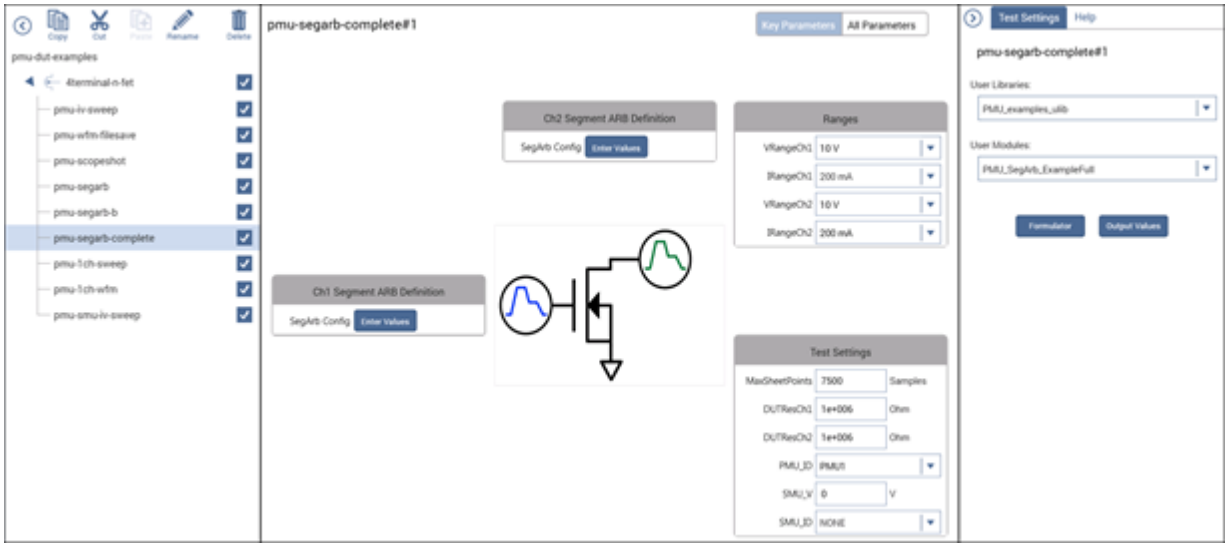
Figure 297: SegARBConfig UTM UI Parameter Configuration dialog box



PMU_SegArb_ExampleFull user module

For this control type, the `PMU_SegArb_ExampleFull` user module is used in the example. This module is included in the `pmu-dut-examples` project as the UTM `pmu-segarb-complete`, shown below. This figure 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.

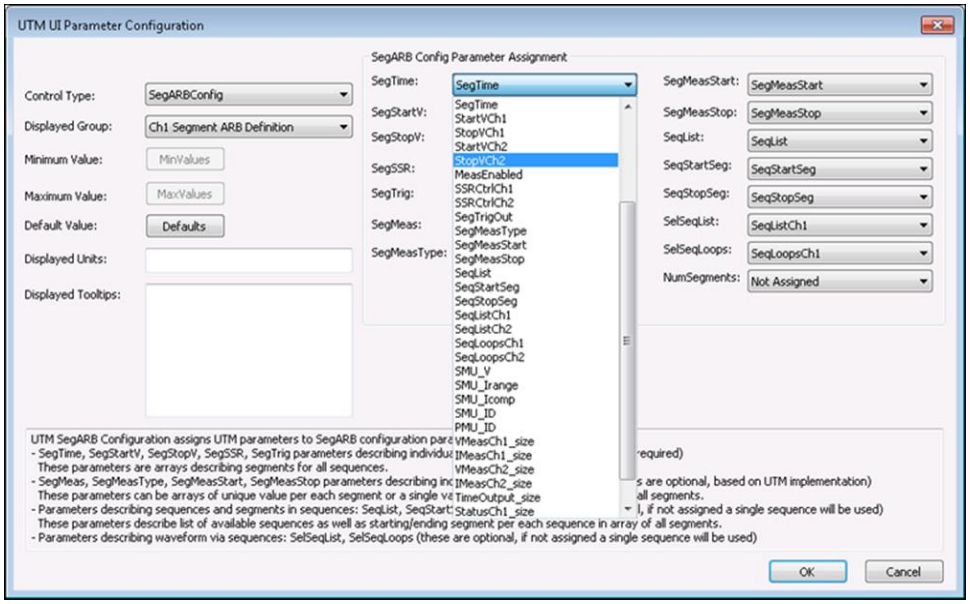
Figure 298: UTM Configure pane for PMU_SegArb_ExampleFull



SegARBConfig multiple parameters

In addition to the settings that are configured like the other control types, such as displayed group, displayed units, and 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 (shown in the figure in SegARBConfig (on page 6-166) and the figure below). Also, each parameter assignment involves many parameters (see PMU_SegArb_ExampleFull user module (on page 6-167)). This is in contrast to other controls. For example, the `InputArray` control has a single parameter in its parameter configuration dialog box.

Figure 299: SegARBConfig Parameter Configuration listing variables



SegARBConfig control parameters

The following tables summarize all the parameters in the `SegARBConfig` control. In the tables, each parameter is shown mapped to its parameter target. As mentioned above, the Segment Arb functionality is primarily in two LPT commands: `seg_arb_sequence` and `seg_arb_waveform`. In addition, a few parameters are unique to the `SegARBConfig` control.

SegARBConfig parameter names mapped to LPT function seg_arb_sequence

In this table, the variable name is in the function seg_arb_sequence (on page 14-140).

Parameter name	Variable name in function	Type	Required in SegARBConfig?	Description
`SegTime`	*Time*	Double array	Yes	Time duration for each segment
`SegStartV`	*StartV*	Double array	Yes	Segment start voltage
`SegStopV`	*StopV*	Double array	Yes	Segment stop voltage
`SegSSR`	*SSR*	Integer array	Yes	Solid State Relay control, per segment
`SegTrig`	*Trig*	Integer array	Yes	Trigger output state for each segment
`SegMeasType`	*MeasType*	Integer array or Integer[1]	No	Measurement Type: None, spot mean, or waveform
`SegMeasStart`	*MeasStart*	Double array or Double[1]	No	Start point for measurement window, for each segment; from 0 to 1 (100%)
`SegMeasStop`	*MeasStop*	Double array or Double[1]	No	Stop point for measurement window, per segment; from 0 to 1 (100%)
`NumSegments`	*NumSegments*	Integer	No	Number of segments in a sequence

[1] If using a single value instead of an array, the `SegARBConfig` control automatically assigns the value to each segment.

SegARBConfig parameter names mapped to SegARBConfig

In this table, the variable name is in the function `SegARBConfig`.

Parameter name	Variable name in function	Type	Required in SegARBConfig?	Description
`SegMeas`	*SegMeas*	Integer array	No	Measurement Enabled[1]
`SeqList`	*SeqList*	Integer array	No	List of defined sequences in SegARBConfig
`SeqStartSeg`	*SeqStartSeg*	Integer array	No	Start segment array value in SegARBConfig
`SeqStopSeg`	*SeqStopSeg*	Integer array	No	Stop segment array value in SegARBConfig

[1] `SegMeas` parameter turns a measurement on or off for each segment and is independent of the `SegMeasType` (based on implementation in user module).

Parameter targets mapped to LPT function seg_arb_waveform

In this table, the variable name is in the function seg_arb_waveform (on page 14-144).

Parameter name	Variable name in function	Type	Required in SegARBConfig?	Description
SelSeqList	*Seq*	Integer array	No	List of sequences that define the Seg-Arb waveform
SelSeqLoops	*SeqLoopCount*	Integer array	No	Array of loops values for each sequence in the Seg-Arb waveform

Except for the NumSegments parameter, the SegARB Config Parameter Assignment parameters are arrays. 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, the UTM UI displays it as a ListBox. You must associate each Segment Arb parameter with the appropriate variables in the user module. If not, an error will result.

The other parameters are optional and based on the implementation of the Segment Arb mode in a particular user module. Optional parameters mean 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 waveform.

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 14-140) function and the multi-sequence waveform using the seg_arb_waveform (on page 14-144) function. These sequencing parameters allow data of each sequence to be stored in 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 with nine segments and sequence two with 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 "SegARBConfig uses SeqList, SeqStartSeg, and SeqStopSeq to store sequence," below). This mapping applies to all array-parameters in the seg_arb_sequence. It is then used 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 "SegARBConfig uses SeqList, SeqStartSeg, and SeqStopSeq to store sequence," below), 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 rows are in the table before the end of the data.

Figure 300: Segment Arb UI UTM configuration for channel 1

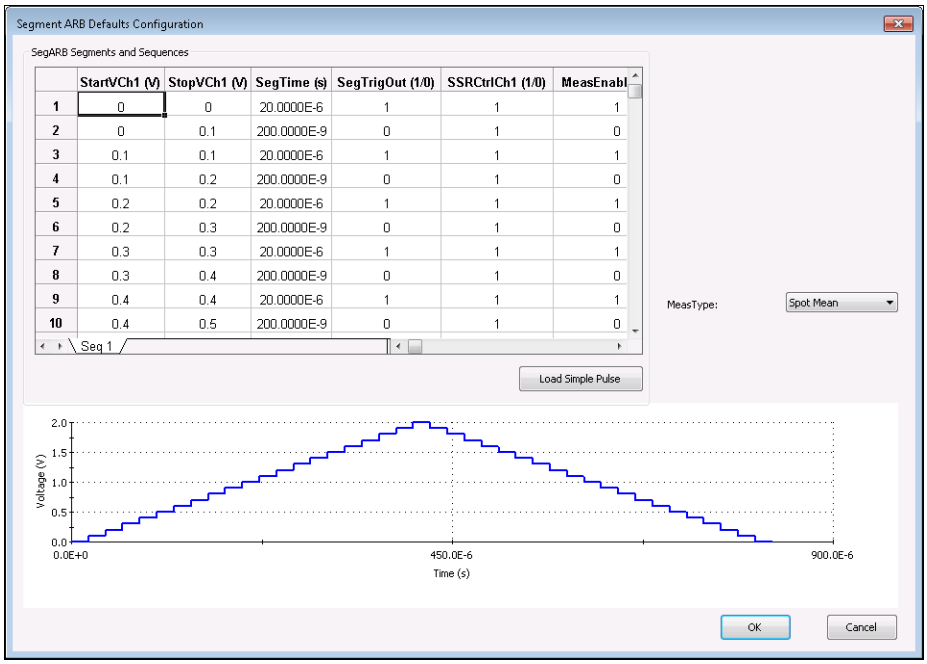


Figure 301: Two-sequence Segment Arb waveform

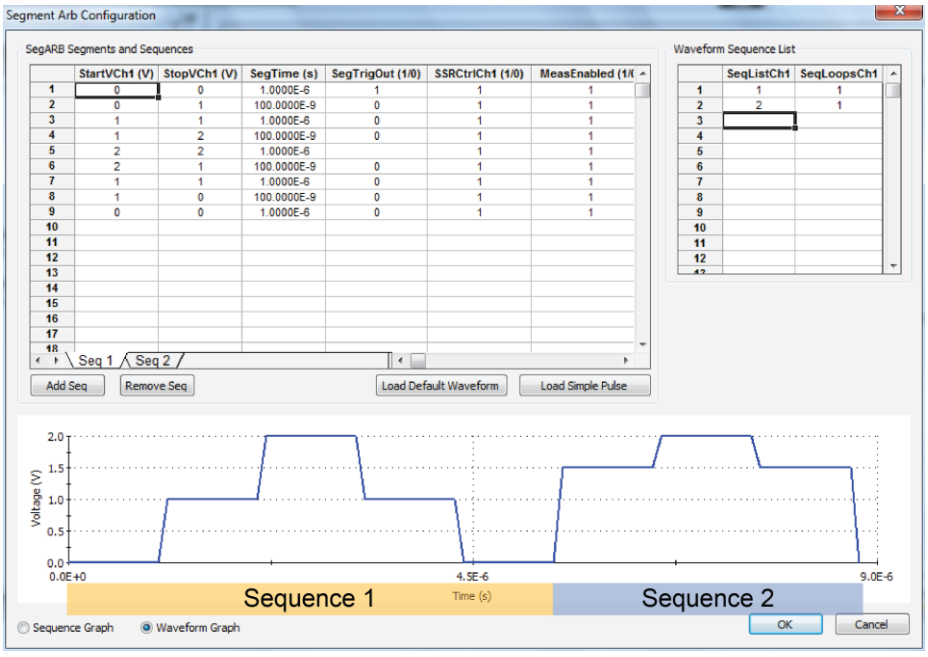
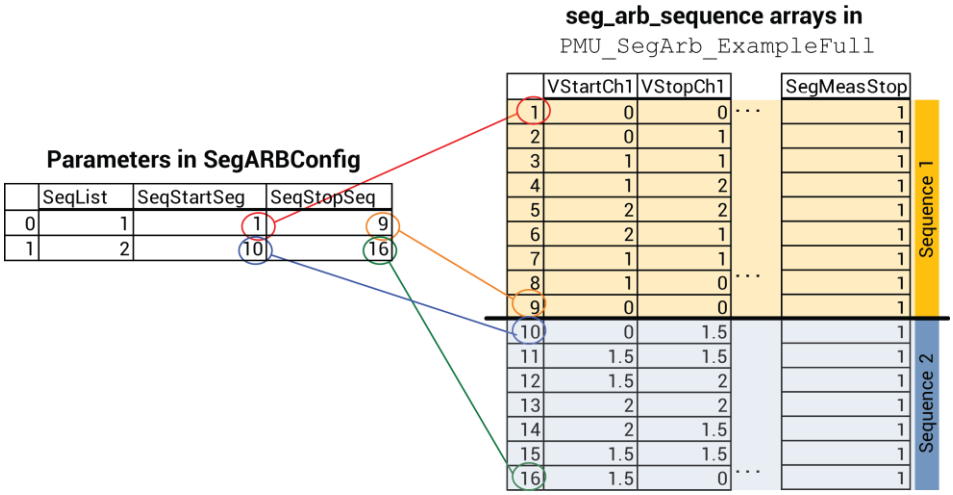


Figure 302: SegARBConfig uses SeqList, SeqStartSeg, and SeqStopSeq to store sequence



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)`

Assigning SegARBConfig to a group

As with other Control Types, to appear in the UTM Key Parameters pane, the `SegARBConfig` must be assigned to a group. If a user module has more than one Segment Arb® waveform channel, place a `SegARBConfig` control for each channel in a separate group. Choose which channel is configured by selecting the specific variable name.

The figures below show the `SegARBConfig` for user module channel 1 and channel 2. For this example, selecting the variable named `StartVCH1` configures the `SegStartV` for channel 1, where selecting the variable named `StartVCH2` configures the `SegStartV` for channel 2.

Be sure to select appropriate variables for the `SegARBConfig` parameter assignment for each channel. 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 14-144)).

Figure 303: Channel 1 SegARBConfig parameter configuration

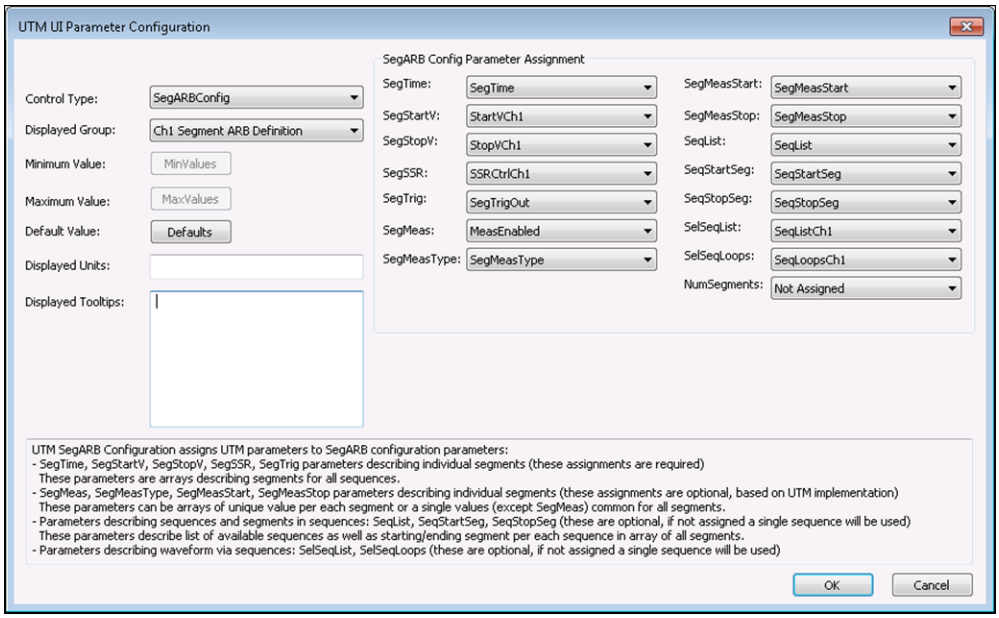
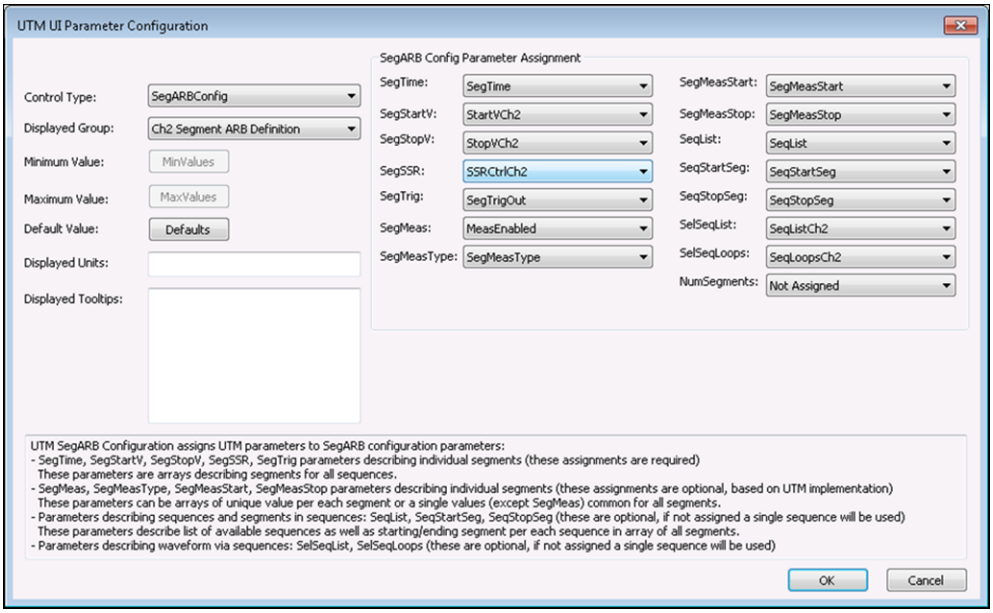


Figure 304: Channel 2 SegARBConfig parameter configuration



Starting with default waveforms

The `SegARBConfig` dialog box can supply default waveforms. Select the **Defaults** button shown in "Channel 2 SegARBConfig parameter configuration" in Assigning SegARBConfig to a group (on page 6-173).

You can use the created defaults to get started with a new UTM without having to input any parameter values. Figure "Segment Arb Defaults Configuration" below shows the created blank Segment Arb Default dialog box. To get a basic default waveform, select the **Load Simple Pulse** button. This results in the four segment pulse shown in "UTM UI Editor Segment Arb Defaults Configuration (after pressing Load Simple Pulse)" below.

NOTE

The stop voltage of a segment must equal the start voltage of the next segment.

Figure 305: Segment Arb Defaults Configuration

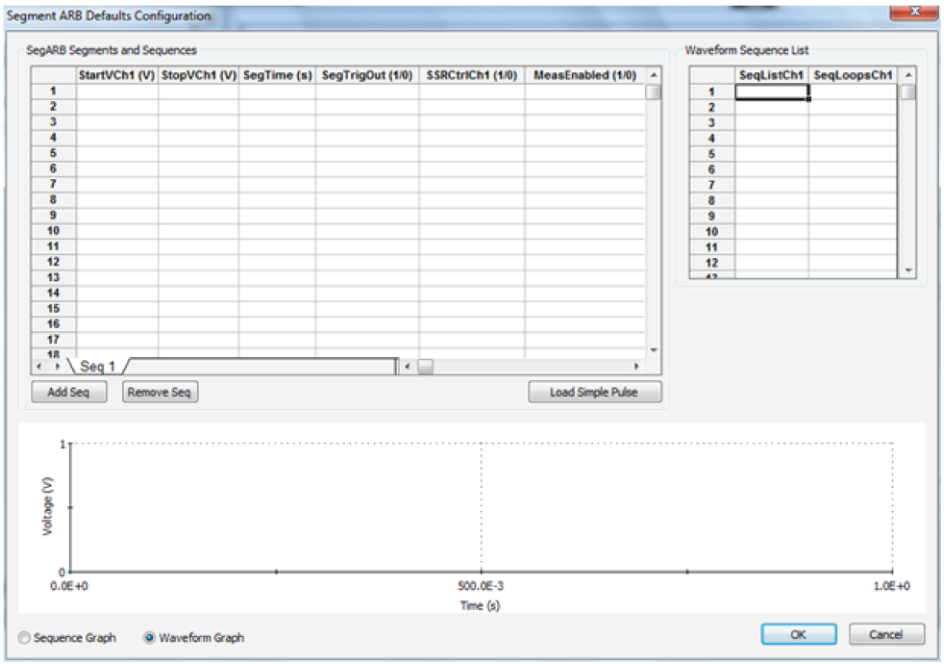
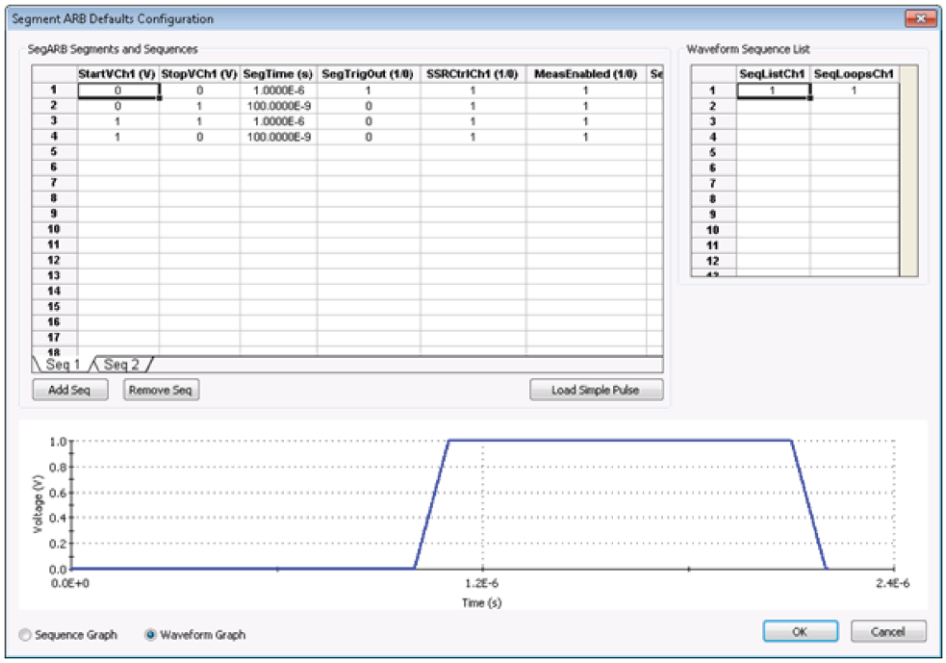


Figure 306: UTM UI Editor Segment Arb Defaults Configuration (after pressing Load Simple Pulse)



Defaults for this example

The following figures 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 307: UTM UI Editor Segment Arb Defaults Configuration for channel 1

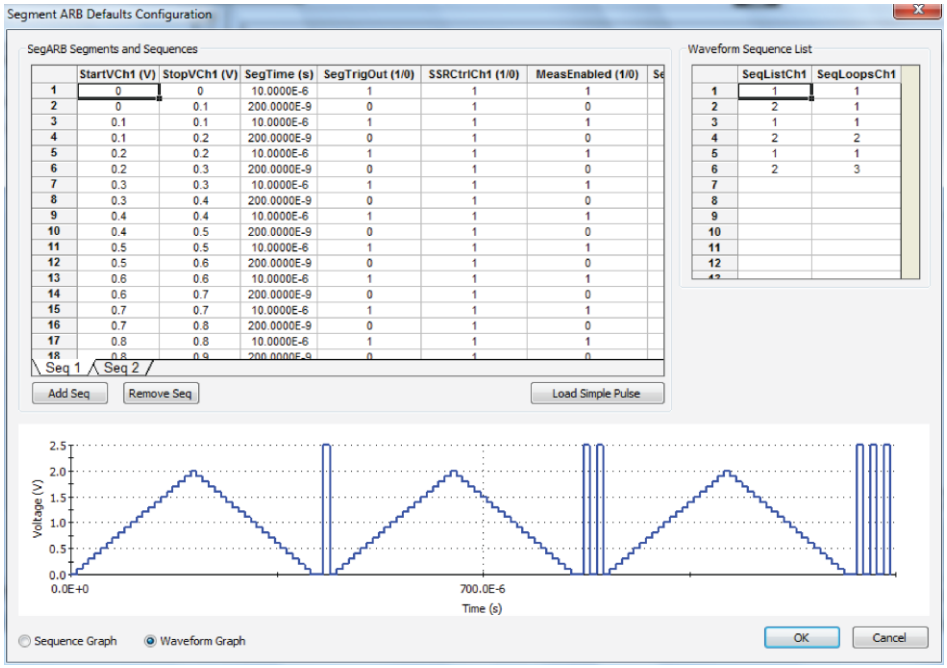
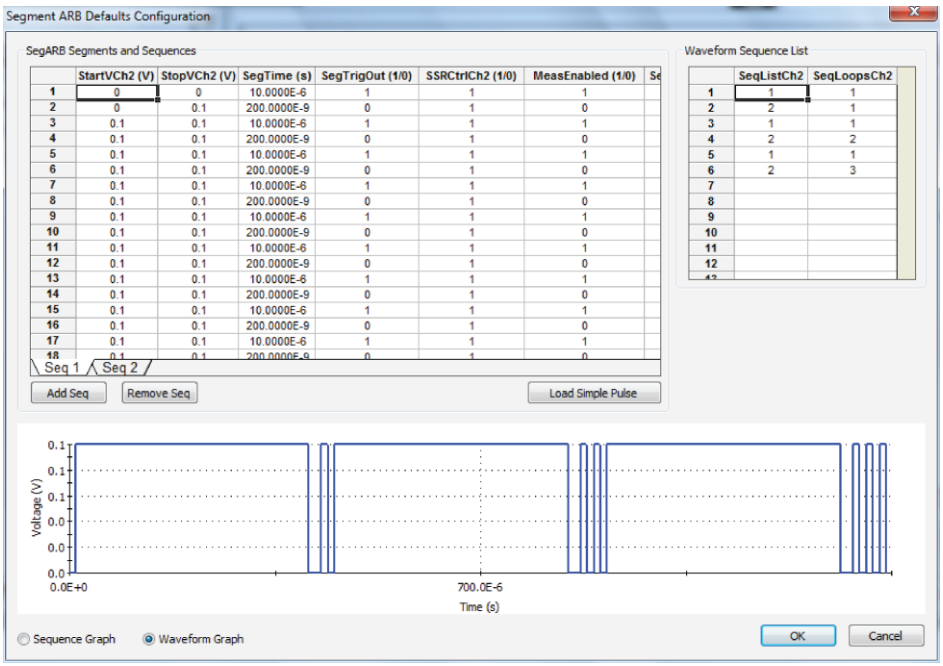


Figure 308: UTM UI Editor Segment Arb Defaults Configuration for channel 2

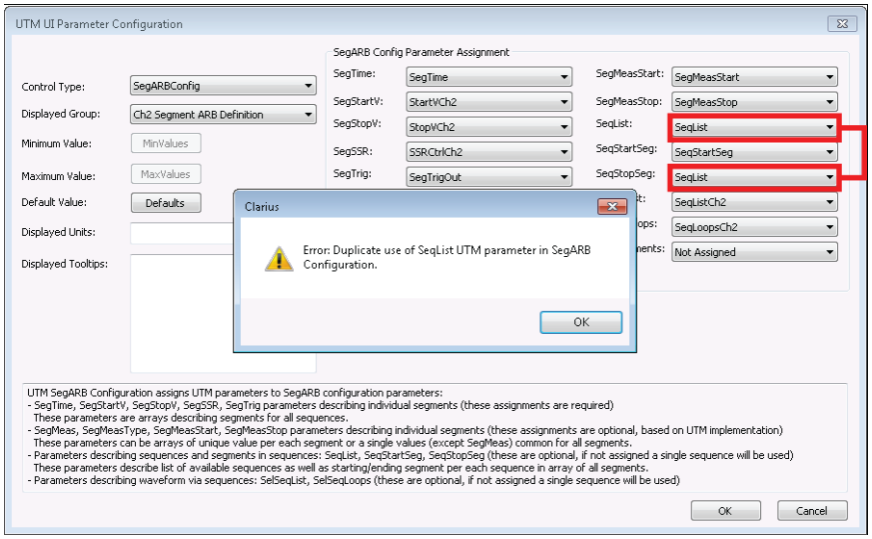


Error checking

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 missed assignments of parameters, the correct configuration relies on the selection of correct parameters chosen, as shown in the figures in Assigning SegARBConfig to a group (on page 6-173). For example, the configuration checks for duplicate parameters specified in a single channel.

The following figure shows an error when using the parameter `SeqListCh1` (in red boxes) twice in the Parameter Assignment.

Figure 309: Error as the result of duplicate use of parameter



Complete a change

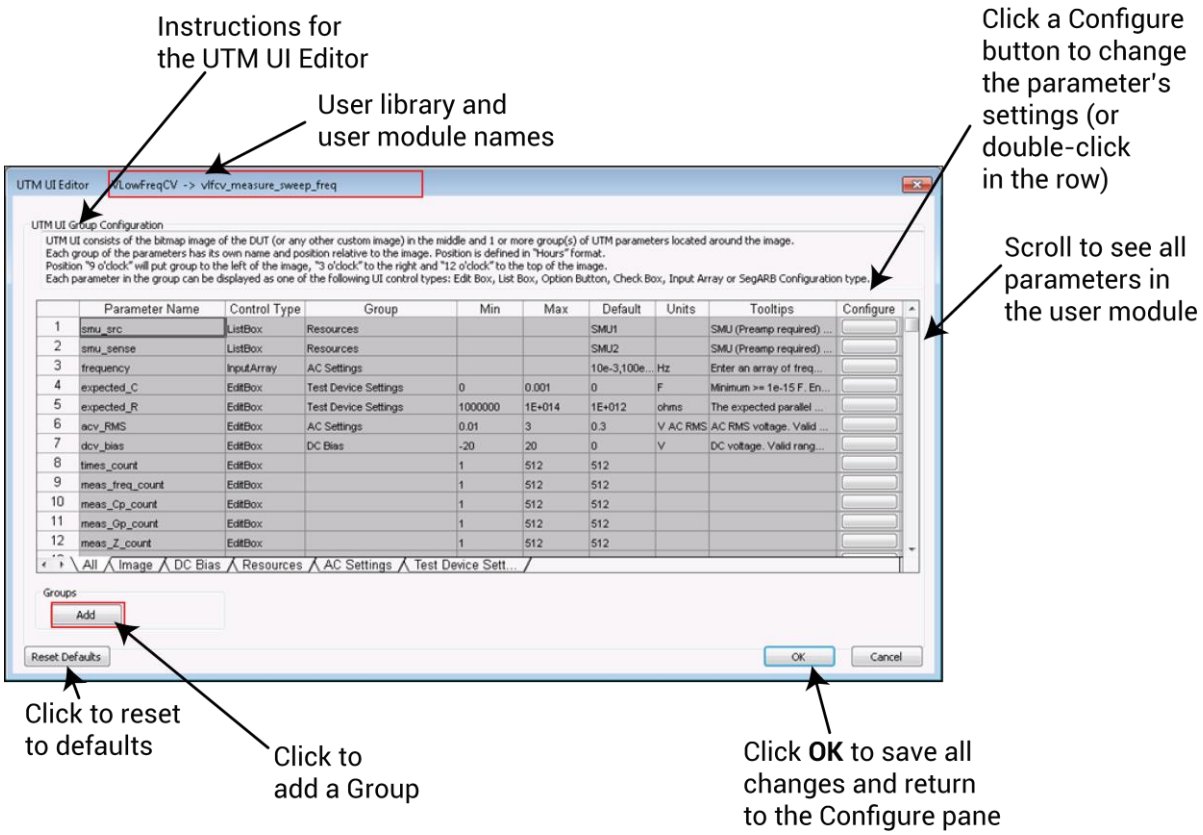
Select **OK** to save any changes.

Select **Cancel** to exit the dialog box without saving any changes.

UTM UI Editor

An example of the UTM UI Editor is shown below.

Figure 310: UTM UI editor



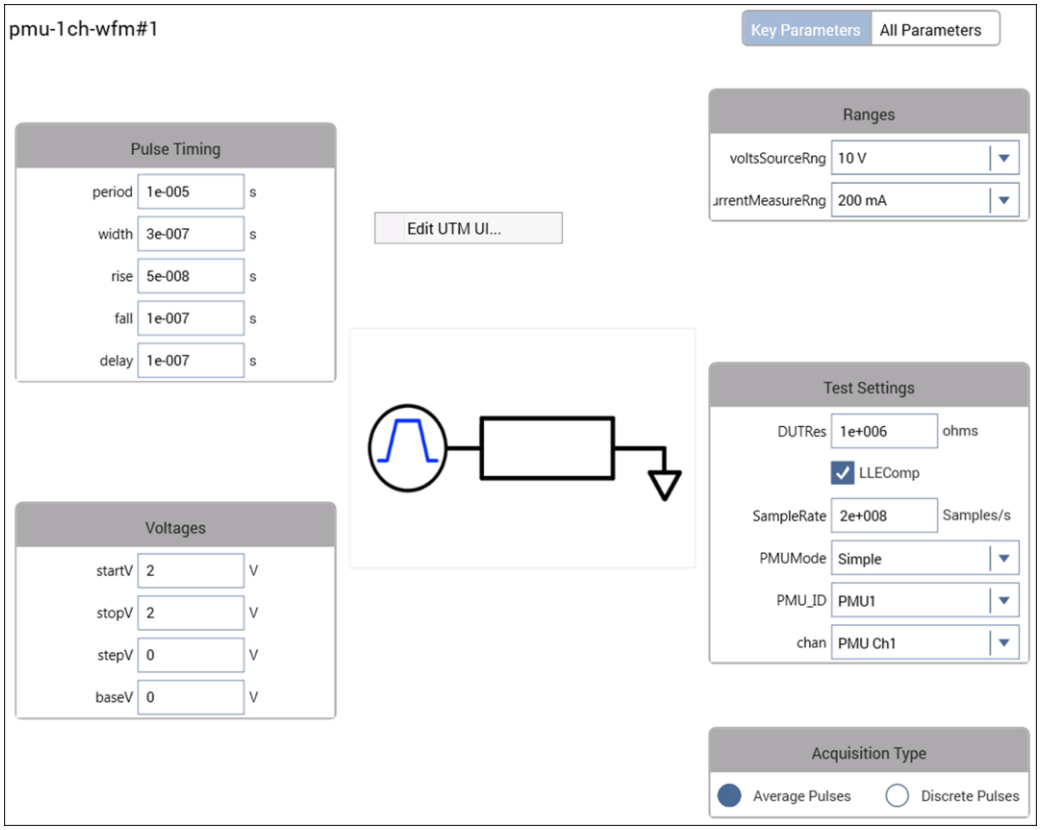
Example of using the editor

This example demonstrates edits to the `pmu-1ch-wfm` UTM from the `pmu-dut-example` project.

To edit the UI for this test:

1. Make sure the UTM UI Editor is enabled. See Allow access to the UTM UI editor (on page 6-145).
2. Open the `pmu-dut-examples` project from the Project Library.
3. Select the `pmu-1ch-wfm` test.
4. Select **Configure**.
5. Right-click anywhere in the Configure pane to display the Edit UTM UI button, shown below.

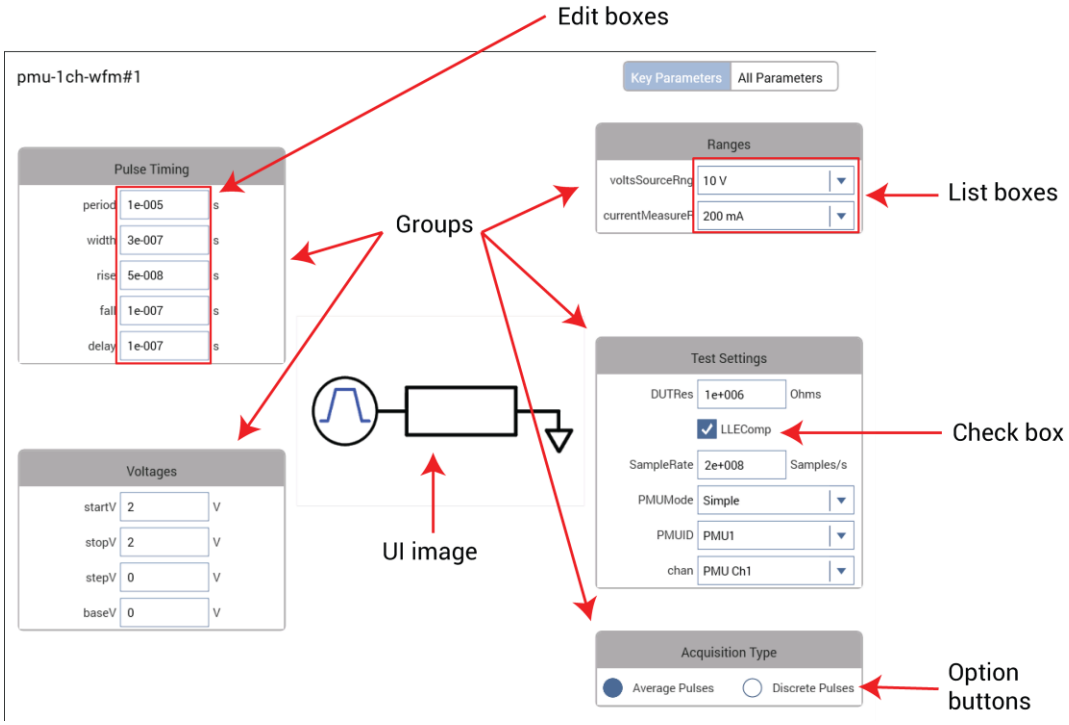
Figure 311: Start the UTM UI editor (right-click the Configure pane)



6. Select **Edit UTM UI**. The UTM UI Editor dialog box opens, as shown in UTM UI Editor (on page 6-180).

The following figure illustrates some of the available controls for the UTM UI view, including groups, edit boxes, list boxes, and check boxes. These controls are configured in the UTM UI Editor.

Figure 312: GUI view element sample



NOTE

The user module defines the Test Description content in the KULT description tab. You cannot use the UTM UI Editor to define the Test Description content. For more information, see the Description tab area (on page 8-8) in KULT.

UTM UI definition file information

The UTM UI 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 file name `user_module_name_GUI_Config.xml`. For example, one factory UTM UI file is `PMU_examples_ulib_GUI_Config.xml`. The user UI file name format is `user_module_name_User_GUI_Config.xml`. Both files are 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 UI definitions for all user modules in the user library. Modifying a UTM UI for a user module that has a factory UTM UI will automatically create a user XML file. If a user definition exists for a user module, it is used for the UTM UI. For UI definitions that are provided with the 4200A-SCS, there is a factory file. If a user module does not have a factory UTM UI definition, creating a definition will create a user file. Do not modify the XML files outside of the UTM UI Editor, as errors or non-functional UTM UI definitions may result.

Reset defaults

On the main screen of the UTM UI Editor, there is a Reset Defaults button (refer to the figure in UTM UI Editor (on page 6-180)). Select this button to overwrite the settings for the user module with the factory defaults from the original UTM UI definition file. If there is no original definition for the user module, one is dynamically generated.

Copy or move UTM UI definitions

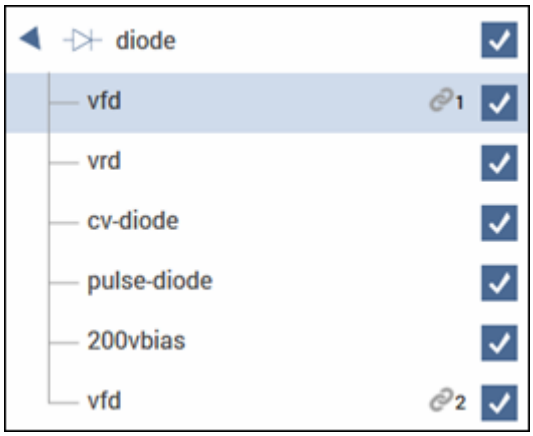
The UTM UI definition consists of one file for each user library. Use the `kultcopy` command to move a user library to another 4200A-SCS. This command copies the user modules (c code), UTM UI definition (XML files), and any bitmaps for the UTM UI (bitmapped files with the `.bmp` extension). For more information, see `kultcopy` in Copying user libraries using kultcopy (on page 8-57).

Link tests or actions

You can use the Linked Copy option to insert multiple instances of test or action in the project tree. When ITMs are linked, Clarius automatically keeps the configurations of the linked ITMs identical. This allows you to have multiple tests that perform identically. When UTMs or actions are linked, the user libraries, user modules, Formulator formulas and constants are identical. Note that parameter settings are not identical for linked UTMs or actions.

When tests are linked, the tests in the project tree display a chain-link icon, as shown in the following figure.

Figure 313: Linked ITMs in the project tree



When using linked copies:

- For ITMs, the settings in the Configure pane (including Formulator formulas and Output Values) are kept in synchronization. When you change a setting in once instance of a linked test, all of the linked tests are changed.
- For UTMs and actions, the user libraries, user modules, Formulator formulas and constants, and Output Values are identical between the tests. Parameter settings can be different.
- The data for each linked test remains independent. The Analyze Run, Calc, and Settings sheet and Analyze graph and graph settings for each linked test are different.

To insert linked tests:

1. Add a test to the project tree (refer to Add a device and test to the project (on page 6-12) or Create a custom test (on page 6-143)).
2. Right-click the test and select **Linked Copy**.
3. Select an item in the project tree that you want the test to follow. The item is highlighted in green if this is a valid place to copy the test. It is red if you cannot copy the test to this location.
4. Select **Paste**. The linked test is added to the project tree.

NOTE

You can create multiple linked copies of tests by using Linked Copy at the device or subsite level. In this case, all ITMs associated with the device are copied with the new device and become linked copies of the tests in the original device. UTMs are copied as independent tests or actions. Note that the devices and subsites do not become linked copies, only the ITMs.

Add actions

Actions allow you to move probers, add user notifications such as beepers and dialog boxes, and change switching options. You can add existing actions or create actions based on user modules. When you create an action, you select a user module from a user library to create the action. Clarius supplies user libraries, or you can create your own using KULT.

NOTE

If you are moving from 4200 KITE to 4200A Clarius, actions replace initialization and termination steps. Actions are more versatile than initialization and termination steps. You can place them in the project wherever they are needed instead of being limited to the top and bottom of the project.

To add an action to the project tree, drag it into the tree where the action needs to occur during the test. For example, if you need to sound a beep after a specific test, drag the `Beeper` action to the project tree under that test.

To create an action:

1. Choose **Select**.
2. Select the **Actions** tab.
3. Drag **Custom Action** to the project tree. The action has a red triangle next to it to indicate that it is not configured.
4. Select **Rename**.
5. Enter a name for the action.
6. Select **Configure**.
7. In the Test Settings pane, select the user library.
8. Select the user module.
9. Set other settings as needed. Refer to the Help pane for information.

Sites

A site includes all of the subsites, devices, and tests in the project. If you set up multiple sites, all sites are identical. They will each have the same type and number of subsites and the sites are repeated across the wafer.

To add a site to the project tree:

1. Choose **Select**.
2. Select the **Wafer Plan** library.
3. Select **Site**.
4. Select **Add**. The site is added to the project tree.

Subsites

A subsite is a collection of devices and their associated tests. You can work with devices and tests as you do in a project that does not include a subsite.

You need to use actions to initiate prober movement between subsites and close matrix channels between devices.

NOTE

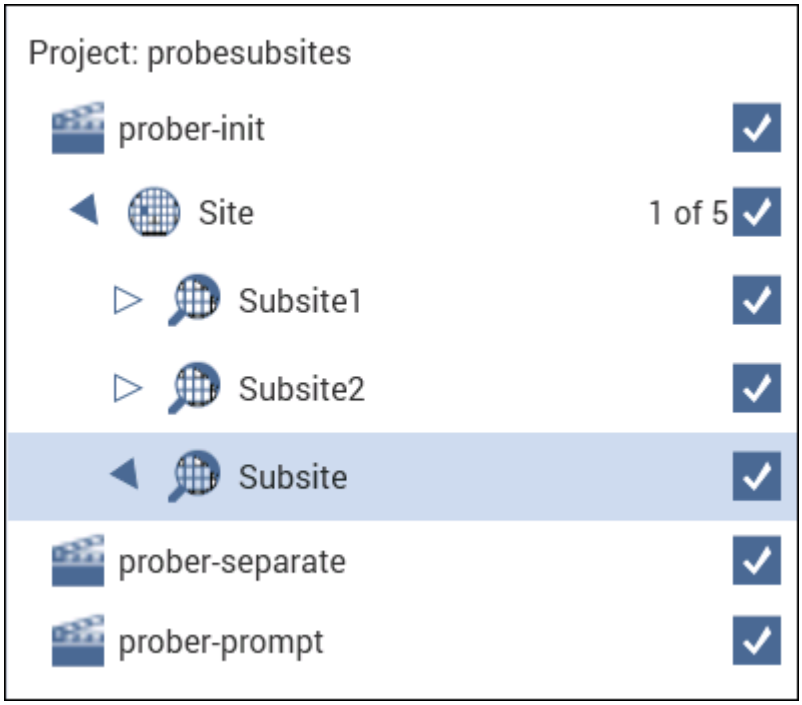
You can add subsites as show below. If you are using a prober, you can also use `probesubsites` from the Project Library to start with a site and subsite template with prober actions.

To add a subsite:

1. Open a project or create a new one.
2. Choose **Select**.
3. From the Wafer Plan tab, drag **Subsite** to the project tree.
4. If needed, select **Rename**, enter a new name, and press **Enter**.

An example of a project tree with several subsites and prober actions are shown in the following figure.

Figure 314: Project tree with subsites example



Configure a complex test

For more complex tests, you can:

- Set advanced test and terminal parameters.
- Set up multiple steps or sweeps to track simultaneously using masters and subordinates.
- Configure actions.
- Configure sites and subsites.
- Set up subsite cycling for stress and measure cycles.

Test and terminal settings

You can access test and terminal settings from the center and right panes of Clarius when Configure is selected.

The most common terminal settings are available in the center pane when Key Parameters is selected. Additional common test settings are available in the right Terminal Settings pane.

Less commonly used terminal settings are available in a dialog box that you open with the Advanced button on the Terminal Settings pane.

To view all terminal settings, select All Parameters from the center pane. In this view, all terminal settings are displayed for every terminal.

Test settings are also displayed in the right pane. Test settings affect all terminals in the selected test. The most common settings are available in the right. Additional settings are available when you click the Advanced button.

For descriptions of all the test and terminal settings, refer to Test and terminal setting descriptions (on page 6-26).

Step or sweep multiple device terminals in the same test

Multiple steps or sweeps in an interactive test module (ITM) must track with regard to step number and duration. For example, you might want to apply multiple steps to multiple device terminals, such as when stepping the biases on two transistor terminals and sweeping voltage or current on the third terminal. In this setup, Clarius automatically sets the step operations to occur simultaneously, with one terminal set up as the master. All other step functions are automatically designated as subordinates. The following figure illustrates this concept.

Figure 315: Master step versus subordinate step

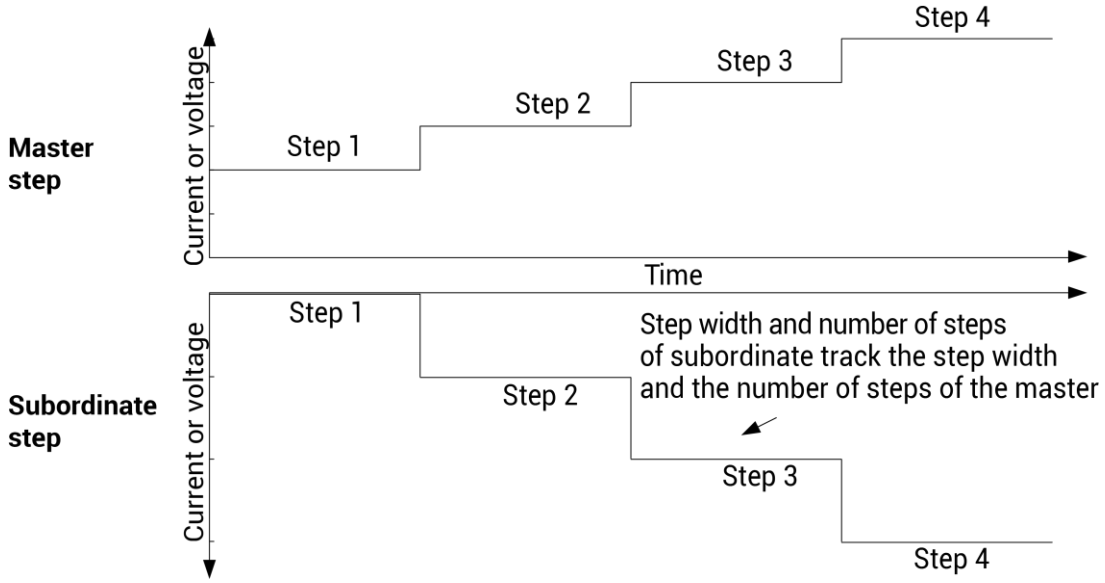


Figure 316: Master and subordinate sweeps

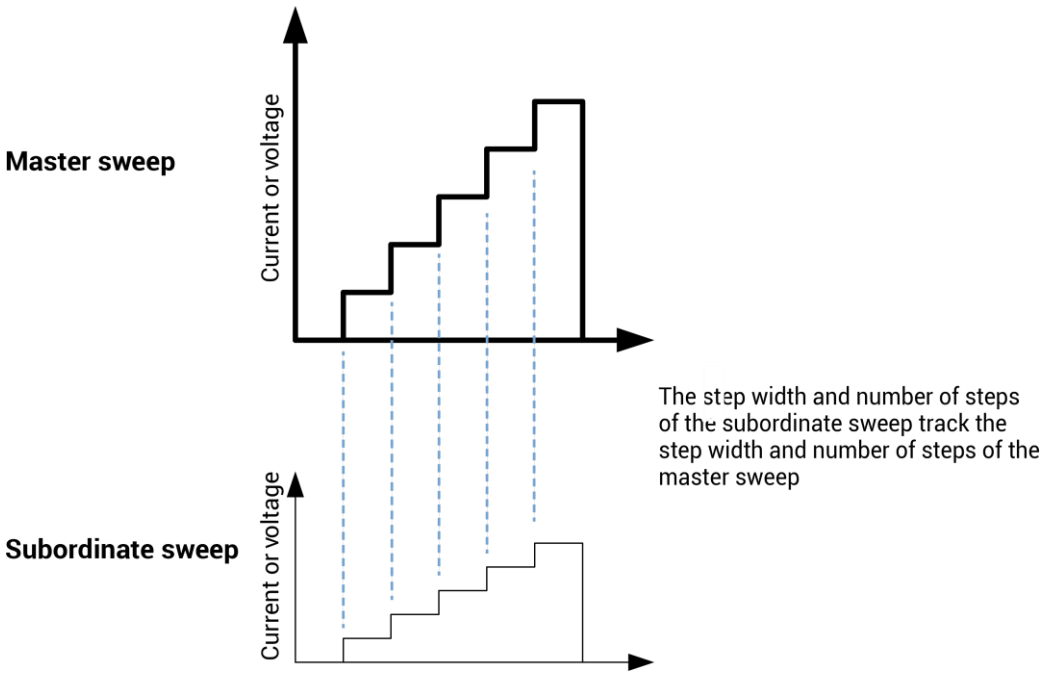
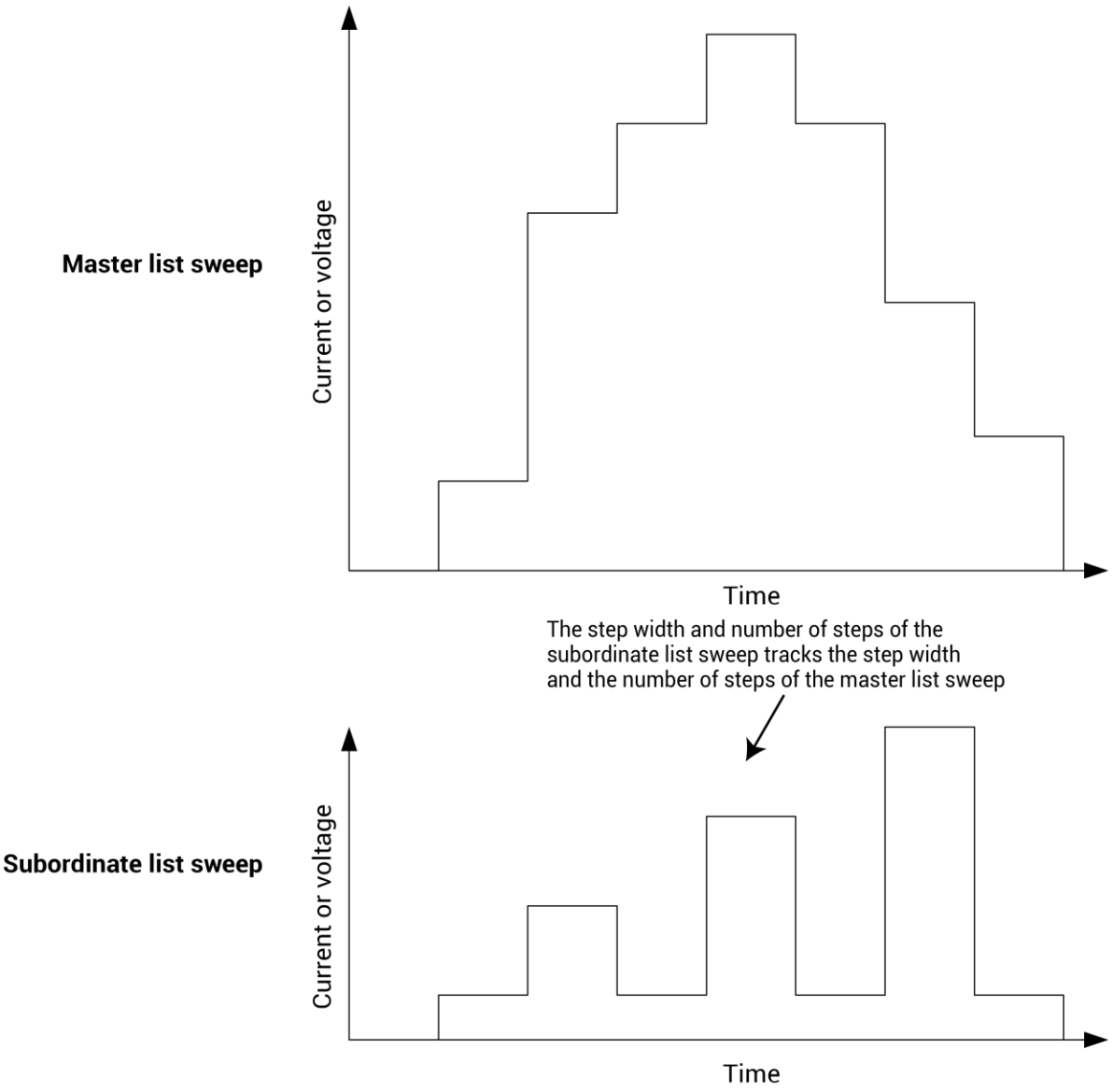


Figure 317: Master list sweeps versus subordinate list sweeps



If you do not specify an instrument to be the master, the first instrument that was assigned to the step or sweep operation mode is assigned to be the master. You can change this designation in the Test Settings pane.

When a master is set, Clarius sets the points and step size values for the subordinate terminal to be the same as the settings for the master terminal. You cannot change the subordinate points value for the subordinate terminal. For list sweeps, the number of points in the list items for the subordinate is changed to match the number of points in the master. If points are added to the master, the last point of the subordinate list is repeated. If points are removed from the master, the same number of points are removed from the end of the subordinate list.

You can have a dual sweep on a subordinate terminal even if the master is not set to dual sweep. In this case, the dual sweep of the subordinate terminal has a total number of steps equal to the number of steps in the master terminal. For example, if the master terminal is set to measure ten points, the subordinate dual sweep will measure five points on the first side of the sweep and five points on the second side of the dual sweep. If the master is set to an odd number of points, the subordinate terminal will repeat the measurement of the last sweep point.

The subordinate SMUs are not automatically set for dual sweep when Dual Sweep is enabled for the master SMU. Dual Sweep must be individually enabled for each SMU.

To specify the master sweep:

1. Select the test.
2. In the right pane, select **Test Settings**.
3. For **Sweep Master**, select the instrument that you want to designate as the master.

Configure actions

Settings for actions depend on the type of action that is selected. Actions can generate dialog boxes to prompt test operator action, control prober movements, and manage switching. You can also create your own actions from user libraries.

You can place actions anywhere in the project tree.

Information for actions is available in the Help pane. To open the Help pane, select the action in the project tree and select **Configure**. Select the Help tab in the right pane.

Configure sites

A site includes all of the subsites, devices, and tests in the project. If you set up multiple sites, all sites are identical. They will each have the same type and number of subsites and the sites are repeated across the wafer.

For sites, you can set the following options:

- **Number of Sites:** The maximum number of sites that can be tested. This is typically set to the number of sites that have been programmed in a prober controller.
- **Start Execution at Site:** The site where project execution starts. This is normally the same as the prober starting site number.
- **Finish Execution at Site:** The site where project execution ends. This must be less than or equal to the number of sites.

NOTE

If you use a semi-automatic prober, understand that a Clarius probe action only triggers movements that are already programmed in the prober controller. Each execution of the action advances the probe to the next site in this programmed sequence. Site numbers are not communicated between the prober and Clarius. Therefore, if you evaluate multiple sites, the range of site numbers that you specify in the Clarius Project window must agree with the sequence of site numbers in the prober controller program.

To configure sites:

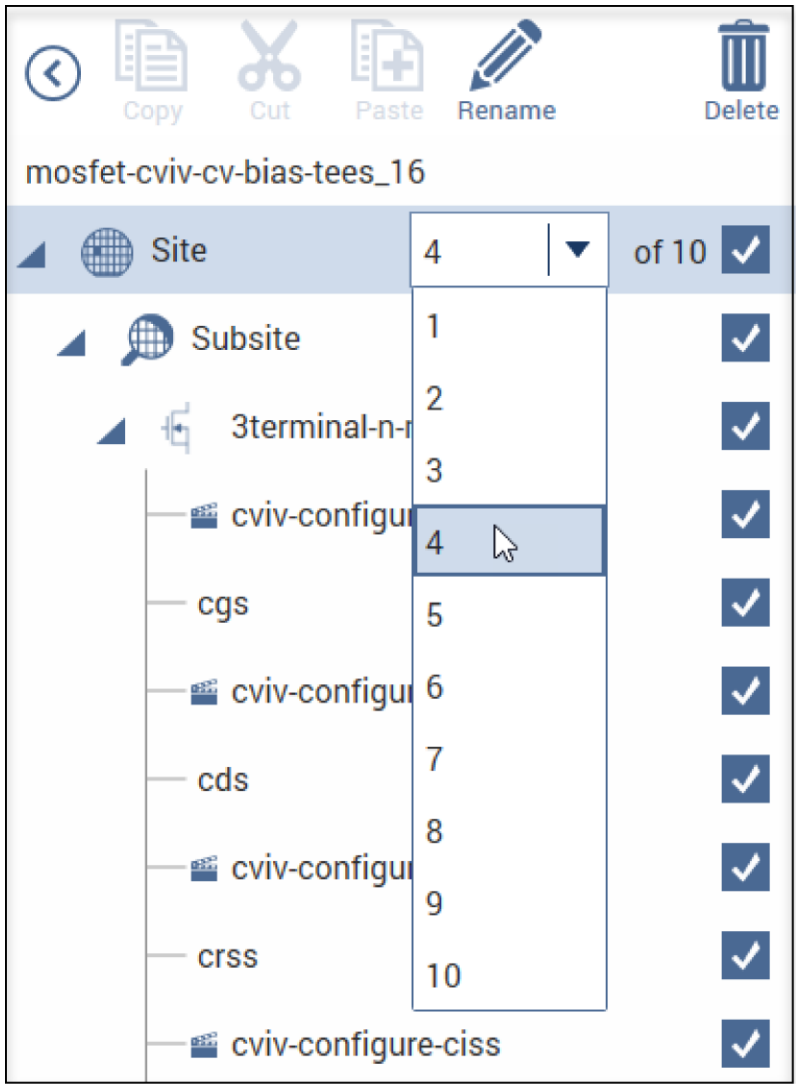
1. Select the site in the project tree.
2. Select **Configure**.
3. Set the **Number of Sites**.
4. Select **Save**.

NOTE

If you are configuring your sites for Segment Stress mode, the maximum number of sites is 999.

5. If there is more than one site, you can select the site where testing should start executing and the site where testing should stop executing.
6. If there is more than one site, you can select which site's tree is displayed. The site that is displayed is shown next to the site name in the project tree. For example, if the current site is set to 4 in a project with 10 sites, "4 of 10" is displayed next to the site name. See the following figure.

Figure 318: Selecting a site



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 actions.

Configure subsite cycling

You can use the 4200A-SCS to stress test DUTs using subsite cycling. A Clarius evaluation consists of pre-stress tests at a subsite, followed by alternate cycles of stressing and retesting. During the evaluation, Clarius can display intermediate numerical and graphical results and status information. Clarius ends the evaluation when the devices degrade beyond specified exit criteria (degradation targets) or when the total stressing time reaches a specified maximum, whichever comes first.

Subsite cycling allows you to cycle through the subsite tests up to 128 times. Clarius 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.

Data and graphs of the subsite cycles are available in the Analyze pane for the subsite.

The measured readings listed in the Analyze pane are output values. An output value is a measurement that is imported from an individual test into the subsite. See Export output values to Analyze sheet (on page 6-224) for details.

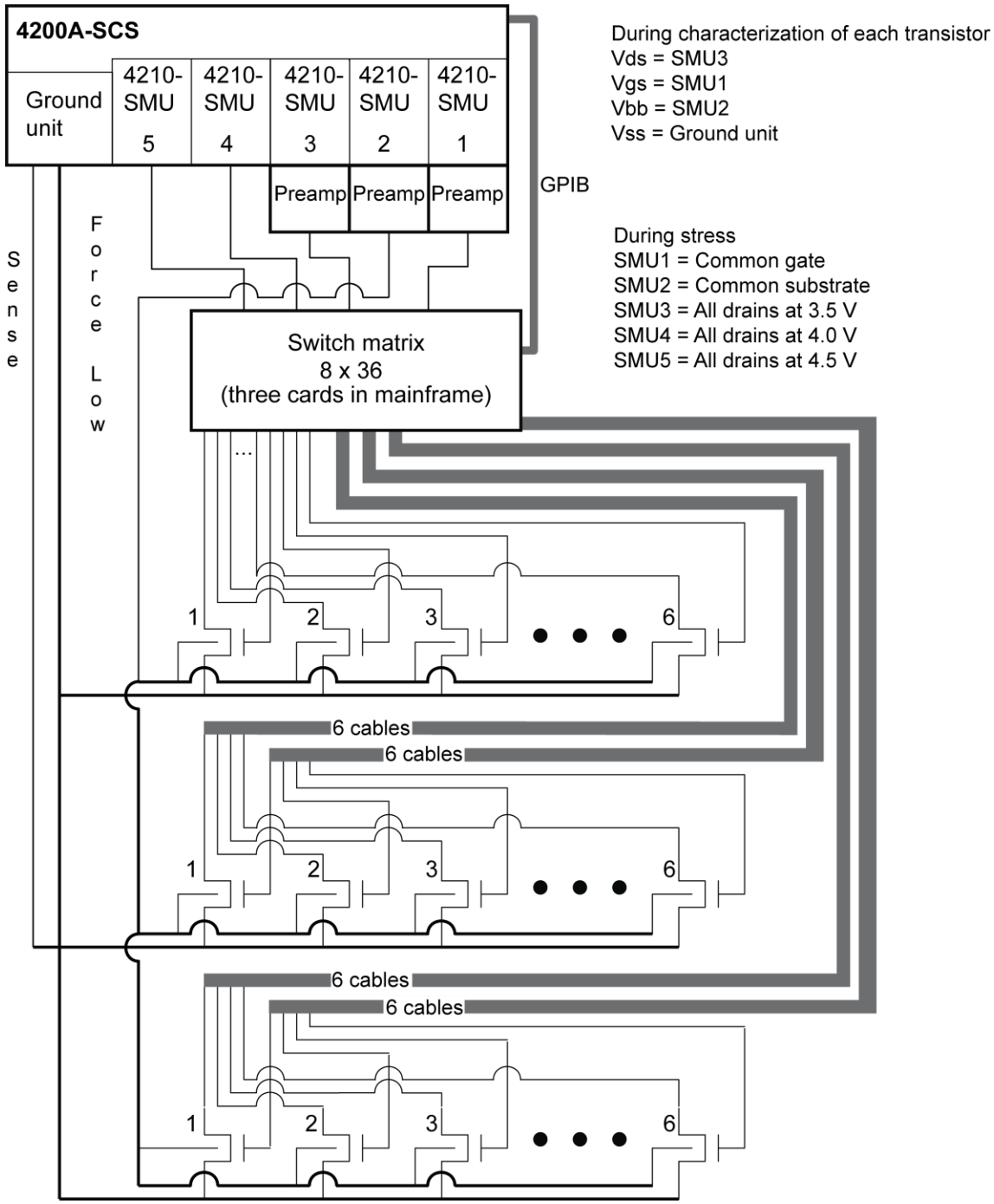
Stress mode integrates stressing with subsite cycling for testing. The first cycle is stress-free. For each subsequent cycle, the devices in the subsite are stressed with voltage or current for a specified period. After the stress period expires, the tests in the subsite are run. Device stressing includes DC voltage stress, DC current stress, AC voltage stress, and segment stressing. 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 cards. Each pulse card has two pulse output channels, each of which can stress one device terminal.

Segment stress is similar to the standard Stress mode, but is done using Segment Arb® pulse mode for stressing. Stress is provided by a Segment Arb waveform generated by a pulse card. Each channel of the pulse card can stress one device terminal. DC bias voltage and current limit for the device can be provided by the SMUs in the system.

Connect devices for stress/measure cycling

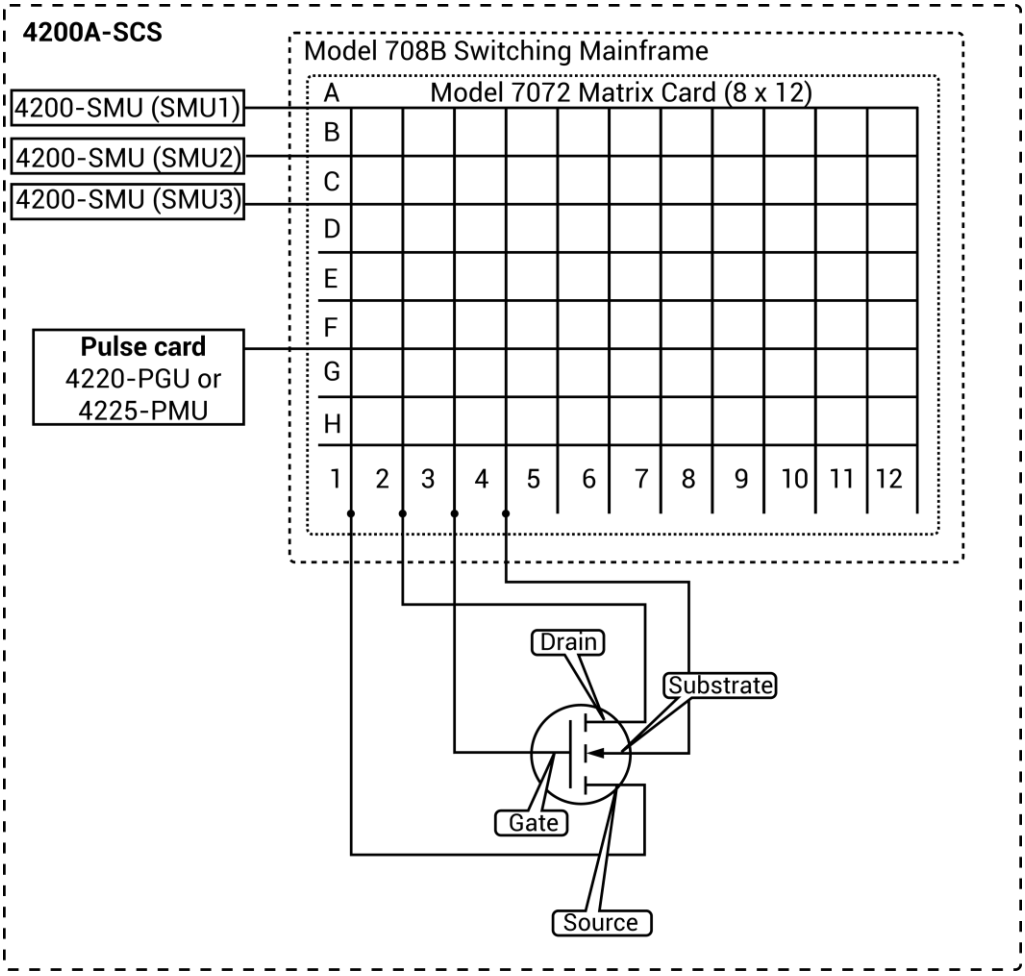
Devices that are stress/measure cycled in parallel are connected through a switching system. The following figure shows an example of connections for an HCI evaluation.

Figure 319: Stress / measure wiring example



Connections for matrix card

Figure 320: AC Pulse stress-measure — hardware matrix card simplified schematic



Connections for pulse card to device under test

Connect the pulse generator to the DUT during stress as shown in the following figures.

Figure 321: AC pulse stress-measure — hardware setup block diagram for stress

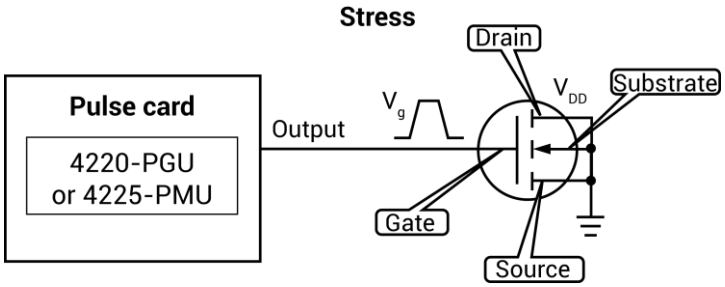
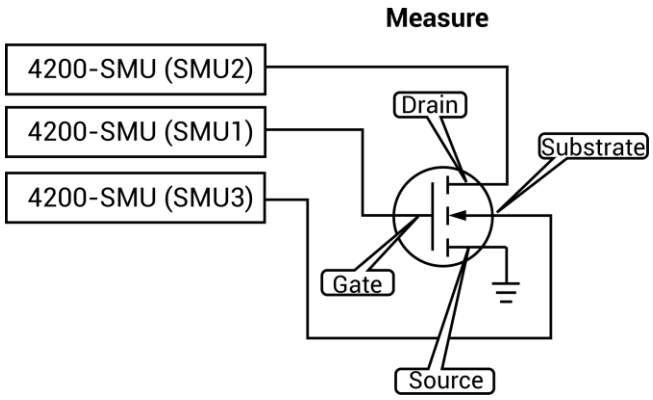
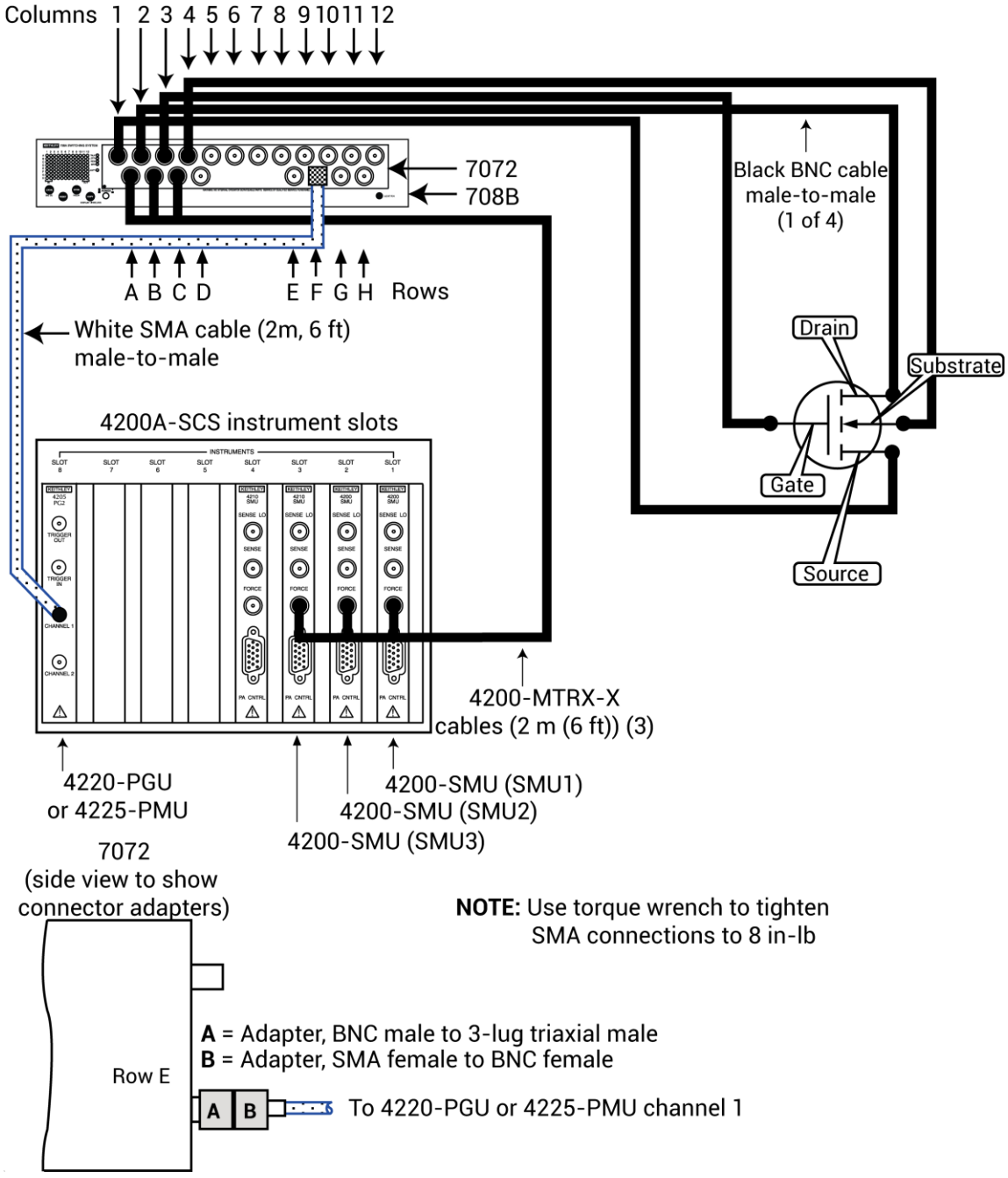


Figure 322: AC Pulse stress-measure — hardware setup block diagram for measure



Connections for system hardware



Set up the Subsite Operation

You can select one of the following Subsite Operations:

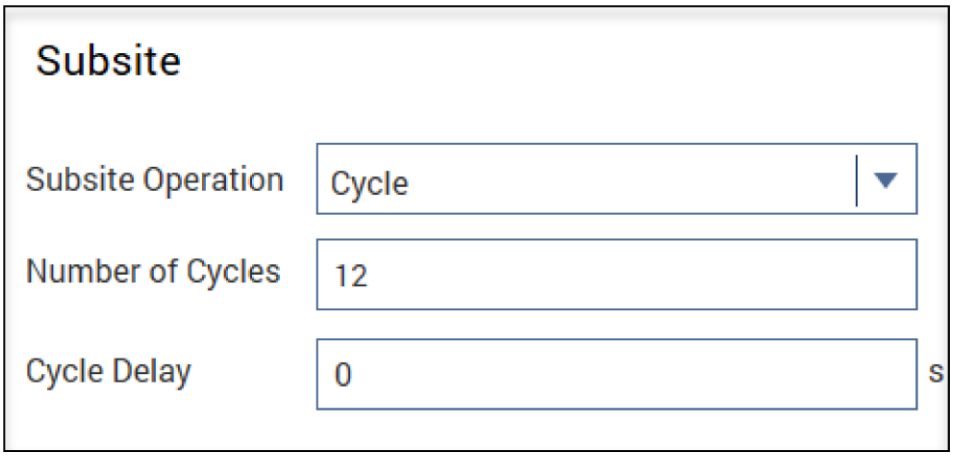
- **None**: No cycling or stressing operation is performed on the subsite.
- **Cycle**: Loops through the subsite tests without stressing the devices.
- **Stress**: Test-stress-test-stress cycles, such as hot carrier injection (HCI) or negative bias temperature instability (NBTI) studies. You can use SMUs to provide bias voltage and current limit for the devices, but you cannot use them to measure stress. This operation can also include segment stress using the Segment Arb® pulse mode.

In the Cycle operation, the subsite test is repeated a specified number of times. There are only measure cycles with no stressing. For each individual test in the subsite, data is acquired for each subsite cycle. For example, if the subsite is cycled five times, there are five sets of data and graphs for each test. You can execute up to 128 cycles.

To set up the Cycle operation:

1. From the project tree, select the subsite.
2. Select **Configure**.
3. In the Subsite pane, select **Cycle** from the **Subsite Operation** menu.
4. Enter the **Number of Cycles**. This is the fixed number of times that you want the subsite to execute.
5. Enter the **Cycle Delay** in seconds.
6. The setup is complete.

Figure 323: Stress Mode Setup pane



To set up the Stress operation:

1. From the project tree, select the subsite.
2. Select **Configure**.
3. Select **Stress** from the **Subsite Operation** menu.
4. See the remaining topics in this section to configure the operation.

If your project is set up to run on more than one subsite, you need to set the stress properties for each subsite separately. This allows you to have different levels of stress on each subsite. After you configure the first site, repeat the steps for the next subsite.

To configure multiple subsites with the same settings, configure the first site, then select **Copy** in the project tree.

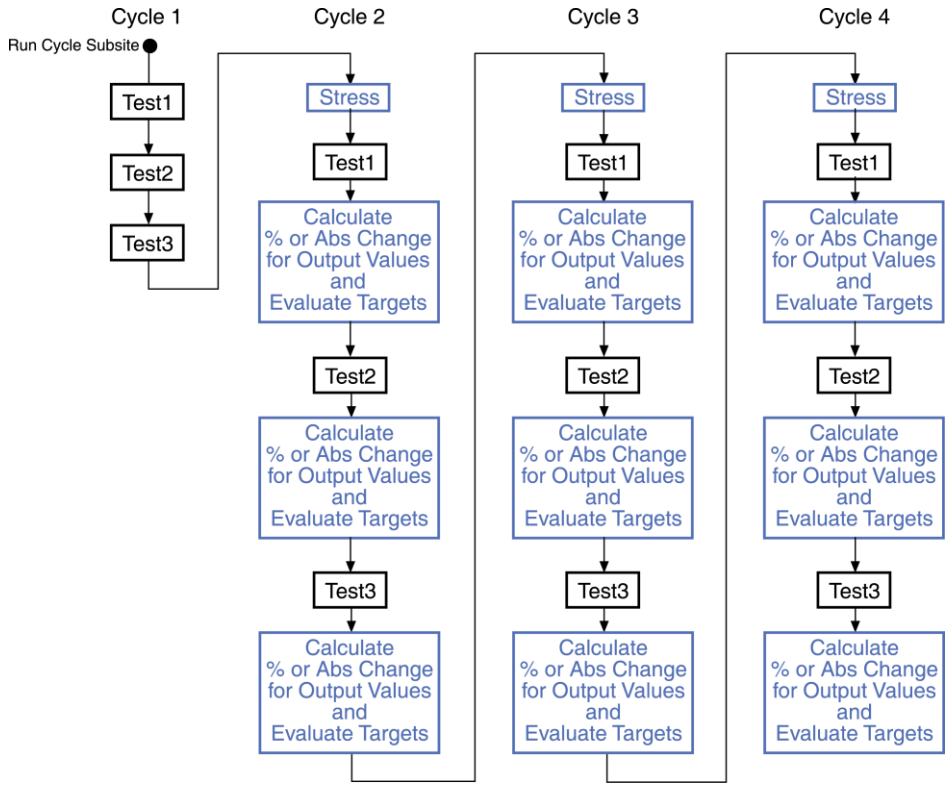
About the Stress operation

In the Stress mode, the subsite test is repeated a specified number of times. For each individual test in the subsite, data is acquired for each subsite cycle. For example, if the subsite is cycled five times, there are five sets of data and graphs for each test. You can execute up to 128 cycles.

The test sequence includes components for stressing, percent change, and target evaluation. The following figure shows an example of a basic testing sequence. The components for stressing, percent change, and target evaluation are shown in blue.

When subsite cycling is started, the first pass through the subsite is a pre-stress cycle. Tests are run with no stressing. At the start of the next cycle, the configured stress (voltage or current) is applied to all devices. After the stress period expires, the stress is removed and enabled tests are run. Each additional stress cycle operates in the same manner. That is, the stress is applied for the specified stress time, then all the enabled tests are run. Notice that after each test is run, the percent absolute (Abs) change and targets are evaluated.

Figure 324: Example of the stress testing sequence (four cycles) for a single device



NOTE

The following information explains stress testing using the Stress mode. Stressing is provided by SMUs or Keithley Instruments pulse cards or both (using the standard pulse mode for AC stressing).

Stressing can also be provided by Keithley Instruments pulse cards using the Segment Arb pulse mode. Refer to Segment stressing (on page 6-210) for supplemental information on using Segment Arb for stress testing.

For stress testing, a Clarius evaluation consists of pre-stress tests at a particular subsite, followed by alternate cycles of stressing and retesting. Clarius performs these cycles automatically when you select Stress mode. During the evaluation, Clarius can display intermediate numerical and graphical results and status information. Clarius 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.

Combined stressing and testing

The following steps summarize an HCI evaluation for the stressing configurations shown in DC Voltage stressing (on page 6-208) and AC Voltage stressing (on page 6-209). Similar operations apply to other types of stress-measure studies.

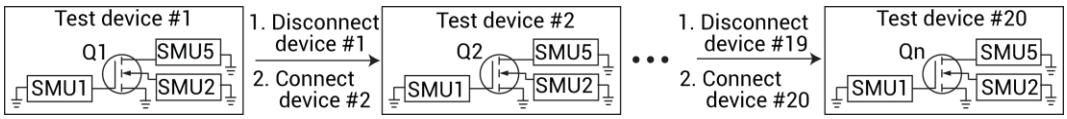
NOTE

For information about AC stress for wafer-level reliability, refer to Wafer-Level Reliability Testing (on page L-1).

Summary of an HCI evaluation:

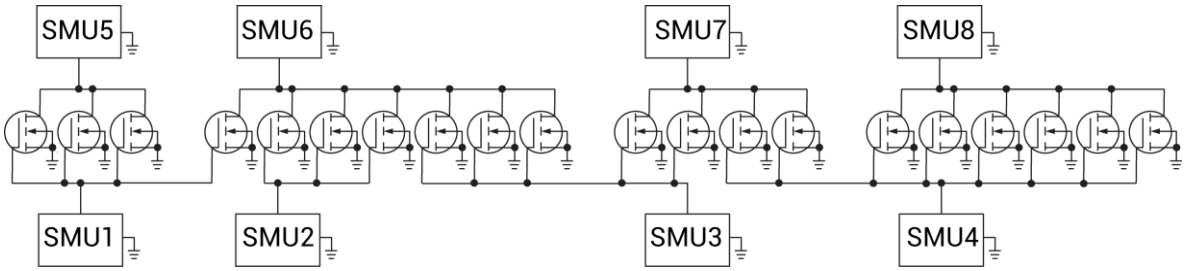
1. Use the switch matrix to automatically connect the SMUs to device 1.
2. Run pre-stress parametric tests on each device individually in device-number sequence, as shown in the following figure.

Figure 325: Pre-stress parametric tests



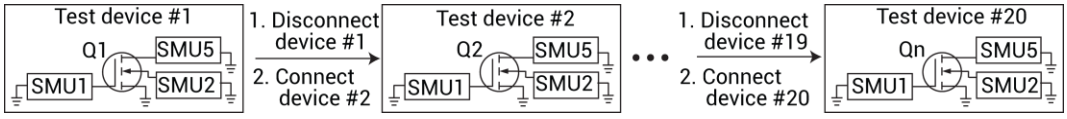
3. Disconnect all devices.
4. Use the switch matrix to automatically connect all devices to SMUs, as determined by the drain and gate voltages that were specified for each device.
5. Run stress cycle 1, which stresses all of the devices simultaneously, as shown in the following figure.

Figure 326: Stress all devices simultaneously



6. Disconnect all devices.
7. Use the switch matrix to automatically connect the SMUs to device 1.
8. Wait for a 10 s delay to promote uniform pre-test decay for all devices.
9. Run test cycle 1, running post-stress parametric tests on each device individually in device-number sequence, as shown in the following figure.

Figure 327: Post-stress parametric tests

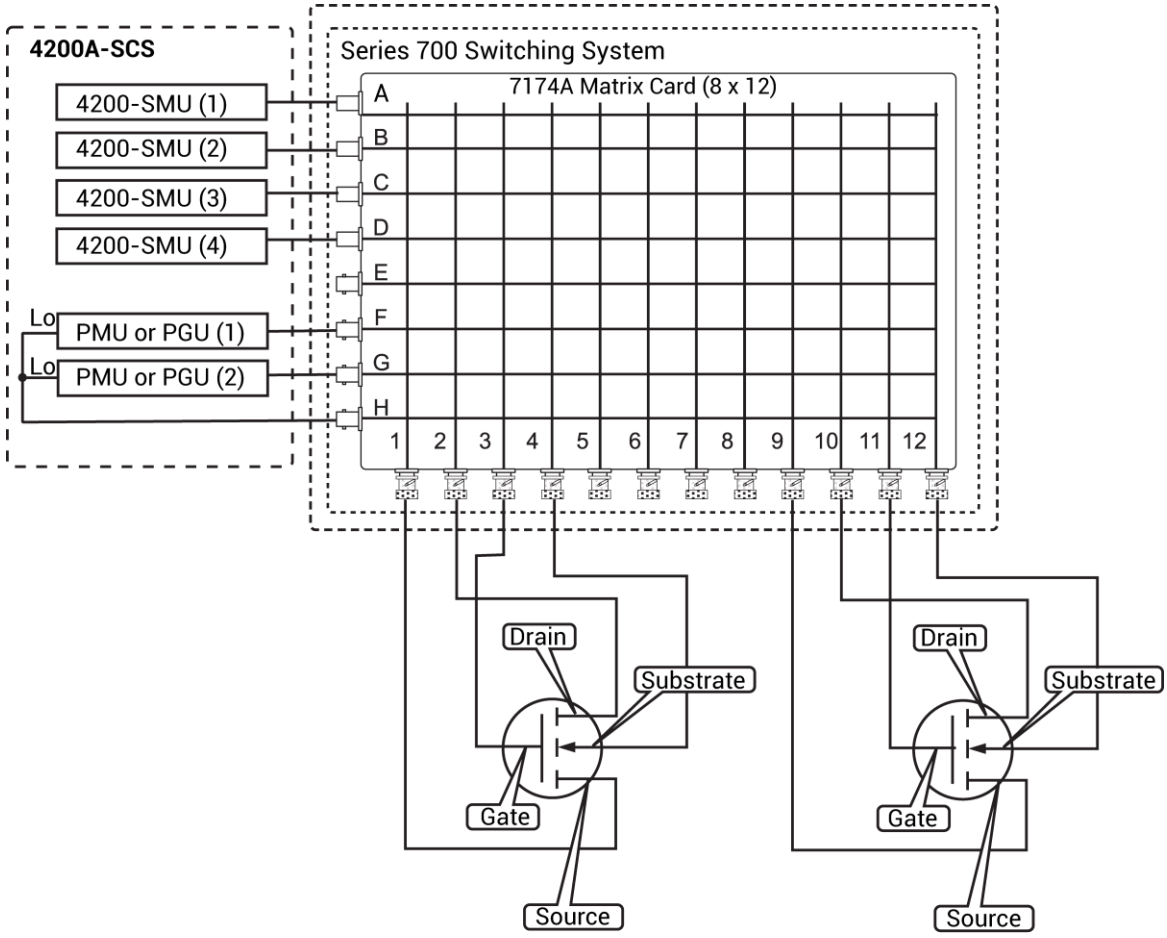


10. Disconnect all devices.
11. Reconnect all devices for stress cycle 2.
12. Run stress cycle 2.
13. Disconnect all devices.
14. Reconnect all devices for test cycle 2.
15. Wait for a 10 s to promote uniform pre-test decay for all devices.
16. Run test cycle 2.
17. Continue with stress and tests cycles (3, 4, … *n*) until a device degrades to all enabed target values or goes into compliance.
18. Stop testing this device but continue stress and test cycles until another device degrades to all target values or goes into compliance.
19. Stop testing this second device, but continue stress and test cycles until one of the folowing occurs:
    - All devices have either degraded to target values or have gone into compliance.
    - Total stress time reaches a user-specified value.

AC and DC voltage stress/measure system with a switch matrix

A switch matrix is supported for a pulse card AC voltage stress/measure system. The following figure shows the use of a switch matrix for an AC and DC voltage stress/measure system. The recommended matrices for this system configuration are the Series 700 Switching Systems. To effectively transmit the higher frequency components of the typical pulse (Segment Arb or Standard), use a high bandwidth switch matrix card, such as the Keithley Instruments 7174A or 7173-50.

Figure 328: Switch matrix for AC and DC voltage stress/measure system



Select a Subsite Mode

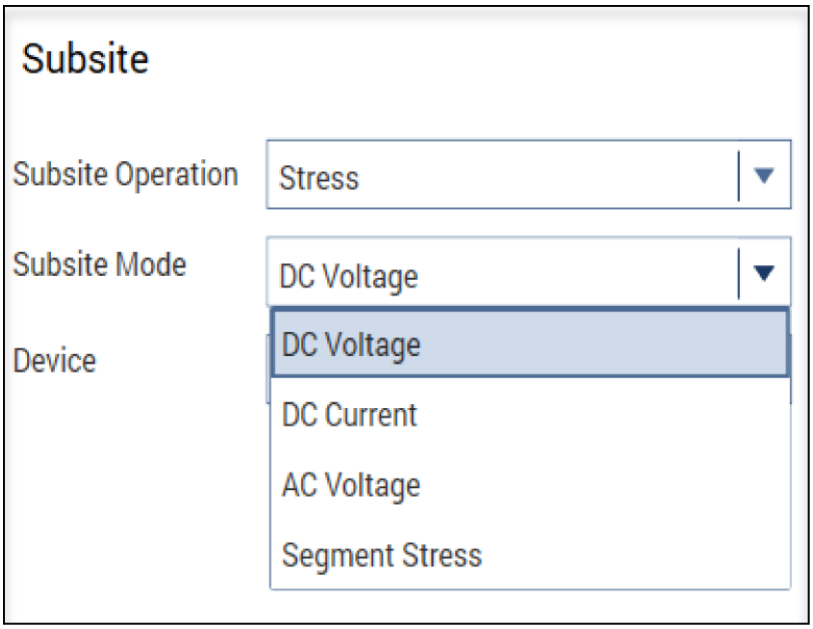
You can select four stress operation subsite modes. The modes are:

- DC Voltage
- DC Current
- AC Voltage
- Segment Stress

To select a Subsite Mode:

1. From the project tree, select the subsite.
2. Select **Configure**.
3. From the **Subsite Operation** menu select **Stress**.
4. Select a **Subsite Mode**.

Figure 329: Selecting a subsite mode



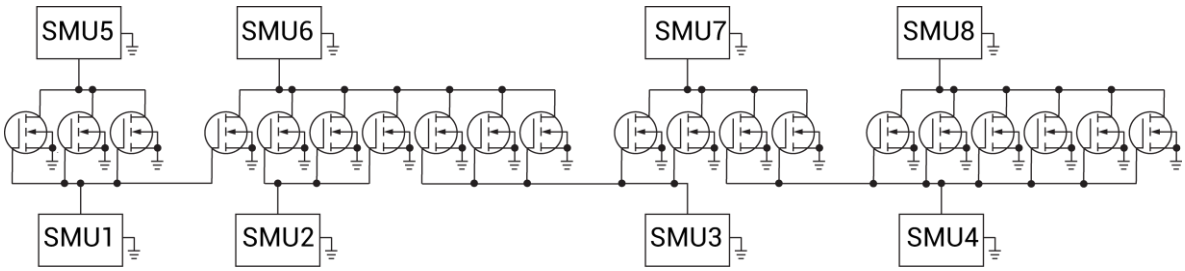
See the following topics in this section for a description of the subsite modes.

DC Voltage stressing

Clarius'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 (HCI) 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 4200A-SCS system contains four SMUs, you can apply a maximum of four unique stress bias voltages (gate voltages plus drain voltages combined). If your 4200A-SCS system has 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. The following figure illustrates a voltage stressing configuration that uses the maximum software and system capabilities.
- If your voltage stress system is using a switch matrix, the 4200A-SCS tries to maximize the amount of SMU sharing in order to allow parallel testing. It determines which pins can share SMUs in the following fashion. If pins from different devices have the same name (for example, gate pin, drain pin) and the like-named pins are assigned the same voltage stress, then when the stress is applied, these pins are automatically connected to the same SMU through the switch matrix. That SMU supplies the voltage stress to all the pins simultaneously.
- Because parallel-connected devices share resources, Clarius monitors stressing resources when Stress Properties are configured. If the requirements exceed the resources, Clarius reports an error.

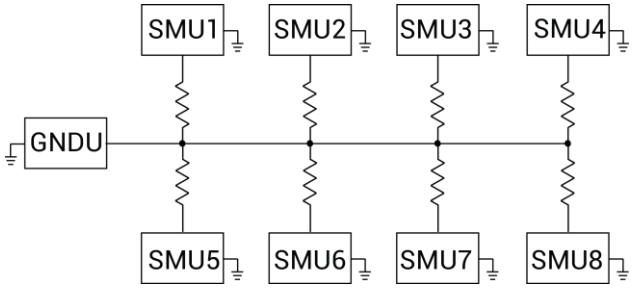
Figure 330: DC voltage stressing: 20 parallel-connected devices stressed at eight gate and drain voltages



DC 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 a system with eight SMUs, up to eight devices can be current stressed, as shown in the figure below.

Figure 331: EM test: Eight devices being current stressed by eight SMUs

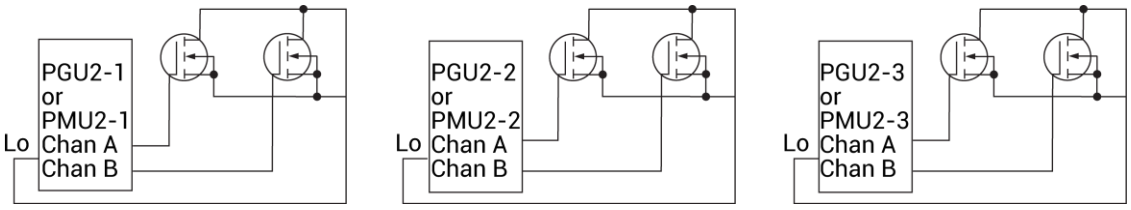


AC Voltage stressing

You can use four Keithley Instruments pulse cards to AC voltage stress eight devices (one device pin for each pulse channel). The figure below shows a Keithley Instruments pulse card providing AC voltage stress for six devices.

Parallel-connected devices cannot be AC voltage stressed using the pulse card. As shown in the figure, each pulse card channel can only stress one pin of one device.

Figure 332: AC voltage stressing: Six devices stressed at the gates with six pulse outputs



Segment stressing

Segment stress testing consists of two phases:

- During a measure phase, the system makes measurements on the DUT.
- During a stress phase, the Keithley pulse card provides stress using Segment Arb waveforms, and the SMUs provide voltage bias and current limit. There are no measurements made during the stress phase.

For Segment Arb stressing, the waveform period is the fundamental unit of time for stressing. In the setup pane, the term "stress counts" is used to specify the number of times the Segment Arb waveform will stress the device. For example, assume the stress count is three, and the waveform period is four seconds. For that stress cycle, the Segment Arb waveform will stress the device three times for a total stress time of twelve seconds.

In a typical stress/measure test system that uses a switch matrix to automate the stress and measure phases of the test:

- During a measure phase, the switch matrix connects the instruments that will make the measurements on the DUT. The Keithley Instruments pulse card is disconnected from the DUT during a measure phase.
- During a stress phase, the switch matrix connects the pulse generator to the DUT. It also connects SMUs that will be used for device pin grounding or biasing.

NOTE

If your system contains 4225-RPMs, you cannot use SMUs during segment stresses. You must disconnect all RPMs from the 4200A-SCS and update the RPM configuration in KCon to enable DC biasing during subsite segment stress.

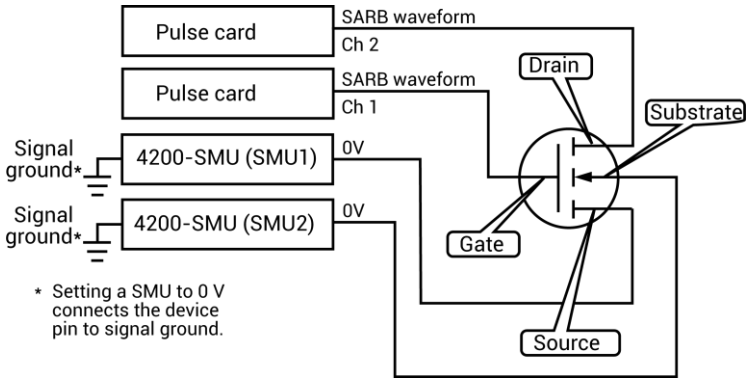
NOTE

To effectively transmit the higher frequency components of the typical pulse, a high-bandwidth switch matrix card should be used (for example, Keithley Instruments 7174A).

The stress phase example figure below shows an example of how a DUT can be stressed using Segment Arb waveforms. During a stress phase, the matrix connects the channels of the Keithley Instruments pulse card to the drain and gate of the DUT. The pulse generator stresses the drain and gate by outputting Segment Arb waveforms.

Two 4200-SMUs (SMU1 and SMU2) are connected to the substrate and source terminals of the DUT. They are set to 0 V to effectively ground the terminals.

Figure 333: Segment stressing: Stress phase example



Select and configure a Device

From the Subsite pane, you can configure the devices in your subsite. You can assign the device terminals to an instrument or function, and also set the stress conditions.

Depending on the Subsite Mode and the instruments you have connected to your 4200A-SCS, you can assign the terminals of the device you have selected to one of the following types of instruments or functions:

- GNDU
- SMU
- PGU
- PMU
- PIN x (for test systems with a switch matrix card)
- NONE (not connected)

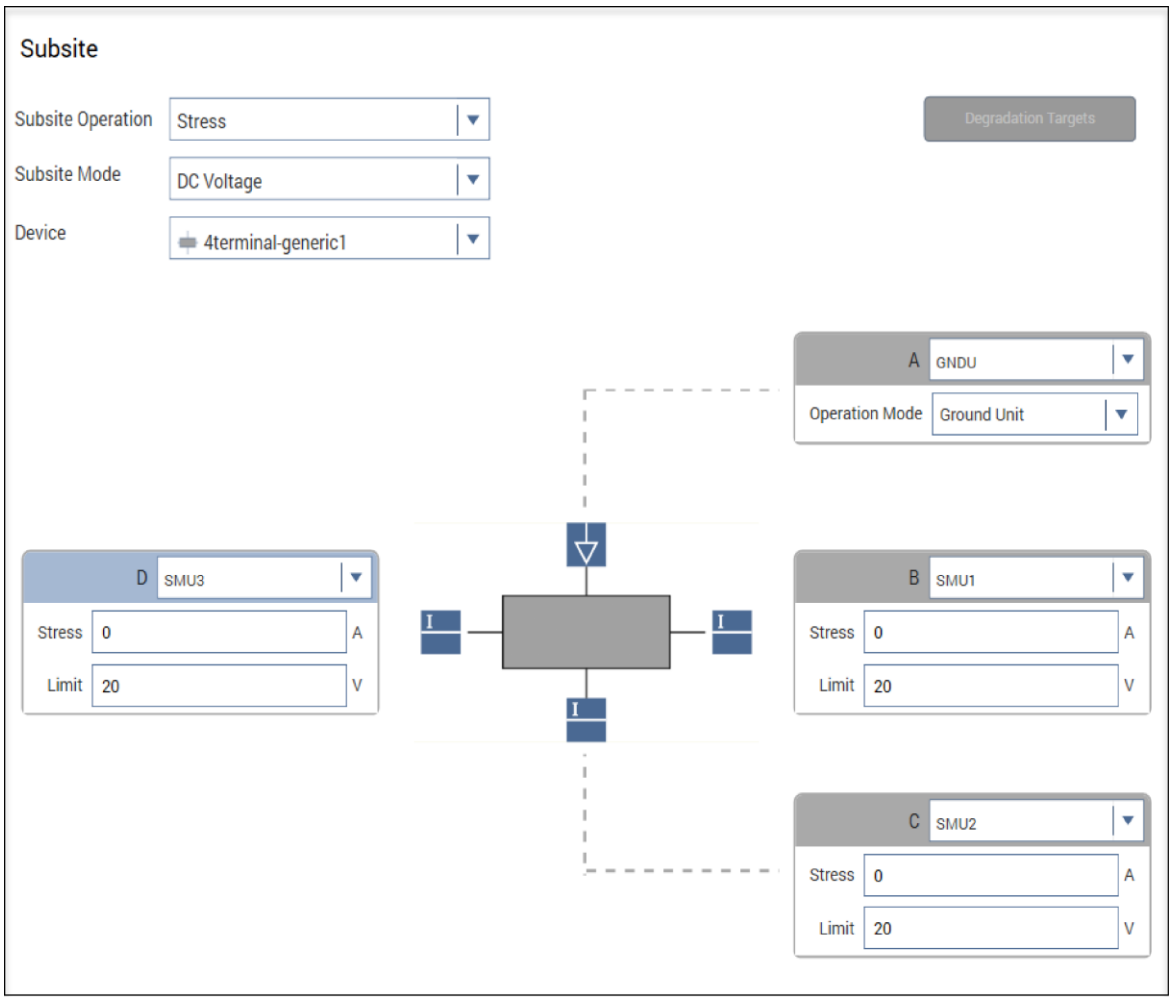
NOTE

If a device terminal is set to GNDU, its Operation Mode is always Ground Unit.

To select and configure a device:

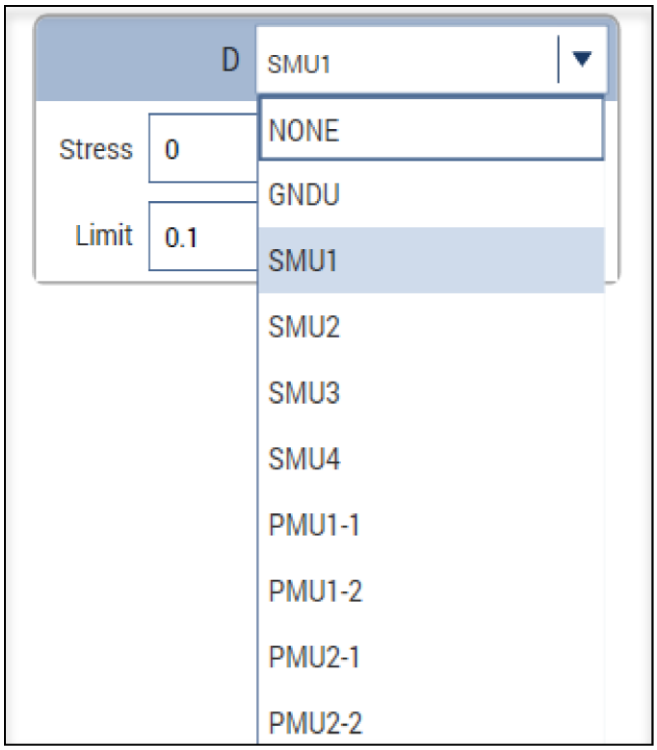
1. From the Device menu, choose a device to configure for your subsite. A diagram of the device, its terminals, and its terminal connections is displayed. See the following graphic.

Figure 334: Device with terminals assigned to instruments



2. Assign an instrument or function to each terminal of the device, as needed.

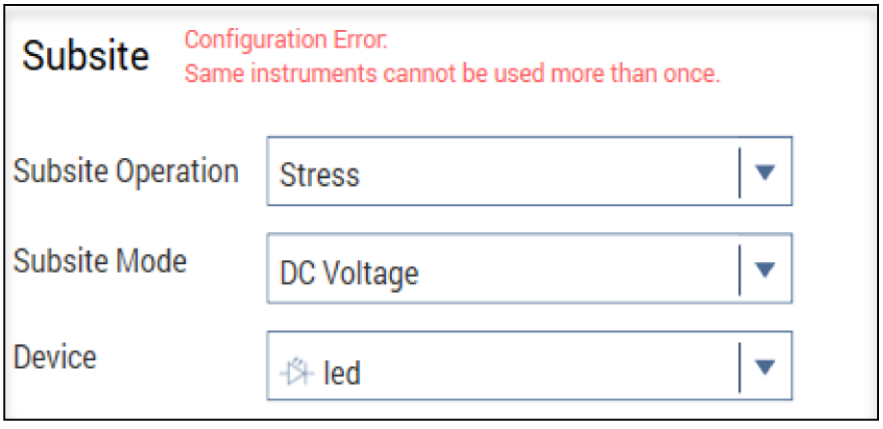
Figure 335: Selecting an instrument or function for the terminal



NOTE

You can only assign one device terminal to an instrument or function at a time. For example, you cannot assign the anode and cathode of a diode to SMU1. If you try to assign multiple terminals, the following error is displayed at the top of the Subsite pane.

Figure 336: Warning for one instrument or function connected to multiple device terminals



3. Specify the device terminal stress settings. The available settings depend on the subsite mode and the instrument or function assigned to the terminal. For more information, see Device terminal options for subsite modes (on page 6-214).

NOTE

For additional device configuration options, see Configure the Terminal Settings (on page 6-221).

Device terminal options for subsite modes

You see different terminal stress settings depending on the subsite mode and the instrument or function you assigned to a terminal. NONE and GNDU are always available regardless of the subsite mode.

For example, if you select DC Voltage as a subsite mode and assign SMU1 to the Gate of a device terminal, you can adjust the stress in volts and limit in amperes. However, if you select the DC Current subsite mode, you will specify the stress in amperes and limit in volts.

The following are device terminal fields you may see:

- **Operation Mode** - Applies to GNDU only. This field cannot be changed.
- **Stress** - The terminal voltage or current stress.
- **Duty Cycle** - The time, as a percentage of the pulse period, that the pulse is on (pulse width).
- **Instrument** - Only available when your test system includes a switch matrix.

Device pin connections for matrix cards

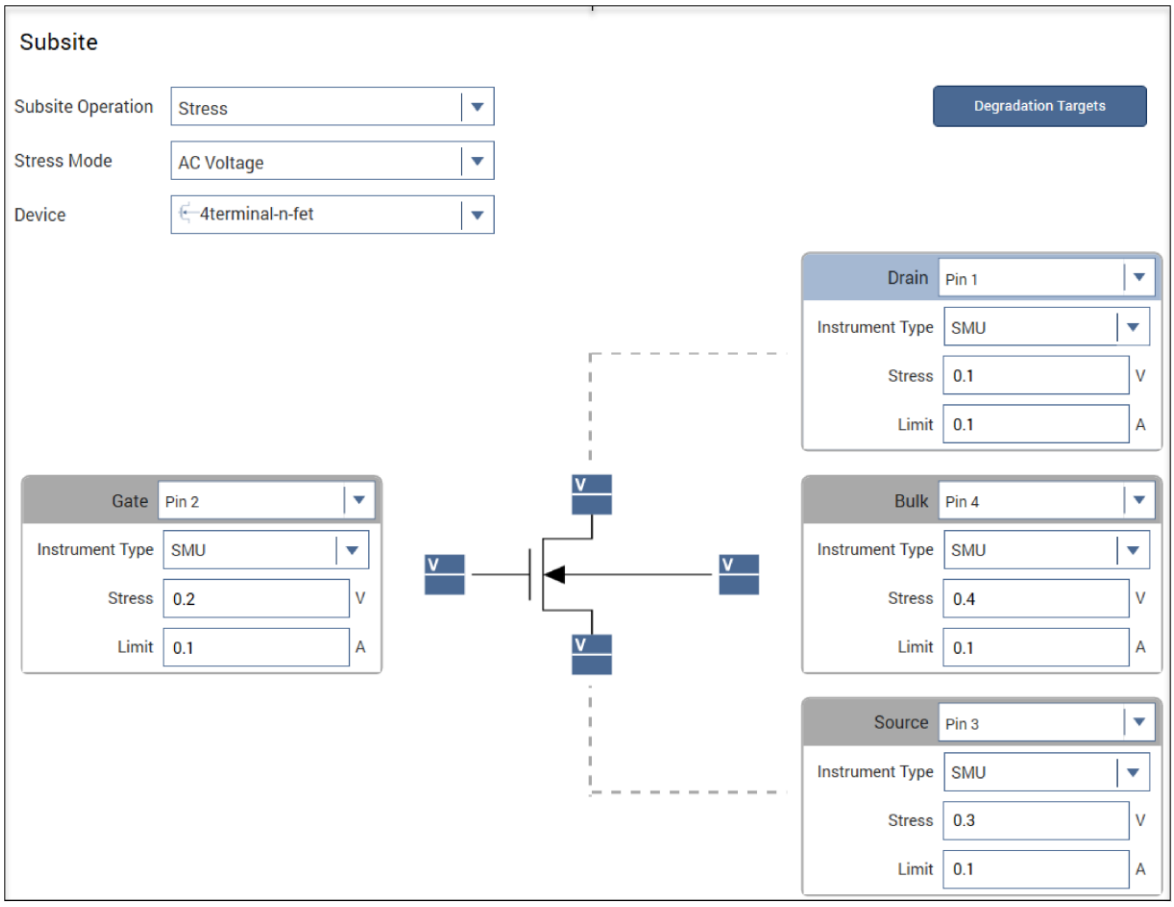
If you have a switch matrix connected to your test system, you will select the pin connection numbers and instrument type when you configure the DUT. The pin number assignments for the device must match the physical connections to the matrix card.

You can also specify the instrument for each terminal:

- **SMU** - This option is available if at least one SMU is connected to the switch matrix.
- **VPU** - Available when at least one PMU or PGU is connected to the matrix and the Stress Mode is AC Voltage or Segment Stress.
- **GNDU** - Available if the 4200A-SCS GNDU is connected to the matrix.
- **NONE** - No instrument is connected to the device terminal.

The following figure shows a subsite device terminal configuration in Clarius with a switch matrix connected to the test system.

Figure 337: Terminal configuration in a test system with a switch matrix



Import KPulse Segment Arb waveform files

If you exported a Segment Arb (SARB) waveform file from KPulse, you can import it when in Segment Stress subsite mode. You must assign a terminal of a device to use a PMU before you can select a SARB waveform file.

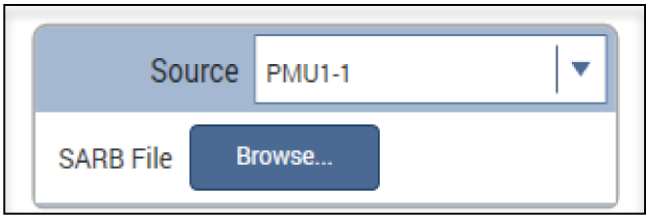
SARB waveform files have the extension `.ksf` and are normally stored at the following location:

```
C:\s4200\kiuser\KPulse\SarbFiles
```

To import a SARB file:

1. In the Subsite pane, select a device terminal.
2. Select a PMU.
3. Select **Browse**.

Figure 338: Selecting a SARB file



4. Select a file.
5. Select **Open**.

Configure the Stress Settings

To configure the Stress Settings for the subsite, select the Stress Settings pane. The Subsite Mode Stress Settings can include values for any of the following:

- Stress timing and counts
- Stress delays
- The terminal power on and power off sequence
- Pulse times

The following topics in this section describe the Stress Settings you may see when configuring your subsite.

Measurement Timing

When you select the Stress operation, you can configure the measurement timing for the Subsite Mode stress cycles. You can select:

- Linear: After the first stress cycle, all stress times are identical.
- Log: After the first stress cycle, all stress times increase logarithmically.
- List: You set the stress cycle times.

NOTE

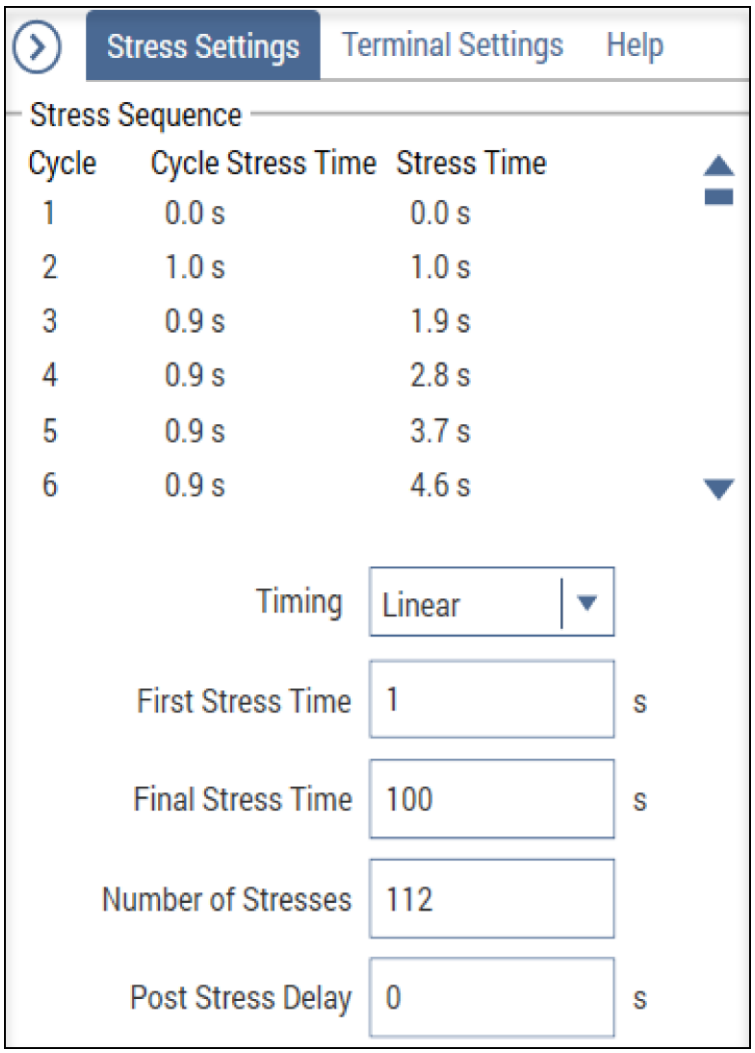
Information that is entered when Linear or Log timing is selected is shown when List is selected. You can use those settings as a starting point when you set up your list.

To set up the Linear and Log timing modes:

1. Select the Stress Settings pane.
2. From the Timing menu, select **Linear** or **Log**.
3. In **First Stress Time**, set the time in seconds that devices are stressed during the first cycle.
4. In **Final Stress Time**, set the time in seconds that devices are stressed during the last cycle.
5. Select a total number of stresses:
   a. If you have selected **Linear** timing, set the **Number of Stresses.** This is the total number of stresses, up to 128.
   b. If you have selected **Log** timing, set the **# Stresses/Decade**. You can select a maximum of 10 per decade. There can be up to 128 stresses for all decades combined.
6. If needed, enter the **Post Stress Delay** in seconds. This is the delay after each stress cycle. It allows the device to reach equilibrium before the next measurement.

Clarius uses the values you entered to calculate the cumulative stress times for the cycles. The times are displayed in the Stress Settings pane in seconds. See the following figure.

Figure 339: Set timing for linear or log measure mode

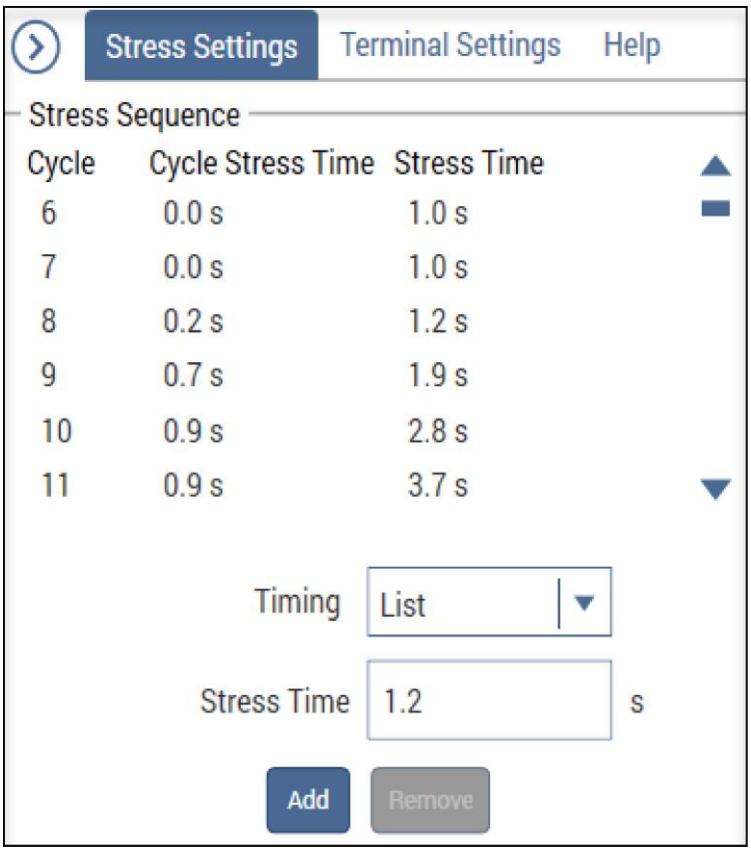


To set up the List timing mode:

1. Select the Stress Settings pane.
2. From the Timing menu, Select **List**.
3. Enter a **Stress Time** in seconds.
4. Select **Add** to add the time to the Stress Times list.
5. Continue adding stress times as needed.
6. To remove a stress time, select the time in the list and select **Remove**.

Clarius uses the values you entered to calculate the cumulative stress times for the cycles. The times are displayed in the Stress Times box in seconds. See the following figure for an example.

Figure 340: Set timing for list measure mode



Power On and Off Sequence

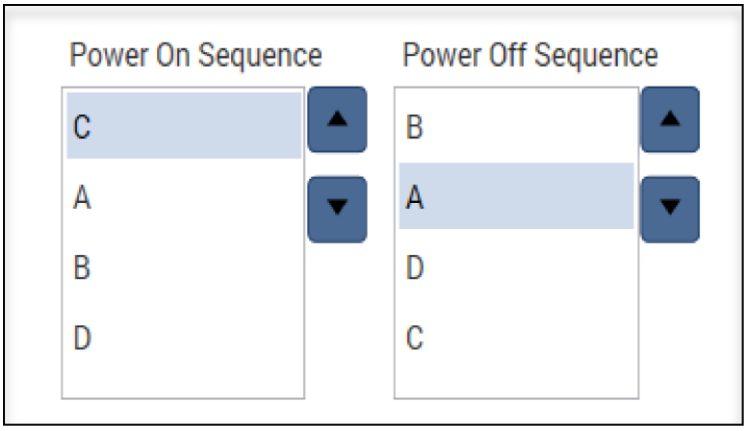
You can set the sequence that the instruments follow for powering and powering down your test device. The names for device terminals (such as drain, gate, and source) and the enabled fields for those terminals are set automatically by Clarius. The terminal names correspond to the terminal names used by the tests for the device.

NOTE

This setting is only available when using the DC Voltage or DC Current subsite modes.

The **Power On Sequence** determines the order that the instruments follow for powering the device. The **Power Off Sequence** determines the order that the instruments follow for powering down the device. You can select the arrow icons to adjust the sequence. See the following graphic.

Figure 341: Adjusting the power on and power off sequence for a four-terminal device



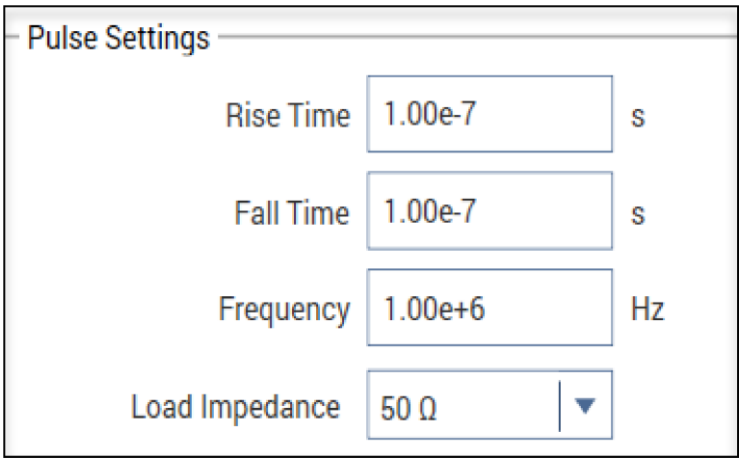
Pulse Settings

NOTE

Pulse Settings are only available when the Subsite Mode is AC Voltage and a PMU or PGU has been assigned to a terminal of a device in the Subsite pane.

To access Pulse Settings settings in AC Voltage Subsite Mode, select the **Stress Settings** pane. You can adjust the rise time, fall time, frequency, and the impedance of the load.

Figure 342: Pulse Settings dialog



Configure the Terminal Settings

The Terminal Settings pane lets you further configure the terminals of a devices. You can adjust the stress conditions specified when you selected a device, and, depending on the subsite mode and the instrument or function assigned to the terminals, perform the following:

- Turn the measurement on and off
- Specify the measurement range
- Specify the stress (voltage) low

NOTE

In the Segment Stress subsite mode, you can only upload a SARB file. See Import KPulse Segment Arb waveform files (on page 6-216) for more information.

See the following graphics for examples.

Figure 343: Terminal settings for a diode assigned to a PMU in DC Voltage subsite mode

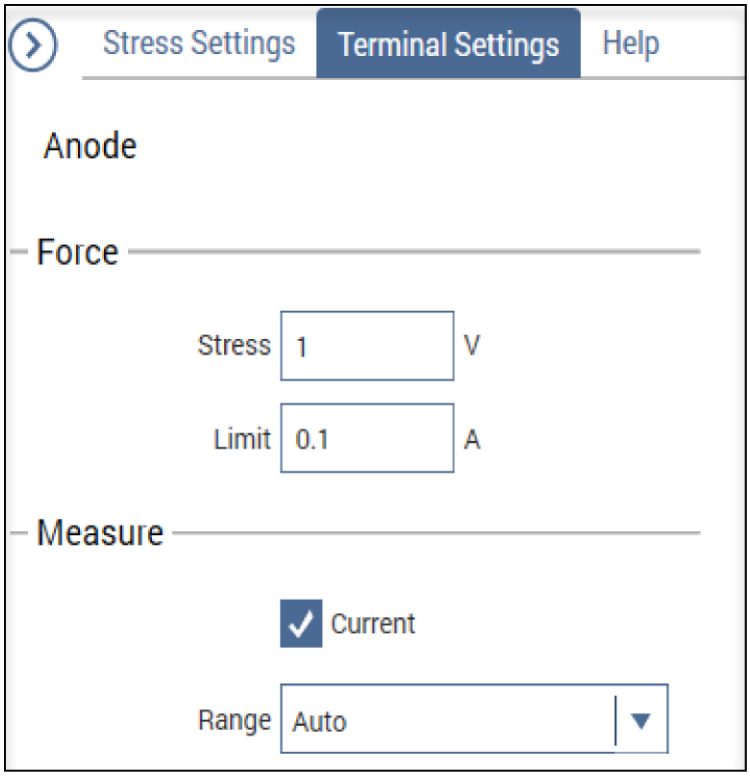
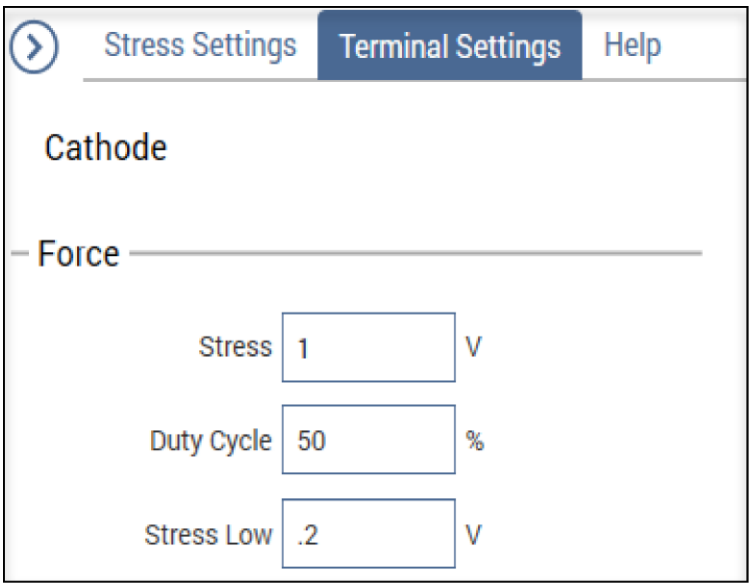


Figure 344: Terminal settings for a diode assigned to a PMU in AC Voltage subsite mode

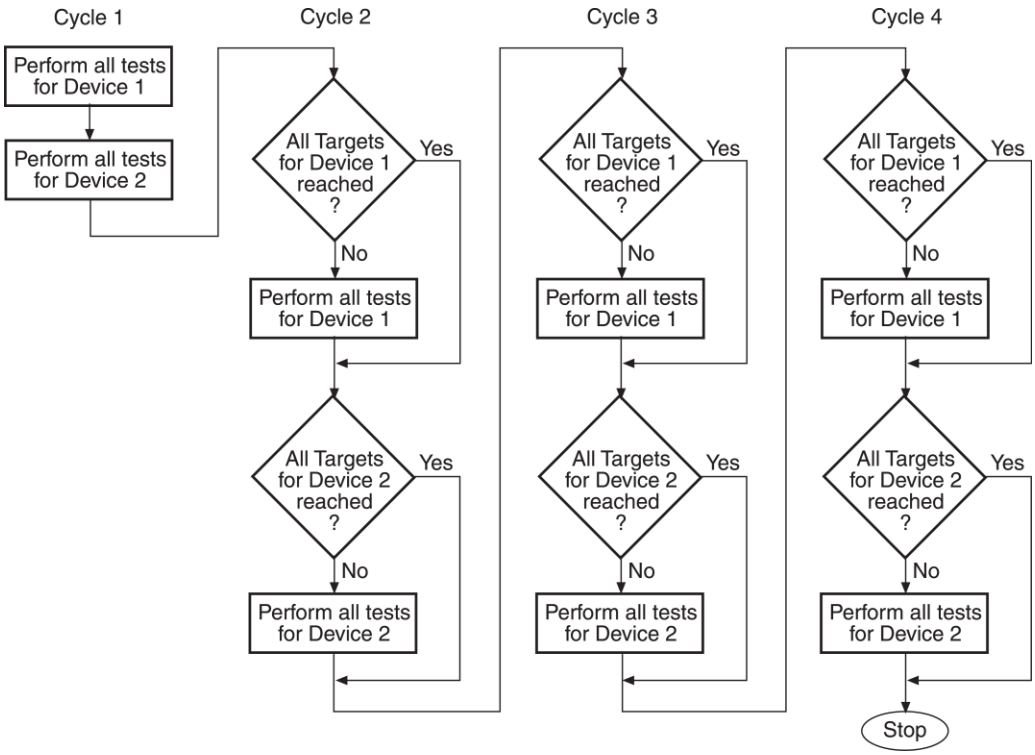


Degradation targets

You can enable an output value as a target and assign it a target value (in % change or absolute value). When all targets for a device are reached, that device is no longer tested. Subsequent cycles bypass the device tests that reached all its targets. The subsite stops when a target on each device is reached or the last subsite cycle is completed.

The testing process for target evaluation is shown in the following flowchart. As a simple example, assume all the targets for both devices are reached after the first cycle of the subsite test. Following the flowchart shows that the tests for cycles 2, 3 and 4 are not performed. The subsite test stops. The graph that is plotted is degradation versus stress time.

Figure 345: Target evaluation process (example for two devices, four cycles)



The Degradation Targets option is only available when the Subsite Operation is Stress and there is at least one output value defined in the tests in your subsite. To enable an output value as a target, refer to Export output values to Analyze sheet (on page 6-224).

When you have defined at least one output value, you can select Degradation Targets from the Subsite panel.

Figure 346: Degradation Targets option

To configure degradation targets:

1. Select **Degradation Targets**.
2. Select **Add Another Target**.
3. Select an output value.
4. Enter a change percentage.
5. Select **Add Another Target** or **OK** to save.

Export output values to Analyze sheet

For subsite cycling, you can export output values from tests into the subsite Analyze sheet. Each time a subsite is cycled, the measurements for the output values are placed in the subsite Analyze sheet. If, for example, the subsite is cycled five times, there are five measured readings for each output value.

NOTE

You must define at least one output value to define Degradation targets (on page 6-223).

To select the values to be exported:

1. Select the test in the project tree.
2. Select **Configure**.
3. In the right pane, select the **Test Settings** tab.
4. Select **Output Values**.
5. Select the values to export into the subsite Analyze sheet.
6. Select **OK**.

Run a complex test

This section describes how to run individual tests, devices, subsites, and sites. It also describes how to run the stress modes.

NOTE

If Clarius detects an above-normal temperature condition at any SMU, it protects system outputs by preventing or aborting a test run and reporting the condition in the message area of the Clarius window. If the condition occurs when a test is attempted, Clarius prohibits execution. If the condition occurs during a test, Clarius aborts the test.

Run projects, subsites, devices, and tests

You can execute an entire project or individual parts of the project.

While a test is running, you can view test data in the Analyze pane.

The Message area at the bottom of the center pane of Clarius displays the start time, stop time, and total execution time of the components that were run.

Run projects

Executing an entire project runs its components, including sites, subsites, devices, tests, and actions, in the order in which they appear in the project tree.

Clarius only runs items that are selected in the project tree and are at a lower level than the highlighted item.

When you run a project, the data from each test is inserted into its own Analyze Run sheet. Each new run creates a new Run History, with its own sheet. For more about worksheets, refer to Analyze data (on page 6-232).

NOTE

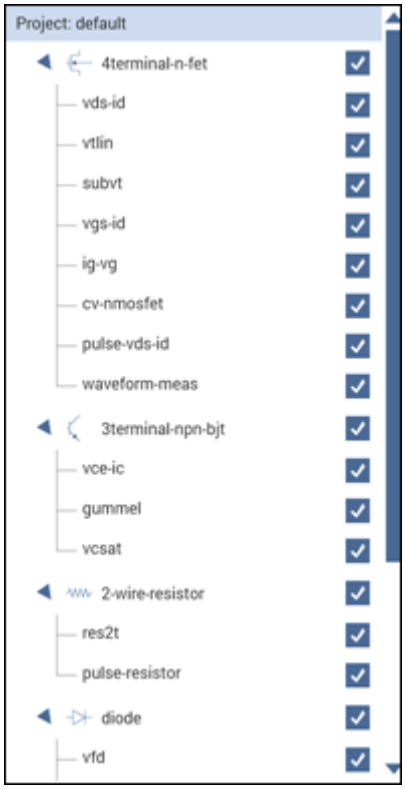
To abort a test, select **Stop**. All test and action execution stops immediately.

The following example uses the Demo Project to demonstrate how to run a project.

To run all objects in a project:

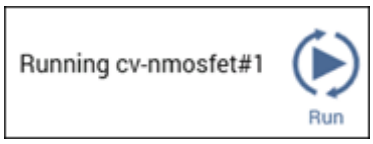
1. Open the **Demo Project**. Refer to Open a project (on page 6-19).
2. Make sure the check boxes are selected for all items in the project tree.
3. Highlight the project name, as shown in the figure below.

Figure 347: Run a project



4. Select **Run**. The Run icon changes as shown below. The active test is listed to the left of Run. The Stop icon changes to red.

Figure 348: Run icon while a test is running



When the test completes, a beep sounds and the run arrows around the Run icon are no longer displayed.

Run individual devices

You can run the tests for a single device in the project tree.

When you run the tests for a single device, the tests are run in the order in which they appear in the project tree. Only the tests that have checkboxes selected are run. In the following example, all the tests for the diode device are run except for `vrd`.

NOTE

To abort a test, select **Stop**. All test and action execution stops immediately.

The following example uses the Demo Project to demonstrate how to run tests for a device.

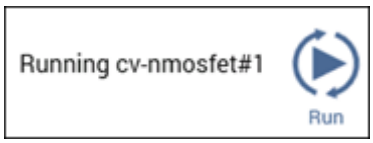
To run tests for a device:

1. Open the **Demo Project**. Refer to Open a project (on page 6-19).
2. Make sure the check boxes are selected for all items under the `diode` device except for `vrd`, as shown in the figure below.
3. Highlight the device name, **diode**, as shown in the figure below.

Figure 349: Run tests for a single device

4. Select **Run**. The Run icon changes as shown below. The active test is listed to the left of Run. The Stop icon changes to red.

Figure 350: Run icon while a test is running



When the test completes, a beep sounds and the run arrows around the Run icon are no longer displayed.

Run an individual test

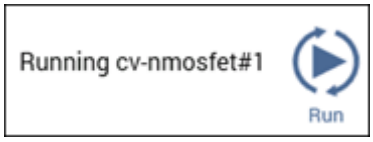
NOTE

To abort a test, select **Stop**. All test and action execution stops immediately.

To run an individual test in the project tree:

1. In the project tree, make sure the check box for the test is selected.
2. Highlight the test.
3. Select **Run**. The Run icon changes as shown below. The active test is listed to the left of Run. The Stop icon changes to red.

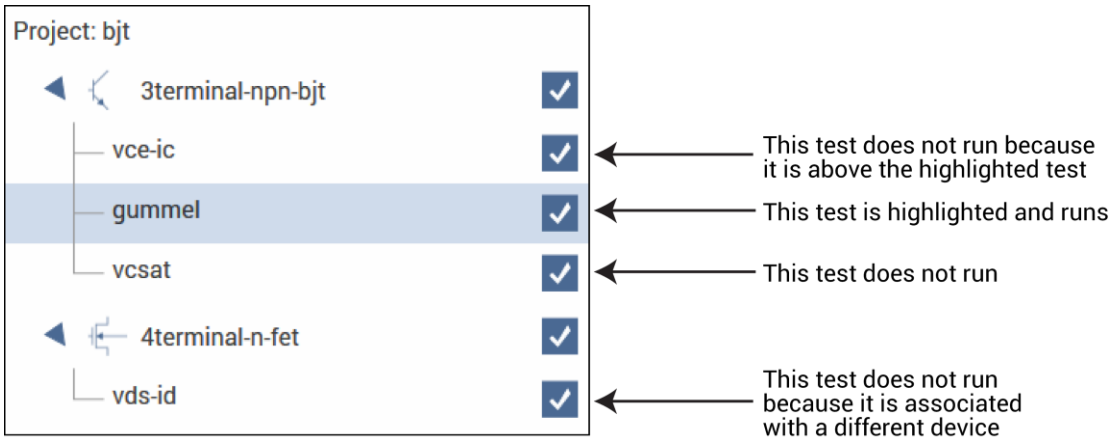
Figure 351: Run icon while a test is running



When the test completes, a beep sounds and the run arrows around the Run icon are no longer displayed.

In the following example, only the `gummel` test runs. Even though the other tests are selected, they are at the same level as the `gummel` test in the hierarchy.

Figure 352: Run specific tests



Run an individual subsite

When you run an individual subsite, only the components that are assigned to it run in the order in which they appear in the project tree.

When you run the components for a subsite, the actions and tests are run in the order in which they appear in the project tree. Only the devices, actions, and tests that have check boxes selected are run.

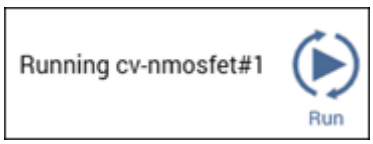
NOTE

To abort a test, select **Stop**. All test and action execution stops immediately.

To run an individual subsite:

1. Make sure the check boxes are selected for all items in the subsite that you want to include.
2. Highlight the subsite name.
3. Select **Run**. The Run icon changes as shown below. The active action or test is listed to the left of Run. The Stop icon changes to red.

Figure 353: Run icon while a test is running



When the test completes, a beep sounds and the run arrows around the Run icon are no longer displayed.

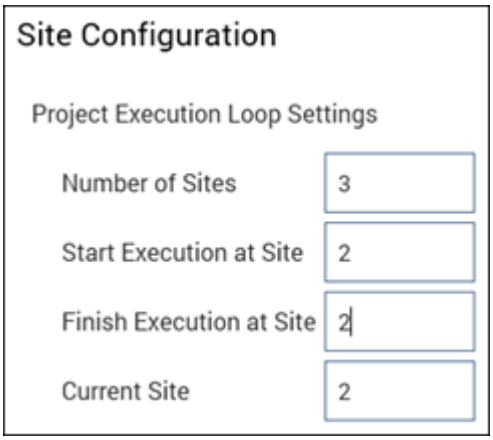
Run a single site

Running a project runs all the sites that are defined for the project. However, you can run a single site if needed.

To run a single site:

1. In the project tree, select the site.
2. Select **Configure**.
3. Set **Start Execution at Site** and **Finish Execution at Site** to the site you want to run. In the following example, only Site 2 is run when you select **Run**.

Figure 354: Settings to run Site 2 only



4. Set **Current Site** to the site that you want to run.
5. Select **Run**. The Run icon changes as shown below. The active site, actions, and tests are listed to the left of Run. The Stop icon changes to red.

Figure 355: Run icon while a test is running

When the site completes, a beep sounds and the run arrows around the Run icon are no longer displayed.

Cycle a subsite

Subsite cycling allows you to repeatedly cycle through the subsite tests. The data for every repeated test is acquired and placed in its Analyze Stress tab.

Measured readings (output values) can be exported from individual tests into the subsite.

To run cycling for a single subsite:

1. Set up the subsite as described in Configure Subsite Cycling (on page 6-195).
2. In the project tree, select the subsite.
3. Select **Run**.

To run cycling for multiple subsites:

1. Set up the subsite as described in Configure Subsite Cycling (on page 6-195).
2. In the project tree, select the project.
3. Make sure the subsites you want to run are checked in the project tree.
4. Select **Run**.

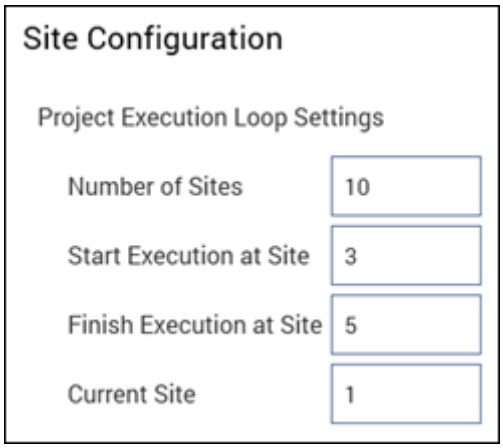
Multi-site execution

Running a project runs all the sites that are defined for the project. However, you can run a subset of the sites if needed.

To run some sites:

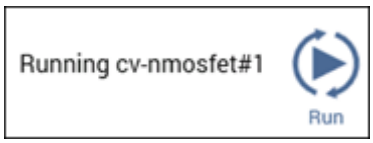
1. In the project tree, select the site.
2. Select **Configure**.
3. Set **Start Execution at Site** and **Finish Execution at Site** to the sites you want to run. In the following example, executing the site will run sites 3, 4, and 5.

Figure 356: Multi-site test sequence



4. Select **Run**. The Run icon changes as shown below. The active sites, actions, and tests are listed to the left of Run as they are executed. The Stop icon changes to red.

Figure 357: Run icon while a test is running



When the site completes, a beep sounds and the run arrows around the Run icon are no longer displayed.

Analyze data

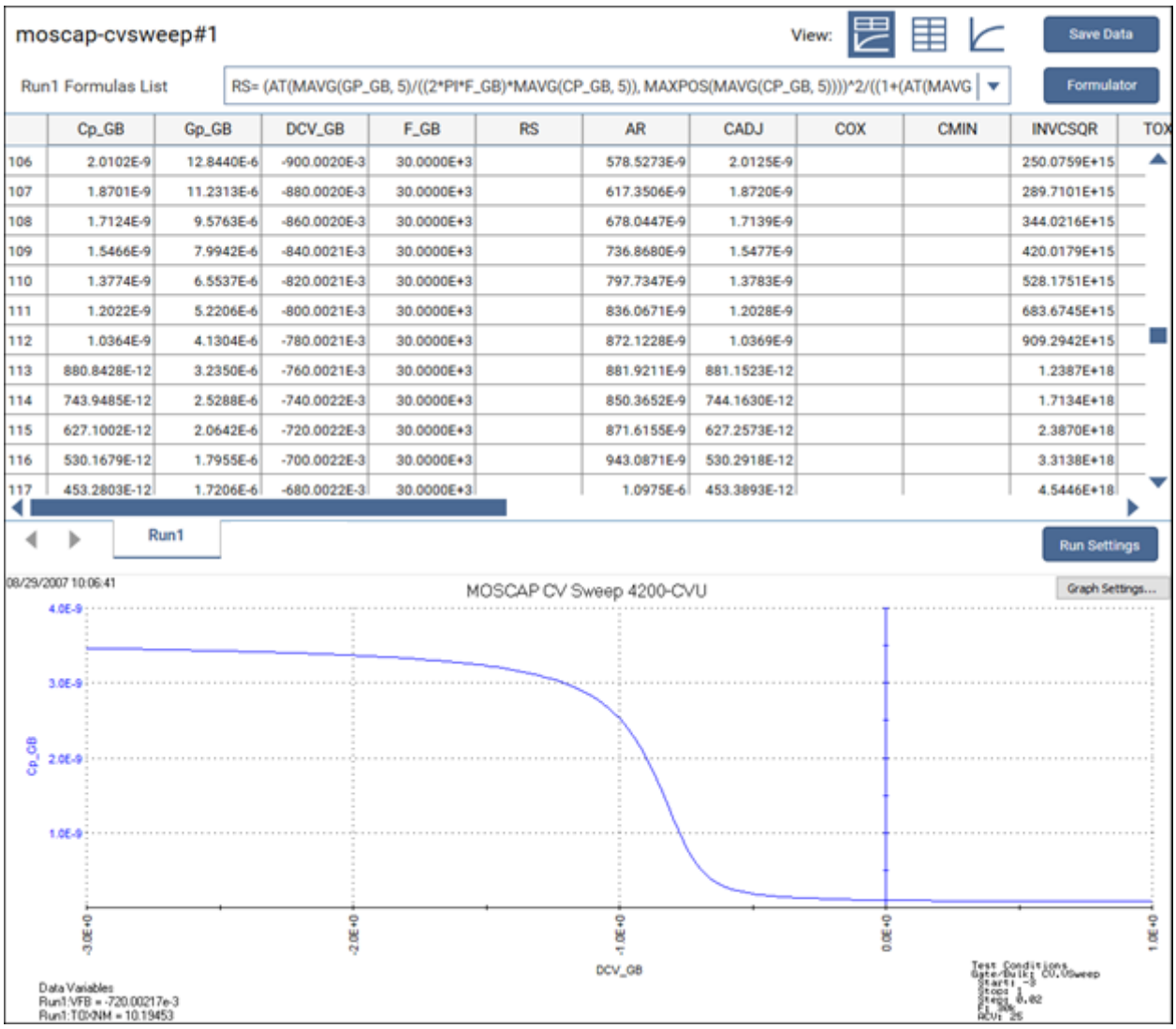
When you run tests for projects or individual subsites, devices, and tests, Clarius records the data in the Analyze pane. You can display the data in a spreadsheet and as a graph. You can also select data for specific tests to display at the project level. Select **Analyze** to view the spreadsheet and graph.

You can change the display to show only the spreadsheet or only the graph using the View buttons in the upper right of the pane.

While a test is running, you can watch the data populate the Run sheet and graph.

You can use the Formulator to have Clarius extract additional parameter information from the data, using formulas that you create. The Formulator calculation results are placed in the Run sheet, in addition to the raw data. Refer to The Formulator (on page 6-290) for more information.

Figure 358: Analyze pane



Spreadsheet

The Analyze spreadsheet is a spreadsheet that is compatible with Microsoft® Excel®. It can contain one or more Run sheets, which correspond to the test runs selected in its Run History. Data is recorded in real time as the test executes. The Run sheets also record data generated by the Formulator. The Run sheets are read-only.

NOTE

If you have tests that were created in earlier versions of Clarius, you may also have a Calc sheet. Calc sheets were used for custom, test-specific data analysis. Calc sheets created for earlier versions are maintained in the present version of Clarius as read-only sheets.

Run sheet

In the Analyze pane, the Run sheet numerically displays data for a test in a worksheet that is compatible with Microsoft® Excel®. There is a Run sheet for every run of every test for each site. Each column contains the results for one test parameter or for a Formulator calculation. Each column heading identifies the data in that column. Headings are assigned by the data source:

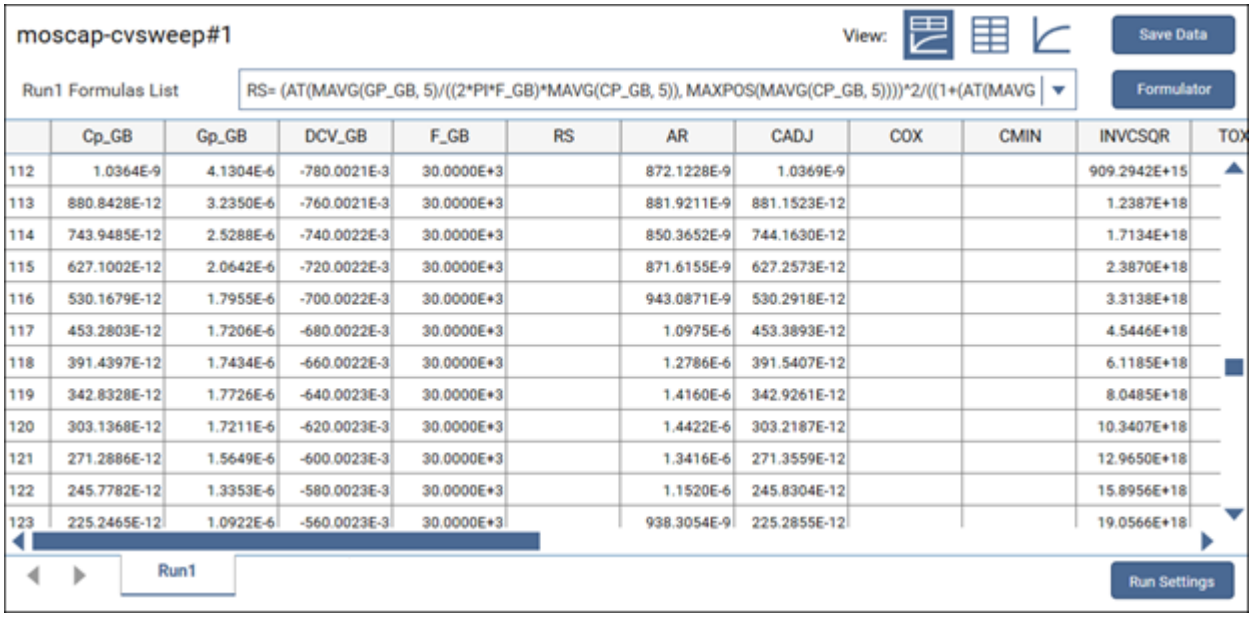
- For ITMs, Clarius assigns headings
- For UTMS, the KULT programmer assigns headings
- For Formulator calculations, the name defined in the Formulator determines the headings

The contents of the Run sheet are read-only.

If a project contains multiple instances of a test under the same name, each instance generates its own data. Tests are numbered in the order in which they are added to the project. Ensure that you select the correct instance of the test in the project tree to view the data for that test.

The figure below shows a Run sheet that contains data generated by the `moscap-cvsweep` test.

Figure 359: Analyze Run sheet



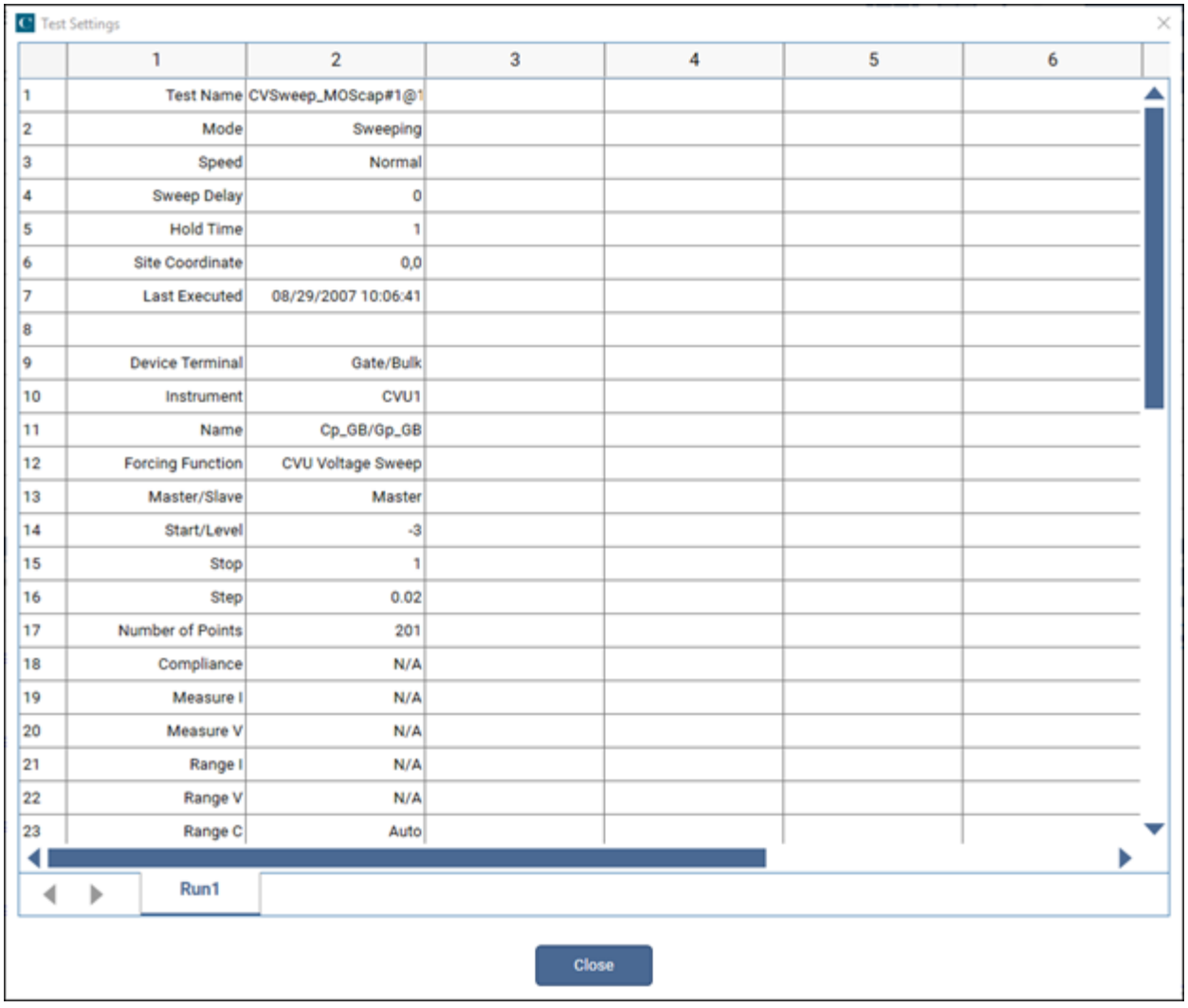
	Cp_GB	Gp_GB	DCV_GB	F_GB	RS	AR	CADJ	COX	CMIN	INVCSQR	TOX
112	1.0364E-9	4.1304E-6	-780.0021E-3	30.0000E+3		872.1228E-9	1.0369E-9			909.2942E+15	
113	880.8428E-12	3.2350E-6	-760.0021E-3	30.0000E+3		881.9211E-9	881.1523E-12			1.2387E+18	
114	743.9485E-12	2.5288E-6	-740.0022E-3	30.0000E+3		850.3652E-9	744.1630E-12			1.7134E+18	
115	627.1002E-12	2.0642E-6	-720.0022E-3	30.0000E+3		871.6155E-9	627.2573E-12			2.3870E+18	
116	530.1679E-12	1.7955E-6	-700.0022E-3	30.0000E+3		943.0871E-9	530.2918E-12			3.3138E+18	
117	453.2803E-12	1.7206E-6	-680.0022E-3	30.0000E+3		1.0975E-6	453.3893E-12			4.5446E+18	
118	391.4397E-12	1.7434E-6	-660.0022E-3	30.0000E+3		1.2786E-6	391.5407E-12			6.1185E+18	
119	342.8328E-12	1.7726E-6	-640.0023E-3	30.0000E+3		1.4160E-6	342.9261E-12			8.0485E+18	
120	303.1368E-12	1.7211E-6	-620.0023E-3	30.0000E+3		1.4422E-6	303.2187E-12			10.3407E+18	
121	271.2886E-12	1.5649E-6	-600.0023E-3	30.0000E+3		1.3416E-6	271.3559E-12			12.9650E+18	
122	245.7782E-12	1.3353E-6	-580.0023E-3	30.0000E+3		1.1520E-6	245.8304E-12			15.8956E+18	
123	225.2465E-12	1.0922E-6	-560.0023E-3	30.0000E+3		938.3054E-9	225.2855E-12			19.0566E+18	

If #REF is displayed, there is a problem with the data, such a divide by zero error or a test that was run with no device under test (DUT) attached.

All data in the worksheets of the Analyze spreadsheet can be exported to Microsoft Excel format. Refer to Save test results and graphs (on page 6-254).

The Run Settings button displays the test configuration information from the Configure pane for the test run selected in the Run History pane. These settings are read-only. You can right-click to copy data from the Run Settings dialog box. An example is shown in the following figure.

Figure 360: Run Settings example



	1	2	3	4	5	6
1	Test Name	CVSweep_MOScap#1@1				
2	Mode	Sweeping				
3	Speed	Normal				
4	Sweep Delay	0				
5	Hold Time	1				
6	Site Coordinate	0,0				
7	Last Executed	08/29/2007 10:06:41				
8						
9	Device Terminal	Gate/Bulk				
10	Instrument	CVU1				
11	Name	Cp_GB/Gp_GB				
12	Forcing Function	CVU Voltage Sweep				
13	Master/Slave	Master				
14	Start/Level	-3				
15	Stop	1				
16	Step	0.02				
17	Number of Points	201				
18	Compliance	N/A				
19	Measure I	N/A				
20	Measure V	N/A				
21	Range I	N/A				
22	Range V	N/A				
23	Range C	Auto				

To access the Analyze Run sheet for a test:

1. In the project tree, select the test.
2. Select **Analyze**. The sheet and graph are displayed.
3. To hide columns in the graph, select the columns you want to hide. Right-click and select **Hide**.
4. To display the columns, right-click and select **Unhide** to choose from a list of columns, or select **Unhide All** to display all columns.
5. If data is from a CVU, right-click to select the **CVU Data Type**.

Formulas List of the Run worksheet

If a column in the Run worksheet contains the results of Formulator calculations, you can display the equation that was used to get the results.

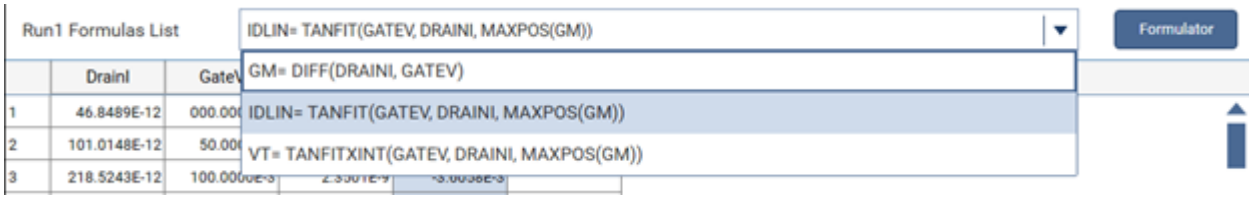
The `#REF` notation in a cell indicates that the Formulator could not calculate a valid value. This can occur if a Formulator function needs multiple rows as arguments, if a calculated value is out of range, or if a divide by zero is attempted.

A column contains multiple instances of `#REF` if the Formulator function requires multiple preceding cells for the calculation. For example, if the `MAVG` function is using five data points to calculate a moving average of a column that contains five values, the first two and last two cells contain `#REF`.

To display the formula:

1. Select the **RUNn Formulas List** box. A list of formulas is displayed.
2. Select the formula you want to display.
3. If you need to make changes to the formula, select **Formulator** to open the formula in the Formulator.

Figure 361: Displaying a Formulator equation using the formula box



Terminal Settings pane (Analyze)

When Analyze is selected, the Terminal Settings pane displays the settings for the presently selected test.

For information on the settings that are available, refer to Test and terminal setting descriptions (on page 6-26).

Run History

When a test is selected in the project tree and Analyze is selected, the right pane includes a Run History tab.

If the test has not yet been run, this displays sample data and a sample graph in the Analyze pane.

If you ran the test, the latest test is displayed at the top of the Run History pane. The data and graph from this test is displayed in the Analyze pane.

You can have up to 10,000 run histories for each test.

Each run history entry includes:

- A timestamp that shows the date and time when the test was run.
- The execution time.
- Rating stars that you can use to flag specific tests.
- Notes. Select the **More** link and select the text box to add notes about the run. Select **Enter** when the notes are complete.

Figure 362: Run history pane

NOTE

If you brought in data from a 4200 project, each set of appended data is shown as a separate run history.

Changing the name of a test run

You can change the name of the run. If you change the name of a run, the new name is displayed in the following locations:

- The data tab name in the Analyze pane.
- In the Graph Settings, the Legend, Data Variables, and graph configuration dialog boxes.
- In the Formulator dialog list under Formula Set, which allows you to use formulas from this run in a new run.
- In the Excel tab name and in the Settings sheet if you save the data grid to a Microsoft Excel file using Save Data.
- In the test library if you add the test to the test library.

The names for test runs:

- Must be less than 19 characters.
- Must be unique; multiple tests cannot have the same name. The name comparison is not case-sensitive. For example, `Best` and `best` are considered the same name.
- Must not use Clarius reserved names, including `Data`, `Calc`, `Settings`, and `Run1` to `Run9999`.

To change the name of a test run:

1. In the Run History pane, select **More**.
2. In the Run number text box, type the new name.
3. Select information in the text box that contains the run number, then select **Enter**. The name you entered is displayed, followed by the run number.

Changing the display of Run Histories

You can select up to 128 Run Histories. Each run has a separate tab in the Analyze sheet and all selected runs are graphed. By default, the latest Run History data and graph is displayed in the Analyze pane. Select a different Run History to view that information in the Analyze pane.

To delete a run history, right-click the Run History and select **Delete**. To delete all run histories, including the sample data, right-click in the Run History pane and select **Delete All**.

You can change the sort order of the Run Histories using the options at the bottom of the Run History pane. Select **Date**, **Name**, **Checked**, or **Stars** to sort the Run Histories in ascending or descending order using that option.

If you have more than one pane of run histories, use the arrow buttons to move between the panes.

To select specific run histories, highlight a run history, right-click and select **Select**. To select a range, highlight another run history and right-click and select **End Select**. To delete the selected run histories, right-click and select **Delete Selected**. To clear the selections, right-click and select **Clear Select**.

You can change the runs that are graphed using the **Graph Settings > Define Graph** option. Refer to Change the graph settings (on page 6-259) for detail. To clear the selections, right-click and select **Unclear All**.

Searching for a Run History

You can search for a run history by date.

To search for a run history by date:

1. At the bottom of the Run History pane, enter the date.
2. Select **Date Search**.

Copy the settings of a Run History to the Configure screen

You can copy the settings in a Run History to the Configure pane. All settings in the Configure pane are replaced by the settings from the selected Run History.

To copy the settings:

1. In the Run History pane, right-click the test run.
2. Select **Load Configuration**.
3. You can copy the configuration from the selected Run History to the Configure pane.

Analyze test data at the project level

You can select test data to display at the project level. This allows you to compare data from different tests in your project.

You can use the Formulator at the project level. The Formulator allows you to make data calculations on test data and on the results of other Formulator calculations. Refer to The Formulator (on page 6-290) for information on creating formulas.

You can also graph data. Refer to Graph (on page 6-255) for information on setting up the graph.

Save the project to preserve the Formulator and graph settings with the project.

Select data for the project-level Analyze pane

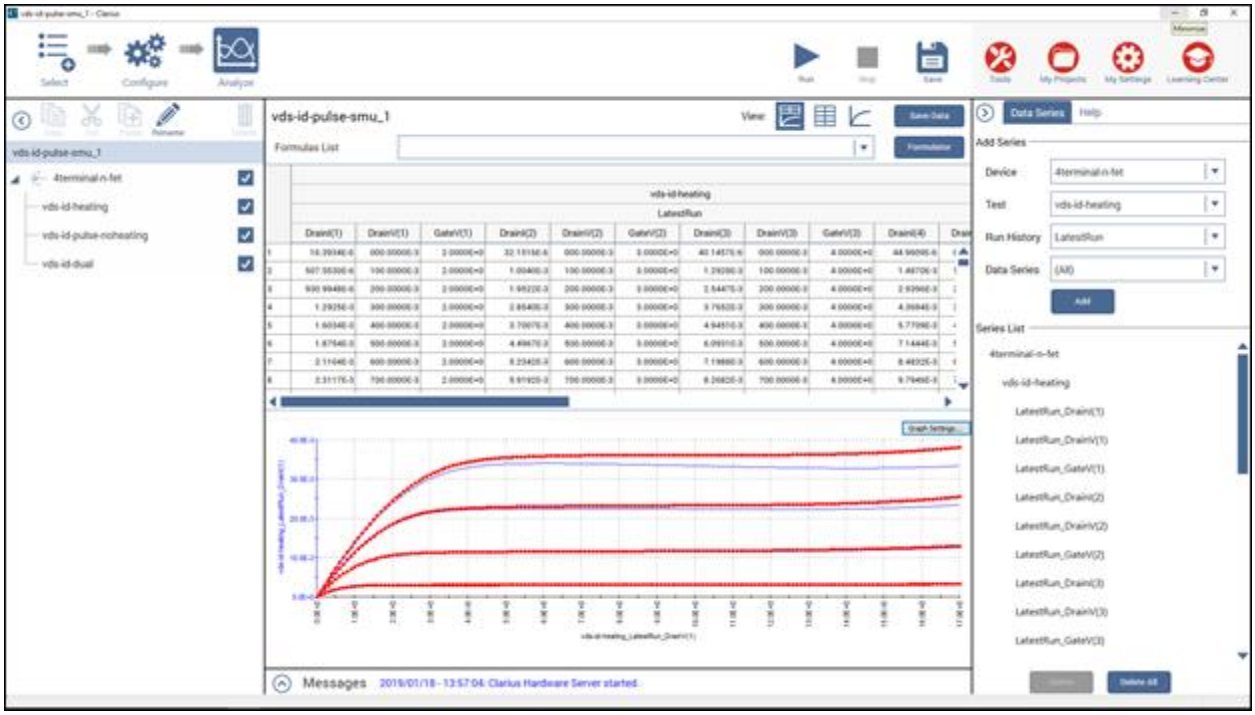
You can select up to 255 columns of data to display in the project-level Analyze pane.

To select test data to display at the project level:

1. In the project tree, select the project.
2. In the right pane, select **Data Series**.
3. From **Device**, select the device that contains the data.
4. From **Test**, select the test that contains the data.
5. From **Run History**, select the run you that contains the data. To always display the most recent data, select **LatestRun**.
6. From **Data Series**, select All to display all available data or a specific set of data.
7. Select **Add**.

The data is displayed in the Analyze pane. The selected data series are also listed in the Series List in the right pane. An example of a project with a graph set up is shown in the following figure.

Figure 363: Analyze pane for a project



Remove data from the project-level Analyze pane

You can remove data from the project level Analyze display. The data is removed from the project Analyze pane. The original data remains intact.

To remove all data from the project level Analyze display:

1. In the right pane, select **Delete All**.

To remove all data for a device from the project level Analyze display:

1. In the right pane, select the device or test.
2. Select **Delete**. All data for the device or test is removed.

To remove data for specific run histories from the project level Analyze display:

1. In the right panel, select the run histories from the Series List. You can use Ctrl+click to select multiple individual run histories. You can use Shift+click to select a series of run histories.
2. Select **Delete**.

Measurement status

Many tests provide status information for the measurements in the Analyze pane of Clarius. For the 4210-CVU, 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 errors.

When a measurement error occurs, the data values in the flagged data row are color-coded to identify the fault type as follows:

Red	Measurement timeout
Magenta	Measurement overflow
Orange	Auto Balance Bridge (ABB) not locked

Figure 364: Status tab showing faults (example)

	Cp_AB	Gp_AB	DCV_AB	F_AB	CVU1S
2	-158.6073E-9	181.4187E-3	-4.8000E+0	1.0000E+6	00080002
3	-158.5251E-9	181.2184E-3	-4.6000E+0	1.0000E+6	00080002
40	-282.2661E-12	3.7564E-3	2.8000E+0	1.0000E+6	03010002
41	-283.2877E-12	3.7541E-3	3.0000E+0	1.0000E+6	03010002

Placing the cursor on a flagged `CVU1S` cell opens a window that summarizes the error.

Status codes

The 16 basic codes used for 4210-CVU measurement status are listed in table below. Each code is represented as a 32-bit hexadecimal value (0x).

CVU measurement status codes (CVU1S)

#	Code	Description
1	000000*mr*	No faults
2	8xxx00*mr*	Measurement timeout occurred
3	01xx00*mr*	CVH1 current measurement overflow
4	02xx00*mr*	CVH1 voltage measurement overflow
5	03xx00*mr*	CVH1 I and V measurement overflow
6	08xx00*mr*	CVH1 ABB not locked
7	09xx00*mr*	CVH1 ABB not locked, I measurement overflow
8	0Axx00*mr*	CVH1 ABB not locked, V measurement overflow
9	0Bxx00*mr*	CVH1 ABB not locked, I and V measurement overflow
10	xx0100*mr*	CVL1 I measurement overflow
11	xx0200*mr*	CVL1 V measurement overflow
12	xx0300*mr*	CVL1 I and V measurement overflow
13	xx0800*mr*	CVL1 ABB not locked
14	xx0900*mr*	CVL1 ABB not locked, I measurement overflow
15	xx0A00*mr*	CVL1 ABB not locked, V measurement overflow
16	xx0B00*mr*	CVL1 ABB not locked, I and V measurement overflow

As shown in this table, the *mr* value is the last two digits of each code.

mr value		
	00	Lowest range (1 µA) used for the impedance measurement
	01	Middle range (30 µA) used for the impedance measurement
	02	Highest range (1 mA) 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 (on page 4-27) 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 4210-CVU card. Try resetting the hardware and running the project test again. If this error reoccurs, contact Keithley Instruments.

To reset the hardware:

1. Select **Start**.
2. Type **resethw**.
3. Select the instruments that need reset.
4. Click **Reset.**

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 was not locked when the measurement was made. The readings and calculation results may not be accurate.

Subsite cycling Analyze sheets

Spreadsheet data for the subsite is acquired in the Analyze sheet for the subsite. The Analyze spreadsheet is a spreadsheet that is compatible with Microsoft® Excel®. It can contain the following sheets:

- Calc: Provides a spreadsheet that you can use for custom, test-specific data analysis. If there are multiple same-named instances of a test in a project, the Calc worksheet equations are unique for each instance. Cells in the Calc worksheet may be hot-linked to cells in the Run and Settings worksheets.
- Settings: Documents the test configuration and site number. This worksheet is read-only.

To display subsite data:

1. In the project tree, select the subsite.
2. Select **Analyze**.

Stress/measure mode Analyze sheet

The following figure shows an example sheet for a subsite that has one device. Analyze spreadsheet columns include:

- Column A: The cycles that were run.
- Column B: The stress times (in seconds) for all cycles. The stress for the first cycle is 0.0 seconds. This is the no-stress cycle for HCI testing.
- Column C: The measured readings for the first output value, I_{DOFF} reading for the ID#1 test.
- Column D: Starting with the second cycle, lists the percent change between each post-stress I_{DOFF} reading and the pre-stress I_{DOFF} reading in the first cycle. The percent change value is calculated as:
  % Change = ABS[(Post-Stress Rdg – Pre-Stress Rdg) / Pre-Stress Rdg x 100]

For the example in the following figure, percent change I_{DOFF} for the second cycle is calculated as:

% Change I_{DOFF} = ABS[(82.2013e-15 – 291.1666e-15) / 291.1666e-15 x 100]
= ABS[–208.9653e-15 / 291.1666e-15 x 100]
= ABS[–0.7176 x 100]
= 71.8

- Column E: The target value that was assigned to the output value in the Subsite Stress Properties. A target value of 0.0 indicates that the target for I_{DOFF} is disabled. A target is reached when the % change value equals or exceeds the target value.
- Starting with Column F, every three columns provide readings for another output value, the percent change, and the target value.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	
1	Cycle	Stress	id#1	% Change	Target	id#1	% Change	Target	id#1	% Change	Target	igleak#1	% Change	Target	vte
2	Index	Time	IDOFF	IDOFF	Value	IDLIN	IDLIN	Value	IDSAT	IDSAT	Value	IGLEAK	IGLEAK	Value	VTE
3	1	0.00	308.1823E-15		0.000	269.9511E-6		0.000	4.6286E-3		0.000	-6.8007E-12		0.000	926
4	2	10.00	74.6224E-15	75.786		276.2232E-6	2.323		4.6160E-3	0.271		168.0886E-15	102.472		921
5	3	21.54	102.9612E-15	66.591		279.2826E-6	3.457		4.6100E-3	0.402		169.3108E-15	102.490		919
6	4	46.42	177.5246E-15	42.396		284.6862E-6	5.458		4.5991E-3	0.636		202.7564E-15	102.981		915
7	5	100.00	133.8737E-15	56.560		292.0504E-6	8.186		4.5843E-3	0.956		213.4304E-15	103.138		910
8	6	215.44	171.1995E-15	44.449		297.4943E-6	10.203		4.5738E-3	1.183		238.4493E-15	103.506		906
9	7	464.16	188.5646E-15	38.814		299.3781E-6	10.901		4.5711E-3	1.241		235.6590E-15	103.465		905
10	8	1000.00	75.7835E-15	75.409		299.9014E-6	11.095		4.5715E-3	1.233		231.4591E-15	103.403		904
11	9	2154.43	131.7894E-15	57.237		299.8906E-6	11.091		4.5733E-3	1.195		202.8820E-15	102.983		904
12	10	4641.59	96.0319E-15	68.839		301.8061E-6	11.800		4.5714E-3	1.235		162.6893E-15	102.392		903
13	11	10000.00	45.7663E-15	85.150		302.9184E-6	12.212		4.5712E-3	1.239		104.9323E-15	101.543		902
14	12	20000.00	96.5853E-15	68.660		302.4817E-6	12.051		4.5738E-3	1.184		59.8365E-15	100.880		902

4terminal-n-fet | Stress | Calc | Settings

Stress/measure mode Analyze graph

The graphs for the stress/measure mode plot degradation (in %) versus the stress times. Each data point in the graph represents the device degradation (% Change) for tests after each stress cycle (stress time).

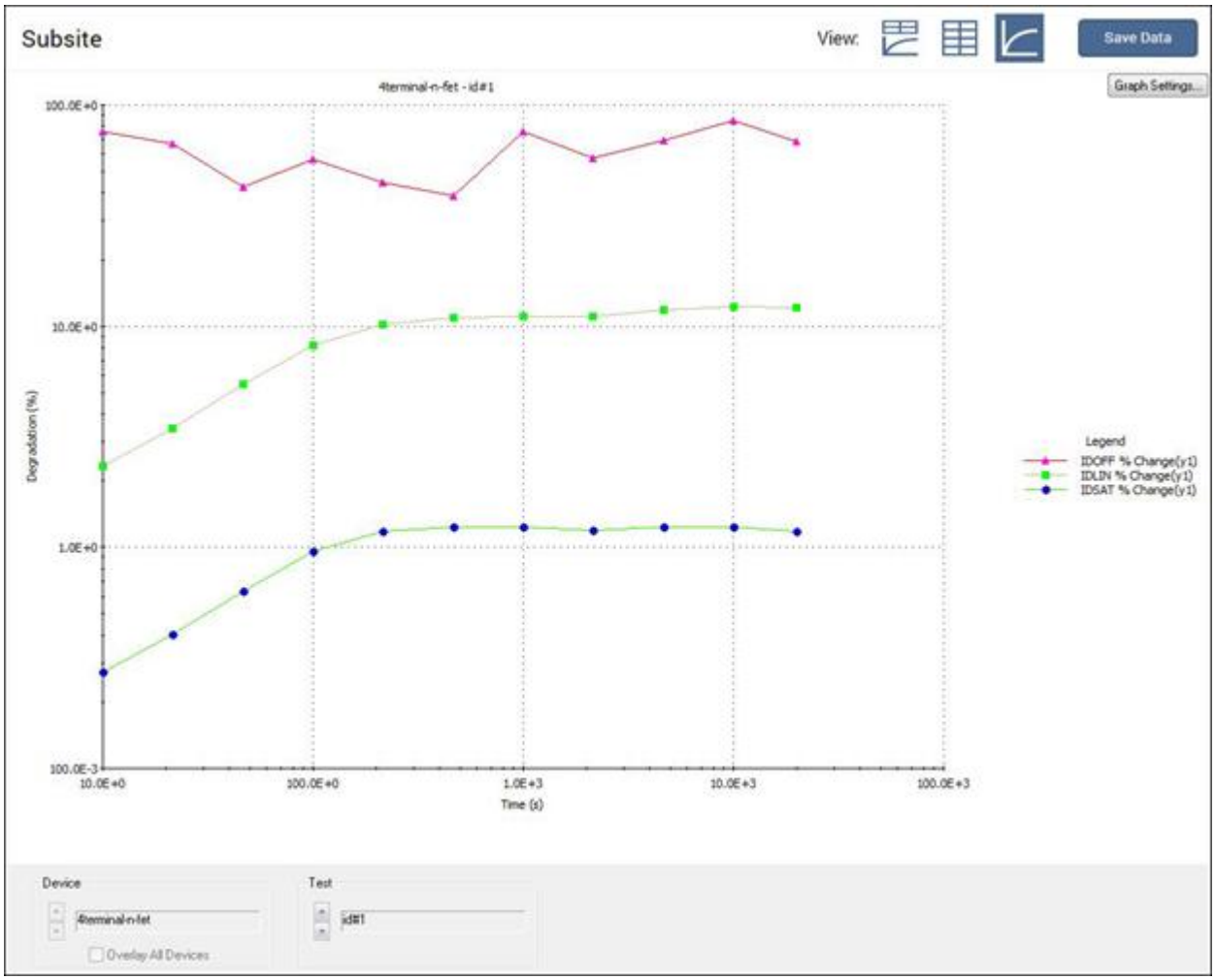
The graph below traces for test `id#1` for the `4terminal-n-fet` device. The three traces are for Output Values I_{DOFF}, I_{DLIN}, and I_{DSAT}.

The options at the bottom of the graph allow you to change which device and test data is graphed. The options are:

- **Device:** Select the device for which to display data. For a single-device subsite, this option is not available.
- **Overlay All Devices:** Select this option to display all the graph traces for all devices that were measured by the selected test. For a single-device subsite, this option is not available.
- **Test:** Select the test for which to display data.

The output values for each test can be graphed as shown in the following figure.

Figure 365: Stress/measure mode graph

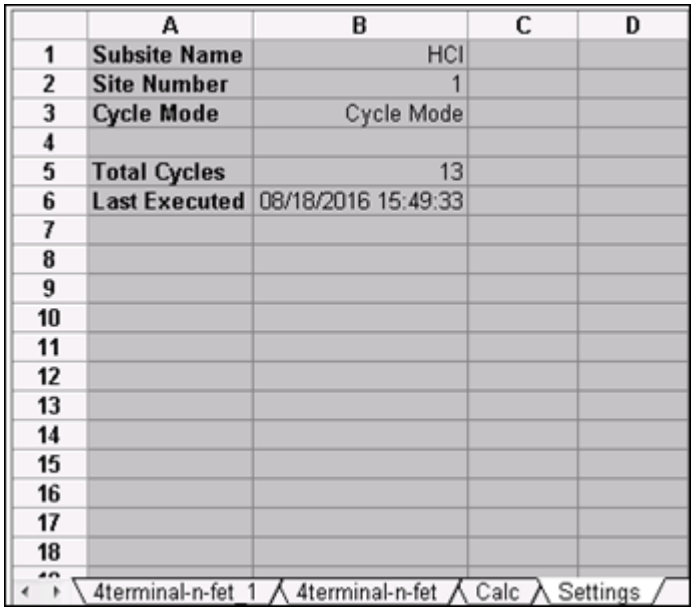


Subsite Settings sheet

The Settings sheet displays information about the subsite cycling setup. It also lists the output values for each device and test. To display the Settings sheet, select the **Settings** tab on the subsite Analyze sheet.

An example of the Settings sheet for the Cycle mode is shown in the figure below.

Figure 366: Analyze Subsite Settings sheet for Cycle mode

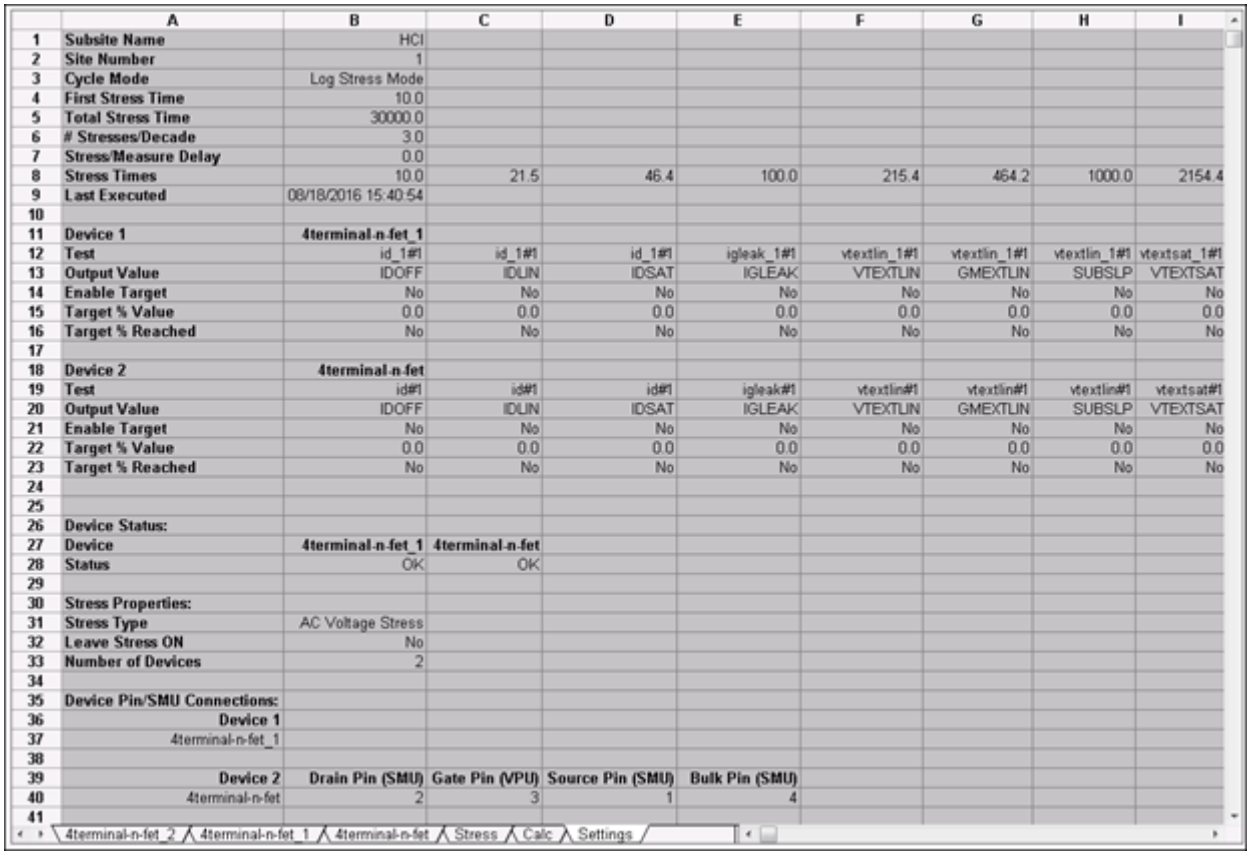


An example of the Settings sheet for the Stress/Measure mode is shown in the following figure. It is similar to the Settings sheet for the Cycle mode, but it includes information on targets. For each enabled target, the target value is listed. After subsite cycling, it also indicates if targets have been reached.

The top rows show the subsite cycling setup.

The next set of rows show the output values and target information. It lists the target percentage values that indicate if the target was reached.

Figure 367: Subsite Analyze Settings sheet for Stress/Measure mode



	A	B	C	D	E	F	G	H	I
1	**Subsite Name**	HCI							
2	**Site Number**	1							
3	**Cycle Mode**	Log Stress Mode							
4	**First Stress Time**	10.0							
5	**Total Stress Time**	30000.0							
6	**# Stresses/Decade**	3.0							
7	**Stress/Measure Delay**	0.0							
8	**Stress Times**	10.0	21.5	46.4	100.0	215.4	464.2	1000.0	2154.4
9	**Last Executed**	08/18/2016 15:40:54							
10									
11	**Device 1**	**4terminal-n-fet_1**							
12	**Test**	id_1#1	id_1#1	id_1#1	igleak_1#1	vtextlin_1#1	vtextlin_1#1	vtextlin_1#1	vtextsat_1#1
13	**Output Value**	IDOFF	IDLIN	IDSAT	IGLEAK	VTEXTLIN	GMEXTLIN	SUBSLP	VTEXTSAT
14	**Enable Target**	No	No	No	No	No	No	No	No
15	**Target % Value**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	**Target % Reached**	No	No	No	No	No	No	No	No
17									
18	**Device 2**	**4terminal-n-fet**							
19	**Test**	id#1	id#1	id#1	igleak#1	vtextlin#1	vtextlin#1	vtextlin#1	vtextsat#1
20	**Output Value**	IDOFF	IDLIN	IDSAT	IGLEAK	VTEXTLIN	GMEXTLIN	SUBSLP	VTEXTSAT
21	**Enable Target**	No	No	No	No	No	No	No	No
22	**Target % Value**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
23	**Target % Reached**	No	No	No	No	No	No	No	No
24									
25									
26	**Device Status:**								
27	**Device**	**4terminal-n-fet_1**	**4terminal-n-fet**						
28	**Status**	OK	OK						
29									
30	**Stress Properties:**								
31	**Stress Type**	AC Voltage Stress							
32	**Leave Stress ON**	No							
33	**Number of Devices**	2							
34									
35	**Device Pin/SMU Connections:**								
36	**Device 1**								
37	4terminal-n-fet_1								
38									
39	**Device 2**	**Drain Pin (SMU)**	**Gate Pin (VPU)**	**Source Pin (SMU)**	**Bulk Pin (SMU)**				
40	4terminal-n-fet	2	3	1	4				
41									

4terminal-n-fet_2 | 4terminal-n-fet_1 | 4terminal-n-fet | Stress | Calc | Settings

Calc sheet

The Calc sheet is available when the Subsite Analyze pane is displayed.

The Calc sheet allows you to:

- Hot-link and copy values and information from the Run and Settings sheets, including Formulator calculations.
- Do additional data analysis or scratch pad calculations.
- Graph the calculation results using the graph. Any Calc sheet column with an entry in the first row is automatically available in the graph as a potential plot variable. Refer to Define data to be graphed (on page 6-256) to graph Calc sheet data.

For all cells that contain hot-linked data or data-derived values from a test, Clarius calculates in real-time as the test is run.

NOTE

Avoid placing numbers or other information that you do not want to display as parameter names in the first row of a Calc sheet. The Data Series option of the graph definition uses this row for assigning data series.

You can enter formulas and perform calculations in the Calc sheet. The Calc sheet is provided under the assumption that most users are already familiar with the use of spreadsheets.

Before performing calculations with the Calc sheet, review the available Calc mathematical functions in Calc sheet function definitions (on page 6-344).

The Calc sheet is compatible with Microsoft® Excel®.

To open a Calc sheet:

1. Select the subsite.
2. Select **Analyze**.
3. Select the **Calc** sheet.

Link Run and Settings worksheet cells to Calc worksheet cells

You can link cells from the Run worksheets to the Calc worksheet. When the contents of linked worksheet cells change, the content of the corresponding worksheet cells change to match.

To link the contents of a worksheet cell:

1. Identify the worksheet and cell number that you want to link.
2. In the Calc worksheet, select the cell where you want to create the link.
3. To link:
   - To a cell in a Run sheet, enter `=TabName`*`X`*`!`*`CellNumber`*, where `TabName` is the name of the spreadsheet tab, *`X`* is the number of the Run worksheet, and *`CellNumber`* is the cell. For worksheet 4terminal-n-fet, cell number A2, you would enter `=4terminal-n-fet!A2`.
   - To a cell in the Settings worksheet, type in `=Settings!`*`CellNumber`*, where *`CellNumber`* is the cell number.
4. Press **Enter**. The formula is replaced by the data from the Run or Settings worksheet.

Once you have a link to a cell, you can link to cells that are adjacent below or to the right of the linked data.

To link to adjacent cells:

1. Select the Calc worksheet cell that contains the linked data and the cells to which you want to link adjacent cells.
2. Right-click the cell and select **Fill Down** or **Fill Right**. The new data is displayed immediately when you release.

Calc menu option

You can right-click the Calc spreadsheet to access options for working with the spreadsheet.

The options are described below.

Cut: Remove content from a cell into the clipboard.

Copy: Copy content from a cell into the clipboard.

Paste: Paste content from the clipboard into the selected cell.

Insert: Insert cells, rows, or columns into the Calc sheet.

Delete: Remove cells, rows, or columns from the Calc sheet.

Clear: Clear formatting, values, or both from the selected cells.

Format Cells: Allows you format the selected cells, including fonts, number types, alignment, borders, and patterns.

Fill Down: Use the Fill Down command to fill a selected range of cells with the contents of the top cell in the column. Select the cells you want to use as the original and the cells below that cell. Right-click and select Fill Down. The content of the top row of cells is filled into the selected cells.

Fill Right: Use the Fill Right command to fill a selected range of cells with the contents of the left-most cell in the column. Select the cells you want to use as the original and the cells to the right of that cell. Right-click and select Fill Right. The content of the left column of cells is filled into the selected cells.

Hide column: Hide the column from view.

Unhide column: Restore the column that was hidden with Hide.

Settings worksheet

The Settings worksheet records test configuration information from the Configure pane for the last execution of a test. The Settings worksheet is read-only, but you can link any of its contents to the Calc worksheet.

Save test results and graphs

You can save test results and graphs.

The sheet data is saved in a format that is compatible with the Microsoft® Excel® application.

You can save graphs to `jpg`, `bmp`, `png`, or `gif` format.

To save test results and graphs:

1. Select **Analyze**. The test results are shown as data in a spreadsheet and on the graph, as shown in the figure below.
2. Select **Save Data**. The Save Test Data As dialog box is displayed.
3. If you would like to use the same name for graphs and sheets, enter the name in **Common Filename** and select **Populate**. The names are changed. No change is made to the file locations.
4. If you are saving the data in the sheet, select the file location and name in the Sheet field.
5. If you are saving a graph, select the file locations and graph names in the Graph1 and Graph2 fields.
6. If you are saving a graph, select the **Graph File Format**.
7. To save the information:
   - In the Run sheet: Select **Save Sheet**.
   - To save the information in the graph, select **Save Graph1** or **Save Graph2**.
   - To save both the data and the graphs: Select **Save All**.

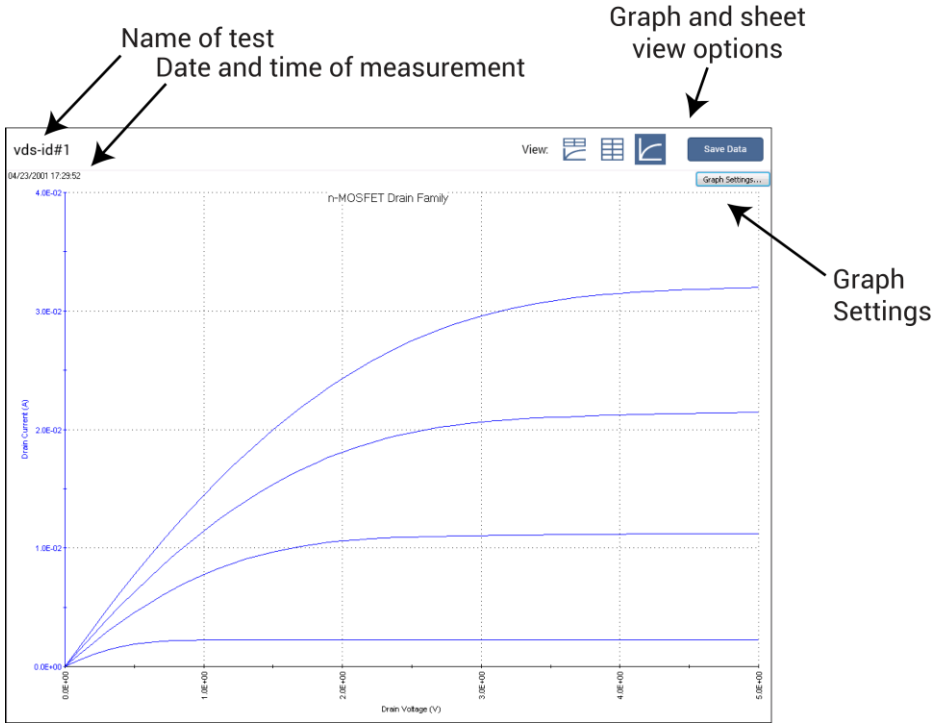
Graph

The Analyze graph allows you to create and export graphs of the test and test analysis results. The graph provides you with flexible plot-data selection, formatting, annotation, and numerical coordinate display using precision cursors.

The graph displays the data from the Run and Calc sheet tabs for the selected Run History. Each run updates the graph to display the latest set of data.

You can change the format of the graph using the Graph Settings.

Figure 368: Example of an Analyze graph

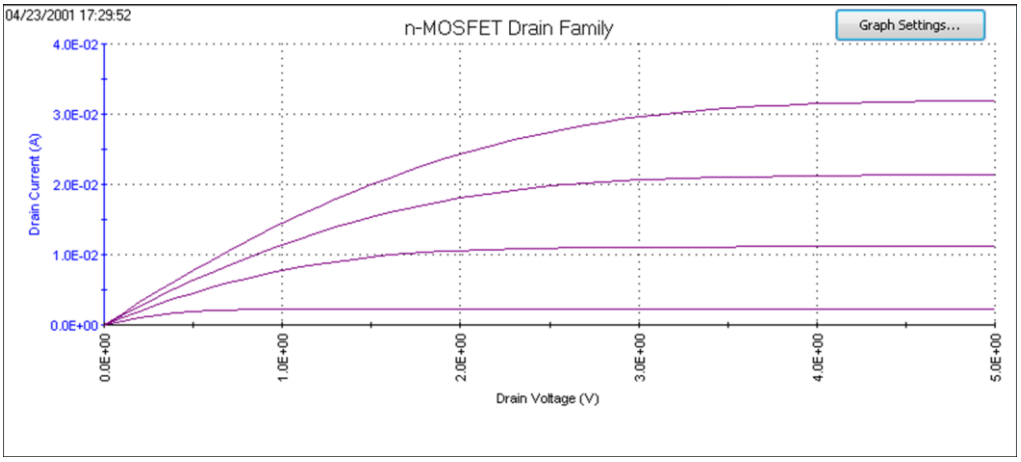


Open a graph

To open a graph:

1. In the project tree, select a test.
2. Select **Analyze**. The graph is displayed at the bottom of the center pane. The time and date when the data was generated are displayed in the upper left corner.
3. To enlarge the graph, select the Graph view option.

Figure 369: n-MOSFET drain family of curves graph example



Define data to be graphed

The Graph Definition dialog box displays the data series that you can show on the graph. The names of the data series are from the first row of the Run spreadsheet. If there are multiple parameters with the same name, an * is displayed after the parameter name in the Graph Definition dialog box.

The Sheet column shows you whether the data came from the Run tab or the Calc tab.

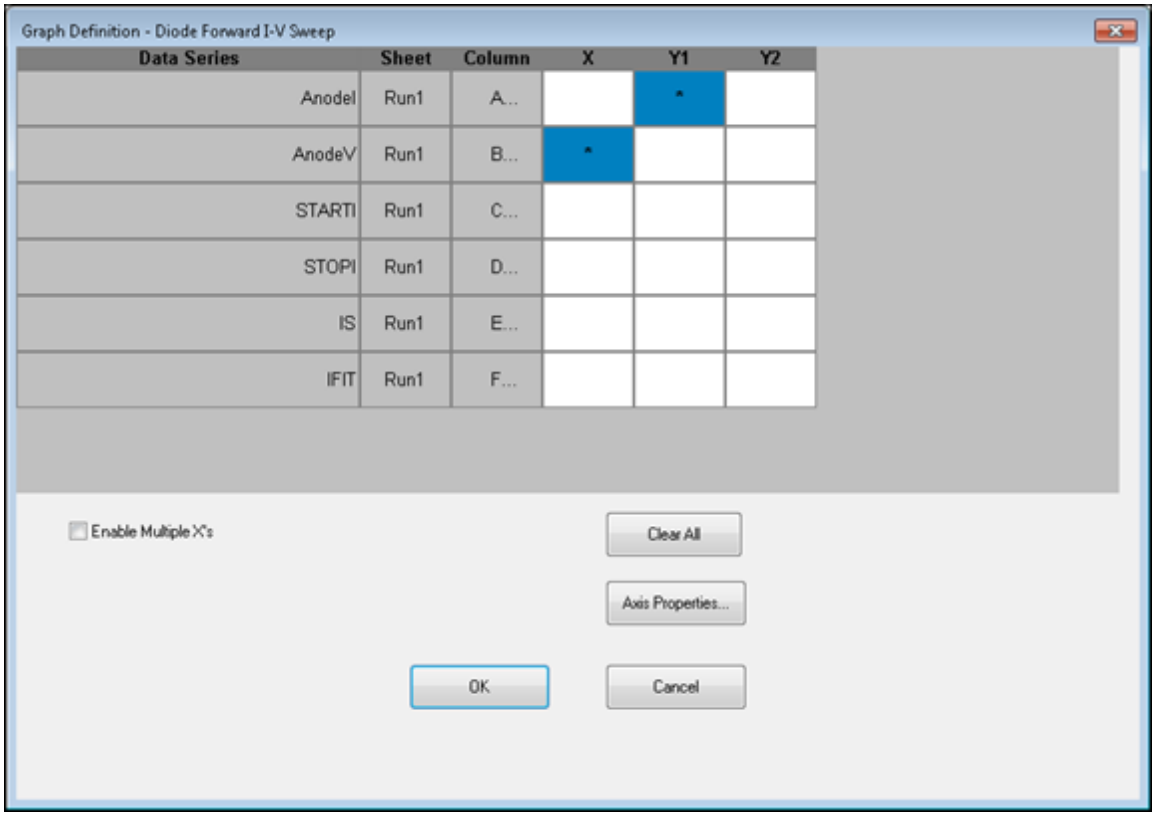
Column shows you the column where the data in the sheet the data came from.

You can use Axis Properties to change the axis. See Define the axis properties (on page 6-260) for information.

To define a graph:

1. Select **Graph Settings**.
2. Select **Define Graph**. An example of the Graph Definition dialog box is shown in the following figure.

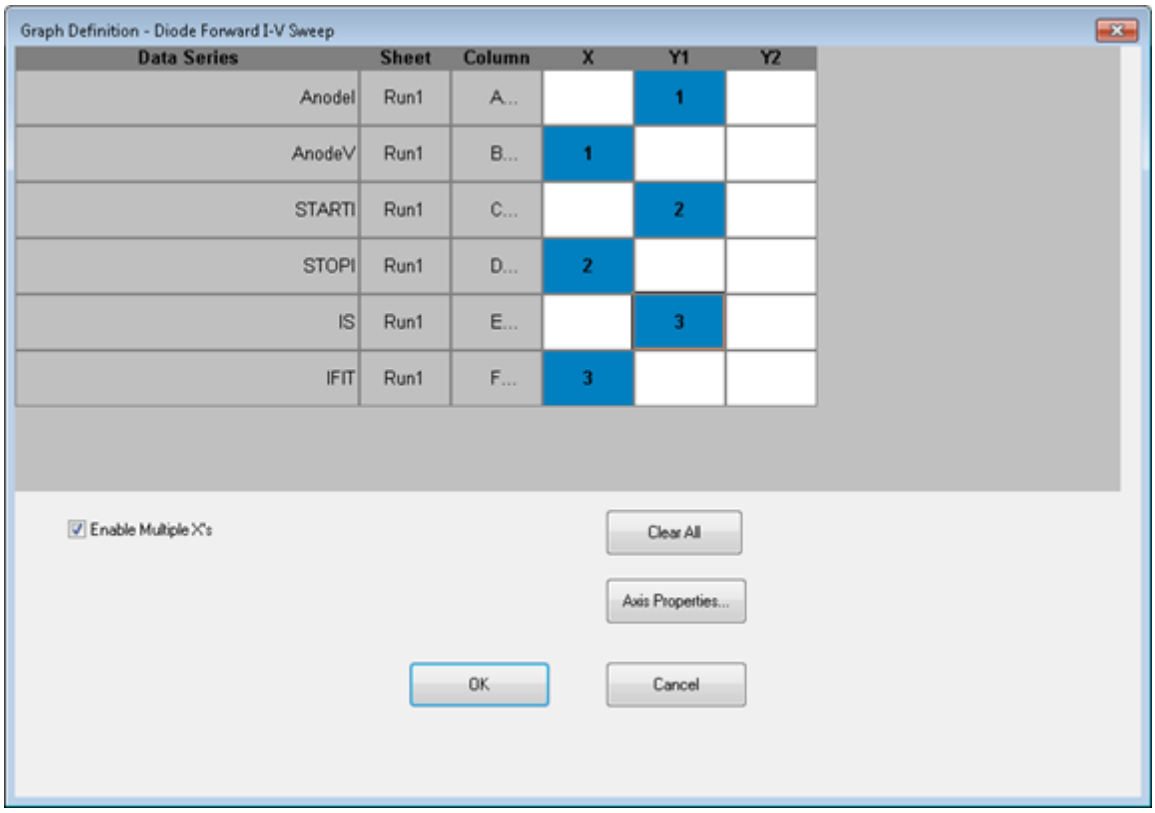
Figure 370: Graph Definition tab for vfd test



3. For each data series, select the axis on which to plot the parameter. The axes are:
   - **X:** X axis.
   - **Y1:** Y axis on the left side of the graph.
   - **Y2:** Y axis on the right side of the graph. The Y2 axis can have a different scale and label than the Y1 axis.

4. If the test does not define a family of curves, you can select **Enable Multiple X's**, as shown in the following figure. If you select multiple X's:
   - Select a Y for each X. The number in the cell indicates the relationships.
   - To change the number, click the cell until the correct number is displayed.

Figure 371: Graph Definition dialog box with multiple X's selected



5. Click **OK**.
6. To clear the settings, select **Clear All**.

View plot coordinates and data series properties

When you select a data point on any graph using the mouse or other pointing device, Clarius displays the following information about the point:

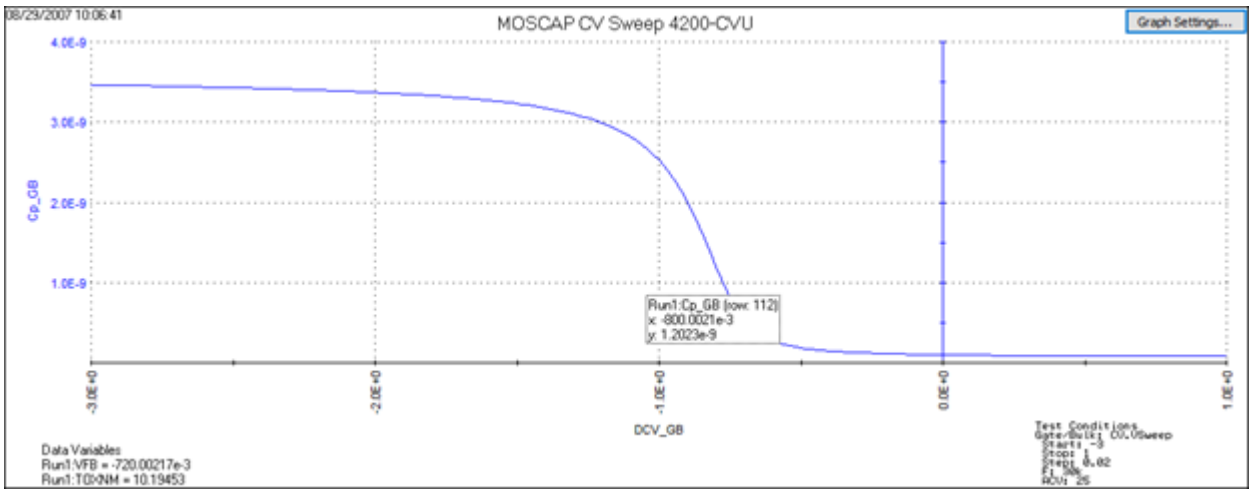
- Data series.
- Run sheet row number.
- Coordinates to four decimal places.

This feature allows you to check information about any point on the graph.

To display the information:

1. Place the default graph cursor over the plot line at approximately the location of the data point.
2. Move the cursor along the plot line until it is over the data point. The cursor changes to the pointer cursor and the coordinates are displayed, as shown in the following figure.

Figure 372: Data point display



3. To display additional information about the data series used for this point, right-click. The Data Series Properties dialog box is displayed. Refer to Change the display of the series data (on page 6-285) for information.

Change the graph settings

You can access the settings for the graph by selecting the **Graph Settings** button. You can also access these settings by right-clicking the graph.

Dual Graph

Select Dual Graph to display two graphs, one with the Y Axis as the left axis and one with the Y1 Axis as the left axis.

Auto Scale

Automatically scales all axes one time.

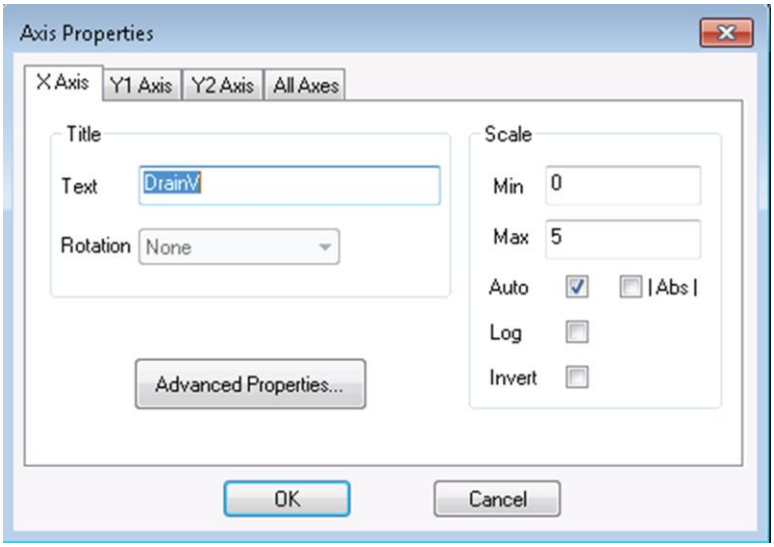
To change how axes are scaled, refer to Define the axis properties (on page 6-260).

Define the axis properties

To change the properties of the graph axes:

1. Select **Analyze**.
2. Select **Graph Settings**.
3. Select **Axis Properties**. The dialog box shown below opens.

Figure 373: Graph Axis Properties dialog box



The tabs of the Axis Properties dialog box are:

- **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 properties that are common to all axes.

X, Y1, and Y2 Axis options

The options for the X Axis, Y1 Axis, and Y2 Axis are described in the following table.

Option	Description
Title Text	The title for the axis. Defaults to the row 1 column heading in the Run tab.
Title Rotation	The direction of the title text. Select the angle from the list.
Scale Min	The minimum scale value for the axis. This value is only applied if Auto is cleared (no autoscaling).
Scale Max	The maximum scale value for the axis. This value is only applied if Auto is cleared (no autoscaling).
Scale Auto	Optimizes the scale of an axis to show all of the data, based on the largest value. Works with both linear and log scales. It may change the way a logarithmic scale is displayed.
Scale |Abs|	Makes all data in the graph display to the positive portion of the graph.
Scale Log	When selected, changes the axis to be shown logarithmically. |Abs| is automatically applied when Log is selected.
Scale Invert	When selected, changes the direction of the data on an axis. For example, if Invert is selected for the X axis, data values decrease from left to right instead of increasing.

Advanced Axis Properties

The Advanced Axis options for the X Axis, Y1 Axis, and Y2 Axis are described in the following table.

Option	Description
Annotation Type	Sets number type used for the axis labels: ■ **Normal:** The labels are in simple decimal notation, such as `30.0`). ■ **Scientific:** The labels are in scientific notation (for example, `3.0E+01` instead of `30.0`). ■ **Engineering:** The axis labels are in engineering notation (for example, `300E-3` instead of `0.30`). ■ **Engineering Symbol:** Displays units with the label, such as `30.0 Volts (V).`
Engineering Symbol	When Engineering is selected, the Annotation is displayed in simple decimal notation with an engineering unit. If you select Auto, the symbol is added automatically. If Auto is cleared, you can select the symbol to be used from a list, such as `30000.0 mV`.
Precision	Specifies the number of decimal points in the labels.
Placement	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 graph. You can select: ■ **Auto:** Clarius determines where the labels are placed. ■ **Origin:** The axis is placed at the origin. This option is intended for a bipolar axis (an axis that has both positive and negative scale values). If an axis is not bipolar, the Origin is the same as Min). ■ **Min:** For X-axes, the axis is placed at the bottom. For Y-axes, the axis is placed to the left of the graph. ■ **Max:** For X-axes, the axis is place at the top. For Y-axes, the axis is placed to the right of the graph.
Rotation	The alignment of the labels of the axis. All angles are specified relative to the X axis. The default rotations place the labels perpendicular to the axes.
Color	Specifies the color of the labels of the axis.
Grid Lines	Specifies if the graph has grid lines at the major tick marks of the axis.
Major	Specifies the spacings between the individual labels on the axis and between the individual tick marks and grid lines, in terms of actual plot units. If Auto is cleared, you can specify the tick spacing manually. For example, if the **X** axis range is 5 V, you could set **0.2** to space the labels and major tick marks 0.2 V apart. 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.
Tick per Major	Specifies the number of ticks to be placed between the major ticks on the axis. If the Auto check box is checked, the Tick per Major combo box is automatically set to 1. Otherwise, you can set the Tick per Major value from 1 to 4.
Auto	When selected, Clarius automatically calculates and implements the major tick spacing for the axis.

Settings for all axes

The All Axes tab includes options that affect all axes. The options are described in the following table.

Option	Description
Run Autoscale	This setting affects any axes that are set to autoscale. Select: ■ **Real-Time:** Autoscaling occurs as data is acquired. This option applies to all raw-data parameters and some calculated parameters. ■ **End of Test:** Autoscaling occurs at completion of the test. If the Auto option is cleared 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 options.
Auto Scale All	Set all axes to autoscale. If any axes were set to manual scale, the Min and Max values are replaced by the values set by Autoscale.
Manual Scale All	Sets all axes to manual scale. The Max and Min settings for all axes are fixed at the values that were last set by autoscale.
Display Font	The font for the text.

Numerically displaying plot coordinates using cursors

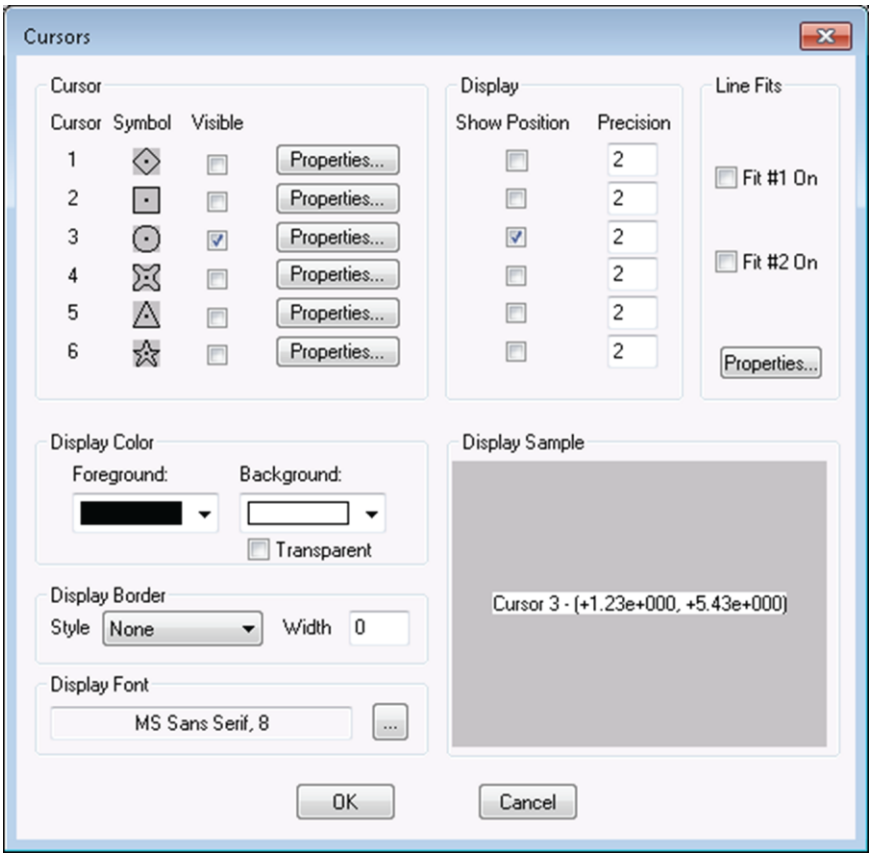
You can display the precise numerical coordinates of data points on a plot using cursors. When you move a cursor, it precisely tracks the plot to which it is attached. Wherever you stop a cursor, a displayed text block indicates the X,Y coordinates of the stopping point.

Configure cursors

To display cursors:

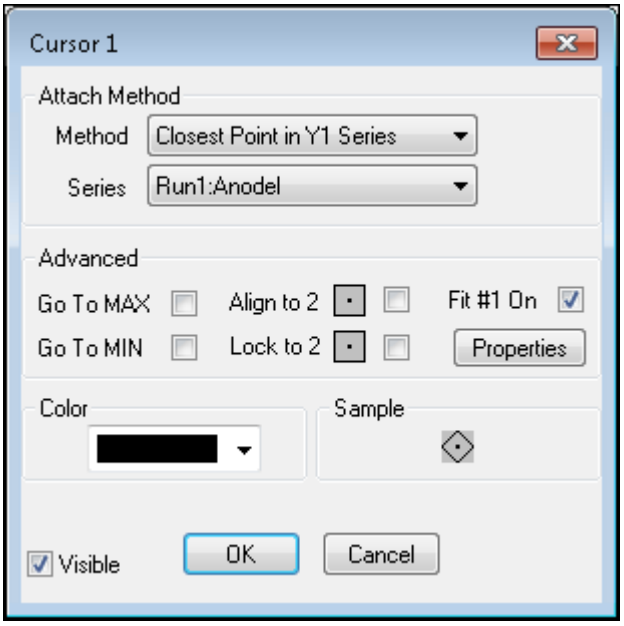
1. Select **Analyze**.
2. On the graph, right-click and select **Cursors**. The dialog box shown below opens.

Figure 374: Graph Cursors dialog box



3. In the Cursor list, select **Visible** for the cursors you want to display. **Show Position** in the Display area is automatically selected for each cursor that has Visible selected.
4. Select **Properties**. The Cursor dialog box for the selected cursor opens, as shown below. The **Sample** area displays the cursor that you are configuring, including the color.

Figure 375: Cursor # dialog box



5. Select the **Attach Method**. You can choose:
   - **Free Floating**
   - **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 for **Series**.
   - **Closest Point in Y2 Series:** Only allows you to attach the selected cursor to the specific Y2 axis plot that is selected for **Series**.
6. If you selected a **Closest Point in Series** option, select **Series** and choose the plot to which you want the attach the cursor.
7. Select the **Color** for the cursor.
8. If you do not want the cursor to display immediately, clear **Visible**. You can restore the cursor later — the cursor keeps its configuration.
9. Click **OK**. If Visible is selected, the cursors and their related text blocks are displayed 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.

Position cursors on the graph

To position cursors:

1. Drag the cursor to its position on the graph.
2. If you selected **Closest Point in Any Series** and the cursor is not on the correct point, drag the cursor from the present point to the correct point until it attaches to the point. On the correct point, drag the cursor to the correct position.

Advanced cursor options

The options in the Advanced area of the Cursor dialog box includes options that place the selected cursor at special locations on the graph.

After the cursor moves to the option you selected, the option is cleared and you can manually position the cursor.

You can select the following options:

NOTE

The Align to <CursorNumber> and Lock to <CursorNumber> check box options are enabled only when both cursors 1 and 2, both cursors 3 and 4, or both cursors 5 and 6 are active.

- **Go To MAX:** Places the cursor at the maximum-Y data point of the plot to which the cursor is attached.
- **Go To MIN:** Places the cursor at the minimum-Y data point of the plot to which the cursor is attached.
- **Align to #:** Aligns the cursor to the same X axis value as the next cursor. The Align To option is disabled if a subsequent cursor is not available (Visible is cleared).
- **Lock to #:** Locks the position of the cursor relative to the position of the next cursor. For example, the next cursor is cursor 3 if the first is 2. The cursor tracks the movement of the next cursor, and the relative X distance between the two cursors remains constant. Note that the next cursor does not track the movement of the previous cursor. The Lock To option is disabled if a subsequent cursor is not available (Visible is cleared).
- **Fit On:** For information on the Fit On options, see Line fits between cursors (on page 6-267).

Line fits between cursors

You can fit lines to test result graphs for one or two line fits between existing cursors. When the line is fitted, the graph displays:

- The fitted line.
- The fit parameters.
- The data point at which a tangent line is fitted to the plot or the starting and ending data points (data range).
- The data-point coordinates. Tangent or starting and ending data points are defined by cursors.

The results on the Graph line fits are similar to the results with the corresponding Formulator functions, as shown in the table below.

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 and Formulator tools each provide specific advantages. For example, Graph fits help you visualize "what if" trials on various data points, while Formulator fit results can be used directly in other calculations.

Line fit examples

The following figures illustrate the linear and regression line fit types.

Figure 376: Linear fit example

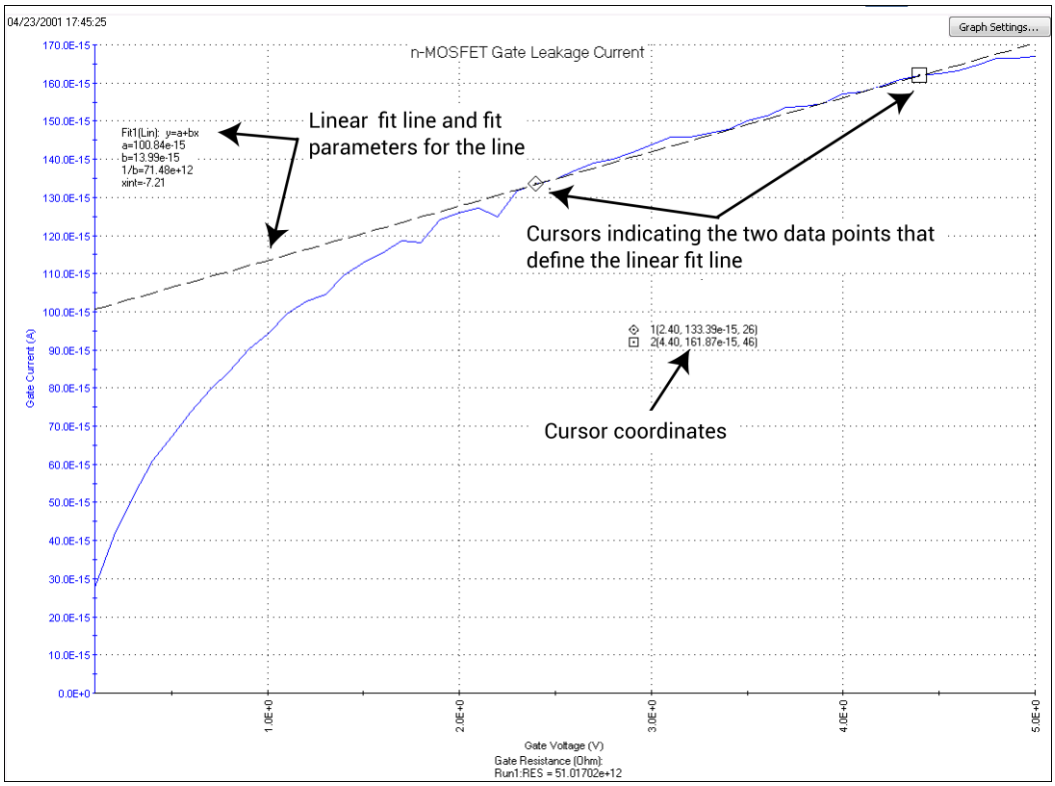
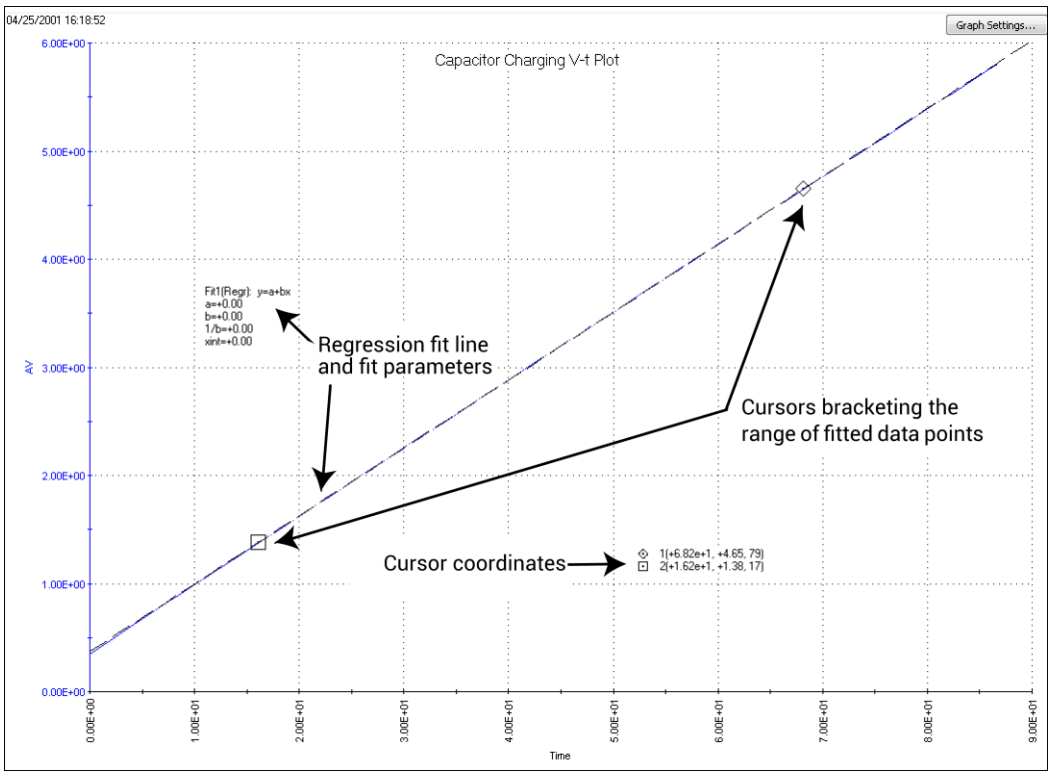


Figure 377: Regression fit example



Perform line fits

Plots of the fit lines appear as dashed lines, and fit parameter and cursor coordinate displays indicate appropriate numerical values.

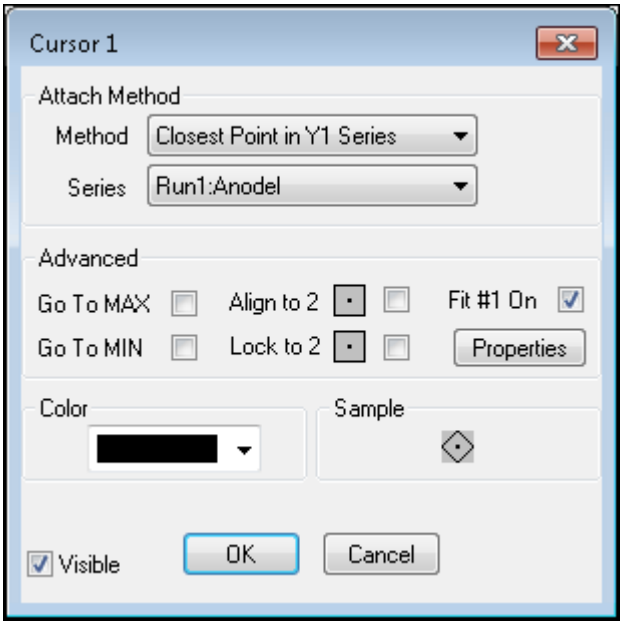
NOTE

Fit #1 is always associated with cursors 1 and 2. Fit #2 is always associated with cursors 3 and 4. Line fits are not available for cursors 5 and 6.

To initiate a line fit:

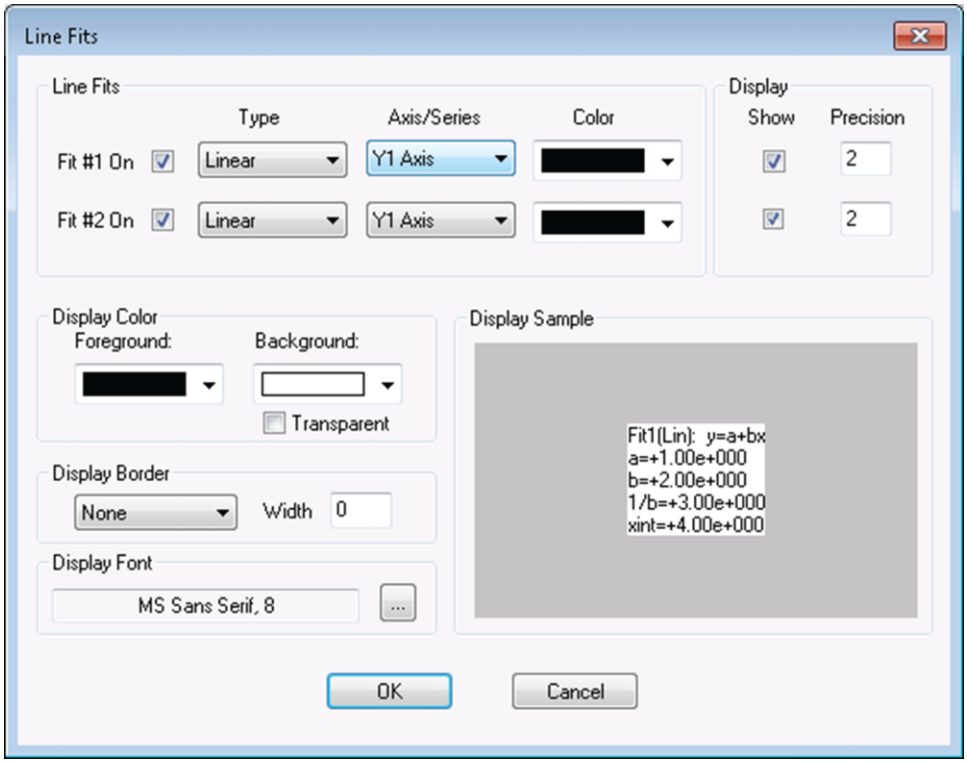
1. Select **Analyze**.
2. On the graph, right-click and select **Cursors**.
3. In the Cursor list, select **Visible** for the cursors you want to display.
4. Select **Properties** next to the cursor. The Cursor dialog box for the selected cursor opens, as shown below.

Figure 378: Cursor # dialog box



5. Select **Fit # On**.
6. Select **Properties** under Fit # On. The Line Fits dialog box is displayed, as shown in the following figure. The options are described in the following table.

Figure 379: Line Fits dialog box



7. Select **OK** when changes are complete. The graph is displayed with the:
   - Line fit cursors at the origin or the Y axis
   - Fit parameters
   - Cursor coordinates
8. Adjust the cursor locations as follows. Refer to Position cursors on the graph (on page 6-266) and Advanced cursor options. (on page 6-266)

NOTE

Positively specify each cursor location. If the initial location for a cursor (for example, the origin) is also the final location, inform Clarius by moving the cursor away from that location and then back again.

9. Drag the fit parameters and cursor coordinates to the locations needed for your project.

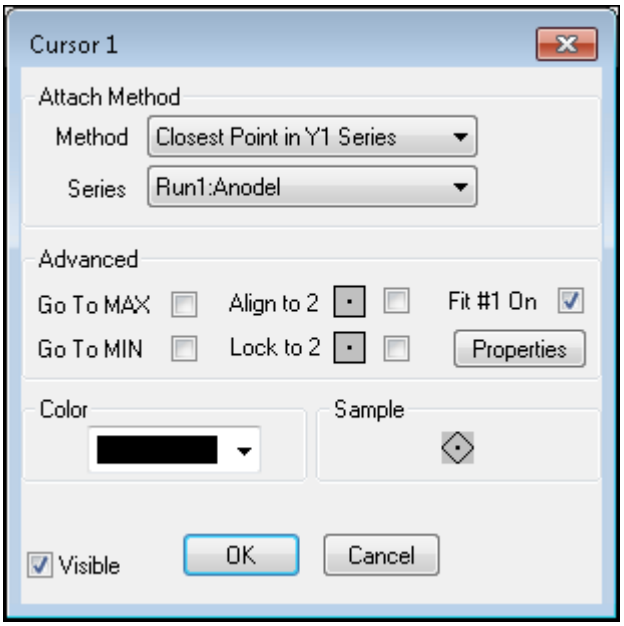
Option	Description
Fit # On	Enable or disable Fit #1 or Fit #2.
Type	The type of line fit to apply: ■ **Linear:** Chord line of the form y = a + bx, drawn between two graphically defined data points. ■ **Regression:** Regression line of the form y = a + bx for a graphically defined range of data points. ■ **Exponential:** Regression line of the form y = a·ebx for a graphically defined range of data points. ■ **Log:** Regression line of the form y = a + b·log10(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.
Axis/Series	The data series for which the fit is to be made. Two cursors will be attached to the specified data curve. If the Type is set to Linear, you can also select a Y axis. This results in the display of free-floating fit cursors that you can position anywhere on the graph. Fit parameters reflect the scale of the selected Y axis.
Color	The color of the fit line.
Display - Show	Select to display the fit parameters. Clear to hide them.
Display - Precision	The precision of the fit line.
Foreground	The color of the text.
Background	The background color.
Transparent	Select to display the box with a transparent background. Clear to make the background solid. Note that selecting Transparent sets the background color selection to light gray.
Display Border	Changes the type of outline around the box. Width must be set to a value other than 0 in order for the border to be displayed.
Width	The width of the display border (0 to 20).
Display Font	The font of the text.

Use existing cursors for line fits

To use existing cursors for line fits:

1. Right-click a cursor. The Cursor dialog box for the selected cursor opens, as shown below.

Figure 380: Cursor # dialog box



2. Select **Fit # On**.
3. Select **Properties** under Fit # On.
4. Refer to Perform line fits (on page 6-269) for information on the options.

View information about the cursor-specified data

When you select a cursor, Clarius displays the following information next to the cursor:

- The cursor number.
- The data series.
- The Run worksheet row number.
- The cursor coordinates.

Interpolate data on the graph

You must select a cursor before the move or interpolation key sequences become active for that cursor.

To add data on the graph:

- Select a cursor, hold the Alt key, and use the arrow keys to find the point for which you want to interpolate data.

NOTE

Note that the highlighted cursor is between points and the label has a * before the data to indicate it is an interpolated value.

- To step between data points as listed in the Run sheet, hold the Ctrl key and use the arrow keys.
- You can select cursors using the Tab key. Press the Tab key to select the next cursor if more than one cursor is displayed.

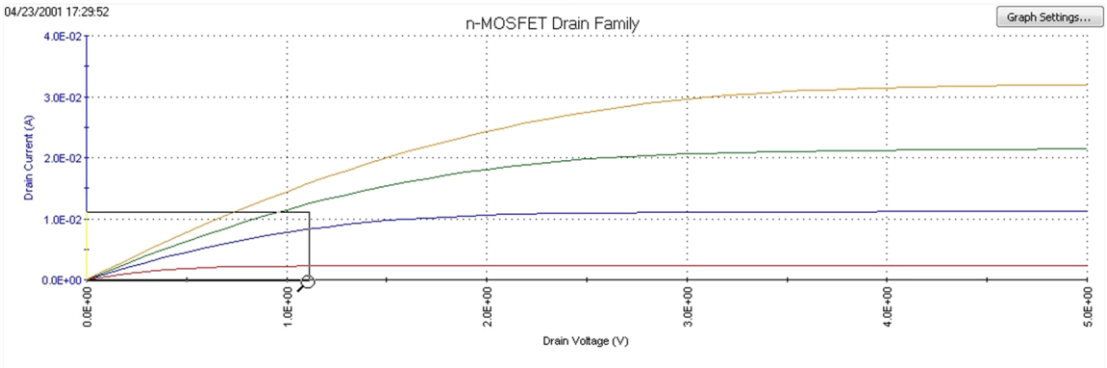
If you hold the Ctrl or Alt and Arrow key for more than a second, the cursor moves more quickly. The cursor moves 1 pixel at a time in normal mode and 5 pixels at a time in fast mode.

Zoom

To enlarge an area of the graph:

1. Click the graph where you want to zoom. A magnifying glass is displayed.
2. Drag the magnifying glass over the data you want to enlarge, as shown in the example here.

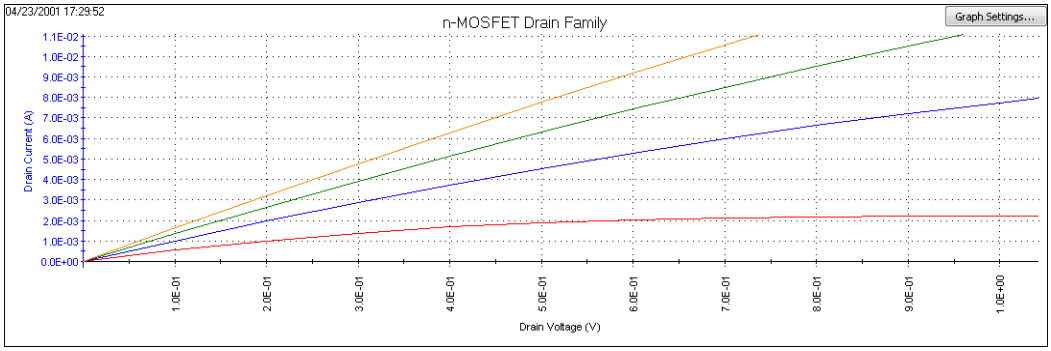
Figure 381: Zoom in on an area of the graph



The axis scales adjust automatically. By zooming in multiple times, you can observe a small portion of the graph.

An example of the zoom area in the graph above is shown here.

Figure 382: Zoom in on an area of the graph - results



To make the graph smaller, right-click the graph and select **Zoom Out**.

NOTE

Zooms are temporary characteristics of the graph and cannot be saved.

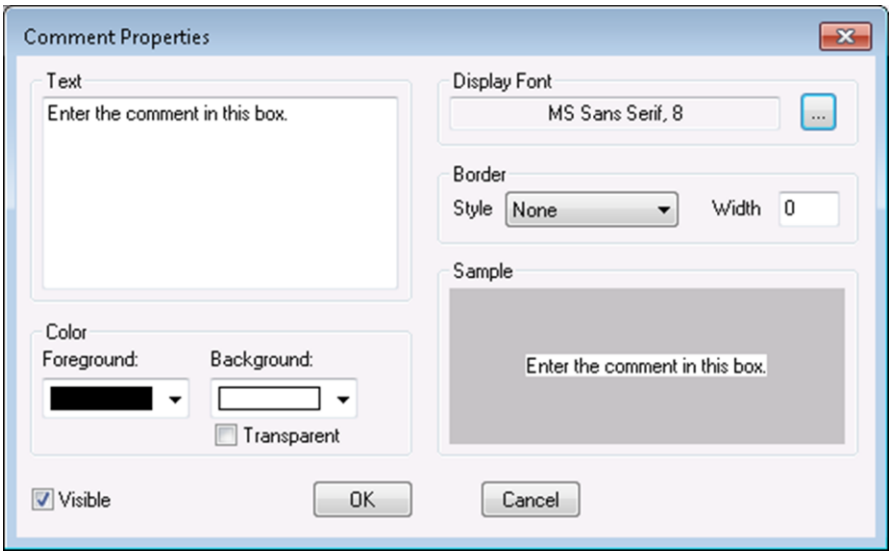
Add a comment

You can add a comment that appears on the graph.

To add a comment:

1. Right-click the graph.
2. Select **Comment**. The Comment Properties dialog box is displayed, as shown here.

Figure 383: Graph Comment Properties dialog box



3. In the Text box, enter the comment.
4. Change the appearance of the comment as needed. See the table below for descriptions of the options.
5. Click **OK**. The comment displays on the graph in the upper left corner.
6. If needed, drag the comment to a new location on the graph.

Option	Description
Text	The comment. Comments can be up to 272 characters long.
Display Font	The font of the text.
Foreground	The color of the text.
Background	The background color.
Transparent	Select to display the box with a transparent background. Clear to make the background solid. Note that selecting Transparent sets the background color selection to light gray.
Border	Changes the type of outline around the box. Width must be set to a value other than 0 in order for the border to be displayed.
Width	The width of the display border (0 to 20).
Visible	Select to display the comment. Clear to hide the comment. The settings are maintained when the comments are hidden.

Display data variables

Opens the Data Variables dialog box, from which you can configure the display of up to four data variables, along with the corresponding names. Data variables are extracted parameters or other values from the heading row of the Run sheet. For example, you can display calculated, single-value extracted parameters, such as curve slopes and saturation values. The Data Variables menu item also toggles the display of the data variables.

If you select multiple data variables, all selected values are displayed together in a single text block, which may be anywhere in the graph.

To display values from the Run tabs on the graph:

1. Select **Analyze**.
2. On the graph, right-click and select **Data Variables 1** or **Data Variables 2**. Both options have the same choices.
3. From the **Sheet:Column** list, select the sheet and column of data you want to display. You can select up to four items. The items you select are displayed in the Data Variables list.
4. Change the appearance of the data as needed. See the table below for descriptions of the options.
5. Click **OK**. The data variables are displayed on the graph.
6. Drag the data variable box to a new location on the graph if needed.

Option	Description
Precision	Sets the precision of the displayed values.
Show Most Recent Data	Select whether to show the most recent data or run history data.
Text	The heading that is displayed for the data.
Display Font	The font of the text.
Foreground	The color of the text.
Background	The background color.
Transparent	Select to display the box with a transparent background. Clear to make the background solid. Note that selecting Transparent sets the background color selection to light gray.
Border	Changes the type of outline around the box. Width must be set to a value other than 0 in order for the border to be displayed.
Width	The width of the display border (0 to 20).
Visible	Select to display the data variables. Clear to hide the data variables. The settings are maintained when the data variables are hidden.

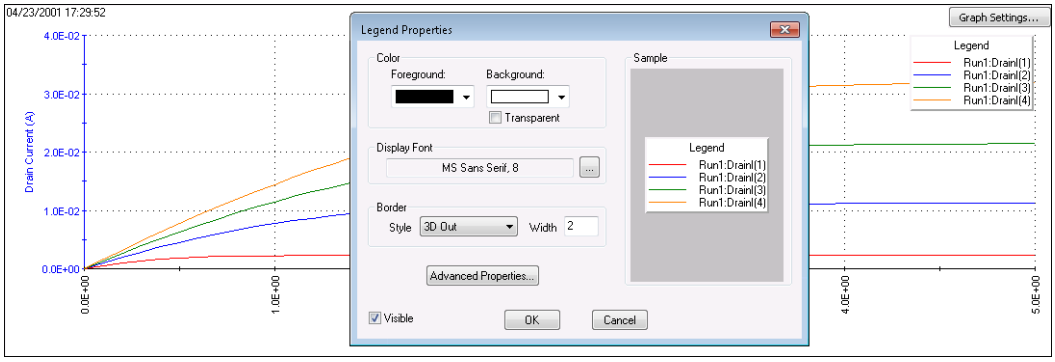
Add a legend

You can display a legend that describes each series of data.

To add a legend:

1. On the graph, right-click and select **Legend**. The legend is displayed in the upper right corner of the graph.
2. Right-click the legend to display the Legend Properties dialog box. An example of a Legend and the dialog box are shown in the figure below.

Figure 384: Graph legend and Legend Properties dialog box



3. Change the appearance of the legend as needed. See the table below for descriptions of the options.
4. Click **OK**.
5. If needed, drag the legend to a new location on the graph.

Option	Description
Foreground	The color of the text.
Background	The background color.
Transparent	Select to display the box with a transparent background. Clear to make the background solid. Note that selecting Transparent sets the background color selection to light gray.
Display Font	The font of the text.
Border	Changes the type of outline around the box. Width must be set to a value other than 0 in order for the border to be displayed.
Width	The width of the display border (0 to 20).
Visible	Select to display the legend. Clear to hide the legend. The settings are maintained when the legend is hidden.
Advanced Properties	This button opens a dialog box that allows you to change the names of the Series in the legend. To change the names, enter the new names in the **Custom Name** column and select **Use Custom Series Names**. Note that if you set a custom name, you cannot return to the original name.

Display test conditions

You can display the primary test conditions that were used to product the data in the graph.

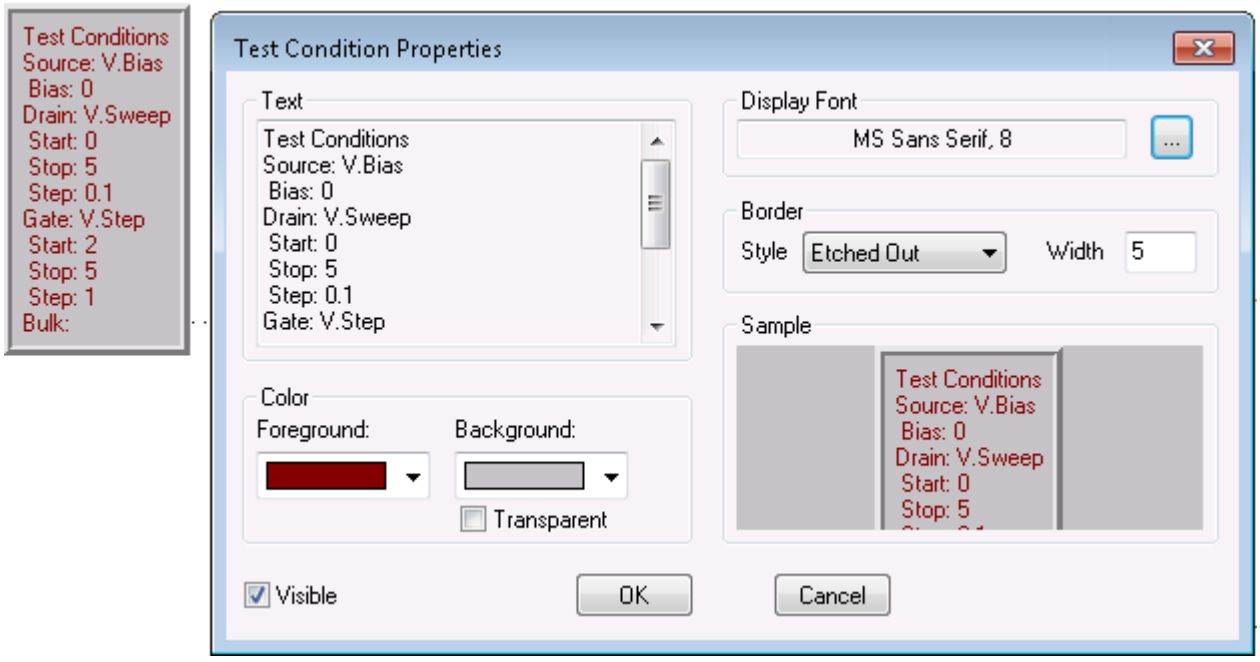
The following table lists the test conditions that are displayed. For each terminal of the DUT, the Test Conditions item displays the name, the applied operation mode, and the corresponding test conditions.

Displayed test conditions	
Operation mode	**Listed test conditions**
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

To display test conditions:

1. On the graph, right-click and select **Test Conditions**. The list is displayed in the upper right corner of the graph.
2. Right-click the test conditions to display the Test Condition Properties dialog box. The conditions and dialog box are shown in the next graphic.

Figure 385: Example of test conditions and test condition properties



3. Change the appearance of the test conditions as needed. See the table below for descriptions of the options.
4. Click **OK**.
5. If needed, drag the test conditions to a new location on the graph.

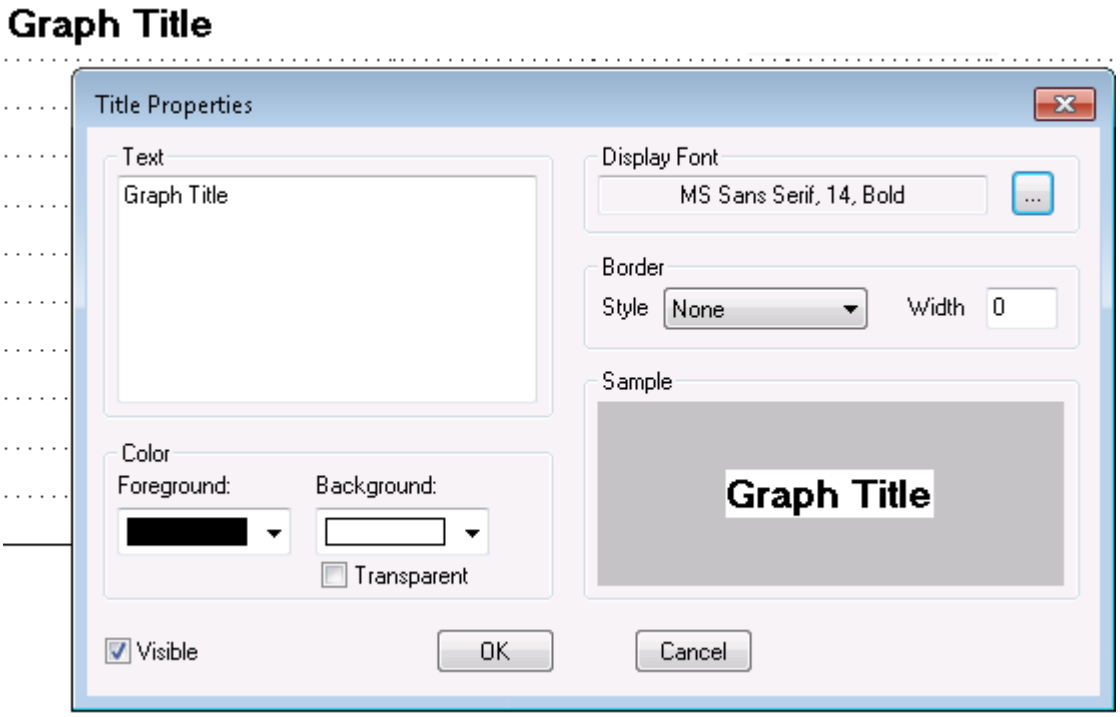
Option	Description
Text	The text that will be displayed. You cannot change this text.
Foreground	The color of the text.
Background	The background color.
Transparent	Select to display the box with a transparent background. Clear to make the background solid. Note that selecting Transparent sets the background color selection to light gray.
Display Font	The font of the text.
Border Style	Changes the type of outline around the box. Width must be set to a value other than 0 in order for the border to be displayed.
Border Width	The width of the display border (0 to 20).
Visible	Select to display the test conditions. Clear to hide the test conditions. The settings are maintained when the test conditions are hidden.

Add or update a title

To add or update the title of the graph:

1. On the graph, right-click and select **Title**. The title is displayed at the top of the graph and the Title Properties dialog box is displayed.

Figure 386: Example of graph title and properties



2. Change the appearance of the title as needed. See the table below for descriptions of the options.
3. Click **OK**.
4. If needed, drag the title to a new location on the graph.

Option	Description
Title	The name of the graph.
Foreground	The color of the text.
Background	The background color.
Transparent	Select to display the title with a transparent background. Clear to make the background solid. Note that selecting Transparent sets the background color selection to light gray.
Display Font	The font of the text.
Border Style	Changes the type of outline around the box. Width must be set to a value other than 0 in order for the border to be displayed.
Border Width	The width of the display border (0 to 20).
Visible	Select to display the title. Clear to hide the title. The settings are maintained when the title is hidden.

Change the graph colors

You can change the colors of graph foreground (the plot area) and background (outside the plot area) and determine if the time and date are displayed.

You can also select Monochrome, which changes all options on the graph, including plot lines, titles, and axes, to black and white. You cannot revert to your previous settings after selecting Monochrome.

To change the colors:

1. On the graph, right-click and select **Graph Properties > Graph Area**. The Graph Area dialog box is displayed.
2. To change the color of the plot area, select a color from the **Foreground** list.
3. To change the color of the background, select a color from the **Background** list.
4. To remove the time and date display from the graph, select **Remove Time/Date.**
5. Click **OK**.

Change the display of the series data

You can define the line pattern, shape, color, and width for each series of data on the graph.

To define the data properties:

1. Select **Analyze**.
2. Select **Graph Settings**.
3. Select **Graph Properties**.
4. Select **Series**. The dialog box shown below opens.

Figure 387: Data Series Properties dialog box

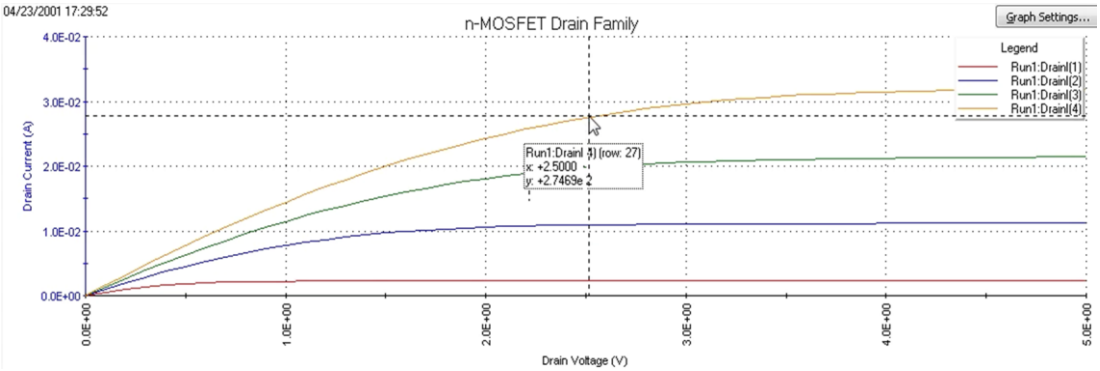
The options for the series are described in the following table.

Option	Description
Series	Select the series of data to which the settings apply.
Pattern	Select the line pattern for the plot line.
Shape	Select the shape that is used for the data points.
Color	Select a color for the plot line.
Width	Select the width of the plot line.

Identify plot coordinates with crosshairs

You can display crosshairs that can be positioned anywhere on the graph. An example is shown in the following figure. The x and y values are displayed when the crosshairs are on a data point.

Figure 388: Graph with crosshairs



To display crosshairs, right-click the graph and select **Crosshair**. Select **Crosshair** again to remove them.

Synchronize graphs for one test executed at multiple sites

Ideally, data from a single test that is gathered at multiple, identically fabricated sites should be plotted on multiple, identically configured graphs. For a single test in the project, you can use the 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 test at the first five sites of a project, Synchronize Graphs identically configures graphs for the Site 1, 2 , 3, 4, and 5 Run sheets.

NOTE

If the project contains multiple instances of a same-named test, you must apply the feature separately each such instance. For example, if the project tree shows both vds-id#1 and vds-id#2 tests, you must apply Synchronize Graphs separately for vds-id#1 and vds-id#2.

CAUTION

The graphs for the selected test 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 each site-specific graph.

To synchronize multi-site graphs:

1. From the project tree, select the site for which you want to configure a master graph.
2. In the project tree, select the test for which the data is to be graphed.
3. Select **Analyze** for the test.
4. On the graph, select **Graph Settings**.
5. Select **Synchronize Graphs.** A caution message appears.
6. 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.

Changing the position of a graph

To reposition the Analyze graph:

1. Select **Analyze**.
2. Right-click the graph.
3. To move the graph, select **Move**. The cursor changes to crossed arrows.
4. Drag the map to the new location.
5. When the location is correct, right-click the graph and select **Move** to turn off the move function.

NOTE

The change in position of the graph is saved with the project.

Change the size of the graph

You can increase or decrease the size of a graph and save it as a property of the graph.

To set the size of the graph and save it:

1. Select **Analyze**.
2. On the graph, select **Graph Settings**.
3. Select **Resize**. The cursor changes to a ruler.
4. Drag the ruler to resize the graph.
5. Select **Save** to save the new graph size.

NOTE

A resized graph remains centered on the **Graph** tab.

Reset graph properties

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.

To reset the graph:

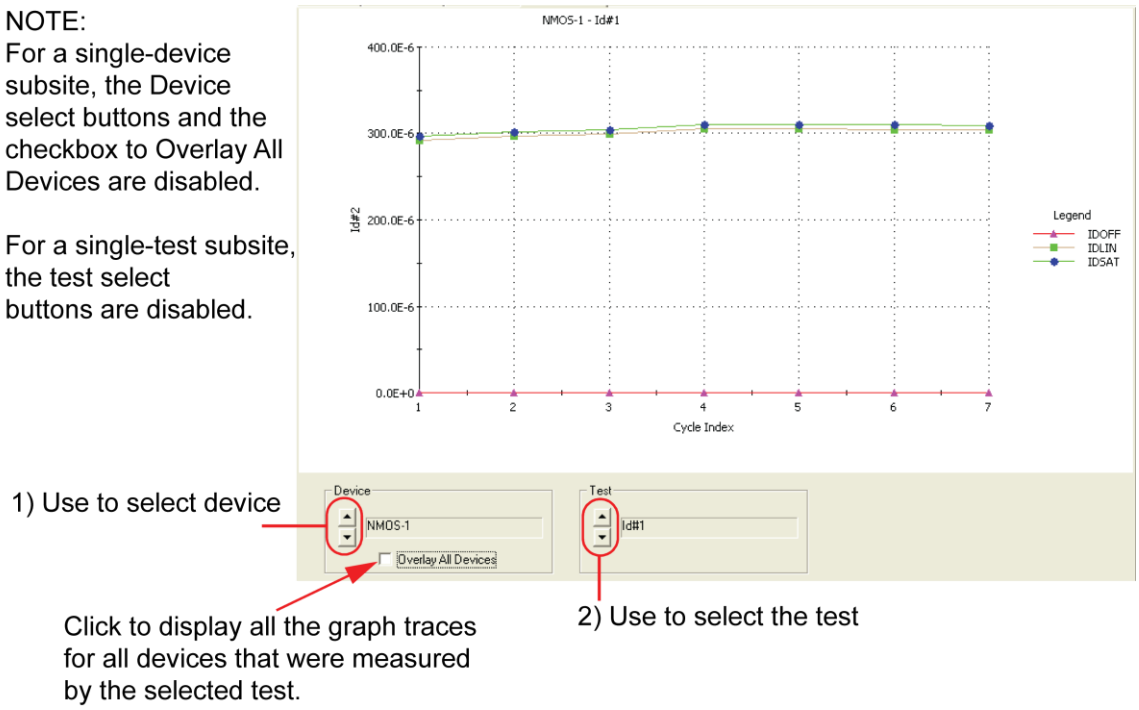
1. On the graph, right-click and select **Reset**.

Cycle mode graphs

The graphs for the Cycle Mode plot output values versus the cycle index. Each data point in the graph represents an output value reading for each subsite cycle. The following figure explains how to display the various graphs.

This figure shows the graph traces for test ID#1 for the NMOS-1 device. The three traces are for the output values IDOFF, IDLIN and IDSAT.

Figure 389: Subsite Analyze graph: Cycle mode



User library descriptions

Keithley Instruments provides several user libraries of user modules. The following topics provide an overview of each of the user libraries.

The KULT user libraries and user modules that are provided with Clarius+ are available in the directory:

```
C:\s4200\kiuser\usrlib\
```

The Formulator

The Formulator allows you to make data calculations on test data and on the results of other Formulator calculations. The Formulator provides a variety of computational functions, common mathematical operators, and common constants. Some of these may be used for real-time, in-test calculations for test data. Others can be used only for post-test data computations. Clarius automatically inserts the Formulator calculation results into the Run sheet, in addition to the raw data.

A formula created by the Formulator is an equation that is made from a series of functions, operators, constants, and arguments.

A formula created using the Formulator calculates any combination of the following:

- Test data.
- Secondary data created by other Formulator formulas.
- Standard constants from the list of constants.

Formulator functions may be limited to specific sets of data. For example:

- Some of the functions operate on only on Run tab columns of values (vectors) only.
- Some operate on single values (scalars) only.
- Some operate on both single values (scalars) and columns of values (vectors).

The results of some calculations may be a column of values (vector) in the Analyze Run sheet or a column that contains only a single value (scalar).

Configure Formulator calculations

The Formulator allows you to do simple in-test calculations on test data and complex post-test data calculations. You can use the following operators and functions for in-test, real-time calculations on test data:

- Operators: +, -, *, /, ^
- General functions: ABS, SQRT, EXP, LOG, LN, DELTA

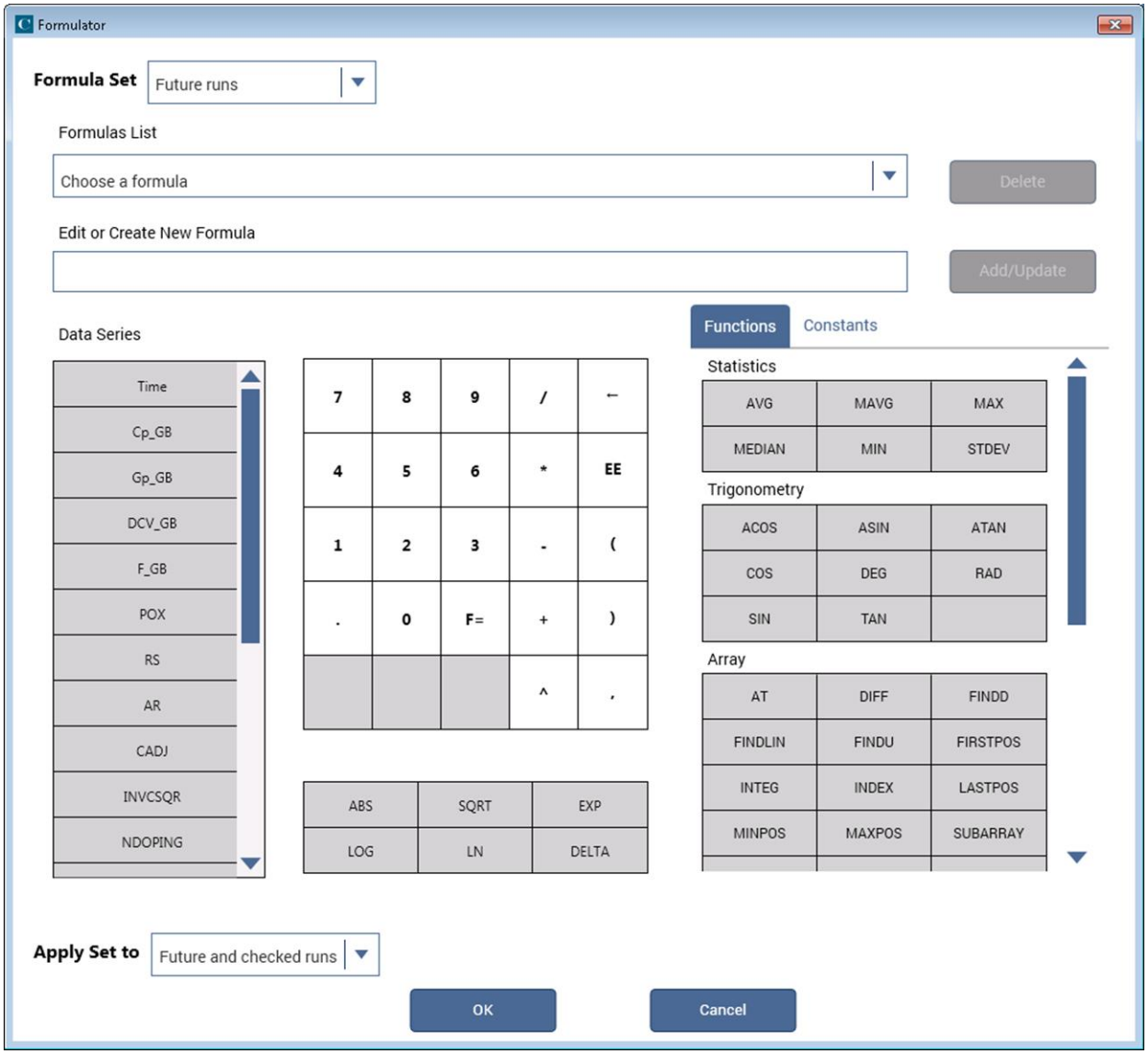
A variety of other functions may be used for post-test calculations. Refer to Formulator function reference (on page 6-296).

Open the Formulator

To open the Formulator:

1. In the project tree, select a test.
2. Select **Configure**.
3. In the Test Settings pane, select **Formulator**. The Formulator dialog box opens, as shown below.

Figure 390: Formulator with no entries



Becoming familiar with the Formulator dialog box

This section summarizes how you can use each Formulator feature.

Formula area

The top area of the Formulator dialog box allows you to manage formulas.

The **Formula Set** box selects whether you want to apply the formula to the Run History that is selected in the Analyze pane or to future runs.

If formulas exist, use the **Formulas List** to open a formula. When a formula is selected, the formula is displayed in the Edit or Create New Formula box.

Use the **Edit or Create New Formula** box to view, edit, or create formulas.

After adding or editing a formula, select **Add/Update** to add the calculation to the Data Series list and the Formulas List. To clear a formula that you do not want to add or update, use the backspace or delete key on the keyboard.

The **Delete** button deletes the formula that is selected in the Formulas List and removes it from the Data Series list.

Data Series

Lists the names of all columns in the Run tab of the Analyze sheet. When you select a data series, the data series is added to the Edit or Create New Formula box.

When you add a formula, it is added to this list and to the Analyze sheet.

Number pad

The number pad displays number keys, mathematical operators, and **F=**. When you select an option from the number pad, the option you selected is added to the Edit or Create a New Formula box at the cursor position.

You can use **F=** in place of a variable name to complete an equation. When you add an equation that uses **F=**, Clarius adds a numeric suffix to the **F**, for example, **F1**, **F2**, **F3**. This is the heading that is used in the Analyze Run sheet for the formula.

For details about the available functions, refer to Using the Formulator functions (on page 6-294).

Functions

You can select functions to include in your formula from the Functions tab to the right of the number pad and the table below the number pad. In the Function tab, use the scroll bar to view all options.

For descriptions of each of the functions, refer to Formulator function reference (on page 6-296).

Constants

The Constants tab provides constants that you can use in the formula. Click the symbol of the constant to add it to the formula.

The definitions of the default constants are:

- **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's constant
- **C** Speed of light
- **KTQ** Thermal voltage

You can edit the values and units of constants in the constants list. Place your cursor in the cell to edit and make changes as needed. Changes are automatically saved for all tests.

To add a new constant to the constants list, click **Add**. Enter the name, value, and unit for the new constant.

To delete a constant, select **Delete**. A list of constants is displayed that you can select from.

Apply Set to

This option determines which Analyze sheets this set of formulas apply to.

- **Future runs:** Only apply this set of formulas to future test runs.
- **Checked runs:** Only apply this set of formulas to test runs that are selected in the Run History pane of the Analyze sheet.
- **Future and checked runs:** Apply this set of formulas to all runs.

Using the Formulator options

You can use the Formulator functions, operators, and constants in combination to create simple or complex analysis equations.

You can nest multiple functions. For example, in one equation you can:

- Calculate a series of moving averages for a column of data (vector) in the Analyze sheet, 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 other functions available in the Formulator are described in the following topics.

Row 1 of an Analyze sheet 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` as the argument for the first value in a vector:

```
[format: FIRSTPOS(DataWorksheetColumn)]
```

Similarly, use the function `LASTPOS` for the last value in the vector:

```
[format: LASTPOS(DataWorksheetColumn)]
```

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`

The fit lines and parameters only display in the graphs. They are not available for use in calculations.

Correspondence between Graph tab and Formulator line fits

Formulator fit functions*	**Corresponding Graph tab line fit**

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.				

Real-time functions, operators, and formulas

A formula that contains only real-time operators and functions is a real-time formula. If a real-time formula is specified as part of a test, it executes for each data point generated by the test immediately after it is generated. The results of a real-time formula may be viewed in the Analyze sheet or plotted during the test in the same way as test data.

The following operators and functions are real-time operators and functions:

- Operators: `+`, `-`, `*`, `/`, `EE`, `^` (exponentiation)
- Functions: `ABS`, `SQRT`, `EXP`, `LOG`, `LN`, `DELTA`, `DIFF`, `INTEG`

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 test is run, the formula executes automatically during each run.
- If a real-time formula is created after a test has been run, the formula executes initially upon adding it to the test and automatically during each subsequent run.

Post-test-only functions and formulas

Some Formulator functions are post-test-only. Post-test-only functions execute only at the end of each run of the test in which the formula is defined. The results of a post-test-only formula may be viewed in the Analyze Run sheet or plotted at the end of a test.

The post-test-only functions are listed in the following table.

AT (on page 6-307)
AVG (on page 6-300)
COND (on page 6-300)
EXPFIT (on page 6-316)
EXPFITA (on page 6-317)
EXPFITB (on page 6-318)
FINDD (on page 6-308)
FINDLIN (on page 6-309)
FINDU (on page 6-310)
FIRSTPOS (on page 6-310)
LASTPOS (on page 6-313)
LINFIT (on page 6-319)
LINFITSLP (on page 6-320)
LINFITXINT (on page 6-321)
LINFITYINT (on page 6-322)
LOGFIT (on page 6-323)
LOGFITA (on page 6-324)
LOGFITB (on page 6-325)
MAVG (on page 6-301)
MAX (on page 6-301)
MAXPOS (on page 6-313)
MIN (on page 6-302)
MINPOS (on page 6-314)
REGFIT (on page 6-329)
REGFITSLP (on page 6-330)
REGFITXINT (on page 6-331)
REGFITYINT (on page 6-332)
SUBARRAY (on page 6-314)
SUMMV (on page 6-315)
TANFIT (on page 6-333)
TANFITSLP (on page 6-334)
TANFITXINT (on page 6-335)
TANFITYINT (on page 6-336)

For example, 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 test has been run, the formula executes automatically at the conclusion of each run.
- If a post-test-only formula is created after a test has been run, the formula executes initially upon adding it to the test and automatically at the conclusion of each subsequent run.

Formulator function reference

Each of the 4200A-SCS Formulator functions is described in the following.

ABS Formulator function

Calculates the absolute value of each value in the designated column (vector) or the absolute value of any operand.

Usage

```
ABS(Value)
```

Value	The name of any column (vector) in the Data Series list or any operand

Details

You can use this function to do calculations in real time (while a test is executing).

Example

`F2 = ABS(GateI)`	Returns the absolute value of the gate current.

Also see

None

SQRT Formulator function

Returns the square root of each value in a designated column (vector) or the square root of any operand.

Usage

```
SQRT(Value)
```

Value	The name of any column (vector) in the Data Series list or any operand

Details

A negative value of `X` returns `#REF` in the Run worksheet.

You can use this function to do calculations in real time (while a test is executing).

Example

`TWO = SQRT(4)`	

Also see

None

EXP Formulator function

Returns the exponential, e^{value}, for each value in a column (vector) or for any operand.

Usage

```
EXP(Value)
```

Value	The name of any column (vector) in the Data Series list or any operand

Details

You can use this function to do calculations in real time (while a test is executing).

Example

```
NEWCURRENT = CURRENT*EXP(ANODEV)
```

Also see

LN Formulator function (on page 6-299)

LOG Formulator function

Returns the base-10 log of each value in a designated column (vector) or the base-10 log of any operand.

Usage

```
LOG(Value)
```

Value	The name of any column (vector) in the Data Series list or any operand

Details

You can use this function to do calculations in real time (while a test is executing).

Example

```
F1 = LOG(DRAINI)
```

Also see

None

LN Formulator function

This command returns the base-e (natural, Napierian) log of each value in a designated column (vector) or the Napierian log of any operand.

Usage

`LN(Value)`

Value	The name of any column (vector) in the Data Series list or any operand

Details

You can use this function to do calculations in real time (while a test is executing).

Example

```
DIODEV = LN(ANODEI)*0.026
```

Also see

EXP Formulator function (on page 6-298)

DELTA Formulator function

This command 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.

Usage

`DELTA(Value)`

Value	The name of any column (vector) in the Data Series list

Details

You can use this function to do calculations in real time (while a test is executing).

Example

```
GM = DELTA(DRAINI)/DELTA(GATEV)
```

Also see

None

COND Formulator function

Returns one of two user-defined expressions, depending on the comparison of two other user-defined expressions.

Usage

```
COND(EXP1, EXP2, EXP3, EXP4)
```

EXP1, *EXP2*, *EXP3*, *EXP4*	Mathematical expressions created using valid Formulator functions, operators, and operands

Details

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.

Example

```
CLIPCURRENT = COND(DRAINI, 1E-6, DRAINI, 1E-6)
```

Also see

None

Statistics

The following Formulator functions provide statistics operations.

AVG Formulator function

Returns the average of all values in the column (vector).

Usage

```
AVG(Value)
```

Value	The name of any column (vector) in the Data Series list

Example

```
LEAKAGE = AVG(GATEI)
```

Also see

MAVG (on page 6-301)

MAVG Formulator function

Returns a new column (vector) consisting of the moving averages of successive groups of data points from another column (vector).

Usage

```
MAVG(V, N)
```

V	The name of any column (vector) in the Data Series list or any operand
N	The number of data points to be averaged in each group

Details

You can configure the number of data points in a group.

If *N* = 3 and *V* contains the 12 values X1, X2, X3, X4, X5, . . . , X10, X11, X12, then MAVG returns a column (vector) that contains 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.

Example

```
FILTER = MAVG(GATEI, 3)
```

Also see

None

MAX Formulator function

Searches all values in a column (vector) and returns the maximum value.

Usage

```
MAX(Value)
```

Value	The name of any column (vector) in the Data Series list

Example

```
MAXGM = MAX(DIFF(DRAINI, GATEV))
```

Also see

None

MEDIAN Formulator function

Searches all values in a column (vector), finds the middle point of that column used, and returns the value.

Usage

```
MEDIAN(Value)
```

Value	The name of any column (vector) in the Data Series list

Also see

None

MIN Formulator function

Searches all values in a column (vector) and returns the minimum value.

Usage

```
MIN(Value)
```

Value	The name of any column (vector) in the Data Series list

Example

```
SMALLESTI = MIN(DRAINI)
```

Also see

None

STDEV Formulator function

Returns the standard deviation of all values in the column (vector).

Usage

```
STDEV(Value)
```

Value	The name of any column (vector) in the Data Series list or any operand

Details

Returns the standard deviation.

Example

```
LEAKAGE = STDEV(GATEI)
```

Also see

None

Trigonometry

The following Formulator functions provide trigonometric operations.

ACOS Formulator function

Returns the arc cosine of each value in a designated column (vector) under Columns or any operand.

Usage

```
ACOS(Value)
```

Value	The name of any column (vector) in the Data Series list or any operand

Details

Returns the value in radians.

Example

```
F1 = ACOS(DRAINI)
```

Also see

None

ASIN Formulator function

Returns the arc sine of each value in a designated column (vector) under **Columns** or any operand.

Usage

```
ASIN(Value)
```

Value	The name of any column (vector) in the Data Series list or any operand

Details

Returns the value in radians.

Example

```
F1 = ASIN(DRAINI)
```

Also see

None

ATAN Formulator function

Returns the arc tangent of each value in a designated column (vector) under **Columns** or any operand.

Usage

```
ATAN(Value)
```

Value	The name of any column (vector) in the Data Series list or any operand

Details

Returns the value in radians.

Example

```
F1 = ATAN(DRAINI)
```

Also see

None

COS Formulator function

Returns the cosine of each value operand.

Usage

```
COS(Value)
```

Value	The name of any column (vector) in the Data Series list or any operand

Details

Returns the value in radians.

Example

```
F1 = COS(DRAINI)
```

Also see

None

DEG Formulator function

The DEG function converts an angle value in radians to degrees.

Usage

`DEG(Value)`

Value	The name of any column (vector) in the Data Series list or any operand

Details

Returns the value in degrees.

Example

```
F1 = DEG(ANGLE)
```

Also see

None

RAD Formulator function

The RAD function converts an angle value in degrees to radians.

Usage

`RAD(Value)`

Value	The name of any column (vector) in the Data Series list or any operand

Details

Returns the value in radians.

Example

```
F1 = RAD(ANGLE)
```

Also see

None

SIN Formulator function

Returns the sine of each value in a designated column (vector) under **Columns** or any operand.

Usage

```
SIN(Value)
```

Value	The name of any column (vector) in the Data Series list or any operand

Details

Returns the value in radians.

Example

```
F1 = SIN(DRAINI)
```

Also see

None

TAN Formulator function

Returns the tangent of each value in a designated column (vector) under **Columns** or any operand.

Usage

```
TAN(Value)
```

Value	The name of any column (vector) in the Data Series list or any operand

Details

Returns the value in radians.

Example

```
F1 = TAN(DRAINI)
```

Also see

None

Array

The following Formulator functions are used to work with arrays.

AT Formulator function

Extracts and returns a single value from a column (vector).

Usage

`AT(Value, POS)`

Value	The name of any column (vector) in the Data Series list or any operand
POS	The row number of column *Value* where the single value is located

Example

```
IDSAT = AT(DRAINI, 36)
```

Also see

None

DIFF Formulator function

For all of the values in two selected columns (vectors), returns a third column (vector) that contains the difference coefficients.

Usage

`DIFF(V1, V2)`

V1	The name of any column (vector) listed under **Columns**
V2	The name of any column (vector) listed under **Columns**

Details

Each coefficient is calculated as follows:

V1/*V2*

Where:

- *V1* = The difference between a pair of adjacent values in the first column.
- *V2* = The difference between the corresponding values in the second column.

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.

You can use this function to do calculations in real time (while a test is executing).

Example

```
GM = DIFF(DRAINI, GATEV)
```

Also see

None

FINDD Formulator function

The find down function searches down the column until it finds a value that matches the user-specified value *X*. FINDD searches a column (vector) *V*, beginning at *START*. Then it returns the row number (index) of that value.

Usage

```
FINDD(V, X, START)
```

V	The name of any column (vector) listed under Columns
X	Any value, which may be the result of another calculations
START	The row number (index) of the starting value for the search

Details

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*.

Example

```
IF = AT(ANODEI, FINDD(ANODEV, 0.7, FIRSTPOS(ANODEV)))
```

Also see

FINDLIN (on page 6-309)
FINDU (on page 6-310)

FINDLIN Formulator function

Find using linear interpolation searches down the column until it finds a value that is closest (but does not exceed) the user-specified value *X*. FINDLIN searches a column (vector) *V*, beginning at *START*. 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.

Usage

```
FINDD(V, X, START)
```

V	The name of any column (vector) listed under Columns
X	Any value, which may be the result of another calculation
START	The row number (index) of the starting value for the search

Example 1

Assume you want to use FINDLIN to locate value 6 in the following array:

```
(Index 1)0
(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.

Example 2

```
IF = AT(ANODEI, FINDLIN(ANODEV, 0.7, FIRSTPOS(ANODEV)))
```

Also see

FINDD (on page 6-308)
FINDU (on page 6-310)

FINDU Formulator function

Find up 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. FINDU searches a column (vector) *V*, beginning at *START*. Then it returns the row number (index) of that value.

Usage

```
FINDU(V, X, STARTPOS)
```

V	The name of any column (vector) listed under Columns
X	Any value, which may be the result of another calculation
STARTPOS	The row number (index) of the starting value for the search

Details

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*.

Example

```
IF = AT(ANODEI, FINDU(ANODEV, 0.7, LASTPOS(ANODEV)))
```

Also see

FINDD (on page 6-308)
FINDLIN (on page 6-309)

FIRSTPOS Formulator function

Returns the row number (index) of the first value in a column (vector), typically the number 2.

Usage

```
FIRSTPOS(V)
```

V	The name of any column (vector) listed under **Columns**

Example

```
STARTOFARRAY = FIRSTPOS(DRAINI)
```

Also see

LASTPOS (on page 6-313)

INDEX Formulator function

Returns a specified number of points starting with a specified value and consecutive values incremented by one.

Usage

```
INDEX(START, N)
```

START	The starting value
N	The number of data points to be included

Example

`INDEX20 = INDEX(5, 20)`	Produces a new column labeled `INDEX20` that contains 20 values, starting with the value of 5 and ending with a value of 24.

Also see

None

INTEG Formulator function

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.

Usage

```
INTEG(VX, VY)
```

VX	The name of any column (vector) listed under **Columns**
VY	The name of any column (vector) listed under **Columns**

Details

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, `INTEG` returns a column (vector) containing `A1` equal to 0 (zero area at the start of a curve, at `X1`) and `A2`, `A3`, `A4`, and `A5` equal to curve areas, as shown in the following graphics.

Figure 391: INTEG Formulator function

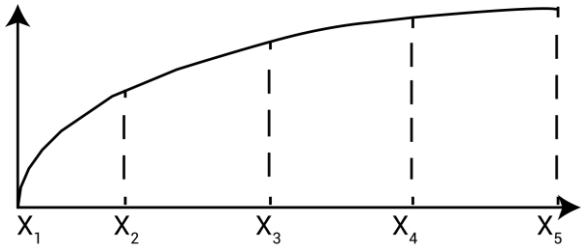


Figure 392: A5 curve area

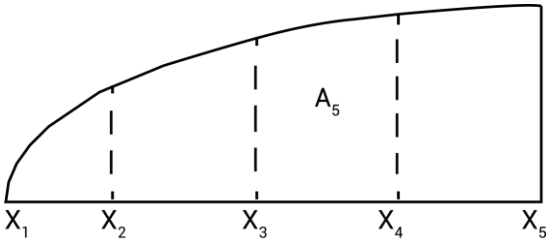


Figure 393: A2, A3, and A4 curve areas

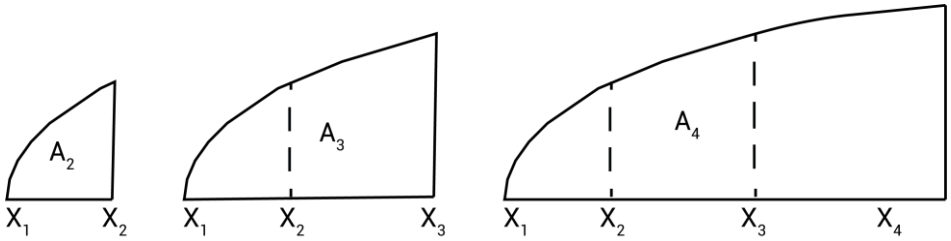
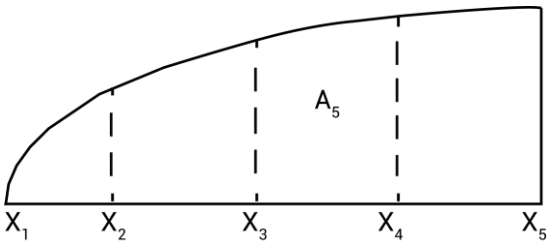


Figure 394: A5 curve area



You can use this function to do calculations in real time (while a test is executing).

Example

```
QBD = INTEG(TIME, GATEI)
```

Also see

None

LASTPOS Formulator function

Returns the row number (index) of the last value in a column (vector).

Usage

`LASTPOS(`*`Value`*`)`

Value	The name of any column (vector) listed under **Columns**

Example

```
NUMSWEEPPTS = LASTPOS(COLLECTORI)
```

Also see

FIRSTPOS (on page 6-310)

MAXPOS Formulator function

Searches all values in a column (vector), finds the maximum value, and returns the row number (index) of the maximum value.

Usage

`MAXPOS(`*`V`*`)`

V	The name of any column (vector) listed under Columns

Example

```
PEAKSTRESS = AT(GATEV, MAXPOS(SUBSTRATEI))
```

Also see

None

MINPOS Formulator function

Searches all values in a column (vector), finds the minimum value, and returns the row number (index) of the minimum value.

Usage

```
MINPOS(V)
```

V	The name of any column (vector) listed under Columns

Example

```
LOCATION = MINPOS(DRAINI)
```

Also see

None

SUBARRAY Formulator function

Returns a new column (vector) containing a specified range of the values from an existing column (vector).

Usage

```
SUBARRAY(V, STARTPOS, ENDPOS)
```

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)

Details

If *STARTPOS* and *ENDPOS* are invalid numbers, the function returns **#REF** as the result.

Example 1

```
SUB1 = SUBARRAY(rESV, 10, 20)
```

Given an existing column (vector), V_{exist}, containing values in rows 2 through 60, you could use SUBARRY to return a new column (vector), Vnew, containing only the values from rows 20 through 40 of V_{exist}.

Also see

None

SUMMV Formulator function

Returns a column (vector) *VY* that consists of moving summation of a column (vector) V.

Usage

```
SUMMV(V)
```

V	The name of any column (vector) listed under Columns

Details

The nth value in *VY* (Y_n) is the sum of the nth 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 = the values in column (vector) V.

Example 1

```
F1 = SUMMV(BASEI)
```

Example 2

```
PSISPSIO = SUMMV((1-CQADJ/COX)*DELTA(VGS))*DOPETYPE
```

Example 3

The following example illustrates the SUMMV function numerically.

V	VY = SUMMV(V)
1.0000	1.0000
2.0000	3.0000
3.0000	6.0000
4.0000	10.0000
•	•
•	•
•	•

Also see

None

Line Fits

The Line Fit Formulator functions allow you to set up different types of line fits.

EXPFIT Formulator function

Performs an exponential fit. 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.

Usage

`EXPFIT(`*`VX`*`, `*`VY`*`, `*`STARTPOS`*`, `*`ENDPOS`*`)`

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

Details

Using the above exponential relationship, returns a new column (vector) containing Y values calculated from all X values in column VX.

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.

Example

```
DIODEI = EXPFIT(ANODEV, ANODEI, 2, LASTPOS(ANODEV))
```

Also see

EXPFITA (on page 6-317)
EXPFITB (on page 6-318)

EXPFITA Formulator function

Performs an exponential fit. 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.

Usage

`EXPFITA(`*`VX`*`, `*`VY`*`, `*`STARTPOS`*`, `*`ENDPOS`*`)`

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

Details

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.

Example

```
DIODEI = EXPFIT(ANODEV, ANODEI, 2, LASTPOS(ANODEV))
```

Also see

EXPFIT (on page 6-316)
EXPFITB (on page 6-318)

EXPFITB Formulator function

Performs an exponential fit. 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.

Usage

EXPFIT(*VX*, *VY*, *STARTPOS*, *ENDPOS*)

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

Details

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.

Example

```
DIODEIDEALITY = 1/(EXPFITB(ANODEV, ANODEI, 2, LASTPOS(ANODEV))*0.0257)
```

Also see

EXPFITA (on page 6-317)
EXPFITB (on page 6-318)

LINFIT Formulator function

Finds a linear equation and returns a new column (vector).

Usage

```
LINFIT(VX, VY, STARTPOS, ENDPOS)
```

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

Details

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.

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.

Example

```
RESISTORFIT = LINFIT (rESV, RESI, FIRSTPOS(rESV), LASTPOS(rESV))
```

Also see

REGFIT (on page 6-329)

LINFITSLP Formulator function

Finds a linear equation and returns the slope.

Usage

```
LINFITSLP(VX, VY, STARTPOS, ENDPOS)
```

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

Details

Finds a linear equation and returns the slope as follows:

- 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`).

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.

Example

```
RESISTANCE = 1/LINFITSLP(rESV, RESI, FIRSTPOS(rESV), LASTPOS(rESV))
```

Also see

REGFITSLP (on page 6-330)

LINFITXINT Formulator function

Finds a linear equation and returns the X intercept.

Usage

```
LINFITXINT(VX, VY, STARTPOS, ENDPOS)
```

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

Details

Finds a linear equation and returns the X intercept as follows:

- 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`).

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.

Example

```
EARLYV = LINFITXINT(COLLECTORV, COLLECTORI, 56, 75)
```

Also see

REGFITXINT (on page 6-331)

LINFITYINT Formulator function

Finds a linear equation and returns the Y intercept.

Usage

```
LINFITYINT(VX, VY, STARTPOS, ENDPOS)
```

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

Details

Finds a linear equation and returns the Y intercept as follows:

- 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`).

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.

Example

```
OFFSET = LINFITYINT(GATEV, GATEI, FIRSTPOS(GATEV), LASTPOS(GATEV))
```

Also see

REGFITYINT (on page 6-332)

LOGFIT Formulator function

Performs a base-10 log-linear fit.

Usage

```
LOGFIT(VX, VY, STARTPOS, ENDPOS)
```

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

Details

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*.

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.

Example

```
GOODFIT = LOGFIT(GATEV, DRAINI, 30, 50)
```

Also see

LOGFITA (on page 6-324)
LOGFITB (on page 6-325)

LOGFITA Formulator function

Performs a base-10 log-linear fit.

Usage

```
LOGFITA(VX, VY, STARTPOS, ENDPOS)
```

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

Details

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 the value of the constant LOGFITA.

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.

Example

```
OFFSET = LOGFITA(GATEV, DRAINI, 30, 50)
```

Also see

LOGFIT (on page 6-323)
LOGFITB (on page 6-325)

LOGFITB Formulator function

Performs a base-10 log-linear fit.

Usage

```
LOGFITA(VX, VY, STARTPOS, ENDPOS)
```

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

Details

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 the value of the constant LOGFITB.

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.

Example

```
FACTOR = LOGFITB(GATEV, DRAINI, 30, 50)
```

Also see

LOGFIT (on page 6-323)
LOGFITA (on page 6-324)

POLY2COEFF Formulator function

Enables quadratic regression line fitting.

Usage

```
POLY2COEFF(VX, VY, STARTPOS, ENDPOS)
```

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

Details

Enables quadratic regression line fitting. It allows a set of data to best fit an equation of the parabola $Y = aX^2 + bX + c$.

The `a`, `b`, and `c` values of the quadratic equation are returned.

The quadratic regression line fit functions are useful for deriving the defect density when you use the drive-level capacitance profiling (DLCP) technique.

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.

Also see

POLY2FIT (on page 6-327)
POLYNFIT (on page 6-328)

POLY2FIT Formulator function

Enables quadratic regression line fitting.

Usage

```
POLY2COEFF(VX, VY, STARTPOS, ENDPOS)
```

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

Details

Enables quadratic regression line fitting. It allows a set of data to best fit an equation of the parabola $Y = aX^2 + bX + c$.

The `a`, `b`, and `c` values of the quadratic equation are returned.

The quadratic regression line fit functions are useful for deriving the defect density when you use the drive-level capacitance profiling (DLCP) technique.

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.

Also see

POLY2COEFF (on page 6-326)
POLYNFIT (on page 6-328)

POLYNFIT Formulator function

`POLYNFIT` (n^{th} order) does polynomial approximation from the 1st order to the 9th order.

Usage

`POLYNFIT(`*`VX`*`, `*`VY`*`, `*`ORDER`*`, `*`STARTPOS`*`, `*`ENDPOS`*`)`

VX	The name of any column (vector) listed under Columns
VY	The name of any column (vector) listed under Columns
ORDER	The order
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

Details

Enables quadratic regression line fitting. It allows a set of data to best fit an equation of the parabola $Y = aX^2 + bX + c$.

The `a`, `b`, and `c` values of the quadratic equation are returned.

The quadratic regression line fit functions are useful for deriving the defect density when you use the drive-level capacitance profiling (DLCP) technique.

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.

Also see

POLY2COEFF (on page 6-326)
POLY2FIT (on page 6-327)

REGFIT Formulator function

Performs a linear regression fit.

Usage

```
REGFIT(VX, VY, STARTPOS, ENDPOS)
```

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

Details

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*.

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.

Example

```
COLLECTORFIT = REGFIT(COLLECTORV, COLLECTORI, 25, LASTPOS(COLLECTORV))
```

Also see

REGFITSLP (on page 6-330)
REGFITYINT (on page 6-332)

REGFITSLP Formulator function

Performs a linear regression fit.

Usage

```
REGFITSLP(VX, VY, STARTPOS, ENDPOS)
```

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

Details

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.

- Returns the value of the slope constant REGFITSLP in the relationship above.

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.

Example

```
COLLECTORRES = 1/REGFITSLP(COLLECTORV, COLLECTORI, 25, LASTPOS(COLLECTORV))
```

Also see

REGFIT (on page 6-329)
REGFITYINT (on page 6-332)

REGFITXINT Formulator function

Performs a linear regression fit.

Usage

```
REGFITXINT(VX, VY, STARTPOS, ENDPOS)
```

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

Details

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.
- Returns the value of the X intercept for relationship above.
  (–REGFITYINT/REGFITSLP).

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.

Example

```
EARLYV = REGFITXINT(COLLECTORV, COLLECTORI, 25, LASTPOS(COLLECTORV))
```

Also see

REGFIT (on page 6-329)
REGFITYINT (on page 6-332)

REGFITYINT Formulator function

Performs a linear regression fit.

Usage

```
REGFITYINT(VX, VY, STARTPOS, ENDPOS)
```

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

Details

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.

- Returns the value of the Y intercept for relationship above (REGFITYINT).

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.

Example

```
OFFSET = REGFITYINT(COLLECTORV, COLLECTORI, 25, LASTPOS(COLLECTORV))
```

Also see

REGFIT (on page 6-329)
REGFITXINT (on page 6-331)

TANFIT Formulator function

Finds a linear equation of the form `Y = a + bX` from two columns (vectors), *VX* and *VY*.

Usage

`TANFIT(`*VX*`, `*VY*`, `*POS*`)`

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) where the tangent is to be found

Details

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*.

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.

Example

```
VTFIT = TANFIT(GATEV, DRAINI, MAXPOS(GM))
```

Also see

TANFITSLP (on page 6-334)
TANFITXINT (on page 6-335)
TANFITYINT (on page 6-336)

TANFITSLP Formulator function

Finds a linear equation of the form `Y = a + bX` from two columns (vectors), *VX* and *VY*.

Usage

`TANFITSLP(`*VX*`, `*VY*`, `*POS*`)`

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) where the tangent is to be found

Details

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`).

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.

Example

```
VTSLOPE = TANFITSLP(GATEV, DRAINI, MAXPOS(GM))
```

Also see

TANFIT (on page 6-333)
TANFITXINT (on page 6-335)
TANFITYINT (on page 6-336)

TANFITXINT Formulator function

Finds a linear equation of the form `Y = a + bX` from two columns (vectors), *VX* and *VY*.

Usage

`TANFITXINT(`*VX*`, `*VY*`, `*POS*`)`

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) where the tangent is to be found

Details

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`).

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.

Example

```
VT = TANFITXINT(GATEV, DRAINI, MAXPOS(GM))
```

Also see

TANFIT (on page 6-333)
TANFITSLP (on page 6-334)
TANFITYINT (on page 6-336)

TANFITYINT Formulator function

Finds a linear equation of the form `Y = a + bX` from two columns (vectors), *VX* and *VY*.

Usage

`TANFITYINT(`*VX*`, `*VY*`, `*POS*`)`

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) where the tangent is to be found

Details

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`).

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.

Example

```
OFFSET = TANFITYINT(GATEV, DRAINI, GMMAX)
```

Also see

TANFIT (on page 6-333)
TANFITSLP (on page 6-334)
TANFITXINT (on page 6-335)

Identify data analysis requirements

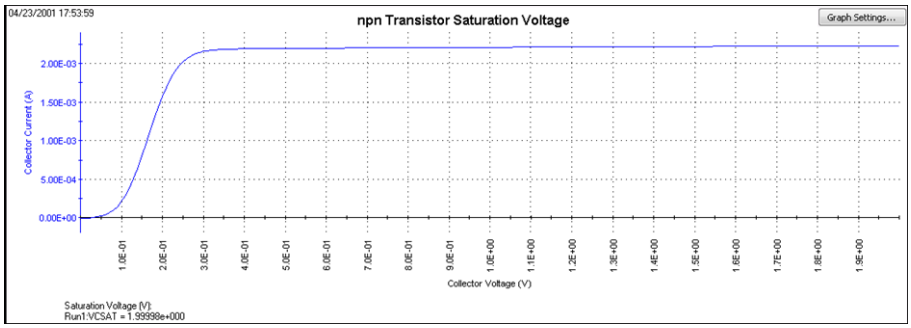
In many cases, you may already know a needed analysis formula, even before running a test. In fact, for a test, you may create a real-time formula in advance so that you can monitor its output during a test, either in the Analyze sheet or graph.

However, in other cases, you may decide to perform an analysis after a review of existing test data. The Formulator functions to use and the data to be included in the Formulator calculations must be evaluated to meet your requirements. The following topics illustrate one such evaluation.

Determining the type of calculation: an example

For example, after looking at the BJT saturation voltage plot below, you need to have a better look at the point where the slope of the saturation plateau becomes constant.

Figure 395: Formulator example data



You might decide to apply the function `REGFIT` to the CollectorV values (and corresponding CollectorI values) between 1 V and 3 V. The line generated by `REGFIT`, when co-plotted with the existing curve, should depart from the plateau at the point of curvature.

The following topics apply the `REGFIT` function to the data in the above figure to illustrate use of the Formulator.

Determining range data 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. These are typically as the row numbers (indices) for the first and last values to be included (refer to Using the Formulator functions (on page 6-294)). 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 Run sheet. For example, referring to the figure below, you might decide to apply `REGFIT` only to the **CollectorV** values between 1 V and 2 V. Looking at the Analyze sheet for this data, you note that the **CollectorV** values between 1 V and 2 V are located between rows (indices) 101 and 201. This is the range information that you need to create the regression analysis equation using `REGFIT`.

Figure 396: Determining the starting and ending row numbers (indices) for the data to be analyzed

	CollectorI	CollectorV	BaseV
97	2.2098E-3	959.9553E-3	699.0722E-3
98	2.2100E-3	969.9862E-3	699.0660E-3
99	2.2102E-3	979.9798E-3	699.0726E-3
100	2.2105E-3	990.0086E-3	699.0747E-3
101	2.2107E-3	1.0000E+0	699.0748E-3
102	2.2110E-3	1.0100E+0	699.0801E-3
103	2.2112E-3	1.0200E+0	699.0759E-3
104	2.2114E-3	1.0300E+0	699.0726E-3
105	2.2117E-3	1.0400E+0	699.0829E-3
106	2.2119E-3	1.0500E+0	699.0737E-3
107	2.2122E-3	1.0600E+0	699.0722E-3
108	2.2124E-3	1.0701E+0	699.0723E-3
109	2.2126E-3	1.0800E+0	699.0801E-3
110	2.2129E-3	1.0901E+0	699.0789E-3
111	2.2131E-3	1.1001E+0	699.0743E-3
112	2.2133E-3	1.1101E+0	699.0740E-3
113	2.2136E-3	1.1200E+0	699.0748E-3
114	2.2138E-3	1.1301E+0	699.0600E-3
115	2.2140E-3	1.1400E+0	699.0608E-3
116	2.2143E-3	1.1500E+0	699.0635E-3
117	2.2145E-3	1.1600E+0	699.0668E-3
118	2.2147E-3	1.1700E+0	699.0674E-3
119	2.2149E-3	1.1800E+0	699.0572E-3
120	2.2152E-3	1.1900E+0	699.0560E-3
121	2.2154E-3	1.2000E+0	699.0518E-3
122	2.2156E-3	1.2099E+0	699.0555E-3
123	2.2158E-3	1.2199E+0	699.0443E-3

	CollectorI	CollectorV	BaseV
175	2.2267E-3	1.7399E+0	698.9456E-3
176	2.2269E-3	1.7499E+0	698.9495E-3
177	2.2271E-3	1.7599E+0	698.9557E-3
178	2.2273E-3	1.7699E+0	698.9453E-3
179	2.2275E-3	1.7799E+0	698.9540E-3
180	2.2277E-3	1.7899E+0	698.9564E-3
181	2.2278E-3	1.7999E+0	698.9256E-3
182	2.2280E-3	1.8099E+0	698.9381E-3
183	2.2282E-3	1.8199E+0	698.9393E-3
184	2.2284E-3	1.8299E+0	698.9346E-3
185	2.2286E-3	1.8399E+0	698.9323E-3
186	2.2288E-3	1.8499E+0	698.9418E-3
187	2.2290E-3	1.8599E+0	698.9480E-3
188	2.2292E-3	1.8700E+0	698.9329E-3
189	2.2294E-3	1.8800E+0	698.9381E-3
190	2.2296E-3	1.8900E+0	698.9332E-3
191	2.2298E-3	1.9000E+0	698.9386E-3
192	2.2300E-3	1.9100E+0	698.9304E-3
193	2.2301E-3	1.9200E+0	698.9313E-3
194	2.2303E-3	1.9300E+0	698.9384E-3
195	2.2305E-3	1.9400E+0	698.9293E-3
196	2.2307E-3	1.9500E+0	698.9235E-3
197	2.2309E-3	1.9600E+0	698.9282E-3
198	2.2311E-3	1.9700E+0	698.9354E-3
199	2.2313E-3	1.9799E+0	698.9368E-3
200	2.2315E-3	1.9900E+0	698.9073E-3
201	2.2317E-3	1.9999E+0	698.9119E-3

Creating an analysis formula

After you have identified the needed Formulator functions and data, create an analysis formula as follows:

1. Enter the left side of the equation. You can use the **F=** option (available from the center number pad) or type in a variable name that contains no spaces and is not the same as a Function name.

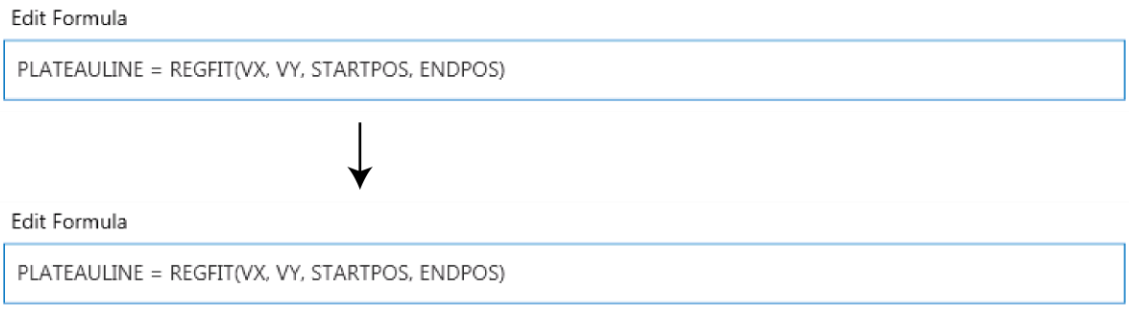
NOTE

Each time that you create an equation with **F=**, the Formulator adds a sequential numerical suffix to the F when you click **Add**. That is, the left side of the first equation is **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 Data Series area, select the area in the formula and click the Data Series item.
   - To insert a constant from the Constants area, click the constant.

For example, to find the regression line for the plateau in the figure in Determining the type of calculation: an example (on page 6-337), enter the equation in the figure below.

Figure 397: Creating the regression formula for the data



Adding an analysis formula to the test

To move from the upper **Formula** box and enter it in the collection of formulas in the lower **Formula** box, click **Add**. If you have previously run the test, this action also executes the new formula for the existing data.

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. (on page 6-344)

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 test 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.
- If you compose a formula containing one or more post test only functions, it executes at the end of each run of the test.

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 **Add**. Thereafter, it executes as listed above.

Viewing analysis results in the Analyze sheet

After executing a new formula, a new column of data containing the results is added to the Analyze sheet. If the formula in the upper Formula box was edited to replace a previous version, the corresponding column in the sheet updates to reflect the changes.

In some cases, a results column contains only a single value.

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.

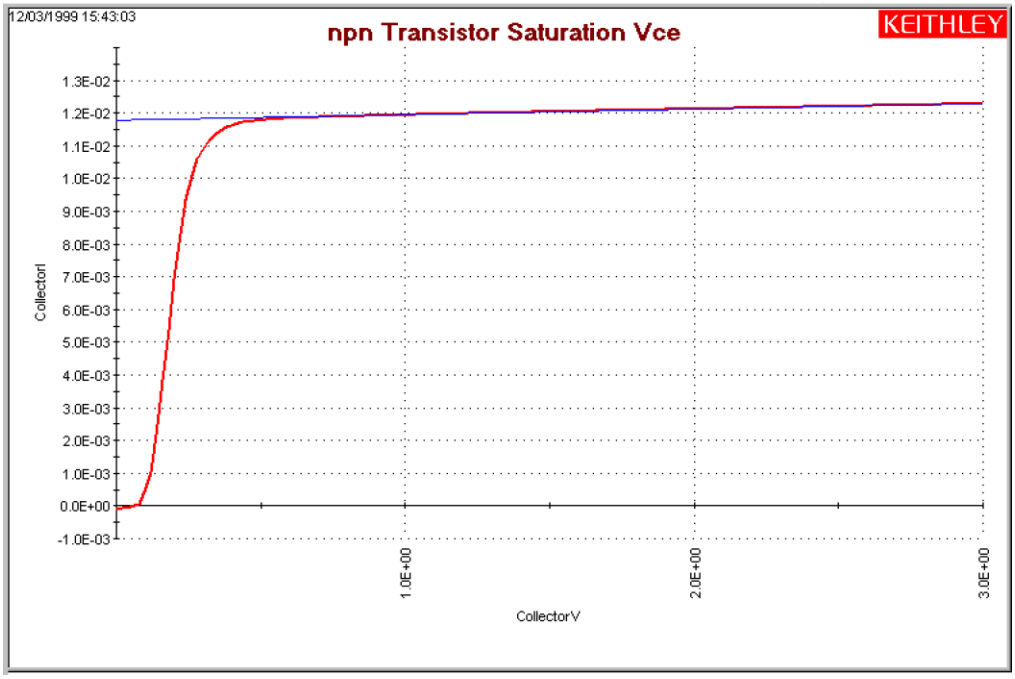
For example, each result of the `DIFF` function is a difference coefficient that is calculated as the ratio `DValues1/DValues2`, where `DValues1` and `DValues2` are differences between values in the present row and values in the previous row. Because no previous row exists before the first row, a valid calculation is not possible for the first row.
Therefore, the Formulator returns the `#REF` notation in the first row.

A column contains 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 contain `#REF`.

Viewing analysis results in the Analyze graph

If a new column (vector) is added to the Analyze sheet after you create or change a formula, it can be plotted in the Analyze graph like any other column (vector). See the following figure.

Figure 398: Added linear regression line to the graph



NOTE

For information about using the Graph tab, refer to Graph (on page 6-255).

Editing Formulator formulas and constants

To edit a Formulator formula:

1. In the lower **Formula** box, double-click the formula to edit. A copy of the formula appears in the upper Formula box
2. In the upper Formula box, edit the formula as needed.
3. Click **Add**.
   - 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, a confirmation dialog box is displayed.
4. Select:
   - **No** if you edited the formula so as to create an additional new formula. Nothing happens to either of the formula boxes. Edit the name of the result variable, then click **Add** again.
   - **Yes** if you edited the formula to update it. The replacement formula appears in the lower Formula box.

Deleting Formulator formulas and constants

To delete a Formulator formula:

1. Select the formula with the cursor.
2. Do either of the following:
   - Click **Delete** in the Formulator dialog box.
   - Press the **Delete** key on the keyboard.

Calc worksheet function definitions

Clarius supports a variety of Calc worksheet functions for the subsite, which are used in the same way as typical spreadsheet functions. These functions are identified below, including their purpose, format, and required arguments.

NOTE

In the Calc worksheet functions, parameter names are shown in Courier font, with the parameter variables shown in italics. For example, `ACOS(`*`Value`*`)`.

ABS Calc worksheet function

This command returns the absolute value of a value.

Usage

```
ABS(Value)
```

Value	Any number

Details

An absolute value does not display a positive or negative sign.

Example

`=ABS(1)` `=ABS(-1)`	Both return a value of `1.0000E+0`.

Also see

SIGN (on page 6-363)

ACOS Calc worksheet function

This command returns the arc cosine of a value.

Usage

```
ACOS(Value)
```

Value	The cosine of an angle, in the range +1 to –1

Details

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()`.

Example

`=ACOS(0.5)` `=ACOS(-0.2)`	Returns `1.0471E+0`. Returns `1.7721E+0`.

Also see

COS (on page 6-349)

ACOSH Calc worksheet function

This command returns the inverse hyperbolic cosine of a value.

Usage

```
ACOSH(Value)
```

Value	Any number equal to or greater than 1

Example

`=ACOSH(1.2)` `=ACOSH(3)`	Returns `622.3625E-3`. Returns `1.7627E+0`.

Also see

ASINH (on page 6-347)
ATAN (on page 6-347)
COSH (on page 6-350)

ASIN Calc worksheet function

This command returns the arcsine of a value.

Usage

```
ASIN(Value)
```

Value	The sine of the resulting angle, ranging from –1 to 1

Details

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()`.

Example

`=ASIN(1)` `=ASIN(0.4)`	Returns `1.5707E+0`. Returns 411.5168E-3

Also see

ASINH (on page 6-347)
PI (on page 6-361)
SIN (on page 6-364)

ASINH Calc worksheet function

This command returns the inverse hyperbolic sine of a value.

Usage

```
ASINH(Value)
```

Value	Any number

Example

```
=ASINH(5.3)
=ASINH(-4)
```

Returns 2.3696E+0.
Returns -2.0947E+0.

Also see

ACOSH (on page 6-346)
ASIN (on page 6-346)
ATANH (on page 6-348)
SINH (on page 6-364)

ATAN Calc worksheet function

This command returns the arctangent of a number.

Usage

```
ATAN(Value)
```

Value	The tangent of the resulting angle

Details

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()`.

Example

```
=ATAN(3.5)
=ATAN(4)
```

Returns 1.2924E+0.
Returns -1.3258E+0.

Also see

ATAN2 (on page 6-348)
ATANH (on page 6-348)
PI (on page 6-361)
TAN (on page 6-367)

ATAN2 Calc worksheet function

This command returns the arctangent of specified coordinates.

Usage

```
ATAN2(x, y)
```

x	The x coordinate
y	The y coordinate

Details

The arctangent is the angle between the x axis and a line with the following end points:

- The origin (0, 0)
- The point at the coordinates (*x*, *y*)
- The angle is returned in radians, ranging between $-\pi$ and π ($-\pi$ is excluded).

Example

`=ATAN2(3, 6)`	Returns `1.1071E+0`.
`=ATAN2(-1, 0.1)`	Returns `3.0419E+0`.

Also see

ATAN (on page 6-347)
ATANH (on page 6-348)
PI (on page 6-361)
TAN (on page 6-367)

ATANH Calc worksheet function

This command returns the inverse hyperbolic tangent of a number.

Usage

```
ATANH(Value)
```

Value	A number between –1 and 1, excluding –1 and 1

Example

`=ATANH(0.5)`	Returns `0.55`.
`=ATANH(-0.25)`	Returns `-0.26`.

Also see

ACOS (on page 6-345)
ASINH (on page 6-347)
TANH (on page 6-367)

AVERAGE Calc worksheet function

This command returns the average of the supplied numbers.

Usage

```
AVERAGE(Value_list)
```

Value_list	A list of numbers separated by commas or a range of number-containing cells in the Calc worksheet

Details

You can average as many as 30 numbers. Text, logical expressions, or empty cells in a cell range are ignored. All numeric values are used, including 0.

The result of `AVERAGE` is also known as the arithmetic mean.

Example

`=AVERAGE(5, 6, 8, 14)` `=AVERAGE(C15:C17)`	Returns `8.2500E+0`. `AVERAGE(C15:C17)` returns the average of the values in cells C15 through C17 of the Calc tab.

Also see

MAX (on page 6-357)
MIN (on page 6-358)

COS Calc worksheet function

This command returns the cosine of an angle.

Usage

```
COS(Value)
```

Value	The angle in radians

Details

If the angle is in degrees, convert the angle to radians by multiplying it by `PI()/180`.

Example

`=COS(1.4444)` `=COS(5)`	Returns `126.0600E-3`. Returns `283.6622E-3`.

Also see

ACOS (on page 6-345)
ASINH (on page 6-347)
ATANH (on page 6-348)
COSH (on page 6-350)
PI (on page 6-361)

COSH Calc worksheet function

This command returns the hyperbolic cosine of an angle.

Usage

```
COSH(Value)
```

Value	Any value

Example

```
=COSH(2.10)
=COSH(0.24)
```

Returns 4.1443E+0.
Returns 1.0289E+0.

Also see

ASINH (on page 6-347)
ATANH (on page 6-348)
COS (on page 6-349)

DAY Calc worksheet function

This command returns the day-of-the-month component of the supplied date and time serial number.

Usage

```
DAY(Serial_number)
```

Serial_number	A date represented as a serial number or text (for example, 06-21-94 or 21-Jun-94)

Details

Needed to extract the day from the serial number created by the NOW function.

Example

```
=DAY(39399)
=DAY("7-21-2016")
=DAY(NOW())
```

Returns 13.0000E+0.
Returns 21.0000E+0..
Returns the present day of the month.

Also see

HOUR (on page 6-352)
MINUTE (on page 6-359)
MONTH (on page 6-360)
NOW (on page 6-361)
SECOND (on page 6-363)
YEAR (on page 6-368)

EXP Calc worksheet function

This command returns the constant e raised to the specified power.

Usage

```
EXP(Value)
```

Value	Any number as the exponent

Details

The constant e is 2.71828182845904 (the base of the natural logarithm).

Example

```
=EXP(2.5)
=EXP(3)
```

Returns 12.1824E+0.
Returns 20.0855E+0.

Also see

LN (on page 6-353)
LOG (on page 6-353)

FIXED Calc worksheet function

This command rounds a number to the supplied precision, formats the number in decimal format, and returns the result as text.

Usage

```
FIXED(Value)
FIXED(Value, Precision)
FIXED(Value, Precision, No_commas)
```

Value	Any number
Precision	The number of digits that appear to the right of the decimal point; if this argument is omitted, a default precision of 2 is used
No_commas	*No_commas* determines if commas separate thousands in the result; send 1 to exclude commas in the result; send 0 or do not define *No_commas* to include separators

Details

If you specify negative *Precision*, *Value* is rounded to the left of the decimal point. You can specify a precision up to 127 digits.

Example

```
=FIXED(2000.5, 3)
=FIXED(2009.5, -1,1)
=FIXED(2009.5, -1,0)
```

Returns 2,000.500.
Returns 2010.
Returns 2,010.

Also see

ROUND (on page 6-362)

HOUR Calc worksheet function

This command returns the hour component of the supplied date and time serial number, specified in 24-hour format.

Usage

```
HOUR(Serial_number)
```

Serial_number	The time as a serial number; the decimal portion of the number represents time as a fraction of the day

Details

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.

Example

`=HOUR(34259.4)`	Returns `9.000E+0`.
`=HOUR(34619.976)`	Returns `23.000E+0`.
`=HOUR(NOW())`	Returns the present hour of the present day.

Also see

DAY (on page 6-350)
MINUTE (on page 6-359)
MONTH (on page 6-360)
NOW (on page 6-361)
SECOND (on page 6-363)
YEAR (on page 6-368)

IF Calc worksheet function

This command tests the condition and returns the specified value.

Usage

```
IF(Condition, True_number, False_number)
```

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

Example

`=IF(A1>10, "Greater", "Less")`	Returns `Greater` if the content of `A1` is greater than 10 and `Less` if the content of `A1` is less than 10.

Also see

None

LN Calc worksheet function

This command returns the natural logarithm (based on the constant e) of a value.

Usage

```
LN(Value)
```

Value	Any positive real number

Example

```
=LN(12.18)
=LN(20.09)
```

Returns 2.4997E+0.
Returns 3.0002E+0.

Also see

EXP (on page 6-351)
LOG (on page 6-353)
LOG10 (on page 6-354)

LOG Calc worksheet function

This command returns the logarithm of a value to the specified base.

Usage

```
LOG(Value)
LOG(Value, base)
```

Value	Any positive real number
base	The base of the logarithm; if *base* is omitted, base 10 is assumed

Example

```
=LOG(1)
=LOG(10)
=LOG(8, 2)
```

Returns 000.0000E-3.
Returns 1.0000E+0.
Returns 3.0000E+0.

Also see

EXP (on page 6-351)
LN (on page 6-353)
LOG10 (on page 6-354)

LOG10 Calc worksheet function

This command returns the base-10 logarithm of a value.

Usage

```
LOG10(Value)
```

Value	Any positive real number

Example

`=LOG10(260)`	Returns `2.4149E+0.`
`=LOG10(100)`	Returns `2.0000E+0.`

Also see

EXP (on page 6-351)
LN (on page 6-353)
LOG (on page 6-353)

LOOKUP Calc worksheet function

This command searches for a value in one range and returns the contents of the corresponding position in a second range.

Usage

```
LOOKUP (Lookup_value, Lookup_range, Result_range)
```

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*, 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	Range of one row or one column that is the same size as the *Lookup_range*

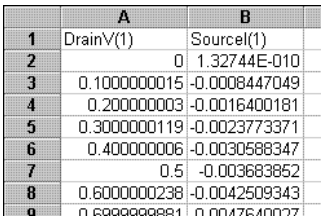
Details

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.

Example

The following examples refer to the Calc worksheet cells illustrated below (these cells were linked to a Run worksheet; refer to Link Run and Settings worksheet cells to Calc worksheet cells (on page 6-252)).

Figure 399: 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

`=LOOKUP (0.5, A2:A8, B2:B8)`	Returns `-0.003683852`.
`=LOOKUP (0.4, A2:A8, B2:B8)`	Returns `-0.0023773371` (see **Details**).
`=LOOKUP (-0.1, A2:A8, B2:B8)`	Returns `#N/A` (see **Details**).

Also see

MATCH (on page 6-356)

MATCH Calc worksheet function

This command compares a specified value against values in a range. The position of the matching value in the search is returned.

Usage

`MATCH(Lookup_value, Lookup_range, Comparison)`

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	The range to search; contains only one row or one column; the range can contain numbers, text, or logical values
Comparison	A 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; see **Details**

Details

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 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 *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 . . .).

When no match is found for *Lookup_value*, `#N/A` is returned.

Example

The following examples refer to the Calc worksheet cells illustrated below (these cells were hot-linked to a Run worksheet, as discussed in Link Run and Settings worksheet cells to Calc worksheet cells (on page 6-252)).

Figure 400: Example Calc worksheet cells

	A	B
1	DrainV(1)	SourceI(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

`=MATCH(0.5,A2:A8,1)`	Returns `6` (the sixth cell relative to cell 2, for example, cell 7).
`=MATCH(0.4,A2:A8,1)`	Returns `4` (the fourth cell relative to cell 2, for example, cell 5).
`=MATCH(0.5,A2:A8,0)` `=MATCH(0.4,A2:A8,0)`	Returns `6` (because an exact match is found). Returns `#N/A` (because an exact match is not found).

Also see

LOOKUP (on page 6-355)

MAX Calc worksheet function

This command returns the largest value in the specified list of numbers.

Usage

`MAX(`*`Value_list`*`)`

`Value_list`	A list of as many as 30 numbers separated by commas

Details

The *`Value_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)` `=MAX(A1:F12)`	Returns `500.0000E+0`. Returns the largest value in this range.

Also see

AVERAGE (on page 6-349)
MIN (on page 6-358)

MIN Calc worksheet function

This command returns the smallest value in the specified list of numbers.

Usage

`MIN(Value_list)`

Value_list	A list of as many as 30 numbers separated by commas

Details

Value_list can contain numbers, logical values, text representations of numbers, or a reference to a range that contains 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

`=MIN(50, 100, 150, 500, 200)` `=MIN(A1:F12)`	Returns `50.0000E+0`. Returns the smallest value in this range.

Also see

AVERAGE (on page 6-349)
MAX (on page 6-357)

MINUTE Calc worksheet function

This command returns the minutes component of the serial number generated by the NOW function.

Usage

```
MINUTE(Serial_number)
```

Serial_number	The time as a serial number; the decimal portion of the number represents time as a fraction of the day

Details

The result is an integer ranging from 0 to 59.

You need to extract minutes from the serial number created by the NOW function.

Example

```
=MINUTE(34506.4)
=MINUTE(34399.825)
=MINUTE(NOW())
```

Returns 36.0000E+0.
Returns 48.0000E+0.
Returns the present minute of the present hour.

Also see

DAY (on page 6-350)
HOUR (on page 6-352)
MONTH (on page 6-360)
NOW (on page 6-361)
SECOND (on page 6-363)
YEAR (on page 6-368)

MONTH Calc worksheet function

This command returns the month component of the supplied date and time serial number or text-formatted date.

Usage

`MONTH(`*`Serial_number`*`)`

`Serial_number`	The date as a serial number or as text (for example, `06-21-15` or `21-Jun-15`)

Details

`MONTH` returns a number ranging from 1 (January) to 12 (December).

You need to extract the month from the serial number created by the `NOW` function.

Example

`=MONTH("07-21-16")`	Returns `7.0000E+0`.
`=MONTH(34626)`	Returns `10.0000E+0`.
`=MONTH(NOW())`	Returns the present month of the present year.

Also see

DAY (on page 6-350)
HOUR (on page 6-352)
MINUTE (on page 6-359)
NOW (on page 6-361)
SECOND (on page 6-363)
YEAR (on page 6-368)

NOW Calc worksheet function

This command returns the present date and time as a serial number.

Usage

```
NOW()
```

Details

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 in the serial number created by the `NOW` function. These other functions can operate on the `NOW` function in a nested format.

Example

`=HOUR(NOW())`	Returns the present hour.

Also see

DAY (on page 6-350)
HOUR (on page 6-352)
MINUTE (on page 6-359)
MONTH (on page 6-360)
SECOND (on page 6-363)
YEAR (on page 6-368)

PI Calc worksheet function

This command returns the value of pi (π), which is approximately 3.1415.

Usage

```
PI()
```

Details

Although `PI` does not use arguments, you must supply the empty parentheses to correctly reference this function.

Also see

COS (on page 6-349)
SIN (on page 6-364)
TAN (on page 6-367)

PRODUCT Calc worksheet function

This command multiplies a list of numbers and returns the result.

Usage

```
PRODUCT(Value_list)
```

Value_list	A list of as many as 30 numbers, separated by commas

Details

Value_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.

Example

`=PRODUCT(1, 2, 3, 4)`	Returns `24.0000E+0`.

Also see

SUM (on page 6-366)

ROUND Calc worksheet function

This command rounds the given number to the supplied number of decimal places.

Usage

```
ROUND(Value, Precision)
```

Value	Any number
Precision	The number of decimal places to which *Value* is rounded

Details

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)` `=ROUND(9899.435, -2)`	Returns `879.2800E+0`. Returns `9.9000E+3`.

Also see

None

SECOND Calc worksheet function

This command returns the seconds component of the supplied date/time serial number

Usage

```
SECOND(Serial_number)
```

Serial_number	The time as a serial number (the decimal portion of the number represents time as a fraction of the day)

Details

Extracts seconds from the serial number created by the NOW function.

Example

```
=SECOND(0.259)
=SECOND(34657.904)
=SECOND(NOW())
```

Returns 58.0000E+0.
Returns 46.0000E+0.
Returns the present second of the present minute.

Also see

DAY (on page 6-350)
HOUR (on page 6-352)
MINUTE (on page 6-359)
MONTH (on page 6-360)
NOW (on page 6-361)
YEAR (on page 6-368)

SIGN Calc worksheet function

This command determines the sign of a specified number.

Usage

```
SIGN(Value)
```

Value	Any number

Details

SIGN returns 1 if the specified number is positive. It returns –1 if the specified number is negative.

Example

```
=SIGN(-456)
=SIGN(456)
```

Returns -1.0000E+0.
Returns 1.0000E+0.

Also see

ABS (on page 6-345)

SIN Calc worksheet function

This command returns the sine of the specified angle.

Usage

```
SIN(Value)
```

Value	The angle in radians

Details

If the angle is in degrees, convert the angle to radians by multiplying the angle by `PI()/180`.

Example

`=SIN(1.5)` `=SIN(4.8)`	Returns `997.4950E-3`. Returns `-996.1646E-3`.

Also see

ASIN (on page 6-346)
PI (on page 6-361)

SINH Calc worksheet function

This command returns the hyperbolic sine of the specified number.

Usage

```
SINH(Value)
```

Value	Any number

Example

`=SINH(1)` `=SINH(3)`	Returns `1.1752E+0`. Returns `10.0178E+0`.

Also see

ASINH (on page 6-347)
PI (on page 6-361)

SQRT Calc worksheet function

This command returns the square root of the specified number.

Usage

```
SQRT(Value)
```

Value	Any positive number

Details

If you specify a negative number, the error `#NUM!` is returned.

Example

`=SQRT(25)` `=SQRT(160)`	Returns `5.0000E+0`. Returns `12.6491E+0`.

Also see

None

STDEVP Calc worksheet function

This command returns the standard deviation of a population based on an entire population of values.

Usage

```
STDEVP(Value_list)
```

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

Details

The standard deviation of a population represents an average of deviations from the population mean within a list of values.

Example

`=STDEVP(4.0, 3.0, 3.0, 3.5, 2.5n 4.0, 3.5)`	Returns `5.0788E-3`.

Also see

VARP (on page 6-368)

SUM Calc worksheet function

This command returns the sum of the supplied numbers.

Usage

```
SUM(Value_list)
```

Value_list	A list of as many as 30 numbers separated by commas

Details

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)` `=SUM(A10:D10)`	Returns `5.0000E+3`. Returns `6.0000E+3` if each cell in the range contains `1500`.

Also see

AVERAGE (on page 6-349)
PRODUCT (on page 6-362)
SUMSQ (on page 6-366)

SUMSQ Calc worksheet function

This command squares each of the supplied numbers and returns the sum of the squares.

Usage

```
SUMSQ(Value_list)
```

Value_list	A list of as many as 30 numbers separated by commas

Details

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 `227.0000E+0`.

Also see

SUM (on page 6-366)

TAN Calc worksheet function

This command returns the tangent of the specified angle.

Usage

```
TAN(Value)
```

Value	The angle in radians

Details

If the angle is in degrees, convert the angle to radians by multiplying the angle by `PI()/180`.

Example

`=TAN(1.5)`	Returns `14.1014E+0`.
`=TAN(45*PI()/180)`	Returns `1.0000E+0`.

Also see

ATAN (on page 6-347)
PI (on page 6-361)
TANH (on page 6-367)

TANH Calc worksheet function

This command returns the hyperbolic tangent of a value.

Usage

```
TANH(Value)
```

Value	Any number

Example

`=TANH(1.5)`	Returns `905.1483E-3`.
`=TANH(1.1)`	Returns `800.4990E-3`.

Also see

ATANH (on page 6-347)
COSH (on page 6-350)
SINH (on page 6-364)
TAN (on page 6-367)

VARP Calc worksheet function

This command returns the variance of a population based on an entire population of values.

Usage

`VARP(Value_list)`

Value_list	A list of as many as 30 numbers, separated by commas

Details

Value_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 `265.3061E-3`.

Also see

STDEVP (on page 6-365)

YEAR Calc worksheet function

This command returns the year component of the supplied date and time serial number or text-formatted date.

Usage

`YEAR(Serial_number)`

Serial_number	The date as a serial number or as text (for example, `06-21-15` or `21-Jun-15`)

Details

Extracts the year from the serial number created by the `NOW` function.

Example

`=YEAR(39328)`	Returns `2.0070E+3`.
`=YEAR("06/21/16")`	Returns `2.0160E+3`.
`=YEAR(NOW())`	Returns the present year.

Also see

DAY (on page 6-350)
HOUR (on page 6-352)
MINUTE (on page 6-359)
MONTH (on page 6-360)
NOW (on page 6-361)
SECOND (on page 6-363)

Tools

The Tools menu includes tools specific to the SMU, CVU, and PMU instruments that are installed in your 4200A-SCS.

The options include:

- **SMU Auto Calibration:** Recalibrates the current and voltage offsets for all source and measurement functions of all SMUs in the system. To maintain SMU performance specifications, you must auto calibrate the 4200A-SCS every 24 hours or any time after the ambient temperature has changed more than ±1 °C. Refer to Calibrate the system (on page 13-5) for instructions.
- **CVU Connection Compensation:** Corrects offset and gain errors caused by the connections between the CVU and the device under test (DUT). Refer to Connection compensation (on page 4-14) for instructions.
- **CVU Real-Time Measure Mode:** Provides a direct real-time user interface to the CVU to help you set up and debug your system. For example, you can use it to confirm that contact has been made with the pads on a wafer. Refer to CVU Real-Time Measure Mode (on page 4-26) for instructions.
- **CVU 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). Refer to Confidence Check (on page 4-27) for instructions.
- **PMU Connection Compensation:** Corrects errors caused by the connections between the 4225-PMU and the DUT. Refer to PMU connection compensation (on page 5-40) for instructions.

Customize Clarius

To customize Clarius, you can:

- Add your own tests, actions, and projects to the Clarius library.
- Adjust the options in My Settings to set up your working environment, modify project execution, and determine the graphing defaults. You can also set up custom GPIB abort settings and view information about Clarius.
- Hide or display the project tree, right pane, and Messages areas of the Clarius window.

Add objects to the library

You can add tests, devices, actions, and projects to the library. The new version of the object includes the settings you made to the object in the Configure pane. Once an object has been added, you can use it to create new objects in the project tree.

You cannot add sites and subsites to the library.

When you copy a project, it includes all test definitions, formulas, graph settings, and selected run histories.

NOTE

When you add objects to the library, you have the option to enter keywords. You can use these keywords to label the object with information that you can use to search for the object, such as including your name or a project name as part of the project.

Add a test to the library

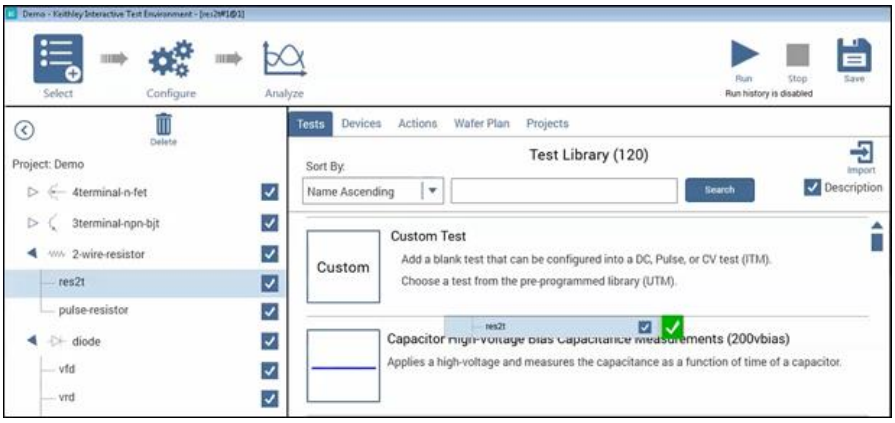
The following steps provide specifics on how to add a test to the library. You can follow the same basic procedure to add devices, actions, and projects. The primary difference is the type of object and which library the object is added to.

When you add a test to the library, Run Histories that are selected on the Analyze pane are included with the new test. However, notes and aliases that were assigned to the Run History items in the original test are not included.

To add a test to the library:

1. In Clarius, set up the test so that it contains the settings you want the new library object to have.
2. Open the **Analyze** pane.
3. Select the run histories that you want to include in the new test.
4. Open the **Select** pane.
5. Drag the test from the project tree to the library. You will see a copy of the test and a checkmark, as shown in the figure below. The test is automatically added to the Tests library, regardless of the tab that is open. When you drop the test, a confirmation dialog box is displayed.

Figure 401: Add a test to the Tests Library



6. Select **OK**. The Library Information Editor is displayed. Refer to Edit information for a library object (on page 6-373) to complete the Library Information Editor.

Add a device to the Device Library

You can copy a device from the project tree to the library to create a new device.

To submit a device to a library:

1. In Clarius, choose **Select**.
2. In the project tree, drag the device into the library. The test is automatically added to the Devices library, regardless of the tab that is open. A confirmation message is displayed.
3. Select **OK**. The Library Information Editor dialog box is displayed. Refer to Edit information for a library object (on page 6-373) to complete the Library Information Editor.

Add an action to the library

The following example provides specifics on how to add an action to the library.

To add an action to the library:

1. In Clarius, set up the action so that it contains the settings you want the new library object to have.
2. Open the **Select** pane.
3. Drag the action from the project tree to the library. You see a copy of the action and a checkmark. The action is automatically added to the Actions library, regardless of the tab that is open. When you drop the action, a confirmation dialog box is displayed.
4. Select **OK**. The Library Information Editor is displayed. Refer to Edit information for a library object (on page 6-373).

Add a project to the library

The following example provides specifics on how to add a project to the library.

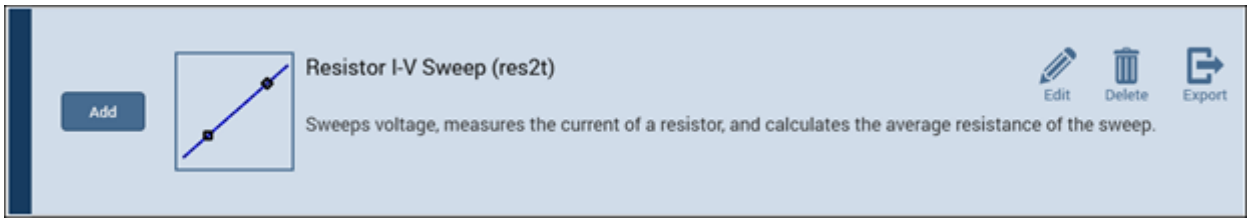
To add a project to the library:

1. In Clarius, set up the project so that it contains the settings you want the new library object to have.
2. Open the **Select** pane.
3. Drag the project from the project tree to the library. You see a copy of the project and a checkmark. The project will be added to the Projects library regardless of the tab that is open. When you drop the project, a confirmation dialog box is displayed.
4. Select **OK**. The Library Information Editor is displayed. Refer to Edit information for a library object (on page 6-373).

Edit a library object you added

You can edit items that you added to a library. Items that can be edited have Edit, Delete, and Export options as shown in the following figure.

Figure 402: Edit, Delete, and Export options for a library item



To edit an item:

1. Select the item in the library.
2. Select **Edit**.
3. Refer to Edit an object in the library (on page 6-374) for the options.

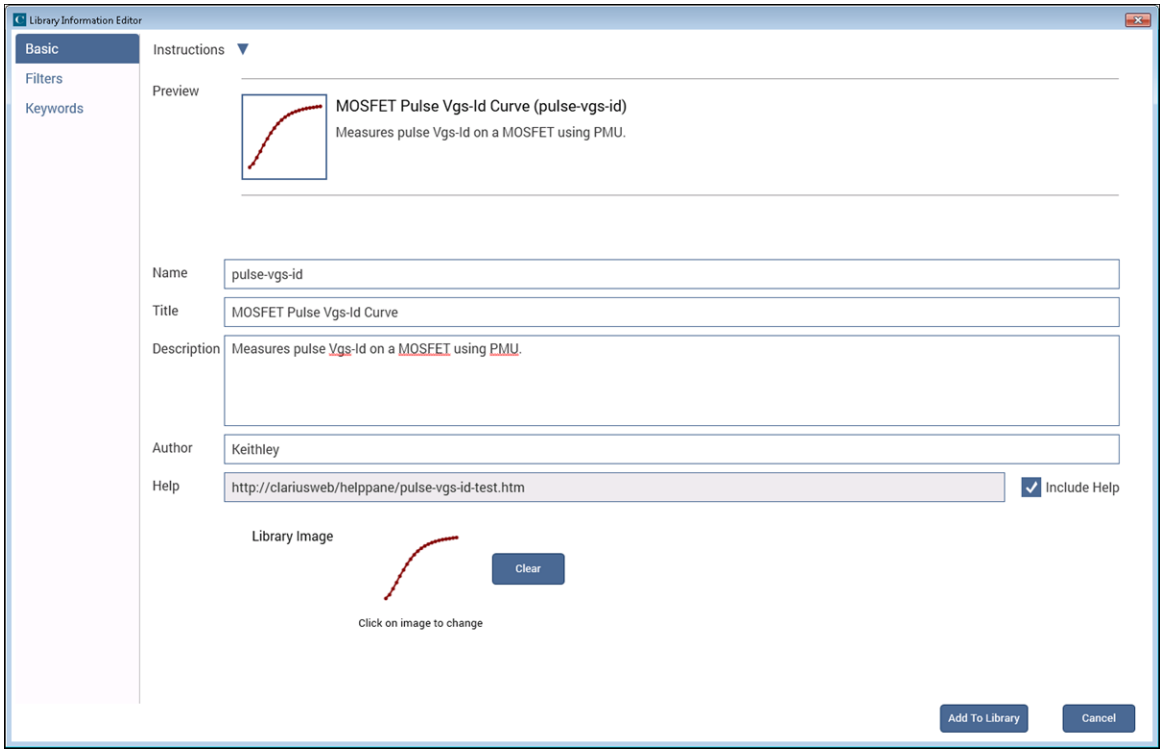
Edit an object in the library

The Library Information Editor allows you to change the information for a new library object. You can change information such as the name, description, graphic, help information, filters, and keywords.

The Library Information Editor is displayed when you drag an object from the project tree to the library. For objects that you created, you can also access it using Edit in the library.

As you make changes in the Basic tab, the Preview area displays the changes as they will appear in the library.

Figure 403: Library Information Editor



To change information for a library object:

1. In the Basic tab, complete the information as needed. Refer to the table below for the options.
2. Select the **Filters** tab. These options set the filters that will cause this item to appear in the library when you select the right-pane filters.
3. Select the filters that help a user find this item in the library.
4. Select the **Keywords** tab. These options determine what you can type in the library Search field to locate this item. You can use the Sort By options at the bottom of the lists to change the order of the entries in the Information Editor. It does not affect the order in the library.
5. Drag a keyword from the left to the right to add a keyword.
6. To remove a keyword, select the keyword and select **Delete**. This does not remove the keyword from the Global Keywords list.
7. To add a keyword, select **New** and type the keyword.

Options in the Information Editor	
Preview	Displays the changes you make as they will appear in the library.
Name	Type the new name. This is the name that is used in the library and the project tree.
Title	Type the title. This is used in the library.
Description	Type a brief description of the item. This is displayed in the library.
Author	Type information that identifies who created this item. This is available only through the Library Information Editor.
Help	Only editable when adding an object from the project tree to the library. This is the help file that is displayed in the right pane when Configure or Analyze is selected. It is also displayed when the item is selected in the library.
Include Help	Only displayed when you add an item from the project tree to the library. Determines if the existing help is included with the new item. If you want to include the help that was associated with the original object, select **Include Help**. Clear **Include Help** to keep the help from displaying (the Help pane will be blank). You cannot change the help link; you can only include or hide it.
Library Image	The image that is displayed in the library. Click the image to select a different image. Images should be 400x400 pixels in `png` format. Larger images display, but anything larger than 400x400 is cut off in the library display. To re-use an image from an older project, you may need to save the existing `bmp` image to `png` format. You can use a tool such as Microsoft® Paint to convert the image. To leave the image area blank, select **Clear**.
Icon Image	Devices only. The image that is displayed in the project tree. Click the image to select a different image. Images should be 80x80 pixels in `png` format. Larger images will display, but anything larger than 80x80 will be cut off in the project tree. To re-use an image from an older project, you may need to save the existing `bmp` image to `png` format. You can use a tool such as Microsoft® Paint to convert the image. To leave the image area blank, select **Clear**.

My Settings

The options in My Settings allow you to set environment settings, run settings, graph defaults, and GPIB abort preferences. You can also view revision and copyright information regarding Clarius.

Specify environment settings

The options in Environment Settings allow you to determine if GPIB devices are reset on startup, if unsettled measurements for PMUs are allowed, if you can edit the user interface for user test modules (UTMs), if site tracking is available for stress testing, and where your projects are stored.

Reset GPIB devices when Clarius starts

Determines if the GPIB devices in the system are reset to their default settings when Clarius starts up.

PMU: Allow unsettled measurements

When this option is selected, all 4225-PMU instrument cards ignore the minimum timing versus measure range relationship. This is only recommended for advanced users, as spot mean results will not be settled, which may cause a variety of operational issues, such as:

- Inconsistent current measure range-changing on the 4225-PMU or 4225-RPM.
- Lack of proper load-line effect compensation (LLEC) for the PMU.
- Lack of correlation between PMU or PMU and RPM results and SMU results.

See PMU minimum settling times versus current measure range (on page 5-52) for additional information.

Allow access to UTM UI editor

Select this option to access the user interface editor for user test modules (UTMs).

Refer to Define the user interface for a user test module (on page 6-144) for detail on how to use the editor.

NOTE

To prevent unintentional changes to the user interface, it is good practice to disable the editor after changes are complete.

Track site during stress test

Select this option to track sites during the stress test.

When this option is selected, during the subsite stress test:

- The site identification box in the project tree is automatically updated to the presently started site. For example, if the run site was set to 3, then after Run is selected, it is automatically reset to 1. It changes to 2 as soon as testing switches to site 2.
- All open subsite views are automatically switched to the presently running site. The Configure and Analyze panes will match the presently running site.
- If the subsite is opened, that also automatically converts to the present site during execution.

You can still change to other sites manually during execution.

My Projects Directory

Use this option to change the directory where your projects are stored. To create a new folder, right-click and select **New Folder**.

Changing the project directory does not affect the open project or previously created projects. If you save the open project, it is saved to its original location.

Existing projects remain in the directory in which they were previously stored. The default location is `C:\s4200\kiuser\projects`.

You can move projects to the new location using Windows Explorer. When moving projects, move the project folder and all its contents.

Specify run settings

These options allow you to specify if a project continues to run after an error, if interlock states are ignored, if autoscroll is active on the Analyze sheet during a test, and if hardware is reinitialized after the run.

Continue to run after an error

Determines if Clarius continues running a project sequence when it encounters errors in a test. Examples of errors are tests for which no SMUs have been specified and tests that are not configured or are improperly configured.

When this option is selected, if Clarius encounters an error, it displays an error message in the message area at the bottom of the Clarius window and continues execution on the next test in the sequence.

Ignore interlock state

If "Ignore interlock state" is selected and the 4200A-SCS interlock circuit is disconnected or otherwise open, Clarius continues to execute tests. However, Clarius automatically limits the output voltage to a safe level, even if a test specifies a higher level.

If "Ignore interlock state" is cleared and the 4200A-SCS interlock circuit is disconnected or otherwise open, Clarius displays a warning message and disables the execution of all tests.

Autoscroll the Analyze sheet during test

When autscroll is selected, if you are viewing the Analyze pane during test execution, the sheet scrolls so that new data is always displayed.

Reinitialize hardware after run

When this option is selected, all instruments in the system return to their default settings after the test completes. The SMU output remains at the last programmed value for a brief time before being reinitialized.

⚠ WARNING

If "Reinitialize hardware after run" is cleared, all outputs remain at their last programmed levels after the test is completed. To prevent electrical shock that could cause injury or death, never make or break connections to the 4200A-SCS while the output is on.

Graph defaults

The My Settings Graph Defaults option allows you to set the defaults that are used for the Analyze graph when graphs are generated from a Run History:

- **Default Line Width:** Sets the line width that is used if a line is displayed.
- **Default Data Display:** Select **Points** to display each data point on the graph, **Lines** to display a line connecting the points, or **Points and Lines** to display both.
- **Y1 Default Most Recent Data Color:** The color that is used for data graphed against the Y1 axis.
- **Y1 Default Older Data Color:** The color that was used for data graphed against the Y1 axis. To have Clarius select the color automatically, select **Auto (cycle through colors)**.
- **Y2 Default Most Recent Data Color:** The color that is used for for data graphed against the Y2 axis.
- **Y2 Default Older Data Color:** The color that was used for data graphed against the Y2 axis. To have Clarius select the color automatically, select **Auto (cycle through colors)**.

Custom GPIB Abort Options

These options allow you to set the operations that occur when a GPIB abort is sent.

You can select that a `*RST` and `DCL` occur when an abort occurs.

If you enable Custom String, you can send a a user-defined GPIB command string to the instrument.

NOTE

While most instruments will 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.

Instruments are added to this list when they are added to the system through KCon. Instruments added as General Purpose Test Instruments are shown. Refer to Use KCon to add equipment to the 4200A-SCS (on page 7-7) for more information.

To add a custom string, select the box to the left of the Custom String box for the instrument.

Project tree display options

You can select whether or not Clarius displays the project tree if you need more workspace. To hide the project tree, click **<** at the top of the tree. Redisplay the project tree by selecting **>**.

Messages display option

You can hide the Messages area at the bottom of the Clarius window. To hide Messages, select v. To display the messages, select ^.

User library descriptions

Keithley Instruments provides several user libraries of user modules. The following topics provide an overview of each of the user libraries.

The KULT user libraries and user modules that are provided with Clarius+ are available in the directory:

```
C:\s4200\kiuser\usrlib\
```

AVMControl user library

The `AVMControl` user library contains a user module that limits the SMU maximum voltage. The next table lists and briefly describes the user module.

AVMControl user module

User module	Description
`SetAVMLevel`	Sets the 4200-SMU or 4210-SMU absolute voltage monitor (AVM) maximum voltage. The AVM is an analog circuit that limits the SMU voltage output regardless of what the SMU is sourcing and measuring. Depending on the voltage that the AVM is set to, the SMU clamps the output voltage to one of the built-in voltage limits. Refer to the Help pane for the voltage limits.

BeepLib user library

The `BeepLib` user library contains several user modules that control the 4200A-SCS beeper. The next table lists and briefly describes the user modules.

The beeper user modules are affected by the Windows operating system audio settings. For example, if sound is muted, the beeper will not sound.

BeepLib user modules

User module	Description
`beep`	Specifies the frequency and duration of the beeper.
`BeepCharge`	Sounds a battle cry through the speaker.
`BeepDown`	Sounds a series of beeps in descending tones through the speaker.
`BeepInfiniteLoop`	This function sounds a series of beeps through the system speaker. The beeps continue until they are terminated.
`BeepUp`	Sounds a series of beeps in ascending tones through the speaker.

chargepumping user library

The `chargepumping` user library contains several user modules to characterize interface and charge-trapping phenomena. The next table lists and briefly describes the user modules.

chargepumping 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.
`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.
`BaseSweep`	The base voltage of the waveform is swept by a 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.
`BaseSweep_2SMU`	Same as BaseSweep user module, except it uses a second SMU to apply a DC bias voltage to the source/drain terminals.
`FallTimeLinearSweep`	Performs a linear sweep of the falling transition time of the pulse. Charge pumping current (ICP) is measured and graphed as a function of the fall time.
`FreqFactorSweep`	With the pulse amplitude, offset voltage, and rise/fall time kept constant, the charge pumping current (ICP) is measured as a function of a multiplier factor controlled frequency sweep of the test frequency.
`FreqLinearSweep`	With the pulse amplitude, offset voltage, and rise/fall time kept constant, the (ICP) is measured as a function of a linear sweep of the test frequency.
`RiseTimeLinearSweep`	Performs a linear sweep of the rising transition time of the pulse. Charge pumping (ICP) is measured as a function of the rise time.

cvivulib user library

The `cvivulib` user library contains user modules for configuring the 4200A-CVIV Multi-Switch. The next table lists and briefly describes the user modules.

cvivulib user modules

User module	Actions in Clarius	Description
`cviv_configure`	`cviv-configure`	This user module configures the CVIV relays and the display for each channel.
`cvu_cviv_comp_collect`	`cvu-cviv-comp-collect`	This user module provides CVU compensation collection for open, short, and load with a 4200A-CVIV.

cvucompulib user library

The `cvucompulib` user library contains user modules for collecting 4210-CVU compensation data. The next table lists and briefly describes the user modules.

NOTE

If your configuration includes a 4200A-CVIV, use the `cvivulib` user library instead of this one.

cvucompulib user modules

User module	Description
`cvu_ConstantsFileSelect`	Selects the constants file that is used for a CVU ITM or UTM. The file must be created using `cvu_OSLcomp_collect`.
`cvu_OSLcomp_collect`	Collects the open, short, and load compensations of the CVU instrument as selected. It generates a file that contains the open, short, and load compensation values to apply to the CVU readings that are returned from an ITM or UTM.

DLCP user library

The `DLCP` user library contains a user module for making C-V measurement for drive-level capacitance profiling (DLCP). The next table lists and briefly describes the user module.

DLCP user module

User module	Description
`ACSweep`	This module allows you to make C-V measurements for drive-level capacitance profiling (DLCP) using the 4210-CVU. During this measurement, the applied AC voltage is sweeping while the capacitance is measured. The total applied voltage (AC and DC) is kept constant. The total applied voltage is defined as the DC voltage minus 1/2 the p-p AC voltage.

flashulib user library

The `flashulib` user library contains user modules for flash memory testing. The next table lists and briefly describes the user modules.

flashulib user modules

User module	Description
`configure_dc_flash`	Disconnects pulse channels by opening the solid-state relay for each pulse channel in the supplied list. This routine should be used before running a DC test when the pulse and DC signals are connected at each DUT terminal.
`double_pulse_flash`	Defines and outputs 1 to 8 waveforms that consist of two pulses that have indepedent widths and levels. The waveforms are defined using line segments (Segment Arb mode). You can define the waveform for just a program or erase pulse or for a waveform that combines program and erase cycles for up to eight independent pulse channels.
`pmu_configure_dc_flash`	Disconnects pulse channels by opening the solid-state relay for each pulse channel in the supplied list. This routine should be used before running a DC test, when the pulse and DC signals are connected together at each DUT terminal.
`pmu_double_pulse_flash`	Defines and outputs 1 to 8 waveforms that consist of two pulses that have indepedent widths and levels. The waveforms are defined using line segments (Segment Arb mode of the 4220-PGU or 4225-PMU). You can define the waveform for just a program or erase pulse or for a waveform that combines both program and erase cycles for up to eight independent pulse channels.
`pmu_single_pulse_flash`	Defines and outputs 1 to 8 waveforms that consists of two pulses that have indepedent widths and levels. The waveforms are defined using line segments (Segment Arb mode of the 4220-PGU or 4225-PMU). You can define the waveform for just a program or erase pulse or for a waveform that combines both program and erase cycles for up to eight independent pulse channels.
`single_pulse_flash`	Defines and outputs 1 to 8 waveforms that consist of two pulses that have indepedent widths and levels. The waveforms are defined using line segments (Segment Arb mode of the pulse card). You can define the wavform for just a program or erase pulse or for a waveform that combines both program and erase cycles for up to eight independent pulse channels.

GateCharge user library

The `GateCharge` user library contains a user module for configuring the 4200A-SCS to measure gate charge of a power MOSFET. The next table lists and describes the user module.

GateCharge user module

User module	Description
`gate_charge`	This module measures the gate charge of a power MOSFET using two source measure units (SMUs).

hivcvulib user library

The `hivcvulib` user library contains user modules for controlling high-voltage C-V measurements. You can use these modules with either one or two 4205-RBT configurations. The next table lists and briefly describes the user modules.

hivcvulib user modules

User module	Description
`multipleSMU_SweepV`	This module allows you to make high voltage C-V measurements up to 400 V using the 4210-CVU, the 4200-SMU, and the 4200A-CVIV. The CVU measures the capacitance, the SMU supplies the DC bias, and the AC and DC signals are coupled through a bias tee connection in the 4200A-CVIV. For a 3-terminal MOSFET, all three SMUs must be sweeping. The gate and the source SMUs must sweep simultaneously to prevent the device from turning on. It's higly recommended that the SMUs at the gate and source have the same start and stop voltages to prevent damage to the device.
`CvsT`	Provides capacitance measurements as a function of time at a user-specified DC bias. You can measure capacitance up to 200 VDC bias with one 4205-RBT and one SMU. Additionally, you can measure capacitance up to 400 VDC bias with two 4205-RBTs and two SMUs.
`SweepV`	Uses one 4205-RBT to sweep a DC voltage across the DUT using the 4200-SMU or 4210-SMU and measure the capacitance using the 4210-CVU. If two 4205-RBTs are used with the `SweepV` module, one SMU sweeps the DC voltage and the other SMU applies an offset DC bias.

Hotchuck_Temptronics3010B user library

This user library controls the temperature of Temptronics 3010B hotchucks. The user module in this library sets the target temperature and waits until the target is reached before exiting.

Hotchuck_Temptronics3010B user module

User module	Description
`Settemp`	This routine controls the Temptronics thermal controller 3010B and other compatible models.

HP8110ulib user library

Use the user modules in the `HP8110ulib` user library to control a Keysight Model 8110A Pulse Generator. These user modules are summarized in the following table. The table also lists the user test modules (UTM) created by Keithley Instruments that use the user modules.

HP8110ulib user modules

User Module	UTM Name	Description
PguInit8110 (on page E-6)	`pgu1-init`	Initializes the pulse generator to the default setup.
PguSetup8110 (on page E-7)	`pgu1-setup`	Sets the output pulse parameters.
PguTrigger8110 (on page E-9)	`pgu-trigger`	Specifies pulse count and trigger start of output.

Hotchuck_Triotek user library

The user module in the `Hotchuck_Triotek` user library is used to control the temperature of the Trio-Tech hot chuck.

Hotchuck_Triotek user module

User module	Description
`SetChuckTemp`	Sets the temperature of the Trio-Tech hot chuck.

HP4284ulib user library

You use the user modules in the `HP4284ulib` user library to control the Keysight 4284A or 4980A LCR Meter. These user modules are summarized in the following table.

HP4284ulib user library

User module	Description
Cmeas4284 (on page C-13)	Makes a single capacitance measurement.
CvSweep4284 (on page C-11)	Makes capacitance versus voltage measurements using a staircase sweep.

HP4294ulib user library

You can use the user modules in the `HP4294ulib` user library to calibrate and control the Keysight Model 4294 IMP meter. Subroutines are provided to perform voltage or frequency sweeps. These user modules are summarized in the following table.

HP4294ulib user modules

User module	Description
`CvSweep4294`	Performs a capacitance versus voltage sweep.
`FISweep4294`	Performs a frequency versus impedance sweep.
`LoadCal4294`	Performs LOAD calibration.
`OpenCal4294`	Performs OPEN calibration.
`PhaseCal4294`	Performs PHASE calibration.
`ShortCal4294`	Performs SHORT calibration.

A Keysight 4294 measurement is valid only if proper calibrations are performed before the measurement is made. 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 Keysight 4294 is added to the 4200A-SCS test system using KCon. For details, see Keithley Configuration Utility (KCon) (on page 7-1).

NOTE

Details on Keysight 4294 operations are provided in the documentation provided by Keysight for the IMP meter.

ki340xulib user library

Used with the Keithley Instruments Series 3400 pulse/pattern generators.

ki340xulib user modules

User module	Description
`PguInit340x`	Initializes the 3401 or 3402 pulse generator to a specific state.
`PguSetup340x`	Defines the pulse timing and voltage settings. Once defined, the pulse can be triggered using the `PguTrigger340x` command.
`PguTrigger340x`	Triggers the pulse (or pulses) defined by the `PguSetup340x` function.

KI42xxulib user library

The `KI42xxulib` user library provides an example subroutine for doing a MOSFET ON resistance (R_{ON}) test routine using the 4200A-SCS LPT library interface.

KI42xxulib user module

User module	UTM name	Description
`Rdson42XX`	`rdson`	Measures the drain to source resistance of a saturated MOSFET.

KI590ulib user library

The user modules in the `KI590ulib` user library are used to control the 590 C-V Analyzer. These user modules are summarized in the table below. Also listed in the table are names of the user test modules (UTMs) and actions in Clarius that use the user modules.

KI590ulib user modules

User module	UTM or action name	Description
CableCompensate590 (on page B-15)	`cable-compensate` in the `ivcvswitch` project	Performs cable compensation using known capacitance source values.
Cmeas590 (on page B-18)	`590-cmeas`	Makes a single capacitance measurement.
CtSweep590 (on page B-21)	`590-ctsweep`	Makes a capacitance versus time measurement.
CvPulseSweep590 (on page B-26)	`590-cvpulsesweep`	Makes capacitance versus voltage measurements using a pulse sweep.
CvSweep590 (on page B-32)	`590-cvsweep`	Makes capacitance versus voltage measurements using a staircase sweep.
DisplayCableCompCaps590 (on page B-36)	`display-cap-file` in the `ivcvswitch` project	Places capacitance source values in a spreadsheet.
LoadCableCorrectionConstants (on page B-39)	n/a	Reads the cable compensation parameters for the range and frequency specified from the cable compensation file and sends these parameters to the 590.
SaveCableCompCaps590 (on page B-40)	`save-cap-file` in the `ivcvswitch` project	Saves entered capacitance source values in a file.

KI595ulib user library

The user modules in the `KI595ulib` user library are used to perform Q/t sweeps and C-V sweeps using the Keithley Instruments 595 Quasistatic C-V Meter. These user modules are summarized in the table below.

KI595ulib user modules

User module	Description
`CVsweep595`	Performs a quasistatic C-V sweep between the start voltage and the stop voltage. The data returned is the source voltage, measured capacitance, Q/t current, and timestamp on each measurement.
`QTSweep595`	Makes 20 Q/t current and capacitance measurements with various time delays that are spaced between 0.07 s and the maximum delay. You can analyze and plot the resulting values to determine the optimum delay time to use during the C-V sweep. The optimum delay time is the time when Q/t reaches the system leakage level.

ki82ulib user library

The user modules in the `ki82ulib` user library control the Model 82 C-V System. They perform simultaneous C-V, C-t, and Q/t measurements and cable compensation. The next table lists the user modules. It also provides the name of tests and actions in Clarius that are based on these user modules.

ki82ulib user modules

User module	Test and action names	Description
Abortmodule82 (on page D-37)	n/a	Puts the three System 82 instruments into a known state when a test is aborted. This function is used by other library modules in the `atexit()` function.
CableCompensate82 (on page D-38)	`cable-compensate` `cablecomp`	Performs cable compensation using known capacitance source values.
CTsweep82 (on page D-41)	`ctsweep`	Performs C-t measurements.
DisplayCableCompCaps82 (on page D-45)	`display-cap-file`	Places capacitance source values in a spreadsheet.
`LoadCableCorrectionConstants82`	n/a	Read the cable compensation parameters and sends them to the 590. This module is for internal use by the `SIMCVsweep82` and `CTsweep82` modules. It is not normally used as a stand-alone module.
QTsweep82 (on page D-47)	`qtsweep`	Performs quasistatic measurement sweep.
SaveCableCompCaps82 (on page D-50)	`save-cap-file` `savecablecompfile`	Saves entered capacitance source values in a file.
SIMCVsweep82 (on page D-53)	`system82-cvsweep` `cvsweep`	Performs simultaneous C-V sweep.

LS336ulib user library

The `LS366ulib` provides user modules that control the Lake Shore Cryotronics 336 Temperature Controller.

LS336ulib user modules

User module	Description
`heaterOff`	Turns off heater 1 and heater 2 and disables setpoint ramping on both outputs.
`setDelay_Dialog`	Either displays a window that contains the message you specified and an OK button, or performs the delay set by `WaitTime`.
`setSweepParams`	Generates the list of temperatures used by `setTemp` when the `setTemp` parameter `FlagMode` is set to `1`. When active, it calculates the temperature profile to be measured from the start temperature, stop temperature, and step points input. It outputs the temperature list to the file.
`setTemp`	Controls key aspects of the temperature controller, including setpoint, heater parameters, and ramp rates, to allow variable temperature electrical measurements. This routine is designed to function inside a subsite cycle test with the `heaterOff` and `setSweepParam` routines.

Matrixulib user library

The `Matrixulib` connects instrument terminals to output pins using a Keithley Instruments Series 700 Switching System. It is for use with switching systems that are configured as a general purpose, low current, or ultra-low current matrix.

Matrixulib user module

User module	Description
`ConnectPins`	Allows you to control your switch matrix.

MultiSegmentSweep_ulib user library

The `MultiSegmentSweep_ulib` contains two user modules that let you run up to a four-segment current or voltage linear sweep.

These modules are only supported for the 42x0-SMU instruments.

MultiSegmentSweep_ulib user modules

User module	Description
`MultiSegmentSweepI`	This module runs up to a four-segment current linear sweep.
`MultiSegmentSweepV`	This module runs up to a four-segment voltage linear sweep.

nvm user library

The `nvm` user library contains user modules that are used for nonvolatile memory tests that use a source-measure and a pulse measure unit. The next table lists and briefly describes the user modules.

For additional detail on working with user modules in the nvm user library, refer to the application note "Pulse I-V Characterization of Non-Volatile Memory Technologies."

For detail on creating a custom user module for nonvolatile memory tests, refer to the read me file in the directory `C:\s4200\kiuser\usrlib\nvm`.

nvm user modules

User module	Description
`dcSweep`	Applies a long signal, either positive or negative. You can specify the rise time, the slew rate, and the time to hold the voltage at the top or bottom.
`doubleSweep`	Creates a waveform that consists of two voltage sweeps: 0 to V1, V1 to 0, 0 to V2 and V2 to 0. The sweeps are generated on PMU1CH1. Channel PMU1CH2 is kept at 0 V and measures current and charge.
`doubleSweepSeg`	Creates a waveform that consists of two voltage sweeps: 0 to V1, V1 to 0, 0 to V2 and V2 to 0. The sweep is generated on PMU1CH1. Channel PMU1CH2 is kept at 0 V and measures current and charge.
`flashEndurance`	Defines pulse sequences for the program/erase, program, and erase pulses. It runs the program/erase sequence a defined number of times by logarithmically spaced numbers of loops. After each iteration, it does program and erase one more time with Vt extraction after each operation.
`flashProgramErase`	Defines waveform for Programming and Erasing pulse for both drain and gate.
`getRes2`	This function returns the resistance of a two-terminal resistor. Voltage `v_force` is forced on the top side of the device; 0 V is forced to the low side. Measure current and reports resistance (V/I).
`pramEndurance`	Runs an endurance test for a PRAM. It runs iterations with a logarithmically spaced number of SET/PULSE loops. Reports DUT resistance after SET/RESET pulse. Also returns the amplitude of the SET current.
`pramSweep`	This function characterizes PRAM devices and produces RI/RV data. A sequence of SET and RESET pulses, followed by the MEASURE pulses, sets and resets the PRAM DUT.
`pulse_test`	Performs pulse testing according to the definition in the nonvolatile memory structure. This function handles all PMU communications and does all nonvolatile memory pulse testing.
`pundEndurance`	This routine performs a device endurance test that runs fatigue pulse trains in between multiple PUND tests. A preliminary PUND test measurment is taken (iteration of 0), followed by the fatigue voltage pulse train. The PUND test is composed of a 17 segment voltage pulse waveform, with two positive pulses to a user-specified Vp followed by two negative pulses to -Vp. Each PUND test calculates P, Pa, U, Ua, N, Na, D, Da, Psw, and Qsw. The fatigue pulse train is made by looping through a 9-segment voltage pulse waveform with one positive pulse to a user-specified V_{fat} and one negative pulse to $-V_{fat}$. The number of times this waveform is repeated between each PUND test is determined by the specified number of loops divided logarithmically by the total number of fatigue pulse trains.

User module	Description
pundTest	This routine performs a pulse V waveform PUND test for FRAM, measuring the full voltage and current waveforms. It also calculates P, Pa, U, Ua, N, Na, D, Da, Psw, and Qsw. The PUND test is composed of a 17-segment voltage pulse waveform with two positive pulses from 0 V to user-specified Vp, followed by two negative pulses to -Vp.
reramEndurance	The reramEndurance routine performs a series of double sweeps using the same parameters used for the single sweeps, as described in the reramSweep routine.
reramForming	This routine slowly ramps a voltage to a specified value while measuring the current constantly to see if the device has formed.
reramFormingCV	This routine slowly ramps a voltage to a specified value while measuring the current constantly to see if the device has formed.
reramSweep	The reramSweep sweep performs a double sweep with a flat section at the peak of each sweep. To test a ReRAM device, choose appropriate values for the two peaks, either positive or negative, and then set the timing you would like to implement.
vt_ext	This function returns the transistor threshold voltage using the maximum Gm method.

OVPControl user library

The user module in the OVPControl user library allows you to set the maximum voltage of the SMU.

SetOVPLevel user module

User module	Description
SetOVPLevel	Sets the 4200-SMU or 4210-SMU overvoltage protection (OVP) maximum voltage. The OVP is an analog circuit that limits the SMU voltage output regardless of what the SMU is sourcing and measuring. Depending on the voltage that the OVP is set to, the SMU clamps the output voltage to one of the built-in voltage limits.

parlib user library

The `parlib` extracts device parameters on bipolar-junction transistors and MOSFETs. Extracted parameters include Beta, resistance, threshold voltage, and V_{ds}-I_d sweeps and V_{gs}-I_d sweeps for MOSFETs.

parlib user modules

User module	Description
`beta`	Measure beta of bipolar transistor at the specified IE and VCB.
`fnddat`	Find data based on an x search or a y search.
`fndtrg`	Decide if TRIGL or TRIGH should be used.
`gamma`	Returns the value of the body-effect parameter gamma obtained from two measurements of the threshold voltage at different substrate bias voltages.
`gm`	Estimate FET conductance (dId/dVg) at V_{ds} and V_{gs}.
`gummel`	This test makes measurements that are similiar to the `gummel` test in the Demo project for 3-terminal pnp BJTs.
`igvg`	This test makes measurements that are similiar to the `ig-vg` test in the Demo project for 4-terminal MOSFETs.
`vceic`	This test makes measurements that are similiar to the `vce-ic` test in the Demo project for 3-terminal BJTs.
`vdsid`	This test makes measurements that are similiar to the `vds-id` test in the Demo project for 4-terminal MOSFETs.
`vgsid1`	This test makes measurements that are similiar to the `vgs-id` test in the Demo project for 4-terminal MOSFETs.
`vtext`	Returns the value of the extrapolated threshold voltage from multiple linear least square fits to the gate characteristics of a FET in the nonsaturated region.

pmuCompulib

The user modules in this library are used to collect and select offset current compensation data.

pmuCompulib user modules

User module	Description
`pmu_Offset_Current_Comp`	Collects offset current compensation data for both channels of the 4225-PMU.

PMU_examples_ulib user library

The user modules in this library are used in the `pmu-dut-examples` project.

The user module in the OVPControl user library (on page 6-392) allows you to set the maximum voltage of the SMU.

PMU_examples_ulib user modules

User module	Description
`PMU_1Chan_Sweep_Example`	This module is a functional programming reference to illustrate the basic commands necessary to perform a pulse I-V (2-level pulse) sweep. It returns voltage and current spot means for pulse amplitude and base by doing a voltage amplitude pulse I-V sweep using one channel of the 4225-PMU.
`PMU_1Chan_Waveform_Example`	This module is a functional programming reference to illustrate the basic commands necessary to perform a pulse I-V (2-level pulse) sweep with waveform capture. It captures a voltage amplitude pulse I-V waveform using one channel of the 4225-PMU. It returns voltage and current samples versus time for a single channel.
`PMU_IV_sweep_Example`	This module is a functional programming reference to illustrate the basic LPT commands that are needed to perform a single Vd-Id sweep. This module performs a voltage amplitude pulse I-V sweep using two channels of a single 4225-PMU. One channel sweeps (drain) while the other uses a fixed pulse amplitude (gate).
`PMU_PulseWaveform_FileSave_Example`	This module allows for a long pulse or time capture (40 s pulse width maximum, 120 s total waveform capture) of an entire pulse to a `*.csv` file using both channels of a single 4225-PMU and the Segment Arb mode. In addition to optionally saving the waveform to a file, a time-averaged version is available in the Analyze sheet.
`PMU_ScopeShot_Example`	Pulse I-V waveform capture using two channels of a single 4225-PMU. The gate channel outputs a pulse train (no change in pulse base or amplitude) while the drain channel outputs a swept pulse amplitude.
`PMU_SegArb_Example`	This module configures multi-segment waveform generation (Segment Arb) on two channels using a single 4225-PMU. It measures and returns the waveform data (V and I compared to time, no spot means).
`PMU_SegArb_ExampleB`	This module configures multi-segment waveform generation (Segment Arb) on two channels using a single 4225-PMU. It measures and returns either waveform (V and I compared to time) or spot mean data for each segment that has measurement enabled.

User module	Description
PMU_SegArb_ExampleFull	This module configures multi-sequence, multi-segment waveform generation (Segment Arb) on two channels using a single 4225-PMU and measures and returns either waveform (V and I versus time) or spot mean data for each segment that has measurement enabled. It also provides a voltage bias by controlling one SMU.
PMU_SMU_Sweep_Example	This user module is an example of how to use the PMU with a SMU. For example, you could use this module to compare performing a test using a PMU to performing that test with a SMU. This user module is based on the module PMU_IV_Sweep_Example.

pmuulib user library

The pmuulib user library contains user modules for configuring the 4225-RPM for the designated PMU channel. The next table lists and briefly describes the user modules.

pmuulib user modules

User module	Description
RPM_configure	This user module configures the 4225-RPM for the designated PMU channel.
RPM_switch	This user module has been deprecated. Use the LPT command rpm_config() (on page 14-139) for any RPM mode switching.

PRBGEN user library

The PRBGEN user library provides test modules to initialize the prober, move to the next site or subsite in the wafer map of the prober, make or break contact between the probes and the wafer, and get the X position and Y position of the prober. It allows Clarius to control all supported probers in the same manner. Clarius projects that use PRBGEN work with any prober supported by Keithley Instruments.

The user modules in the PRBGEN user library are provided as actions in Clarius.

PRBGEN user modules

User module	Clarius action	Description
PrChuck (on page F-14)	prober-contact	Directs the prober to have the probe pins make contact with the wafer or separate the pins from the wafer.
PrInit (on page F-13)	prober-init	Initializes the prober with die size, first coordinate (X and Y), units (mm or mils), and mode information.
PrMovNxt (on page F-17)	prober-move	In learn mode, the PrMovNxt command causes the prober to move to the next site after inking.
PrSSMovNxt (on page F-15)	prober-ss-move	In learn mode, the PrSSMovNxt command causes the prober to move to the next subsite after inking.

QSCVulib user library

The QSCVulib user library provides a user module to do quasistatic C-V sweeps.

QSCVulib module

User module	Clarius test	Description
meas_qscv	ramprate-cvsweep	This test uses two SMUs with preamplifiers to do a quasistatic C-V sweep. The 4200-PA Preamplifiers are required because this test involves sourcing and measuring current in the picoamp range. The SMUs source current to charge the capacitor and measure the voltage, time, and discharge current.

RPM_ILimit_Control user library

The RPM_ILimit_Control user library provides a user module for short-term calibration of the 4225-RPM current clamp. It also provides user modules that support the calibration user module, but which should not be set individually.

RPM_ILimit_Control user modules

User module	Description
Do_RPM_ILimit_Cal	Performs a short-term calibration of the I-clamp of properly-equipped 4225-RPMs.
Get_RPM_ILimit_DAC_Value	Do not set individually. This is used by Set_RPM_ICompliance.
isLimitSupported	Do not set individually. Used by Do_RPM_ILimit_Cal, Get_RPM_ILimit_DAC_Value, Set_RPM_ICompliance, and OpenLimit.
OpenLimit	Do not set individually.
Set_RPM_ICompliance	Do not set individually.

utilities_ulib

The utilities_ulib user library provides a user module to add delays.

utilities_ulib user module

User module	Description
Delay_second	Enter delay time in seconds.

van der Pauw user library

The `vdpulib` user library contains user modules for measuring the surface resistivity and volume resistivity of semiconductor material using the van der Pauw (vdp) technique.

vdpulib user modules

User module	Description
`hall_coefficient`	This module is used to determine the Hall coefficient (RH) and mobility (uH) of a material using four SMUs.
`resistivity_surface`	This module measures surface resistivity using four SMUs.
`resistivity_volume`	This module measures volume resistivity using four SMUs.

VLowFreqCV user library

The `VLowFreqCV` user library contains user modules that are used for very low frequency C-V characterization. The next table briefly describes the user modules.

VLowFreqCV user modules

- `vlfcv_measure`
  Makes a single C-V measurement using two SMUs connected to the device under test (DUT).
- `vlfcv_measure_dual_sweep_bias`
  Performs C-V characterization at multiple DC bias values. This module allows dual sweep, sweeping from a start to stop bias, with one measure point at the stop point before sweeping back to the start point.
- `vlfcv_measure_dual_sweep_bias_fixed_range`
  Performs C-V characterization at multiple DC bias values. This module allows dual sweep, sweeping from the start point to the stop point, with one measure point at the stop point before sweeping back to the start point. It uses a fixed measure range for the SMU for the entire voltage bias sweep. The routine uses the maximum DC bias voltage, `expected_C` and `expected_R` to determine the maximum current for the test and uses this current to set the current measure range for the test.
- `vlfcv_measure_sweep_bias`
  Performs C-V characterization at multiple DC bias values. It makes the same measurements as `vlfcv_measure`, but allows you to make measurements at each point of a linear sweep of the DC bias voltage.
- `vlfcv_measure_sweep_bias_fixed_range`
  Performs C-V characterization at multiple DC bias values. This routine performs the same measurements as `vlfcv_measure`, but allows you to make measurements at each point of a linear sweep of the DC bias voltage. This routine is also similar to `vlfcv_measure_sweep_bias`, except that it uses a fixed current measure range on for the SMU sense for the entire voltage bias sweep. The routine uses the maximum DC bias voltage, `expected_C` and `expected_R` to determine the maximum current for the test and uses this current to set the current measure range for the test.
- `vlfcv_measure_sweep_freq`
  Performs C-V characterization at multiple frequency values. It makes the same measurements as `vlfcv_measure`, but allows you to make measurements at each point of a list sweep of the test frequency.
- `vlfcv_measure_sweep_time`
  Performs C-V characterization a specified number of times, creating a C versus time graph.

wlrlib user library

The user modules in the `wlrlib` user library run linear regression and charge-to-breakdown (Q_{BD}) ramp tests for wafer-level reliability (WLR) testing. These user modules are summarized in the table below.

wlrlib user modules

User module	Description
`llsq1`	Performs simple linear regression.
`qbd_rmpv`	Performs a charge-to-breakdown test using the QBD V-ramp test.
`qbd_rmpj`	Performs a charge-to-breakdown test using the QBD J-ramp test.

For more information, refer to Wafer-Level Reliability Testing (on page L-1).

Winulib user library

The `Winulib` user library provdes user interface routines for operator inputs and prompts, such as abort, retry, and ignore decision prompts.

Winulib user modules

User Module	Clarius Action Name	Description
AbortRetryIgnoreDialog (on page 6-400)	`abortretryignoredialog`	This user module creates a dialog box with Abort, Retry, and Ignore decision prompts.
InputOkCancelDialog (on page 6-402)	`inputokcanceldialog`	This user module creates a dialog box that can prompt for up to four input parameters.
OkCancelDialog (on page 6-404)	`okcanceldialog`	This user module creates a dialog box that provides OK or Cancel decisions.
OkDialog (on page 6-406)	`okdialog`	This user module creates a dialog box that pauses the test sequence to make an announcement (for example, "Test finished") or prompt for an action (for example, connection change).
RetryCancelDialog (on page 6-408)	`retrycanceldialog`	This user module creates a dialog box that presents Retry or Cancel decisions.
YesNoCancelDialog (on page 6-410)	`yesnocanceldialog`	This user module creates a dialog box that contains up to four lines of text and Yes, No, or Cancel decisions.
YesNoDialog (on page 6-412)	`yesnodialog`	This user module creates a dialog box that contains up to four lines of text and Yes and No buttons.

AbortRetryIgnoreDialog user module

This user module creates a dialog box with Abort, Retry, and Ignore decision prompts.

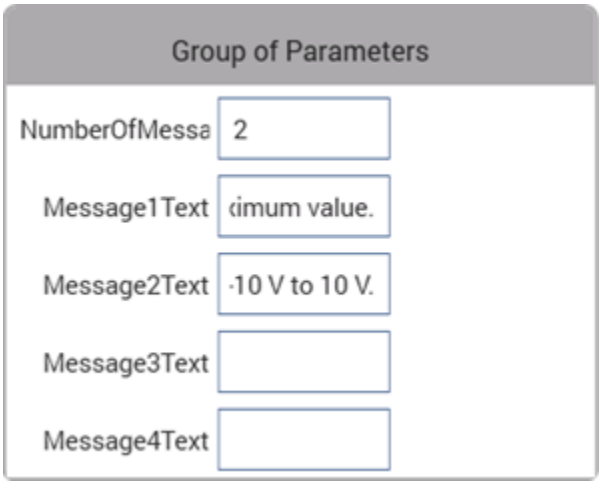
Usage

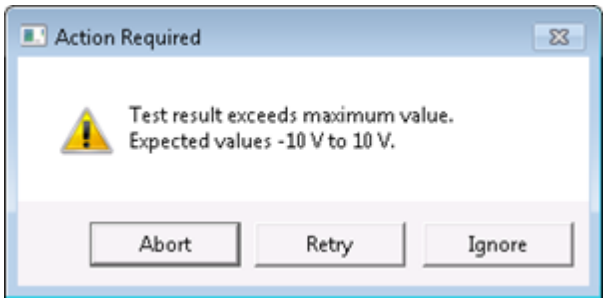
```
status = AbortRetryIgnoreDialog(int NumberOfMessages, char *Message1Text,
    char *Message2Text, char *Message3Text, char *Message4Text);
```

status	Returned values; see **Details**
NumberOfMessages	The number of text lines to display
Message1Text	The text to display on the first line of the dialog box; this line must be less than 40 characters.
Message2Text	The text to display on the second line of the dialog box; this line must be less than 40 characters.
Message3Text	The text to display on the third line of the dialog box; this line must be less than 40 characters
Message4Text	The text to display on the fourth line of the dialog box; this line must be less than 40 characters

Details

You can place up to four lines of text in the dialog box. An example of the entry in Clarius and the resulting dialog box are shown in the following graphics.





Returned values are placed in the Analyze sheet and can be:

- `3`: The Abort button was selected.
- `4`: The Retry button was selected.
- `5`: The Ignore button was selected.
- `-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 Clarius was found. Clarius is not running.

Example

```
status = AbortRetryIgnoreDialog(1, "This is a one line message", "", "", "");
status = AbortRetryIgnoreDialog(4, "Line one", "Line two", "Line three", "Line four");
```

Also see

None

InputOkCancelDialog user module

This user module creates a dialog box that can prompt for up to four input parameters.

Usage

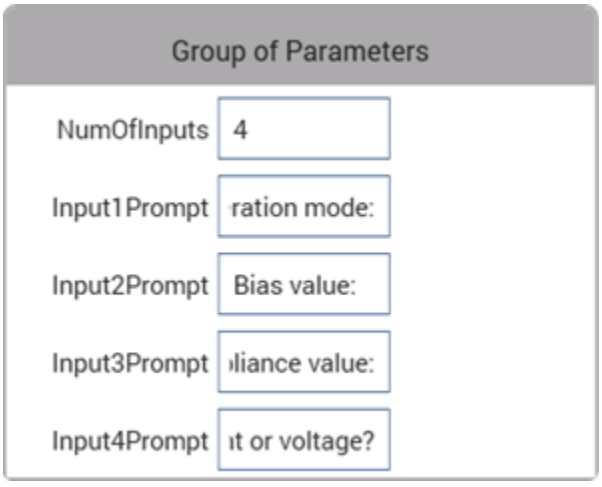
```
status = InputOkCancelDialog(int NumOfInputs, char *Input1Prompt, char *Input1,
   char *Input2Prompt, char *Input2, char *Input3Prompt, char *Input3, char
   *Input4Prompt, char *Input4);
```

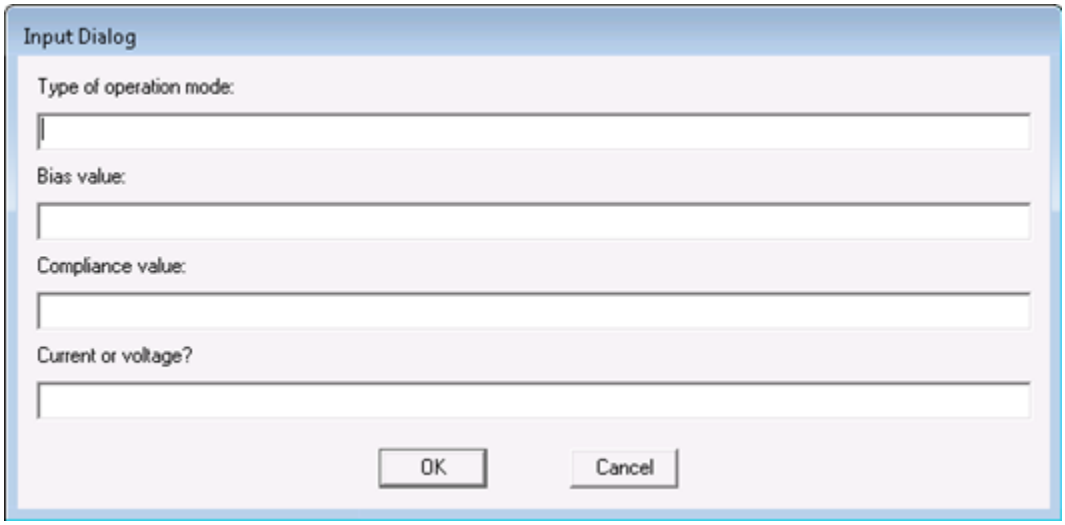
status	Returned values; see **Details**
NumberOfInputs	The number of text lines to display
Input1Prompt	The text to display on the first line of the dialog box; this line must be less than 40 characters.
Input1	A character buffer for the first user input field; any text that the user inputs in the first displayed field is stored here
Input2Prompt	The text to display on the second line of the dialog box; this line must be less than 40 characters.
Input2	A character buffer for the second user input field; any text that the user inputs in the second displayed field is stored here
Input3Prompt	The text to display on the third line of the dialog box; this line must be less than 40 characters
Input3	A character buffer for the third user input field; any text that the user inputs in the third displayed field is stored here
Input4Prompt	The text to display on the fourth line of the dialog box; this line must be less than 40 characters
Input4	A character buffer for the fourth user input field; any text that the user inputs in the fourth displayed field is stored here

Details

`InputOkCancelDialog` displays a dialog box that contains up to four message prompts and four text input fields with OK and Cancel buttons.

There is a separate user-entered prompt message for each input. An example of the entry in Clarius and the resulting dialog box are shown in the following graphics.





Returned values are placed in the Analyze sheet and can be:

- `1`: The OK button was selected.
- `2`: The Cancel button was selected.
- `-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 Clarius was found. Clarius is not running.

Example

```
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);
```

Also see

None

OkCancelDialog user module

This user module creates a dialog box that provides OK or Cancel decisions.

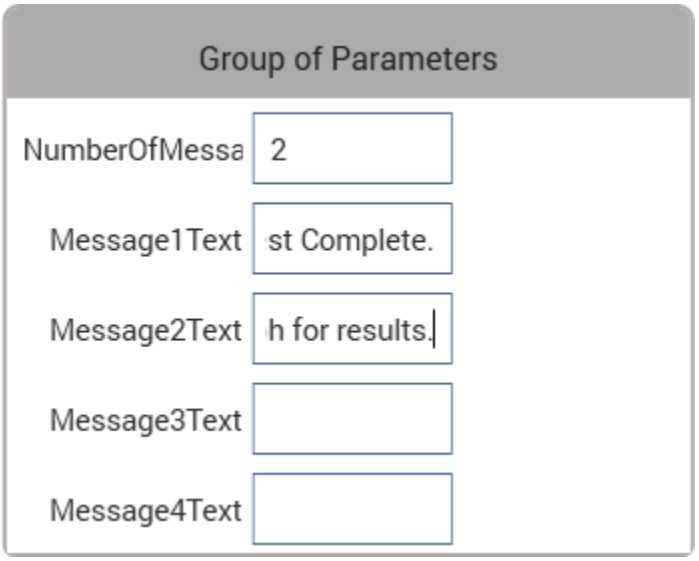
Usage

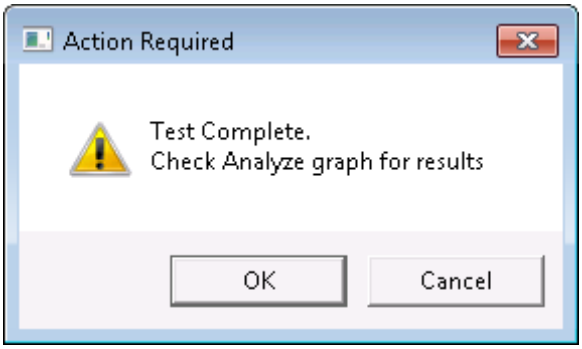
```
status = OkCancelDialog(int NumberOfMessages, char *Message1Text, char *Message2Text,
    char *Message3Text, char *Message4Text);
```

status	Returned values; see **Details**
NumberOfMessages	The number of text lines to display
Message1Text	The text to display on the first line of the dialog box; this line must be less than 40 characters.
Message2Text	The text to display on the second line of the dialog box; this line must be less than 40 characters.
Message3Text	The text to display on the third line of the dialog box; this line must be less than 40 characters
Message4Text	The text to display on the fourth line of the dialog box; this line must be less than 40 characters

Details

`OkCancelDialog` displays a dialog box with up to four text messages with OK and Cancel buttons. Up to four lines of text can be placed in the dialog box. An example of the entry in Clarius and the resulting dialog box are shown in the following graphics.





Returned values are placed in the Analyze sheet and can be:

- `1`: The OK button was selected.
- `2`: The Cancel button was selected.
- `-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 Clarius was found. Clarius is not running.

Example

```
status = OkCancelDialog(1, "This is a one line message", "", "", "");
status = OkCancelDialog(4, "Line one", "Line two", "Line three", "Line four");
```

Also see

None

OkDialog user module

This user module creates a dialog box that pauses the test sequence to make an announcement (for example, "Test finished") or prompt for an action (for example, connection change).

Usage

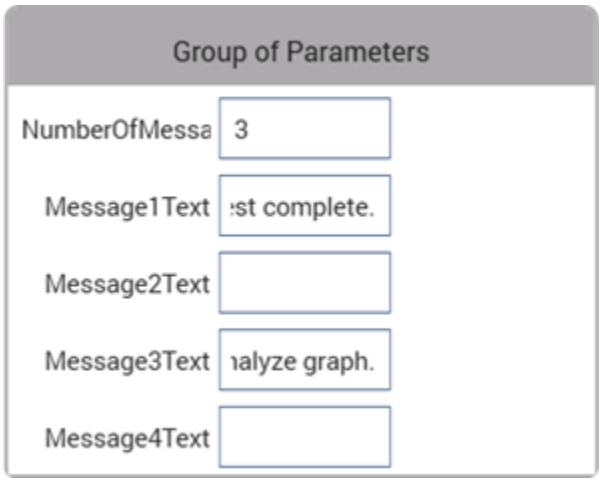
```
status = OkDialog(int NumberOfMessages, char *Message1Text, char *Message2Text, char
    *Message3Text, char *Message4Text);
```

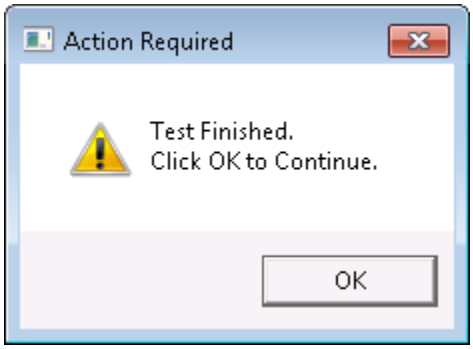
status	Returned values; see **Details**
NumberOfMessages	The number of text lines to display
Message1Text	The text to display on the first line of the dialog box; this line must be less than 40 characters.
Message2Text	The text to display on the second line of the dialog box; this line must be less than 40 characters.
Message3Text	The text to display on the third line of the dialog box; this line must be less than 40 characters
Message4Text	The text to display on the fourth line of the dialog box; this line must be less than 40 characters

Details

Clicking OK continues the test sequence. Up to four lines of text can be placed in the dialog box.

An example of the entry in Clarius and the resulting dialog box are shown in the following graphics.





Returned values are placed in the Analyze sheet and can be:

- `1`: The OK button was selected.
- `-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 Clarius was found. Clarius is not running.

Example

```
status = OkDialog(1, "This is a one line message", "", "", "");
status = OkDialog(4, "Line one", "Line two", "Line three", "Line four");
```

Also see

None

RetryCancelDialog user module

This user module creates a dialog box that presents Retry or Cancel decisions.

Usage

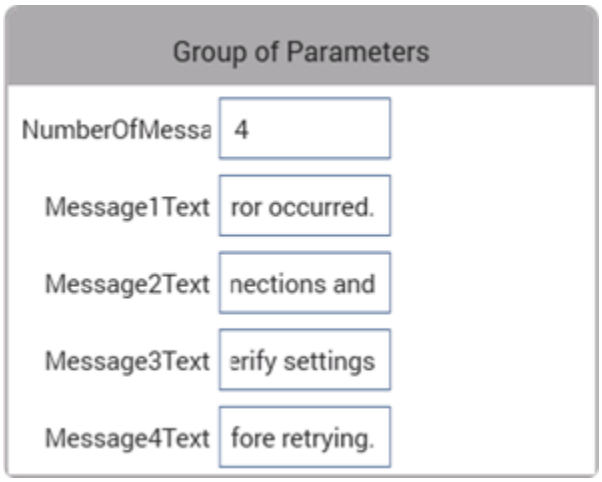
```
status = RetryCancelDialog(int NumberOfMessages, char *Message1Text, char
   *Message2Text, char *Message3Text, char *Message4Text);
```

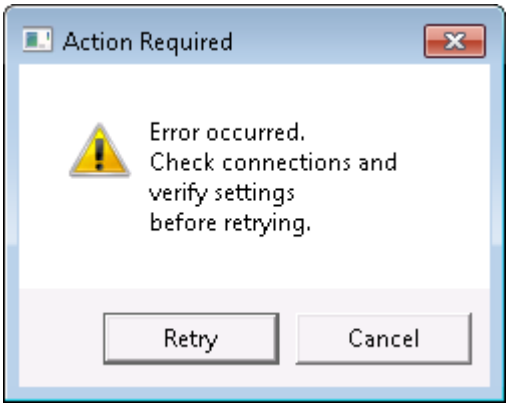
status	Returned values; see **Details**
NumberOfMessages	The number of text lines to display
Message1Text	The text to display on the first line of the dialog box; this line must be less than 40 characters.
Message2Text	The text to display on the second line of the dialog box; this line must be less than 40 characters.
Message3Text	The text to display on the third line of the dialog box; this line must be less than 40 characters
Message4Text	The text to display on the fourth line of the dialog box; this line must be less than 40 characters

Details

`RetryCancelDialog` displays a dialog box that contains up to four lines of text and Retry and Cancel buttons.

An example of the entry in Clarius and the resulting dialog box are shown in the following graphics.





Returned values are placed in the Analyze sheet and can be:

- `2`: The Cancel button was selected.
- `4`: The Retry button was selected.
- `-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 Clarius was found. Clarius is not running.

Example

```
status = RetryCancelDialog(1, "This is a one line message", "", "", "");
status = RetryCancelDialog(4, "Line one", "Line two", "Line three", "Line four");
```

Also see

None

YesNoCancelDialog user module

This user module creates a dialog box that contains up to four lines of text and Yes, No, or Cancel decisions.

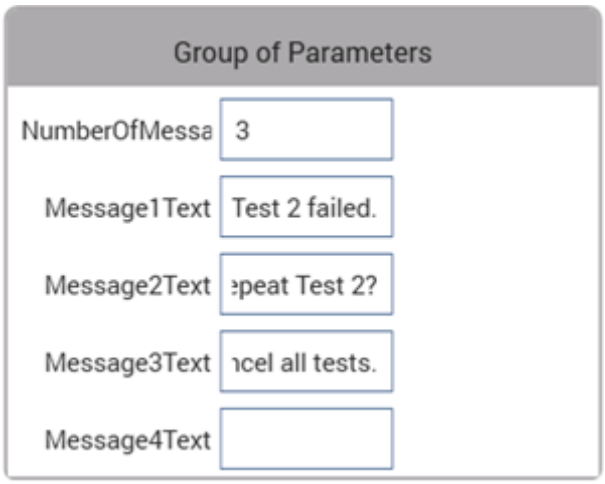
Usage

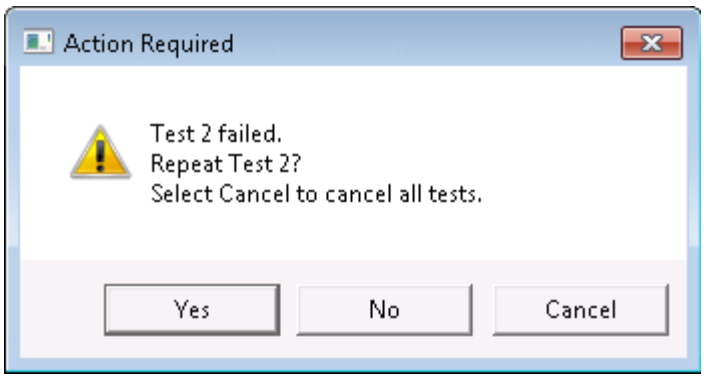
```
status = YesNoCancelDialog(int NumberOfMessages, char *Message1Text, char
   *Message2Text, char *Message3Text, char *Message4Text);
```

status	Returned values; see **Details**
NumberOfMessages	The number of text lines to display
Message1Text	The text to display on the first line of the dialog box; this line must be less than 40 characters
Message2Text	The text to display on the second line of the dialog box; this line must be less than 40 characters
Message3Text	The text to display on the third line of the dialog box; this line must be less than 40 characters
Message4Text	The text to display on the fourth line of the dialog box; this line must be less than 40 characters

Details

An example of the entry in Clarius and the resulting dialog box are shown in the following graphics.





Returned values are placed in the Analyze sheet and can be:

- `2`: The Cancel button was selected.
- `6`: The Yes button was selected.
- `7`: The No button was selected.
- `-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 Clarius was found. Clarius is not running.

Example

```
status = YesNoCancelDialog(1, "This is a one line message", "", "", "");
status = YesNoCancelDialog(4, "Line one", "Line two", "Line three", "Line four");
```

Also see

None

YesNoDialog user module

This user module creates a dialog box that contains up to four lines of text and Yes and No buttons.

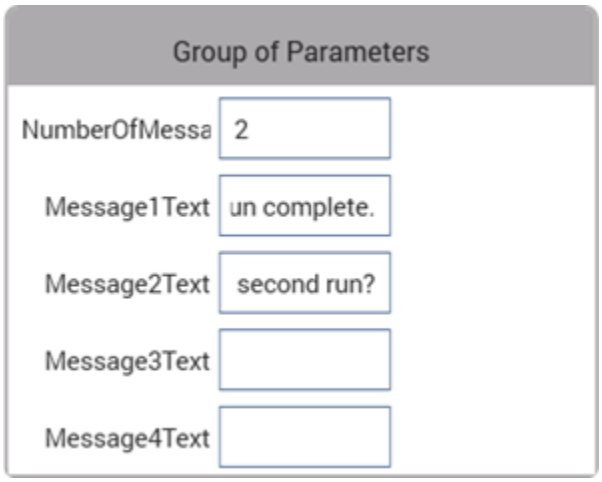
Usage

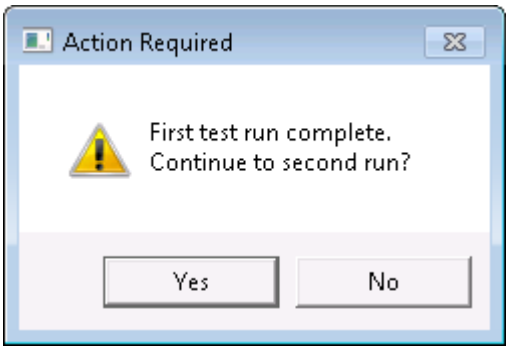
```
status = YesNoDialog(int NumberOfMessages, char *Message1Text, char *Message2Text, char
    *Message3Text, char *Message4Text);
```

status	Returned values; see **Details**
NumberOfMessages	The number of text lines to display
Message1Text	The text to display on the first line of the dialog box; this line must be less than 40 characters
Message2Text	The text to display on the second line of the dialog box; this line must be less than 40 characters
Message3Text	The text to display on the third line of the dialog box; this line must be less than 40 characters
Message4Text	The text to display on the fourth line of the dialog box; this line must be less than 40 characters

Details

An example of the entry in Clarius and the resulting dialog box are shown in the following graphics.





Returned values are placed in the Analyze sheet and can be:

- `6`: The Yes button was selected.
- `7`: The No button was selected.
- `-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 Clarius was found. Clarius is not running.

Example

```
status = YesNoDialog(1, "This is a one line message", "", "", "");
status = YesNoDialog(4, "Line one", "Line two", "Line three", "Line four");
```

Also see

None

Demo Project overview

The Demo Project includes common DC I-V, C-V, and pulse I-V tests for MOSFETs, BJTs, resistors, diodes, and capacitors. These tests serve as examples and are intended to be copied and modified to work for your own devices. All test parameters in the Demo Project were written for standard discrete parts but can be modified for use with other discrete devices or devices on a semiconductor wafer. These tests demonstrate how to configure tests in the Configure pane, how to use Formulator functions for common mathematical calculations and return them to the data sheet, and how to plot the data in a variety of ways.

The Demo project opens when you first start the Clarius application.

The top portion of the project tree for the Demo project is shown in the following graphic.

Figure 404: Demo project (default) in the project tree

4-terminal n-MOSFET tests

By default, the following tests use three source-measure units (SMUs) and one ground unit (GNDU), as shown in the following figure. You can also use four SMUs, one for each device-under-test (DUT) terminal.

Descriptions of the 4-terminal n-MOSFET tests

vds-id	This test generates the standard family of drain current versus drain voltage curves on a FET. For each gate voltage step, the test sweeps the drain voltage and measures the resulting drain current. This test uses either three or four SMUs that are connected to the gate, drain, source, and bulk terminals of the FET.
vtlin	Uses a linear curve fit to find the threshold voltage (V_t) of a MOSFET from the generated drain current versus gate voltage data. This test uses three or four SMUs connected to the gate, drain, source, and bulk terminals of the MOSFET.
subvt	Executes an I-V sweep and calculates the subthreshold voltage (sub-V_t) of a MOSFET and plots drain current versus gate voltage. This test uses three or four SMUs connected to the gate, drain, source, and bulk terminals of the MOSFET.
vgs-id	This test extracts the threshold voltage (V_t) and maximum transconductance (G_m) parameters from a sweep of the drain current versus gate voltage. This test uses either three or four SMUs connected to the gate, drain, source and bulk terminals of a MOSFET.
ig-vg	Measures the gate leakage current of the MOSFET as a function of the sweeping gate voltage. The test determines the gate leakage resistance using a linear line fit.
cv-nmosfet	Measures the capacitance as a function of the gate voltage between the gate terminal and the drain, source, and bulk terminals tied together. Several parameters are extracted, including the flatband capacitance, doping density, flatband voltage, and oxide thickness.
pulse-vds-id	Uses CH1 and CH2 of a PMU to generate a pulse I-V drain family of curves. CH1 outputs a pulse step output to the gate. CH2 outputs a pulsed drain voltage sweep and measures the drain current.
waveform-meas	Uses the waveform capture mode of the PMU to show the time-based response of the drain current and drain voltage of a MOSFET. CH1 outputs a single pulse to the gate. CH2 captures the transient response of the drain current and drain voltage.

3-terminal NPN BJT tests

The tests for this device require three SMUs.

Descriptions of the 3-terminal NPN BJT tests

vce-ic	As the base current is stepped, this test measures the drain current at each point of the drain voltage sweep. Three SMUs are connected to the base, collector, and emitter terminals of the BJT.
gummel	Generates a Gummel plot as it measures both the base current and collector current of a BJT. The currents are measured as a function of the base-emitter voltage. Three SMUs are connected to the base, collector, and emitter terminals of a BJT.
vcsat	At a constant base current, this test plots the collector current as a function of the collector voltage. The data is used by the Formulator to calculate the collector saturation voltage ($V_{ce(sat)}$). Three SMUs are connected to the base, collector, and emitter terminals of a BJT.

Resistor tests

The following tests use two SMUs. It is also possible to use one SMU and the GNDU.

Descriptions of the 2-wire resistor tests

`res2t`	Calculates the average resistance from an I-V sweep of sourcing voltage and measuring current. This test uses two SMUs on either side of the resistor. You can also use one SMU and GNDU if set properly in the Configure pane.
`pulse-resistor`	The resistor is connected between one channel of the PMU and the PMU ground. Applies a pulse voltage sweep to the resistor during the test while the current is measured.
`pulse-high-resistance`	The resistor is connected between CH 1 and CH 2 of the PMU (RPMs). CH1 applies a pulse sweep to the resistor during the test while the current is measured by CH2. The current is derived by averaging 4 pulses. The Formulator multiples the current of CH2 by -1.

Diode tests

Descriptions of the diode tests

`vfd`	While a forward-bias voltage sweep is applied to a diode, this test measures the anode current and plots the data on a semi-log scale. A linear line fit is used to derive the slope of the line.
`vrd`	Sweeps the reverse bias voltage and measures the resulting current of a diode. This test uses two SMUs on either side of the diode. You can also use one SMU and GNDU if they are set properly in the Configure pane. For very low current measurements, it is recommended that you use a SMU with the preamp option.
`cv-diode`	Measures the junction capacitance as a function of an applied voltage sweep. The depletion depth (W) and the doping density (N) are calculated as a function of the C-V data.
`pulse-diode`	Applies a pulse voltage sweep to the anode of a diode and measures the resulting anode current. A single channel of the PMU is used to make the measurement.

Capacitor tests

Descriptions of the capacitor tests

`cap`	Charges a capacitor at a constant current and measures the voltage as a function of time.
`cv-cap`	Measures the capacitance as a function of voltage on a capacitor. The noise is calculated using the standard deviation of the data.
`pulse-cap`	Applies a single voltage pulse to a capacitor and measures the resulting current. A single channel of the PMU is used to make the measurement. One end of the capacitor is connected to PMU1 and the other end is connected to PMU GND.

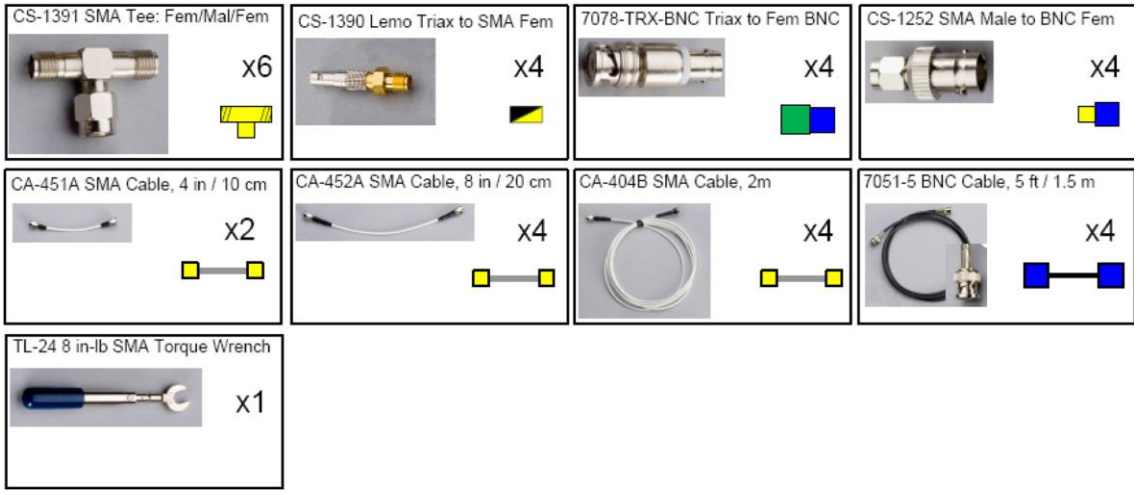
Testing flash memory

Clarius includes several projects that you can use to test floating gate transistors (NOR, NAND) and other types of nonvolatile memory.

To use the flash memory tests, you will need:

- Two pulse cards (four pulse channels).
- At least two SMUs. If your system does not include switching, it is best to have four SMUs to match the number of pulse channels to connect to a three- or four-terminal device. Either 4200-SMUs or 4210-SMUs with SMU preamplifiers removed can be used.
- Interconnecting cables and adapters, shown in the table below.
- 8 in. lb. torque wrench for tightening the SMA connections.

Figure 405: Recommended parts for flash memory testing



Recommended interconnect parts for flash memory testing

Qty.	Description	Comment	Keithley Model Number
6	SMA tee, female – male – female	Trigger, combine SMU and pulse channels	CS-1391
4	LEMO triaxial to SMA adapter	Adapt SMU Force output to SMA for signal interconnect	CS-1390
4	3-slot male triaxial to female BNC adapter	Convert BNC cabling to triaxial for prober or switch matrix connection	7078-TRX-BNC
4	SMA male to BNC female adapter	Adapt Tee to BNC for cabling from instrument to probe manipulators	CS-1252
2	10.8 cm (4.25 in.) white SMA cables	Interconnect for triggering	CA-451A
4	20.3 cm (8 in.) white SMA cables	Interconnect between pulse card and SMU signals	CA-452A
4	2 m (6.6 ft) white SMA cables		CA-404B
4	1.5 m (5 ft) BNC cable	Connect to probe manipulators	7051-5

This configuration permits independent source and measure for each terminal in a typical 4-terminal floating gate transistor.

Flash connection guidelines

All interconnects on instrument chassis are white SMA cables. Cables from the instrument to device are BNC coaxial, except for the direct SMU4 connection, which is black triaxial. Use triaxial to BNC adapters (if necessary) to connect to probe manipulators.

The trigger interconnects are white SMA cables. Cables from the instrument to device are BNC coaxial for the pulse card channels and triaxial for the SMUs. Use triaxial to BNC adapters (if necessary) to connect to probe manipulators.

To test on-wafer devices, there are various ways to connect the SMA cables to the probe manipulators. For the direct connect method or switch method, adapters convert the BNC cabling to the triaxial connector to be compatible with many types of probe manipulators.

Use the supplied torque wrench to tighten each connection as it is assembled. Always connect and torque adapter/cable assemblies before attaching the assembly to the instrument cards.

CAUTION

Non-axial stress on the bulkhead connectors on the SMU or pulse cards could cause damage to the cards installed in the 4200A-SCS chassis. Pre-torque the connections to prevent this damage.

To remove the LEMO triaxial-to-SMA adapter from a SMU, pull on the knurled silver portion of the connector to release the latches from the SMU connector.

CAUTION

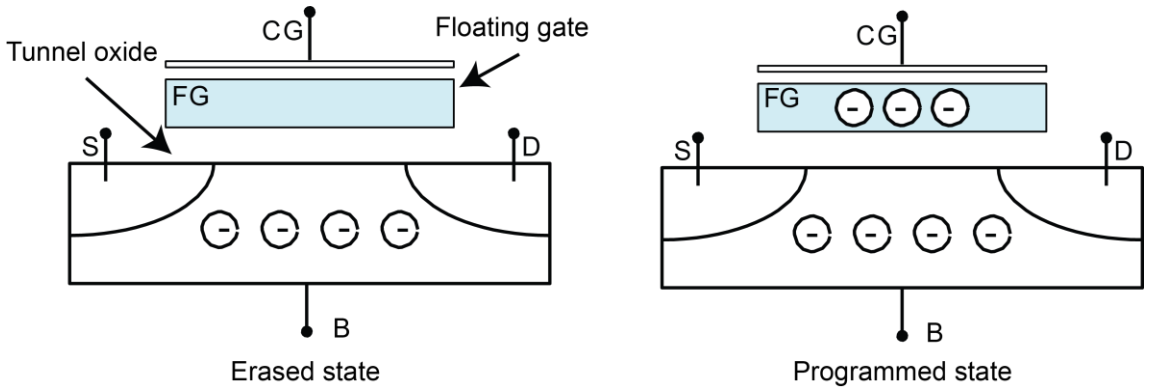
Failure to fully disengage the LEMO adapter latches may result in damage to the adapter and the SMU.

Programming and erasing flash memory

A floating gate (FG) transistor is a modified field-effect transistor with an additional floating gate. The FG transistor is the basic storage structure for data in nonvolatile memory. The floating gate stores charge, which represents data in memory.

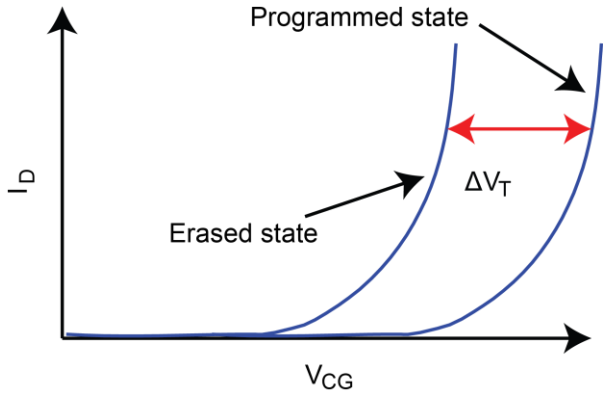
The control gate (CG) reads, programs, and erases the FG transistor. The presence of charge on the gate shifts the voltage threshold (V_T) to a higher voltage, as shown in the following figure.

Figure 406: Cross section of a floating gate transistor in the erased and programmed states



The control gate (CG) reads, programs, and erases the FG transistor. The presence of charge on the gate shifts the voltage threshold (V_T) to a higher voltage, as shown below.

Figure 407: Graph of shifted voltage threshold, VT, due to stored charge on floating gate on a 1 bit (2 level) cell



The Flash transistors tests consist of two parts:

1. Pulse waveforms that program or erase the DUT
2. DC measurements are made to determine the state of the device

This implies switching between two conditions:

1. Pulse resources are connected to the DUT
2. Pulse resources are disconnected and the DC resources are connected to the DUT

The pulses are used to move charge to or from the floating gate. There are two different methods to move charge:

1. Tunneling
2. Hot charge injection (HCI)

The tunneling method is commonly known as Fowler-Nordheim (FN) tunneling, or quantum tunneling, and is a function of the electric potential across the tunneling oxide. HCI is considered a damage mechanism in non-floating gate transistors. HCI is a method that accelerates charges by applying a drain-source field, and then the charges are directed into the floating gate by a gate voltage.

NOTE

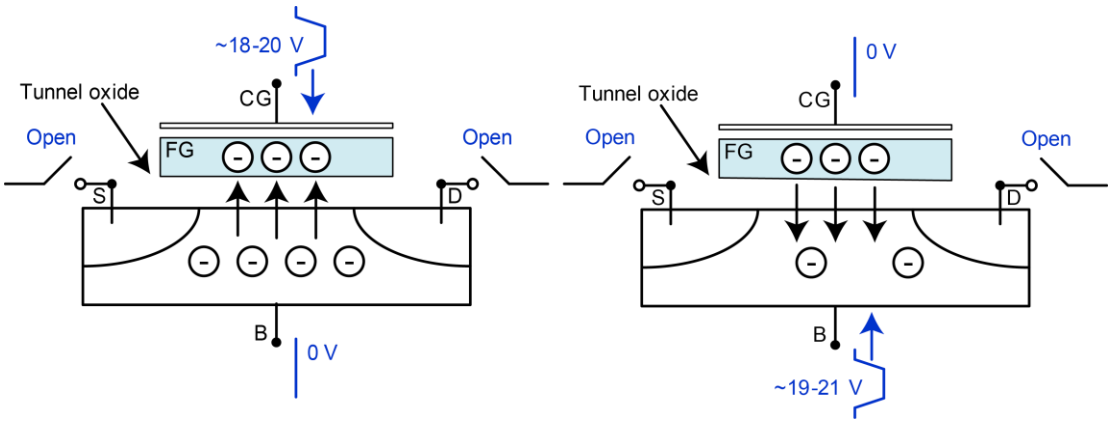
The drain and source are not connected to any test instrumentation.

There are many other ways to provide similar electric fields and balance performance across a variety of parameters: program or erase speed, retention longevity, adjacent cell disturbance, endurance, and others.

The following figure shows examples of tunneling to move charge to and from the FG.

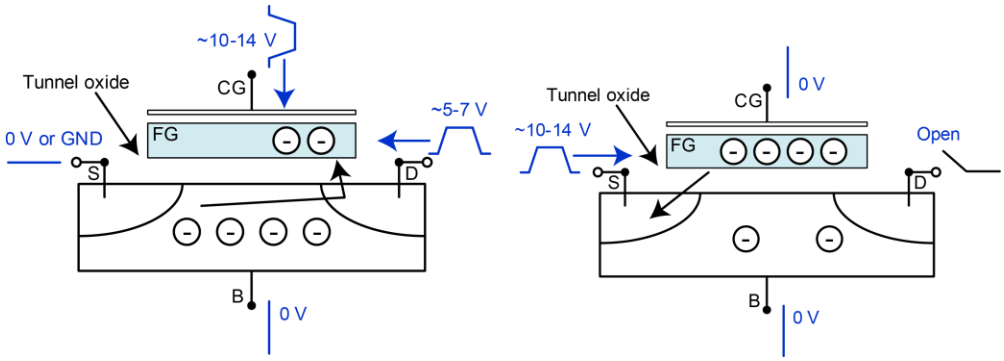
- The electric field and the preferred direction of electron flow are indicated by the black arrows.
- The signal applied to each device terminal is indicated by the blue text and blue features.

Figure 408: Fowler-Nordheim tunneling program and erase



The following figure shows examples of moving charge using HCI. These conditions are examples with approximate voltage values. The pulse width and pulse height will vary depending on device structure and process details.

Figure 409: Hot charge injection (HCI) program and erase



The flash projects support two methods for switching between the pulse and measure phases of the typical flash memory test.

The first is the typical method, using a switch matrix to route the pulse or DC signals to the DUT. Using the switch matrix is more complicated, but provides flexibility for certain tests and test structures that use arrays. The second method uses the on-card isolation relays on both the SMUs and the pulse cards to configure a simpler setup without the external switch matrix.

Because both the SMUs and the pulse cards have isolation relays on the cards, you can configure a simpler setup without the external switch matrix. The advantage of the simpler setup is lower cost. However, the switch matrix provides lower current measurement performance and the flexibility necessary for testing arrays of test structures.

To determine the state of the device, you do a Vg-Id sweep, then perform a calculation to find the voltage threshold (V_T). The shift in V_T represents a change in the amount of charge stored in the floating gate, which indicates the state of the cell from fully programmed (**1**) to fully erased (**0**).

Figure 410: Block diagram of an example flash test setup using a switch matrix

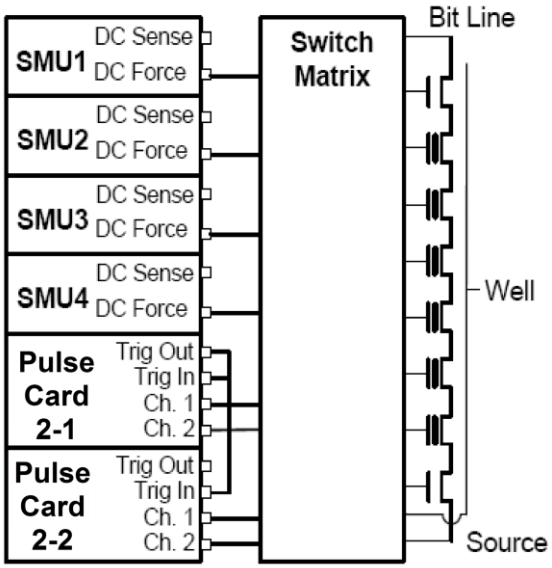
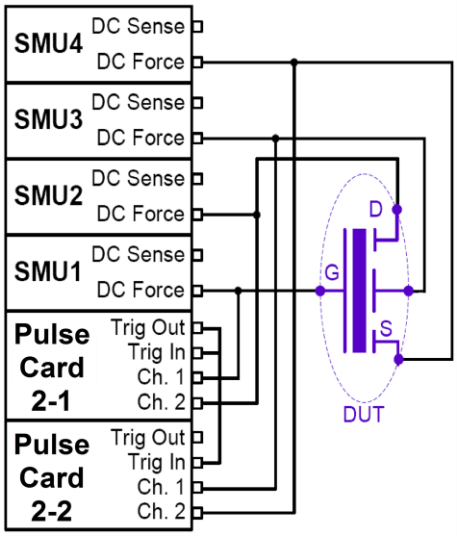


Figure 411: Block diagram of a flash test setup without using a switch matrix (direct connect)



The pulse waveforms are a program pulse, an erase pulse, or a waveform made up of both program and erase pulses. All of these waveforms are implemented by using the Segment Arb capability. For more information about waveforms, refer to Pulse Source-Measure Concepts (on page 5-71). The following waveforms are examples of the different methods and voltage levels for programming and erasing.

Figure 412: Program pulse waveforms for a floating gate DUT with separate pulse waveforms for the DUT gate, drain, source, and bulk

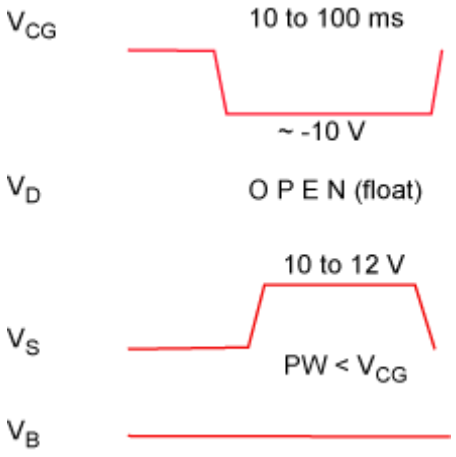
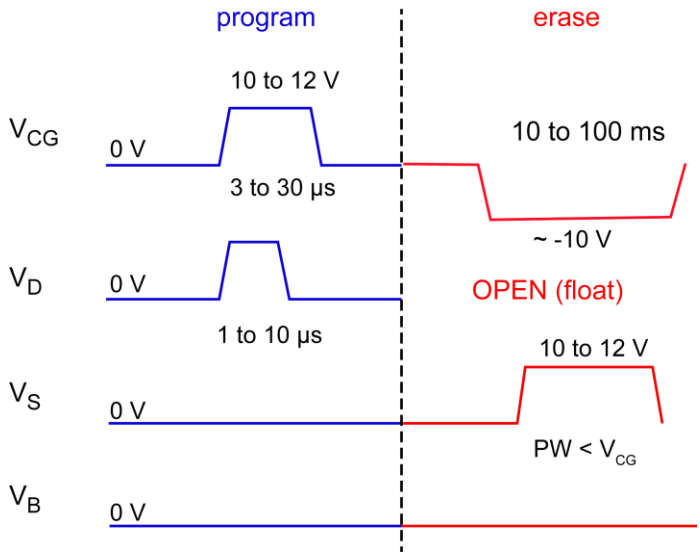
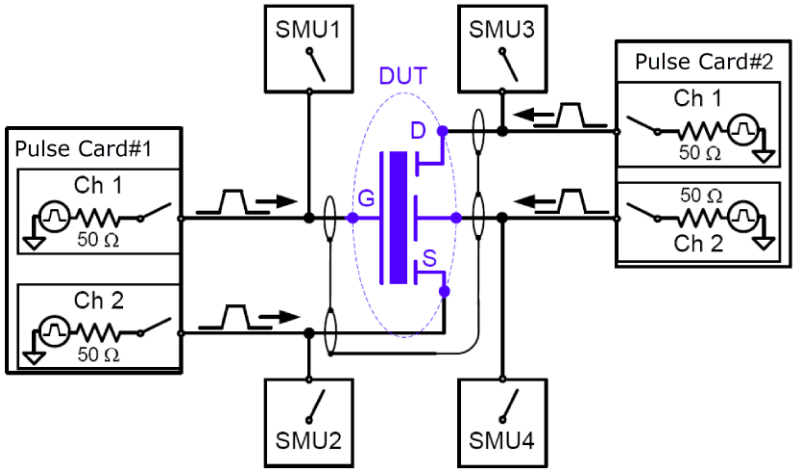


Figure 413: Program + Erase pulse waveforms for a floating gate DUT, with separate pulse waveforms for the DUT gate, drain, source, and bulk



The block diagram for the flash setup is shown in the following figure. To reconfigure from the pulse stress to DC measure phases, activate the switches on the SMU and pulse cards. During the pulse program/erase phase, the relays in the pulse channels are closed and the relays in the SMUs are open. For the DC measure phase, the opposite is true.

Figure 414: Basic schematic of flash testing without a switch matrix

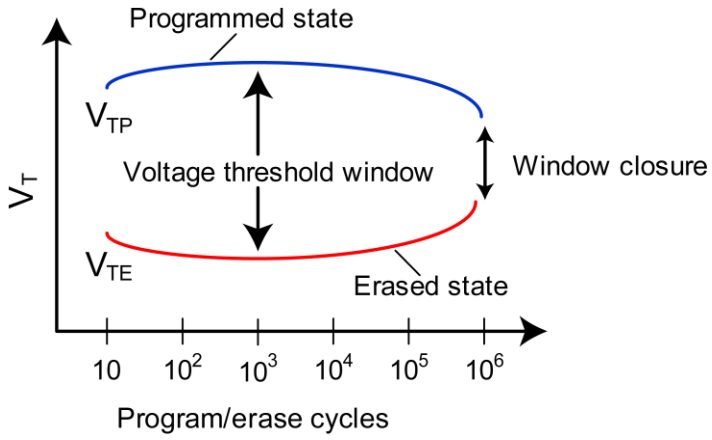


Endurance testing

Endurance testing stresses the DUT with a number of Program+Erase waveform cycles, and then periodically measures both the voltage threshold in the programmed state (VTP), as well as the voltage threshold of the erased state (VTE). These tests to determine the lifetime of the DUT, based on the number of Program+Erase cycles withstood by the device before a certain amount of shift, or degradation, in either the VTP or VTE. The endurance test is performed a set number of program and erase cycles while periodically measuring V_T for both the programmed and erased state.

This figure shows typical degradation on a NOR cell for both VTP and VTE as the number of applied program/erase cycles increases.

Figure 415: Example results of voltage threshold shift in an Endurance test on a NOR flash cell



Connections for endurance testing - no switch matrix

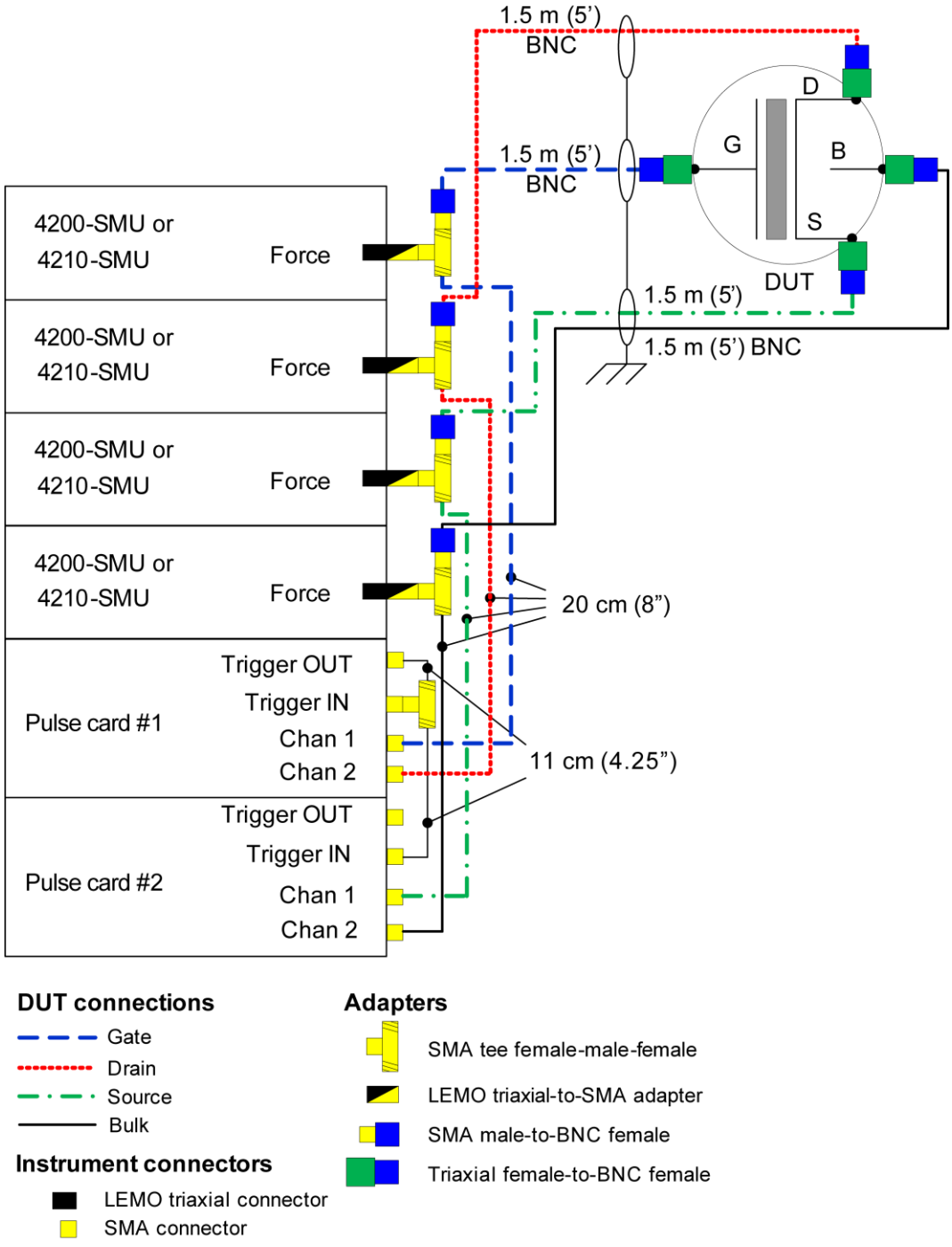
For a direct connect configuration, the minimum number of pulse channels is equal to the number of DUT terminals that need to be simultaneously pulsed, including terminals that must change from connected to disconnected, or open, states for either the program or erase condition.

The following connection configuration does not require a switch matrix. It provides four channels of pulse and four SMUs to permit full characterization of single (non-array) NVM DUT. This connection method is used for both the initial program/erase investigation and endurance testing of a directly-connected DUT.

NOTE

All interconnects on instrument chassis are white SMA cables. Cables from the instrument to device are BNC coaxial. Use triaxial to BNC adapters if necessary to connect to probe manipulators.

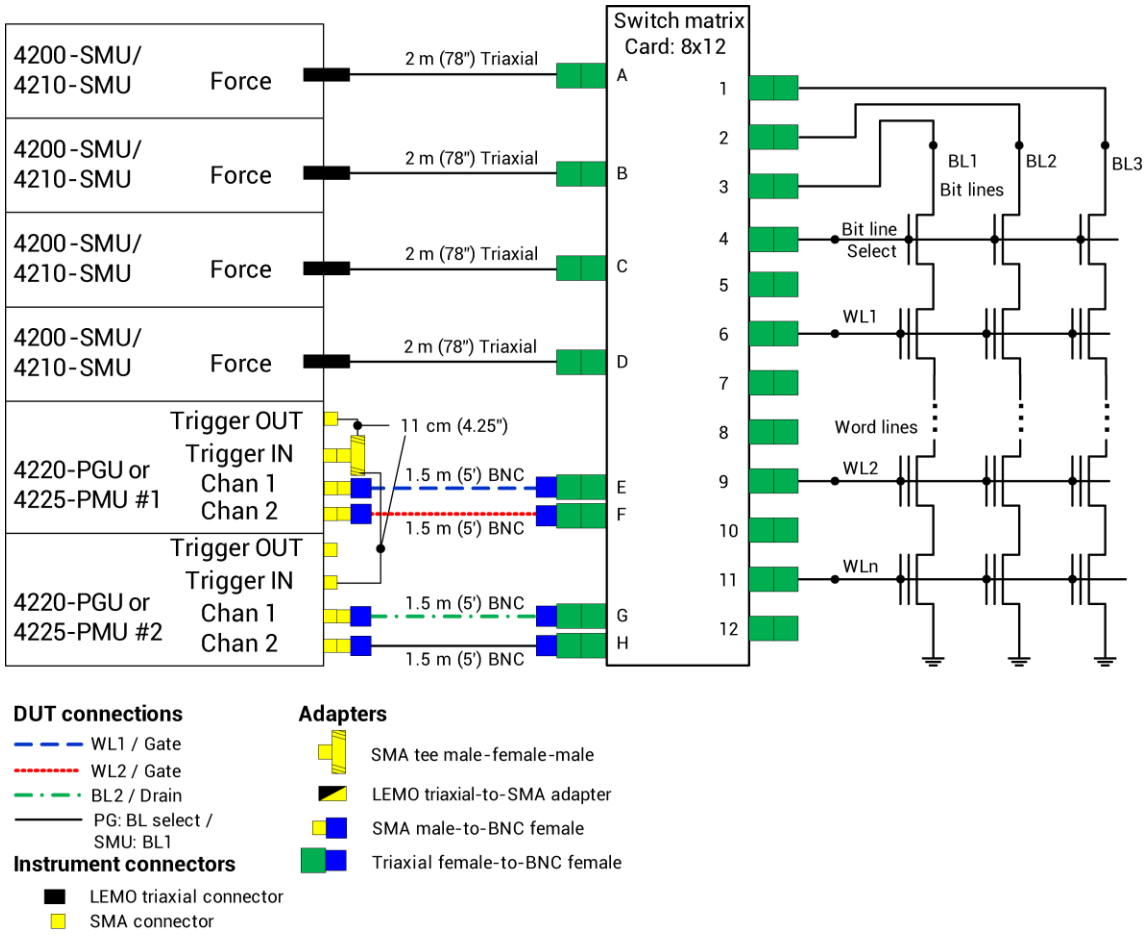
Figure 416: Flash connections - program erase and endurance testing using direct connection to a single stand-alone 4-terminal device



Connections for endurance testing - switch matrix

A switch matrix is recommended for testing array test structures for endurance or disturb.

Figure 417: 4200-900_Flash Switch connections - characterization endurance or disturb testing



Disturb testing

The purpose of the Disturb test is to pulse stress a device in an array test structure, then make a measurement, such as V_T, on a device adjacent to the pulsed device.

The goal is to measure the amount of V_T shift in adjacent cells, either in the programmed or erased states, when a nearby device is pulsed with either a program, erase, or program and erase waveform.

The typical measurement is a V_T extraction based on a Vg-Id sweep, but you can configure any type of DC test. This test is similar to the endurance test, except that the pulsing and measuring are performed on adjacent devices.

The solid-line blue circle indicates the cell to be pulse stressed, and the dotted-line red circles are the adjacent memory cells that will be disturbed by the stressing. This is the stress / measure process:

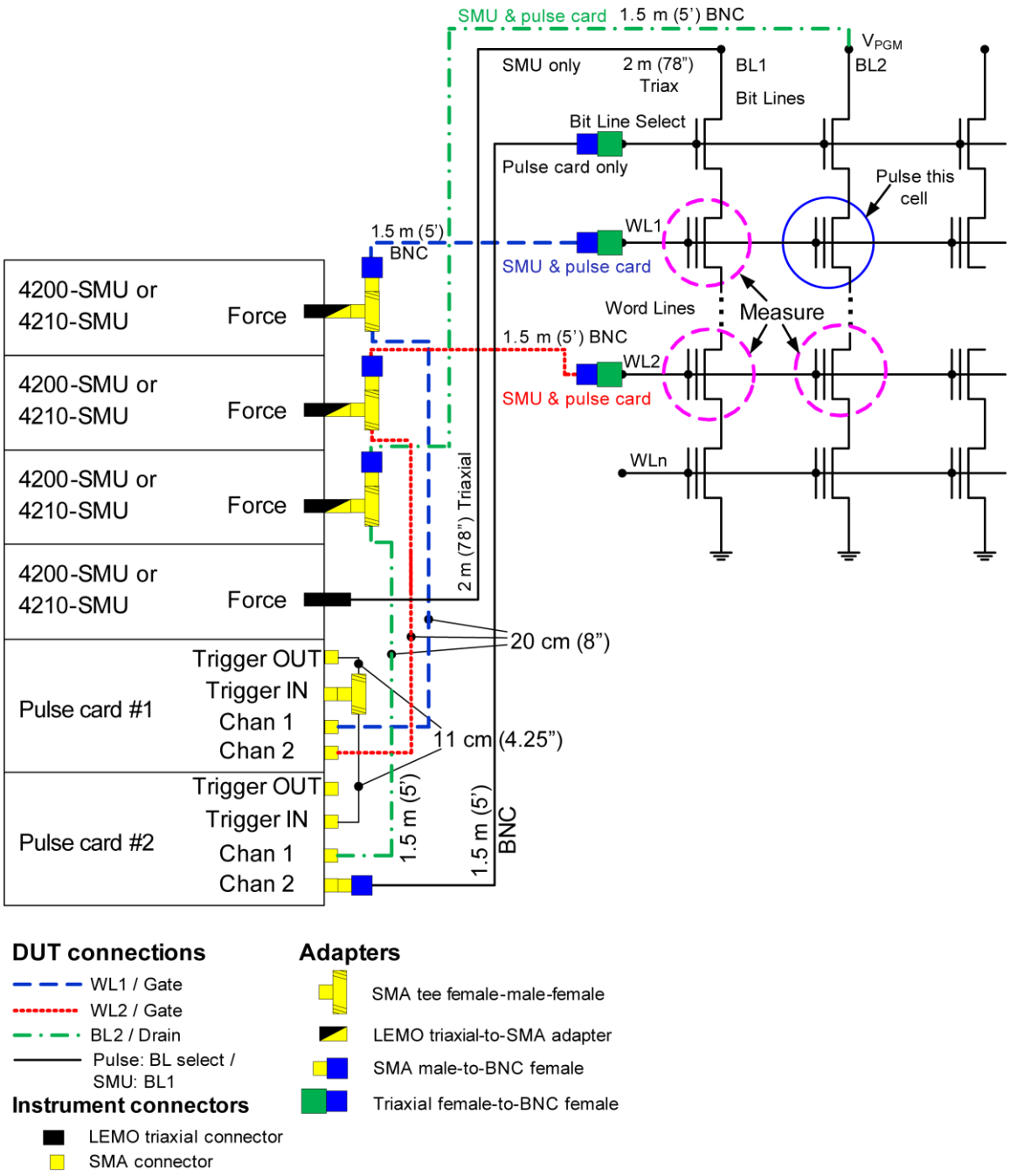
- **Initial test conditions** – SMU4 outputs a DC voltage to turn on the control devices for the array. This connects instrumentation at the top of array to the flash memory cells. SMU2 and SMU3 are set to output 0 V. This ensures that only the Cell 2 is turned on during pulse stressing.
- **Pulse stressing** – The output relay for SMU1 is opened, and the gate and drain of Cell 2 are pulse stressed by Pulse Card 1 (ch 1) and Pulse Card 2 (ch 1).
- **Disturbed cell testing** – The outputs for the pulse cards are turned off and their output relays are opened. SMU1 and SMU2 are then used to perform a DC Vg-Vd sweep on Cell 1 to determine V_T.

Connections for disturb testing

The following figure shows the connections used for disturb testing. A switch matrix is recommended for testing array test structures for both endurance or disturb. However, you can do a limited test of an array structure without using a switch matrix.

The following figure also shows connection to an array test structure, where one of the four SMU+pulse card channels was split. This provides a total of five test signals to provide the minimum necessary channels for the select pins (Bit Line Select, Bit Lines 1 and 2), to the pulse DUT (circled in blue), and the measure DUTs (circled in dashed purple). This configuration allows for pulsing one DUT while performing disturb measurements on the three DUTs labeled Measure.

Figure 418: Flash direct DUT connections - Disturb testing



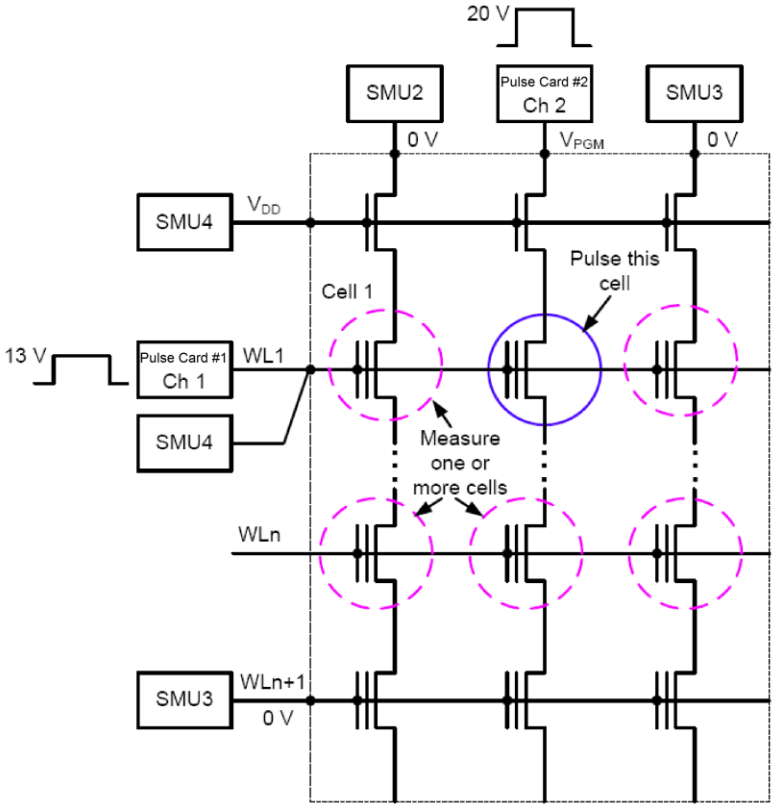
Using a switch matrix

A limitation of the no-switch, direct connect test configuration shown in the figure below is that only three devices can be measured. The test would have to be manually reconfigured or re-cabled to test other devices.

Without a switch matrix, the number of adjacent cells that can be measured is limited. Therefore, it is recommended that a switch matrix be used for disturb testing.

Using a switch matrix allows the flexibility of routing pulse and DC signals without having to make connection changes. Also, this type of structure uses a multi-pin probe card, which provides an additional opportunity for mapping test resources to DUT pins. For example, a SMU can be shared across multiple device terminals where the required voltage is the same.

Figure 419: Disturb testing - configuration to test a single device



Use KPulse to create and export Segment Arb waveforms

This example demonstrates how to create a program and erase waveform using the KPulse flash example file.

You will export the waveforms, which you can import into the Clarius Configure pane for the subsite when Segment Stress/Measure Mode is selected.

NOTE

Each segment pulse waveform must have the same total time. The minimum programmed time for any segment is 20 ns (20E-9 s), but actual output waveform performance is determined by the channel output capability.

To use KPulse to create and export Segment Arb waveforms:

1. If they are open, close Clarius and KCon.
2. Open **KPulse**.
3. Select **File > Load Setup**.
4. Select **Kpulse_Flash_Example_01.kps**.
5. Select **Open**.
6. Select **Edit Segarb**. This opens the edit dialog box for both channels. You can use copy and paste to copy information between the channels.
7. Note that the trigger is set to `1` in the first and fifth segments. These are the first segments in the program and erase pulses in a typical two pulse program/erase waveform. It is recommended that you set trigger to `1` for the first segment of each pulse in a waveform.
8. For each unique waveform, export each to a file. Refer to Exporting Segment Arb waveform files (on page 11-9) for the procedure.

Enter Segment Arb values into UTM array parameters

For tests that are based on user modules, such as the `program`, `erase`, and `fast-program-erase` tests, you need to define the Segment Arb waveforms by entering values into arrays for the tests. These tests are based on user modules. The Segment Arb waveforms have been partially pre-defined to reduce the number of parameters you need to enter.

NOTE

The sign of the PulseVoltages array determines whether the pulse is positive (usually for a `program` pulse) or negative (usually for an `erase` pulse).

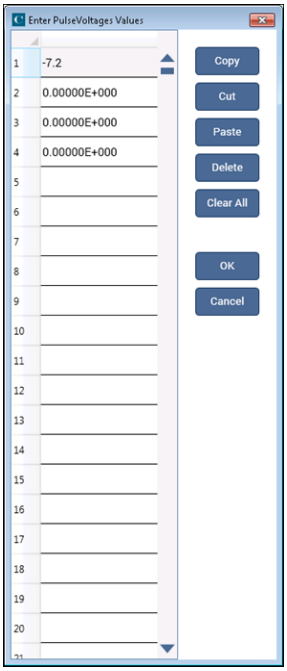
NOTE

The number of parameters and number of pulse channels in the test must be the same. The period of each pulse waveform must be the same. If you accidentally enter a value in a field, delete the value (do not use 0).

To enter Segment Arb values:

1. In Clarius, select the project.
2. Select the test.
3. Select **Configure**.
4. Select the **Enter Values** button for each array. The Enter List Values dialog box opens.

Figure 420: Enter Value dialog box for arrays



5. Enter the values.
6. Select **OK**.
7. For any stress and measure loop tests, such as `endurance` or `disturb`, use KPulse to define and export the waveform files, then import waveforms into the Subsite Stress Properties. Note that if the same waveform is required in the test and the Subsite Stress Properties, you must make sure the waveform information is the same for both.

Refer to the Clarius Help pane for information on the other parameters for each test.

Direct connections to single DUT

Cabling instructions for direct connections to a single DUT are below. Also refer to the figure in Connections for endurance testing - no switch matrix (on page 6-425).

NOTE

In all of the following steps, apply sufficient torque using an 8 in. lb. torque wrench.

These instructions are compatible with the following memory projects:

- `flash-nand`
- `flashdisturb-nand`
- `flashendurance-nand`

Make SMA cable connections to the pulse card:

1. Set up the 4200A-SCS. Refer to Connections and configuration (on page 2-1).
2. Connect a 10.8 cm (4.25 in.) SMA cable to either end of an SMA tee.
3. Connect one of the SMA cables to TRIGGER OUT of the pulse card in the lowest numbered slot.
4. Connect the SMA tee to TRIGGER IN of the first pulse card.
5. Connect the SMA tee to TRIGGER IN of the second pulse card (which should be to the immediate left of the card in the lowest numbered slot).
6. If your system includes four pulse cards, connect the cables and tees as described above to the adjacent cards. There should be three SMA tees used to connect the triggering across the four cards.

Make SMA-to-BNC connections to the pulse and SMU cards:

1. Connect an SMA-to-BNC adapter to one of the female connectors on an SMA tee.
2. Connect the tee to a triaxial-to-SMA adapter.
3. Connect a 20.3 cm (8 in.) SMA cable to the remaining SMA female connector.
4. Connect a 1.5 m (5 ft) black BNC cable to the BNC connection.
5. Repeat these steps three times.
6. Use one of these cable assemblies to connect the SMA to CHANNEL 2 of the pulse card in the left-most slot (pulse card in the slot with the highest number).
7. Carefully insert the LEMO triaxial connector into the Force connector on the SMU in slot 4.
8. Route the BNC cable from SMU4 to the DUT terminal bulk connection. Connect a triaxial-to-BNC adapter, if necessary.
9. Connect the cable to the probe manipulator.

10. Use another one of the cable assemblies to connect the SMA to CHANNEL 1 of the pulse card in the left-most slot (pulse card in the slot with the highest number).
11. Carefully insert the LEMO triaxial connector into the Force connector on the SMU in slot 3.
12. Route the BNC cable from SMU3 to the DUT terminal source connection. Connect a triaxial-to-BNC adapter, if necessary.
13. Connect the cable to the probe manipulator.
14. Use another one of the cable assemblies to connect the SMA to CHANNEL 2 of the pulse card in the right-most slot (pulse card in the slot with the lowest number).
15. Carefully insert the LEMO triaxial connector into the Force connector on the SMU in slot 2.
16. Route the BNC cable from SMU2 to the DUT terminal drain connection. Connect a triaxial-to-BNC adapter, if necessary.
17. Connect the cable to the probe manipulator.
18. Use the remaining cable assemby to connect the SMA to CHANNEL 1 of the pulse card in the right-most slot (the pulse card in the slot with the lowest number).
19. Carefully insert the LEMO triaxial connector into the Force connector on the SMU in slot 1.
20. Route the BNC cable from SMU1 to the DUT terminal gate connection. Connect a triaxial-to-BNC adapter, if necessary.
21. Connect the cable to the probe manipulator.

Direct connections to array DUT for disturb testing

Cabling instructions for direct connections to an array DUT are below. These instructions are compatible with the following projects:

- `flashdisturb-nand`
- `flashendurance-nand`

NOTE

In all of the following steps, apply sufficient torque using an 8 in. lb. torque wrench.

Refer to the drawing in Connections for disturb testing (on page 6-428).

Make the SMA connections:

1. Set up the 4200A-SCS. Refer to Connections and configuration (on page 2-1).
2. Connect a 10.8 cm (4.25 in.) SMA cable to either end of an SMA tee.
3. Connect one of the SMA cables to TRIGGER OUT of the pulse card in the lowest numbered slot.
4. Connect the SMA tee to TRIGGER IN of the first pulse card.
5. Connect the SMA tee to TRIGGER IN of the second pulse card (which should be to the immediate left of the card in the lowest numbered slot).
6. If your system includes four pulse cards, connect the cables and tees as described above to the adjacent cards. There should be three SMA tees used to connect the triggering across the four cards.

Make SMA-to-BNC connections to the DUT arrays:

1. Connect an SMA-to-BNC adapter to a 1.5 m (5 ft) black BNC cable.
2. Connect SMA end of the SMA-to-BNC adapter to CHANNEL 2 of the pulse card in the left-most slot (the pulse card in the slot with the highest number).
3. Route the BNC cable to the DUT array Bit Line Select connection. If necessary, use a triaxial-to-BNC adapter.
4. Connect the cable to the probe manipulator.
5. Insert the LEMO end of a black LEMO triaxial to 3-slot triaxial cable into the Force connection on the left-most SMU in slot 4.
6. Route the triaxial cable from SMU4 to the DUT array BL1 connection.
7. Connect the cable to the probe manipulator.

Make SMA-to-BNC connections to the pulse and SMU cards:

1. Connect an SMA-to-BNC adapter to one of the female connectors on an SMA tee.
2. Connect the tee to a triaxial-to-SMA adapter.
3. Connect a 20.3 cm (8 in.) SMA cable to the remaining SMA female connector.
4. Connect a 1.5 m (5 ft) black BNC cable to the BNC connection.
5. Repeat these steps twice.
6. Connect the SMA of one of the cable assemblies to CHANNEL 2 of the pulse card in the left-most slot (the puse card in the slot with the highest number).
7. Carefully insert the LEMO triaxial connector into the Force connector on the SMU in slot 4.
8. Route BNC cable from SMU4 to the DUT array WL2 terminal. If necessary, use a triaxial-to-BNC adapter.
9. Connect the cable to the probe manipulator.
10. Connect the SMA of one of cable assemblies to CHANNEL 1 of the pulse card in the left-most slot (the pulse card in the slot with the highest number).
11. Carefully insert the LEMO triaxial connector into the Force connector on the SMU in slot 3.
12. Route the BNC cable from SMU3 to the DUT array BL2 connection. If necessary, connect a triaxial-to-BNC adapter.
13. Connect the cable to the probe manipulator.
14. Connect the SMA of one of the cable assemblies to CHANNEL 2 of the pulse card in the right-most slot (the pulse card in the slot with the lowest number).
15. Carefully insert the LEMO triaxial connector into the Force connector on the SMU in slot 2.
16. Route the BNC cable from SMU2 to the DUT array WL2 connection. If necessary, connect a triaxial-to-BNC adapter.
17. Connect the cable to the probe manipulator.
18. Connect the SMA of one of the cable assemblies to CHANNEL 1 of the pulse card in the right-most slot (the pulse card in the slot with the lowest number).
19. Carefully insert the LEMO triaxial connector into the Force connector on the SMU in slot 1.
20. Route the BNC cable from SMU1 to the DUT array WL1 connection. If necessary, connect a triaxial-to-BNC adapter.
21. Connect the cable to the probe manipulator.

Embedded computer policy

CAUTION

If you install software that is not part of the standard application software for the 4200A-SCS, the non-standard software may be removed if the instrument is sent in for service. Back up the applications and any data related to them before sending the instrument in for service.

CAUTION

Do not reinstall or upgrade the Microsoft® Windows® operating system (OS) on any 4200A-SCS unless the installation is performed as part of authorized service by Keithley Instruments. Violation of this precaution will void the 4200A-SCS warranty and may render the 4200A-SCS unusable. Any attempt to reinstall or upgrade the 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 you must not attempt to reinstall or upgrade the operating system, you can restore the hard drive image (complete with the operating system) using the Acronis True Image OEM software tool, described in System-level backup and restore software (on page 12-8).

Section 7

Keithley Configuration Utility (KCon)

In this section:

Keithley Configuration Utility (KCon)

You use the Keithley Configuration Utility (KCon) to manage the configuration of the Keithley Instruments 4200A-SCS and all external system components supported by the Clarius+ applications. You can add, configure, and remove supported switch matrices, external GPIB instruments, and probe stations from the system configuration using KCon. KCon also provides diagnostic and troubleshooting functions.

If you add an instrument to the system, or change connections between the measurement instrumentation and the switch matrix, you must run KCon to update the system configuration.

NOTE

Before starting KCon, ensure that Clarius is not running. You cannot modify the system configuration while Clarius is running.

KCon main window

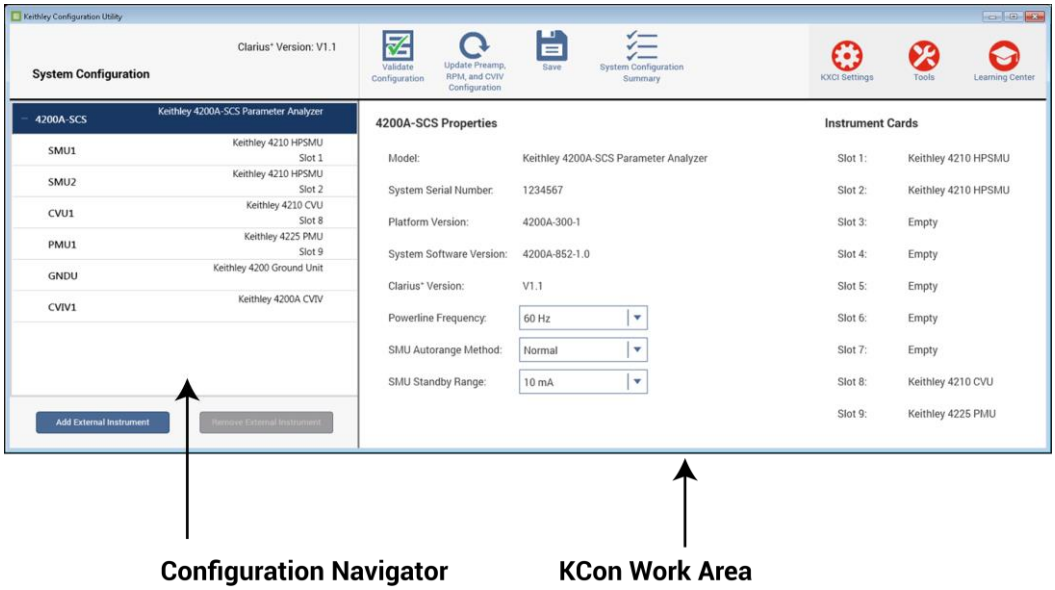
When KCon starts, the KCon main window shown below opens. The left pane is the Configuration Navigator, and the right pane is the Work Area.

The Configuration Navigator provides a tree view of all instruments and equipment in the 4200A-SCS system configuration. To expand or minimize the tree, select the plus (+) and minus (–) symbols, respectively.

The KCon work area displays information about the selected instrument. Each instrument in the system configuration has properties. Selecting a KI System Configuration node in the Configuration Navigator displays a summary of the entire system configuration in the Work Area.

You may need to use the scroll bar to view the entire system configuration.

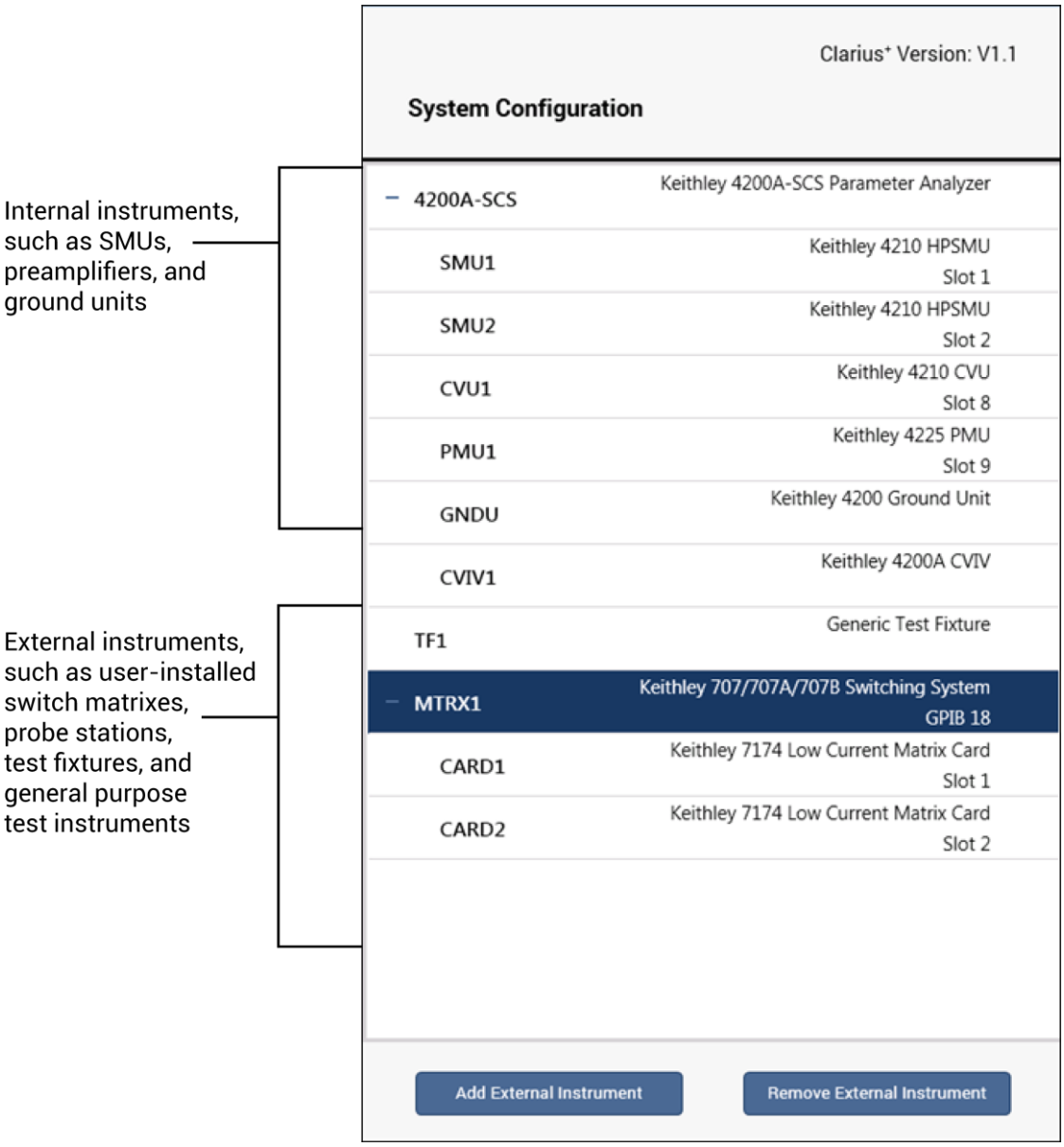
Figure 421: KCon main window



Configuration Navigator

The Configuration Navigator contains each component in the system configuration. Selecting a component in the Configuration Navigator displays the properties associated with the selected component in the Work Area. The figure below shows a typical system configuration with the Configuration Navigator listings expanded.

Figure 422: Configuration Navigator view of the system configuration



Add an external instrument

To add a supported external instrument to the system configuration, select **Add External Instrument**. The supported instrument categories are:

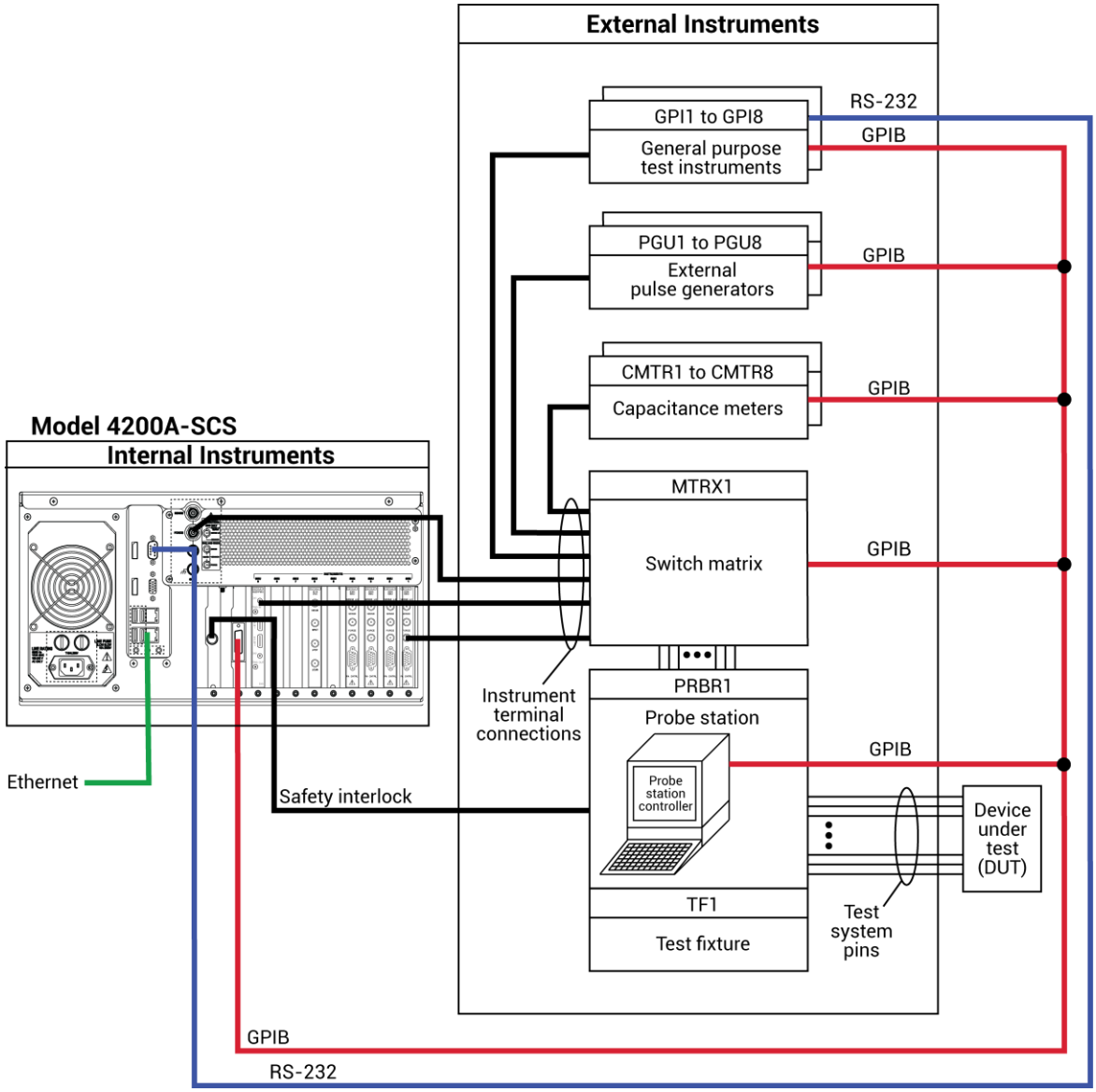
- Switch Matrix
- Capacitance Meter
- Pulse Generator
- Probe Station
- Test Fixture
- General Purpose Test Instrument

Supported external instrumentation and equipment are controlled by Clarius user test modules (UTMs). Keithley Instruments provides libraries of user modules for each supported external instrument (refer to Supported external equipment (on page 7-6)). You can modify these libraries or create your own using the Keithley User Library Tool (KULT).

For additional information regarding external instrumentation and user modules, refer to Keithley User Library Tool (KULT) (on page 8-1).

The following figure shows the relationship between internal and external instrumentation and illustrates each instrument category.

Figure 423: Example system connections



To add an external instrument:

1. Select **Add External Instrument**.
2. Select the instrument. You can only make one selection at a time.
3. Select **OK**. The instrument is added to the Configuration Navigator.

Supported external equipment

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 the table below.

Supported external equipment

Category	Instrument	User library	Additional information
Switch matrix[1]	Series 700 Switching System	`Matrixulib`	Using Switch Matrices (on page A-1)
Capacitance meter[2]	Keithley Instruments Model 590 CV Analyzer	`KI590ulib`	Using a Model 590 CV Analyzer (on page B-1)
	Keithley Instruments Model 595 Quasistatic C-V Meter	`KI595ulib`	Model 595 Quasistatic C-V 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 (on page D-1)
	Keysight Technologies 4284 or 4980 LCR Meter	`HP4284ulib`	Using a Keysight 4284/4980A LCR Meter (on page C-1)
	Keysight Model 4294 LCZ Meter	`HP4294ulib`	HP4294ulib user library reference (on page 6-386)
Pulse generator[3]	Keithley Instruments Model 3402 Pulse Generator	`ki340xulib`	Series 3400 Pulse Pattern Generators User's Manual (document number 3400S-900-01B)
	Keysight Model 8110A Pulse Generator	`HP8110ulib`	Using a Keysight 8110A/81111A Pulse Generator (on page E-1)
Probe station[4]	Karl Suss MicroTec Model PA-200 semiautomatic probe station	`PRBGEN`	Suss MicroTec PA-200 Prober (on page G-1)
	Micromanipulator Model 8860 semiautomatic probe station	`PRBGEN`	Micromanipulator 8860 Prober (on page H-1)
	Manual probe station and fake probe station	`PRBGEN`	Using a Manual or Fake Prober (on page I-1)
	Cascade Summit-12000 probe station	`PRBGEN`	Cascade Summit-12000 Prober (on page J-1)
	Signatone CM500 Prober	`PRBGEN`	Signatone CM500 Prober (on page K-1)
	MPI Probers TS2000, TS2000-DP, TS2000-HP, TS2000-SE, TS3000, and TS3000-SE.	`PRBGEN`	Using an MPI Probe Station (on page M-1)
Test fixture	Keithley Instruments Model LR:8028 Component Test Fixture	Not applicable	Connections and Configuration (on page 2-1)
	Keithley Instruments Model 8007 Semiconductor Test Fixture	Not applicable	Connections and Configuration (on page 2-1)
	Generic test fixture	Not applicable	Connections and Configuration (on page 2-1)

Category	Instrument	User library	Additional information
General purpose test Instruments[5]	Any GPIB controlled or RS-232 controlled instrument or equipment	Created by user	

[1] The 4200A-SCS supports the Keithley Instruments Series 700 Switching System. Only one switch matrix can be in the system configuration at a time.
[2] You can add up to eight supported capacitance meters to the system configuration.
[3] The 4200A-SCS supports a maximum of 16 pulse-generator unit (PGU) channels. You can combine single- and dual-channel pulse generators. 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, see 4200A-SCS power supply limitations (on page 5-59).
[4] For general information about using a probe station, refer to Using a Probe Station (on page F-1).
[5] The 4200A-SCS supports up to eight general-purpose instruments (GPIs). Two-terminal and four-terminal types may be present in the system configuration simultaneously, but the total number of GPIs must be less than eight.

NOTE

Contact Keithley Instruments for the most recent list of supported external equipment.

Add a driver for unsupported equipment

If you need to control an instrument that is not supported by a standard Keithley Instruments driver library, you can create an external instrument driver.

To learn more about creating external instrument drivers for the 4200A-SCS, refer to the following Keithley Instruments Technical Note:

Use KCon to add equipment to the 4200A-SCS

To use the 4200A-SCS to control an external instrument, you must add that instrument to the system configuration.

The next topics describe general settings for adding equipment. Matrices and probers, however, require additional steps.

For detail if you are adding a switch matrix, refer to Use Switch Matrices (on page A-1).

For detail if you are adding a prober, refer to the information for your prober:

- Suss Micro PA-200 Prober (on page G-1)
- Micromanipulator 8860 Prober (on page H-1)
- Manual or Fake Prober (on page I-1)
- Cascade Summit-120000 Prober (on page J-1)
- Signatone CM500 Prober (on page K-1)

Step 1. Exit Clarius and open KCon

To exit Clarius and open KCon:

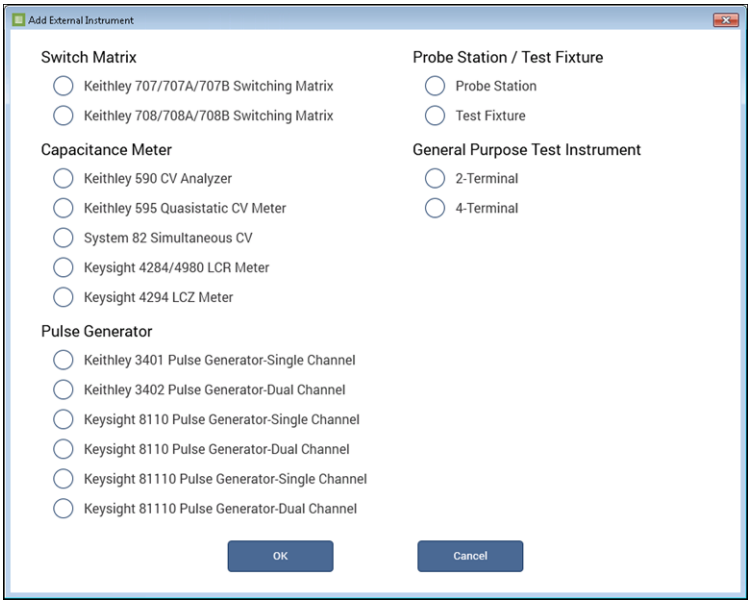
1. Exit Clarius.
2. On the Windows desktop, select the **KCon** icon.

Step 2. Add the equipment

To add equipment to the system configuration:

1. At the bottom of the Configuration Navigator, select **Add External Instrument**.
2. Select the specific instrument. An example of the selection of the System 82 Simultaneous CV Capacitance Meter is shown in the figure below.
3. Select **OK** to add the instrument to the Configuration Navigator.

Figure 424: Add external instrument



For descriptions of the options for your equipment, refer to Instrument properties and connections (on page 7-16).

Step 3. Set GPIB addresses

The GPIB address setting in KCon must match the actual GPIB address of the instruments in the system. See the documentation for the instrument to determine the GPIB Address.

The address for some instruments is briefly displayed during the power-on sequence.

To set the GPIB address for an instrument:

1. In the Configuration Navigator, select the instrument.
2. Select the **GPIB Address** from the list. Addresses that are in use are displayed with asterisks (*) next to them. The range of addresses is 0 to 30 (GPIB address 31 is reserved as the 4200A-SCS controller address). 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.

NOTE

You can programmatically read the GPIB address and other instrument properties from the system configuration using the LPT library `getinstattr` function. Proper use of `getinstattr` allows you to develop user libraries that are independent of the configuration. For more information, refer to Keithley User Library Tool (KULT) (on page 8-1).

Step 4. Save configuration

To save the KCon configuration:

Select **Save**.

Step 5. Validate configuration

This step checks communications between the 4200A-SCS and the attached equipment.

To validate configuration:

Select **Validate Configuration**.

The verification report is displayed on the screen.

Remove an external instrument

To remove an external component from the system configuration:

1. Select the component.
2. Select **Remove External Instrument**. A confirmation dialog box is displayed.
3. Select **OK**.

NOTE

You cannot remove internal instruments.

Validate Configuration

Validate Configuration tests the system configuration to determine if there are configuration conflicts or communication problems between the instrumentation and the 4200A-SCS. This test can validate most of the supported internal and external instruments. However, the following instruments are not automatically verified when the configuration is validated:

- Probe stations
- Test fixtures
- General purpose test instruments

When you run Validate Configuration, KCon communicates with the instruments to verify that the physical configuration matches the KCon defined configuration. The report is displayed in the KCon Work Area. If KCon detects configuration conflicts, it displays corresponding error messages in the Work Area.

NOTE

Clarius automatically validates the configuration when it starts up. If Clarius detects conflicts, it displays an error message that instructs you to resolve the conflicts using KCon.

Update preamplifier, RPM, and CVIV Configurations

After adding or removing a preamplifier, RPM, or 4200A-CVIV, you must update the system by selecting the **Update Preamp, RPM, and CVIV Configuration** option.

Update the preamplifier configuration

For more information on adding or removing preamplifiers, refer to Connections and configuration (on page 2-1).

To update the preamplifier configuration:

1. Remove or reconnect a preamplifier to the SMU.
2. Select the **Update Preamp, RPM, and CVIV Configuration** option.
3. Select **Save**.

Update the RPM configuration

The KCon feature Update Preamp, RPM, and CVIV Configuration must be run:

- Whenever an RPM is connected or disconnected from the 4200A-SCS
- Whenever RPM input connections for a SMU, CVU, or PMU are changed

When using interactive test modules (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 Clarius, the system will detect the instrument connections to the RPM. When an ITM is run, Clarius 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:

For more information on adding or removing RPM connections, refer to Model 4225-RPM (on page 5-7).

To update RPM configuration:

1. Turn off system power.
2. Disconnect line power.
3. Connect the correct RPM to the correct PMU channel.
4. If the system contains SMUs: Connect any SMU to any RPM using either one or two triaxial cables. Repeat for all RPMs in the system.
5. If the system contains a CVU: Connect CVU High to one RPM, and CVU Low to the other RPM.
6. Turn on the system power.
7. Start KCon.
8. Select the **Update Preamp, RPM, and CVIV Configuration** option.
9. Select **Save**.
10. Exit from KCon.

If you are using an interactive test module (ITM) for testing, the RPM automatically selects the PMU, SMU, or CVU. If you are using a user test module (UTM), you must select the correct RPM settings. See Control RPM switching (on page 5-12) for more information.

Update the 4200A-CVIV configuration

For more information on adding or removing 4200A-CVIV connections, refer to the “Configure and install the 4200A-CVIV” section of the *Model 4200A-CVIV User’s Manual*.

To update CVIV configuration:

1. Turn off system power.
2. Disconnect the 4200A-CVIV from the 4200A-SCS.
3. Make connections to the 4200-CVIV.
4. Turn on system power.
5. Start KCon.
6. Select the **Update Preamp, RPM, and CVIV Configuration** option.
7. Select **Save**.

Save

Saves changes to the system configuration. If you do not save the configuration changes, KCon returns to the last saved configuration when you exit.

System Configuration Summary

System Configuration Summary displays the system configuration. From this display, you can save the configuration or print it.

The sections of the system configuration are:

- System Information: Provides a table containing the 4200A-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.

Select **Save Configuration As** to store the file in `html` format. You can view the `html` file in a web browser.

To print the configuration, select **Print Configuration**.

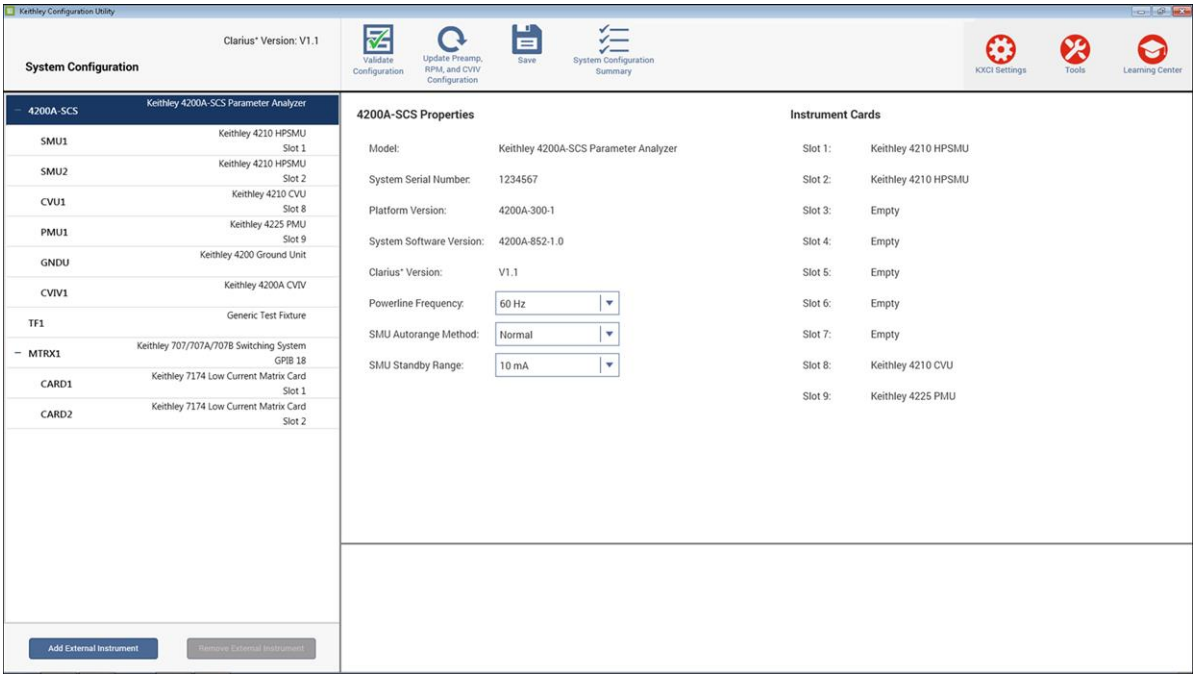
System Configuration properties

Each instrument that is included in the system configuration is displayed in the Configuration Navigator. When you select a Configuration Navigator node, the corresponding attributes or properties are displayed in the Work Area. The following topics describe the properties for each supported instrument.

KI 4200A SCS Properties

When you select KI 4200A SCS in the Configuration Navigator, the Work Area displays the system configuration, as shown in the figure below.

Figure 425: KI System Configuration information



System Properties area

The System Properties area of the Properties displays the 4200A-SCS serial number and other software and platform information. You can change the Powerline frequency (on page 7-15), SMU autorange method (on page 7-15), and SMU standby range (on page 7-15).

Powerline frequency

Set **Powerline Frequency** to **60 Hz** or **50 Hz**. Make sure you set the powerline frequency correctly. If the setting is wrong, the 4200A-SCS cannot properly reject powerline measurement noise.

SMU autorange method

The source measure units (SMUs) in the 4200A-SCS can make smart decisions when changing range. There are many factors the SMUs calculate when they change the range. The SMU autorange method allows you to determine which range change algorithm meets your needs. The available SMU autorange methods are 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 can provide 6-digit precision, but a digit of precision is lost when using the high-speed autorange method. However, 5-digit precision is 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. This is 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 often resolves the problem.

Select the autorange method from the **SMU AutoRange Method** list.

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 0 V, 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 static charges from building up. The default standby range for the SMU is 10 mA.

You may need to select a different standby range 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 seen on the first point of a sweep.

Set standby range by selecting the **SMU Standby Range** from the list.

Instrument Cards area

The Instrument Cards area shows which internal instrument card is installed in each slot.

Instrument properties and connections

Each instrument in KCon has a Properties screen. Properties may include the vendor name, model number, slot number, firmware and hardware versions, serial number, and calibration information for the instrument.

Some instruments include a Run Self Test option. The Self Test button starts the Self Test utility, which runs several internal performance checks to determine if the instrument is operating correctly. When the Self Test process finishes, a pass or fail message is displayed.

If you added external instruments, you can also set the GPIB address from this screen.

If your system configuration includes a matrix, the matrix connections to the measurement terminals are also displayed.

If you added a test fixture, you can select a type of test fixture and define the number of test fixture pins (2 to 72).

NOTE

The number of pins defined in the properties for the test fixture or probe station determines the pins that are available to assign to a switch matrix card column. Make sure the number of pins assigned is appropriate for your system.

NOTE

For supported external instruments, the 4200A-SCS provides user libraries that contain preconfigured data acquisition and control user modules. Refer to Supported external equipment (on page 7-6) for a list of supported equipment and additional information.

The next topics describe general settings for equipment. Matrices and probers, however, require additional steps. For detail if you are adding a switch matrix, refer to Using Switch Matrices (on page A-1).

For detail if you are adding a prober, refer to the information for your prober:

- Suss Micro PA-200 Prober (on page G-1)
- Micromanipulator 8860 Prober (on page H-1)
- Using Manual or Fake Prober (on page I-1)
- Cascade Summit-120000 Prober (on page J-1)
- Signatone CM500 Prober (on page K-1)

Run Self Test

Some instruments include a Self Test option. When you run Self Test, the instrument runs a number of internal performance checks to determine if the instrument is operating correctly. You do not need external equipment to run Self Test.

To run Self Test:

1. Disconnect all connections to the instrument terminals.
2. Select **Run Self Tes**t.
3. Follow the prompts.
4. When the test is complete, a pass or fail message is displayed in the Message area at the bottom of the screen.

Set GPIB addresses

If you added an external instrument, you can set the GPIB address.

In the GPIB Address list, addresses that are in use are displayed with asterisks (*) next to them. You can set addresses from 0 to 30 (GPIB address 31 is reserved as the 4200A-SCS controller address). 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.

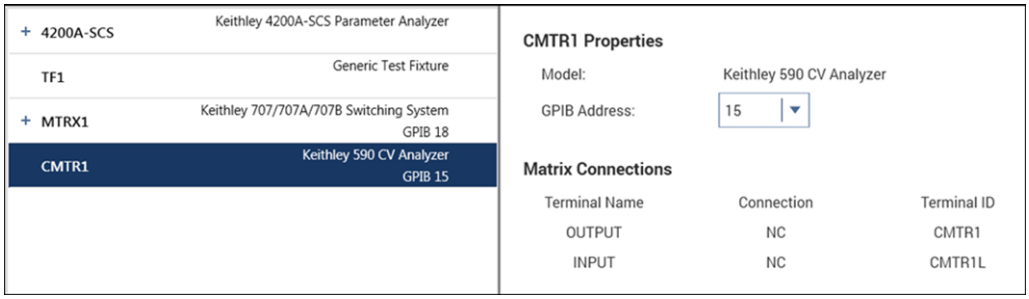
NOTE

You can programmatically read the GPIB address and other instrument properties from the system configuration using the LPT library `getinstattr` function. Proper use of `getinstattr` allows you to develop user libraries that are independent of the configuration. For more information, refer to Keithley User Library Tool (KULT) (on page 8-1).

To set the GPIB address:

1. Select the instrument in the Configuration Navigator. An example of the Keithley 590 C-V Analyzer information is shown in the figure below.
2. From the **GPIB Address** list, select the address.
3. Select **Save**.

Figure 426: Keithley 590 CV Analyzer Properties and Connections screen

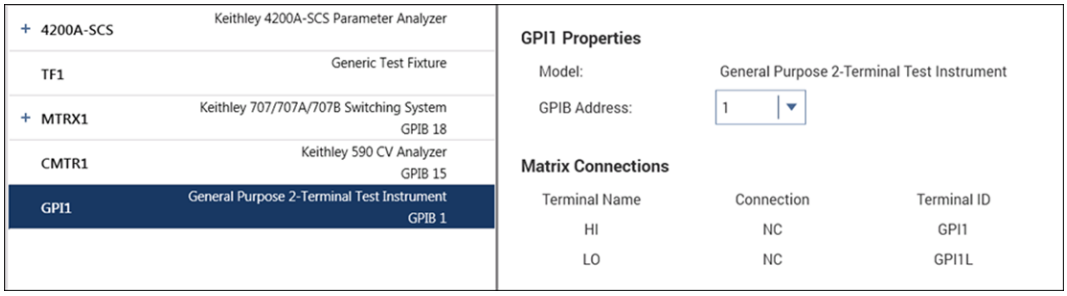


Add a general-purpose instrument

You can add a general-purpose instrument to your system configuration if your application requires an unsupported external instrument. See Supported external equipment (on page 7-6) for a list of supported instruments.

When you select a General Purpose Test Instrument in the Configuration Navigator, its properties and connections are displayed in the Work Area. The properties for a 2-terminal general-purpose instrument are shown in the following figure.

Figure 427: General Purpose Instrument 2-Terminal Properties and Connections



A two-terminal general-purpose instrument is an unsupported external instrument with two terminals (HI and LO), such as a current source. Generally, the instrument HI and LO terminals transmit or receive the instrument stimulus signals.

A four-terminal general-purpose instrument is an unsupported external instrument with four terminals (HI, SENSE HI, LO, and SENSE LO), such as a digital multimeter. Generally, the instrument HI and LO terminals transmit or receive the instrument stimulus signals. The instrument SENSE HI and SENSE LO terminals typically measure the device under test (DUT) response to the instrument stimulus.

NOTE

The SENSE HI and SENSE LO signals are automatically routed through separate (parallel) switch matrix pathways when making connections between the general purpose instrument HI/LO terminals and the test-system pins.

To control the operation of a GPIB or RS-232 general-purpose instrument in a Clarius project, create a user library with KULT and use the LPT library I/O functions (`kib`* and `ksp`*) to communicate with the general-purpose instrument (refer to Keithley User Library Tool (KULT) (on page 8-1) for more information). As with any other instrument in the system configuration, general-purpose instrument terminals can be automatically routed to test system pins using the `ConnectPins` user module in the `Matrixulib` user library. Refer to Using Switch Matrices (on page A-1) for information.

To set the GPIB address, refer to Set GPIB addresses (on page 7-17).

KXCI Settings

If you need to set up the 4200A-SCS as a subordinate on a GPIB or ethernet system, you do the initial setup through the KCon KXCI Settings. This allows you to use an external computer to remotely control the 4200A-SCS over GPIB or ethernet.

You can also use KXCI to set up emulation for Keysight 4145B Semiconductor Parameter Analyzers. In many cases, test programs developed for use with a Keysight 4145B run without modification when they are used with a 4200A-SCS running KXCI.

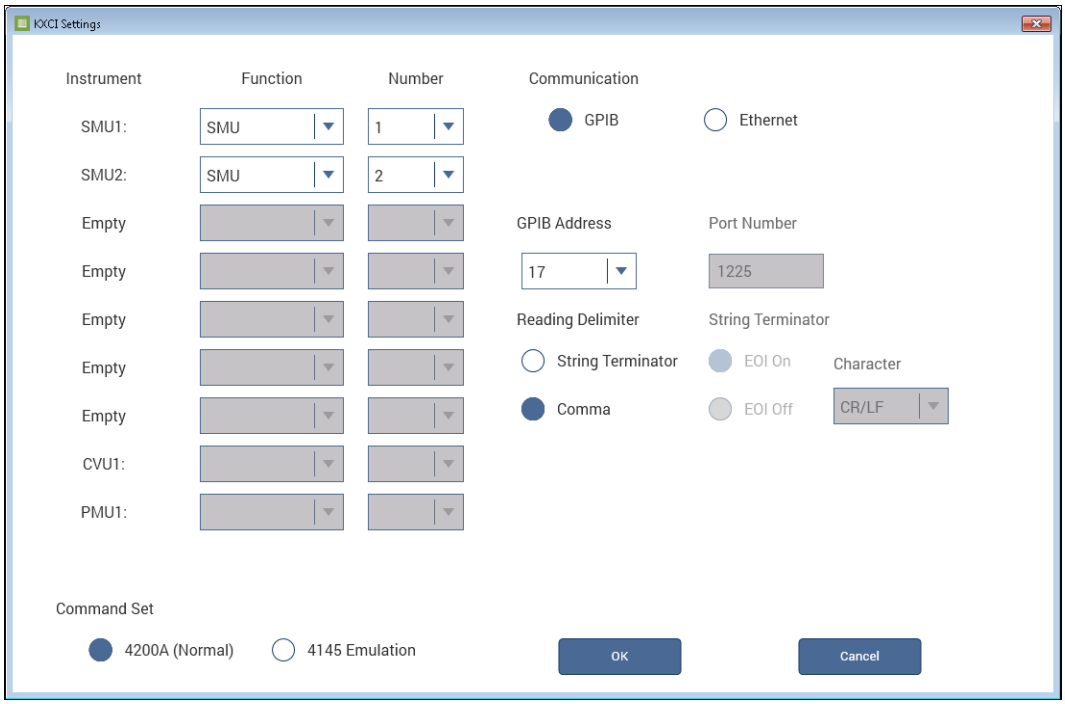
Refer to Keithley External Control Interface (on page 10-1) for detailed information regarding KXCI.

Set up KXCI for GPIB control

To set up GPIB control:

1. Select **KXCI Settings**. The **KXCI Settings** dialog box is displayed, as shown in the following figure.
2. Set Communications to **GPIB**.
3. Set the **GPIB Address**. This is the primary address of the 4200A-SCS when operating under KXCI control. 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.
4. Set the **Reading Delimiter** to determine the output data delimiter characters that are added to the end of each KXCI output message:
   - Select **String Terminator** to use a character.
   - Select **Comma** to terminate output data with a comma (,).
5. If you selected String Terminator, select the type of **Character**:
   - **None** to use no character.
   - **CR** to use a carriage return.
   - **LF** to use a line feed.
   - **CR/LF** to use a carriage return and line feed character sequence.
6. If **String Terminator** is selected, select **EOI ON** or **EOI OFF**. The EOI setting determines if the 4200A-SCS asserts the GPIB End Or Identify (EOI) signal with the last byte of each output data message.
7. Select **OK**.

Figure 428: KXCI Settings dialog box



Set up KXCI for ethernet control

To set up ethernet control:

1. Select **KXCI Settings**. The **KXCI Settings** dialog box is displayed.
2. Set Communications to **Ethernet**.
3. Set the **Port Number** (the default is 1225).
4. Select **OK**.

Setting up KXCI as a 4145B emulator

Although the KXCI GPIB command set is similar to the Keysight 4145B GPIB command set, the 4200A-SCS and Keysight 4145B hardware are different. To use existing 4145B code with the 4200A-SCS, you need to set up the 4200A-SCS SMUs to map to the 4145B instrument numbers. In many cases, test programs developed for use with a Keysight 4145B run without modification when they are used with a 4200A-SCS running KXCI.

The fundamental difference is that the 4200A-SCS hardware is modular, while the Keysight 4145B hardware is fixed, as shown in the following table.

Hardware comparisons

Instrument type	Keithley Instruments 4200A-SCS	Keysight 4145B
Source measure units (SMUs)	2 to 9	4 (fixed)
Voltage monitor (VM)	You can configure any SMU to function as a VM. Up to 9 VMs are possible.	2 (fixed)
Voltage source (VS)	You can configure any SMU to function as a VS. Up to 9 VSs are possible.	2 (fixed)

KCon manages these hardware differences by allowing you to assign source-measure unit, voltage monitor, or voltage source functions to any 4200A-SCS SMU, as shown in the next table.

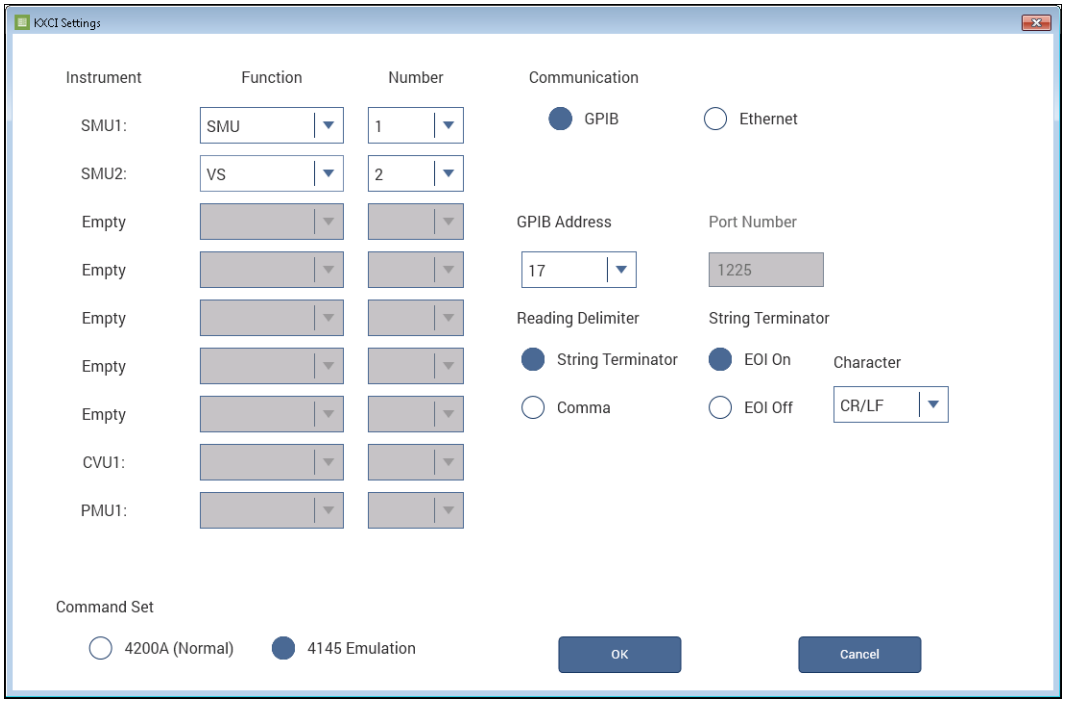
KXCI SMU (Source measure unit) function assignment

Function selection	Description
SMU (source measure unit)	Instructs the Model 42XX-SMU to emulate a Keysight 4145B Source Measure Unit.
VM1...VM8 (voltage monitor)	Instructs the Model 42XX-SMU to emulate the capabilities of a Keysight 4145B VM1 or VM2 and additional voltage monitors (VMs). You can map up to eight VMs to SMUs. You can assign a VM any number from 1 to 8, regardless of the number of SMUs in the system. Each VM number must be unique.
VS1...VS8 (voltage source)	Instructs the Model 42XX-SMU to emulate the capabilities of a Keysight 4145B VS1 or VS2, respectively and additional voltage sources (VSs). You can map up to eight VSs to SMUs. You can assign a VS any number from 1 to 8, regardless of the number of SMUs in the system. Each VS number must be unique.

To set up 4145B emulation:

1. Select **KXCI Settings**. The **KXCI Settings** dialog box is displayed, as shown in the following figure.
2. In the **Function** column, select the 4145 function that the SMU will emulate.
3. In the **Number** column, select the number that is used for the device in your 4145B program.

Figure 429: KXCI Settings for 4145B emulation



Command Set

The Command Set option chooses the control mode through which KXCI runs the 4200A-SCS: 4200A (Normal) or 4145 Emulation.

4200A (Normal) selects the 4200A command set. This command set includes all the 4145 Emulation commands plus an additional 4200A-only command.

In 4145 Emulation mode, the KXCI GPIB command set, status model, and output data are similar in function and format to those supported by the Keysight 4145B Semiconductor Parameter Analyzer, with substantially expanded performance.

The following table summarizes some differences and similarities between the two modes.

Command set comparison

Characteristic	Command set	
	4145 Emulation	**4200A (Normal)**
String reported in response to `ID` query	ID HP4145B 1.1,1.0	KI4200A Vx.x.x (where x.x.x is the version number)
GPIB data resolution	5 digits	7 digits
Maximum number of sweep data points	1024	4096
Possible instrument configurations	8 SMU/VM/VS	
Configuration query	Not supported	`*OPT?` command
Instrument self test	Not supported	SMUs only
Custom A/D control	Not supported	IT4 command options
200 V, 1 A capability	Supported	
1.0 pA source/measure-range capability (with preamplifer on SMU)	Supported	

KXCI always starts in the selected mode.

For additional details about the differences between the 4200A extended mode and 4145 emulation mode, refer to Keithley External Control Interface (KXCI) (on page 10-1).

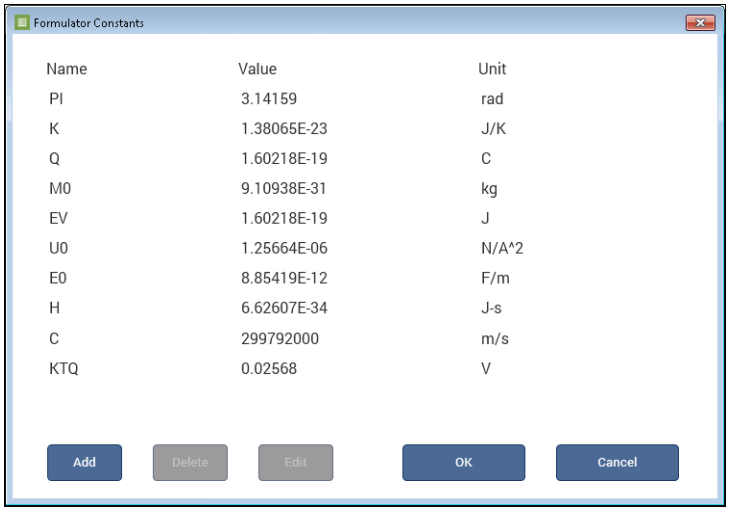
Tools

The options in the Tools dialog box allows you to change Formulator constants, generate technical support files, and get KCon version and copyright information.

Formulator Constants

The Formulator Constants option allows you to access the default Formulator constants that are automatically assigned to new Clarius test modules when they are created. You can edit the default Formulator constants.

Figure 430: Default Formulator constants



The constants include:

- PI: π is the ratio of the circumference to the diameter of a circle
- K: Boltzmann's constant
- Q: The charge of an electron
- M0: Electron mass
- EV: Electron volt
- U0: Permeability
- E0: Permittivity of a vacuum
- H: Planck's constant
- C: Speed of light
- KTQ: The thermal voltage

For additional information, see The Formulator (on page 6-290).

To modify the constants:

1. Select **Tools**.
2. Next to Formulator Constants, select **Open**.
3. To add a constant, select **Add** and complete the fields.
4. To remove a constant, select the constant and select **Delete**.
5. To change a constant, select the constant and select **Edit**.

Generate Technical Support Files

The Technical Support Files option analyzes your 4200A-SCS. KCon stores the analysis results to a USB flash drive. You can then send the results to Keithley Instruments for review.

To generate a technical support file:

1. Insert a USB flash drive into one of the front-panel USB ports.
2. Select **Tools**.
3. Next to Technical Support Files, select **Generate.**
4. Select **Yes**.
5. After the System Audit window appears and indicates a successful analysis, select **OK**. The System Audit window closes.
6. Contact your local Keithley Instruments office, sales partner, or distributor for details on how to send the files.

Technical support personnel at Keithley Instruments will review the analysis information and assess the state of your 4200A-SCS.

About KCon

About KCon displays a dialog box that contains version and copyright information.

KCon Learning Center

The Learning Center provides access to all 4200A-SCS information, such as:

- Instructions in text files and videos.
- The 4200A-SCS Technical Data Sheet.
- Application notes that show examples of how to use the 4200A-SCS for application-specific tasks.

Section 8

Keithley User Library Tool (KULT)

In this section:

Introduction

The Keithley User Library Tool (KULT) is a tool you can use 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, also called 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 Clarius software tool.

KULT provides a simple user interface that helps even a novice programmer to effectively enter code, build a user module, and build a user library. KULT also provides management features for the user library, including menu commands to copy modules, copy libraries, delete modules, 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 4200A-SCS without requiring a software upgrade from Keithley.

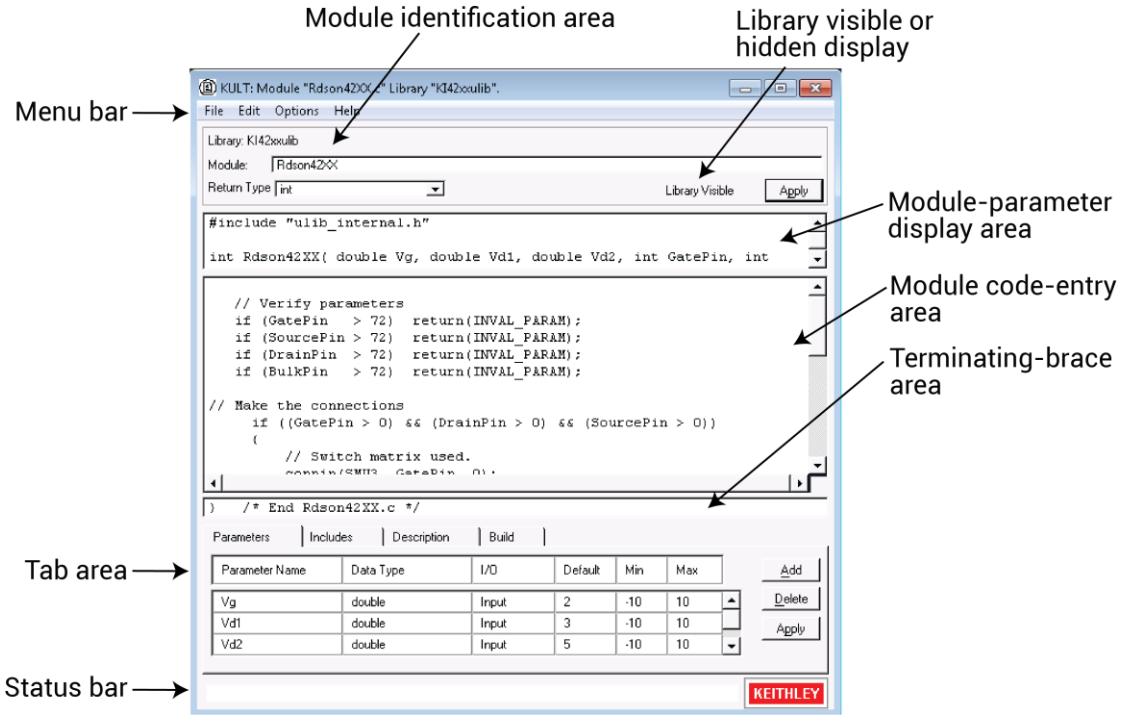
To execute a KULT user module in Clarius, you create a Clarius user test module (UTM) and connect it to the user module. Once this user module is connected to the UTM, the following occurs each time Clarius executes the UTM:

- Clarius dynamically loads the user module and the appropriate user library directory (`usrlib`).
- Clarius 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.

KULT window

The KULT window is shown in the following figure. It provides all the menus, controls, and user-entry areas that you need to create, edit, view, and build a user library and to create, edit, view, and build a user module.

Figure 431: KULT window overview



Each feature of the KULT window is explained in the following sections.

Understanding the module identification area

The module identification area is directly below the menu bar and defines the presently open user library and user module. The components of this area are as follows:

- **Library Name**: Displays the name of the presently open (active) user library.
- **Module Name**: Displays the name of the presently open user module.
- **Return type**: Defines the data type of all codes that are returned by `return(code)` statements in the user module. You can 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 user test module (UTM) is executed by Clarius, the value of the `return(code)` statement is displayed on the Data worksheet in the column labeled with the module name when the test is run.

- **Library Visible / Library Hidden:** Displays whether or not the presently open user library is available to Clarius. To change the hidden/visible status, select or clear the Hide Library option in the Options menu (on page 8-14).
- **Apply:** Updates the presently open user module to reflect additions and changes.

Understanding the module parameter display area

The module parameter area is a display-only area that is directly below the module identification area. In the module-parameter area, KULT displays:

- 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.

Understanding the module code-entry area

The module code-entry area is below the module-parameter area. The module code-entry area is 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 LPT Library Function Reference (on page 14-1).

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 when you click **Apply**.

Understanding the Tab area

The Tab area includes the tabs:

- Parameters
- Includes
- Description
- Build

Parameters tab area

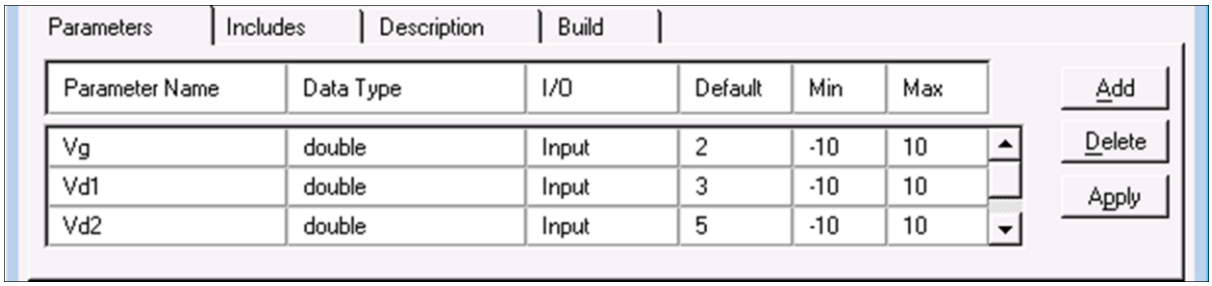
In the Parameters tab, you define and display parameters in the user module call. You can define and display:

- Parameter name
- Parameter data type
- Input or output (I/O) data direction
- Default, min, and max values for the parameter

These options are defined in the following text.

The Parameters tab area is near the bottom of the KULT main screen. An example is shown here.

Figure 432: Parameters tab for the Rdson42XX user module from the KI42XX library



NOTE

You can right-click anywhere in the Parameters tab area to access the Add, Delete, and Apply options.

To add a parameter:

1. Click **Add**.
2. Enter the information as needed.
3. Click **Apply**.

To delete a parameter:

1. Click the parameter name or any of the adjacent fields.
2. Click **Delete**.

To make changes to the parameters:

1. Make changes in the appropriate field.
2. Click **Apply**.

Parameter name field

The parameter name field identifies the parameters that are passed to the user module. These are the same parameters that are 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. Click the arrow at the right of the data type field to choose from a list of 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. Click the arrow to the right of the I/O field to select from the input and output selections.

Default, min, and max fields

The **Default** field specifies the default value for a non-array (only) input parameter.

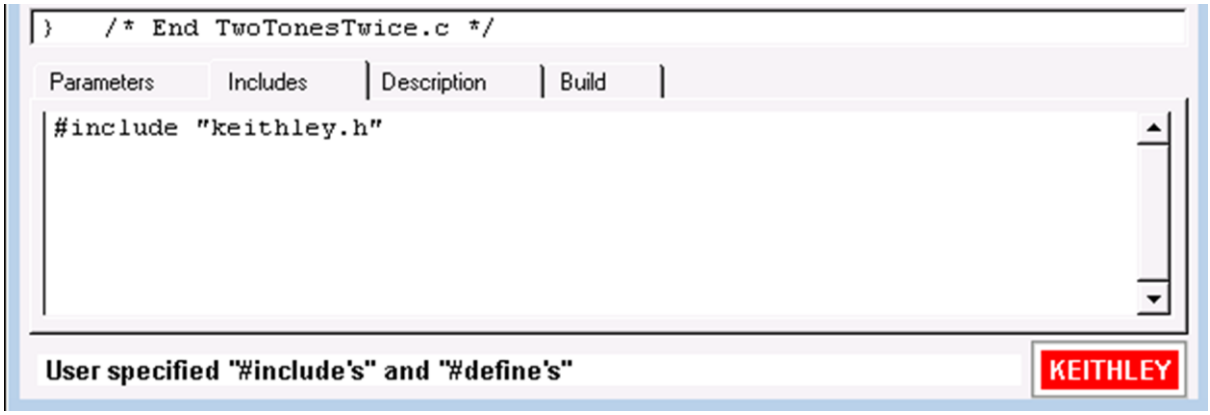
The **Min** field specifies the minimum recommended value for a non-array (only) input parameter. When the user module is used in a Clarius user test module (UTM), configuration of the UTM with a parameter value smaller than the minimum value causes Clarius to display an out-of-range message.

The **Max** field specifies the maximum recommended value for a non-array (only) input parameter. When the user module is used in a Clarius UTM, configuration of the UTM with a parameter value larger than the maximium value causes Clarius to display an out-of-range message.

Includes tab area

The Includes tab, shown below, lists the header files used in the user module. This area can be used to add `#include` and `#define` statements to the presently open user module.

Figure 433: Default Includes tab area



By default, KULT automatically enters the `keithley.h` header file into the Includes tab. 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 want to include `keithley.h`
- You want 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 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, shown below, allows you to enter descriptive information for the presently open user module. Information entered in this area documents the module to the Clarius user and is used to create Clarius user library help.

Figure 434: Description tab area

CAUTION

Do not use C-code comment designators (/*, */, or //) in the Description tab area. When the user-module code is built, KULT also evaluates the text in this area. C-code comment designators in the Description tab area can be misinterpreted, causing errors.

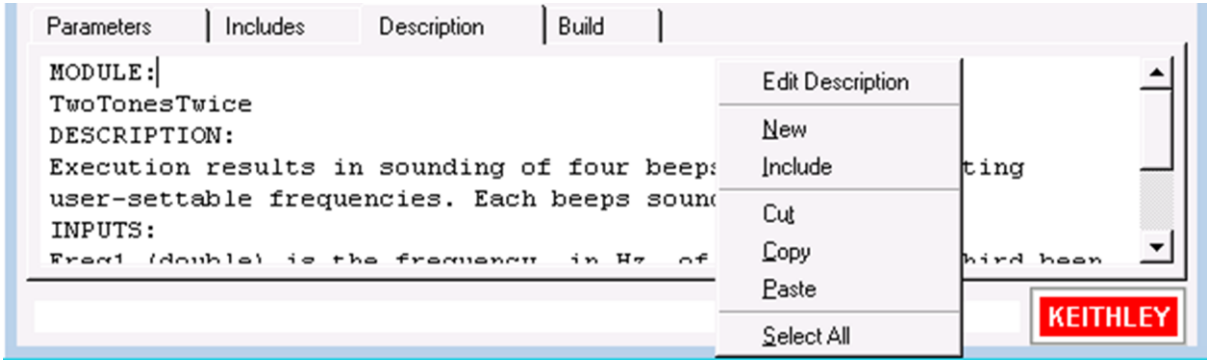
NOTE

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 Clarius UTM definition document.

To enter a description:

1. Click in the **Description** tab area.
2. Enter the description.
3. Right-click in the **Description** tab area to open the menu shown here.

Figure 435: Edit menu for the Description tab area



The edit menu commands are:

- **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. Refer to Include.
- **Cut**: Removes highlighted text from the Description tab and copies it to the clipboard. The text on the clipboard can be restored to new locations, in or out of KULT, using the paste function.
- **Copy**: Copies highlighted text from the description tab area to the clipboard. The text on the clipboard can be placed at new locations, in or out of KULT, using the paste function.
- **Paste**: Places text from the 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 build operation of the user library. When you click a build 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 helps you correct errors.

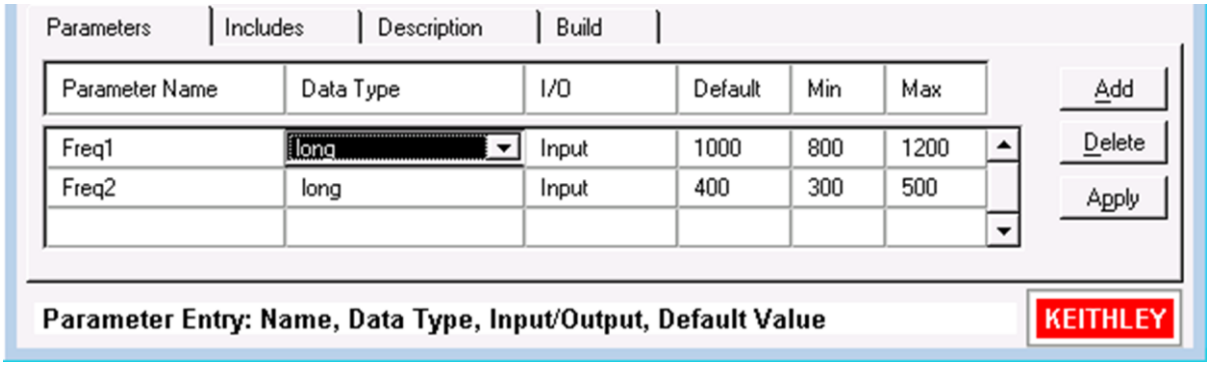
If no errors are found, the Build tab area displays:

No Errors/Warnings Reported. Compilation/Build was Successful.

Understanding the status bar

The status bar at the bottom of the KULT dialog box displays a description of the area where the cursor is located. For example, if the cursor is in the Parameters tab area, the status bar describes that area, as shown in the following figure.

Figure 436: Example of description in status bar



Understanding the menus

This section describes the menus on the menu bar, which is at the top of the KULT dialog box.

File menu

All user libraries are stored in the `C:\s4200\kiuser\usrlib` directory. This directory is referred to as Clarius/KULT user-library directory. It is the active user-library directory, which is where Clarius and KULT look for user libraries and user modules.

The File menu includes options to work with libraries.

New Library

The New Library menu option creates a new user library.

To create a new user library:

1. Click **New Library**. The Enter library dialog box opens.
2. Name the new user library.
3. Click **OK**. This initializes and opens the new user library in place of the presently open library.

Open Library

Opens an existing user library in place of the presently open library.

To open a library:

1. Click **Open Library** to display the open library list.
2. Select an existing user library.
3. Click **OK** to open the selected library.

Copy Library

Creates a copy of the currently open user library.

To copy a library:

1. Click **Copy Library**. The Enter Library dialog box opens.
2. Name the new user library into which to copy the presently open library.
3. Click **OK** to copy the presently open user library into the new library.

Delete Library

Deletes an existing user library and all of its contents.

To delete a library:

1. Click **Delete Library**. The list of libraries is displayed.
2. Select the user library to be deleted.
3. Click **OK** to delete the selected library.

New Module

Creates a new user module. When you create a new user module, module information in the KULT window is cleared.

The name of the new module must not duplicate the name of any existing user module or user library in the entire collection of user libraries.

To create a new user module:

1. Click **New Module**. This clears module information in the KULT window.
2. Enter a new user-module name in the Module text box.
3. Click **Apply** to initialize the new user module.

Open Module

Opens an existing user module.

To open a module:

1. Click **Open Module**. The Open Module list is displayed.
2. Select an existing user module.
3. Click **OK** to open the selected module in place of the currently open module.

Save Module

Saves the open user module.

Copy Module

Creates a copy of the open user module.

The name of the new module must not duplicate the name of any existing user module or user library in the entire collection of user libraries.

To copy the user module:

1. Click **Copy Module**. The list of libraries opens.
2. Select the user library in which to copy the presently open user module.
3. Clicking **OK.** The Enter New Module dialog box opens.
4. Enter a unique user-module name.
5. Click **OK**. The presently open module is copied into the selected library, under the new name. The presently open module remains open.

Delete Module

Deletes a user module from the open user library.

To delete a user module:

1. Click **Delete Module**. The KULT: Library `[OpenLibraryName]` list is displayed.
2. Select the module to be deleted.
3. Click **OK**. The selected module is deleted. The open module continues to be displayed, even if it is the module that you deleted.

NOTE

The executable user-library file, a dynamic link library (DLL), contains the deleted module until you rebuild the library. Refer to Building the user library to include the new user module (on page 8-29).

Print Module

Prints a text file that contains all 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 Microsoft® Windows® editing commands.

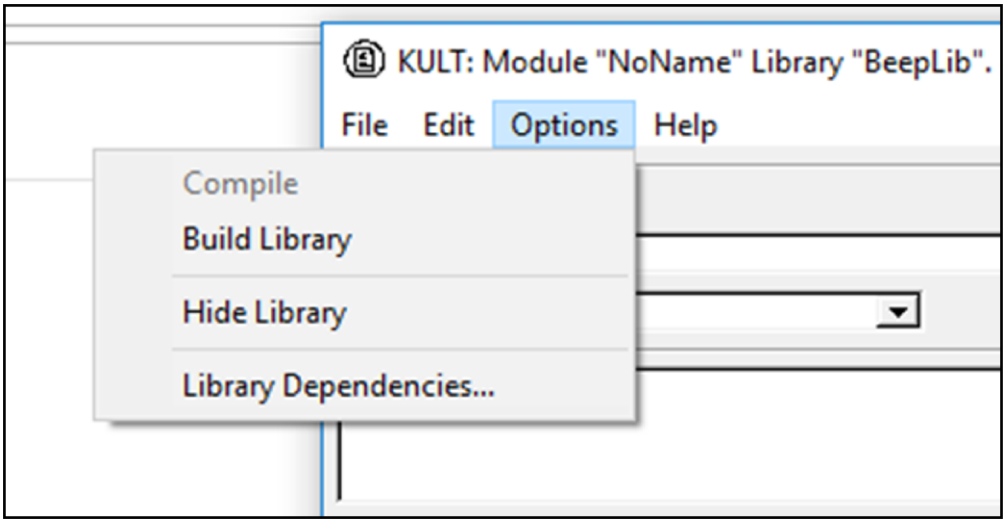
Edit menu commands:

- **Cut**: Removes highlighted text and copies it to the clipboard. The text on the clipboard can be restored to new locations, in or out of KULT, using the paste function.
- **Copy**: Copies highlighted text to the clipboard. The text on the clipboard can be placed at new locations, in or out of KULT, using the paste function.
- **Paste**: Places the text from the clipboard to 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 here.

Figure 437: KULT Options menu



Options menu commands:

- **Build Library:** When selected, adds the open user module (or updates changes) to the open user library. All of the modules in the open user library, and any libraries on which the open module depends, are linked together. A dynamic link library (DLL) is created that is accessible using user test modules (UTMs) in Clarius.

NOTE

The `C:\s4200\kiuser\usrlib\<library name>\build` folder is created when you run the `bld_lib` subcommand or select the **Build Library** menu option. This folder can be safely deleted for debugging purposes.

NOTE

Some Keithley Instruments-supplied user libraries contain dependencies. If 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 Working with interdependent user modules and user libraries (on page 8-66)).
Otherwise, the Build Library function will fail. For example, `ki82ulib` depends on `KI590ulib` and `Winulib`. You must specify these dependencies before rebuilding `ki82ulib` after making changes.

- **Hide Library:** When selected, causes the present user library to be unavailable to Clarius. 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 selected, displays the Library Dependencies list, where you specify each user library that is called by and that must be linked to the open user library. You must make selections individually; do not hold down the control or shift key to make multiple selections.

NOTE

The `C:\s4200\kiuser\usrlib\<library name>\build` folder is created when you run the `bld_lib` subcommand or select the **Build Library** menu option. This folder can be safely deleted for debugging purposes.

Help menu

The Help menu contains online help information about KULT:

- Contents: Allows access to the online KULT manual and other 4200A-SCS reference information.
- About KULT: Displays the software version.

Develop and use user libraries

Clarius includes user libraries of user modules that contain precoded user modules for commonly used external instruments. You can use these as-is, customize them, or create new ones. Most user modules contain functions from the Keithley-supplied Linear Parametric Test Library (LPT Library) and ANSI-C functions. All user modules are created and built using KULT.

Additionally, using KULT, you can program custom user modules in C. The LPT Library contains functions that are designed for parametric tests. However, any C routine that can be built using KULT can be used as source code for a user module.

A user library is a dynamic link library (DLL) of user modules that are built and linked using the Keithley User Library Tool (KULT).

A user module is a C-language function that:

1. Typically calls functions from the LPT library and ANSI-C functions.
2. Is developed using the Keithley User Library Tool (KULT).

Copy user modules and files

You can use the KULT zip (on page 8-64) and unzip (on page 8-65) subcommands to copy user libraries and other files. See Performing other KULT tasks using command-line commands (on page 8-59) for more information.

The `KULTArchive.exe` utility is installed on your 4200A-SCS. You can copy this utility to a Model 4200 or 4200A-SCS to archive or unzip a user library for use with an earlier version of Clarius. This utility is located at `C:\S4200\sys\bin\KULTArchive.exe`.

If you will use the `KULTArchive.exe` utility with a Model 4200, you must install the Microsoft Visual C++ Redistributable. This file is available on your 4200A-SCS at `C:\s4200\sys\Microsoft\Microsoft Visual C++ 2017 Redistributable\c_redistx86.exe`.

Usage

```
kultarchive [subcommand]
```

Where:

- `<subcommand>` is the zip or unzip operation.

KULTArchive zip subcommand

```
zip -l<library_name> [password] <zipfile_name>
```

The `<library_name>` user library is created in the active user-library directory.

The `[password]` parameter is optional.

Example for zip without password

```
kultarchive zip -l<Library1> C:\temp\myzip.zip
```

KULTArchive unzip subcommand

```
unzip [-dest_path] [password] <zipfile_name>
```

Where:

- `[-dest_path]` is the target directory where the file will be unzipped.
- `[password]` is required if the file was compressed using the password parameter in the `zip` subcommand.

The `<zipfile_name>` archive is unzipped in the active user-library directory unless the `[-dest_path]` parameter is specified. The `[-dest_path]` parameter should not be used when you import a user library.

Example for unzip with password

```
kultarchive unzip -password -pw1234 C:\temp\myzip.zip
```

Enabling real-time plotting for UTMs

To enable real-time plotting in a UTM, you use the following LPT library functions:

- PostDataDouble() (on page 14-35)
- PostDataInt() (on page 14-37)
- PostDataString() (on page 14-37)

The first parameter in the three functions is the variable name, defined as `char *`.

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

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* (on page 8-18)**

- 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
- Building the user module
- Finding code errors
- Building the user library to include the new user module
- Finding build errors
- Checking the user module

Tutorial 1: Creating a new user library and user module

KULT is a tool that helps you develop user libraries. Each user library is comprised of one or more user modules. Each user module is created using the C programming language.

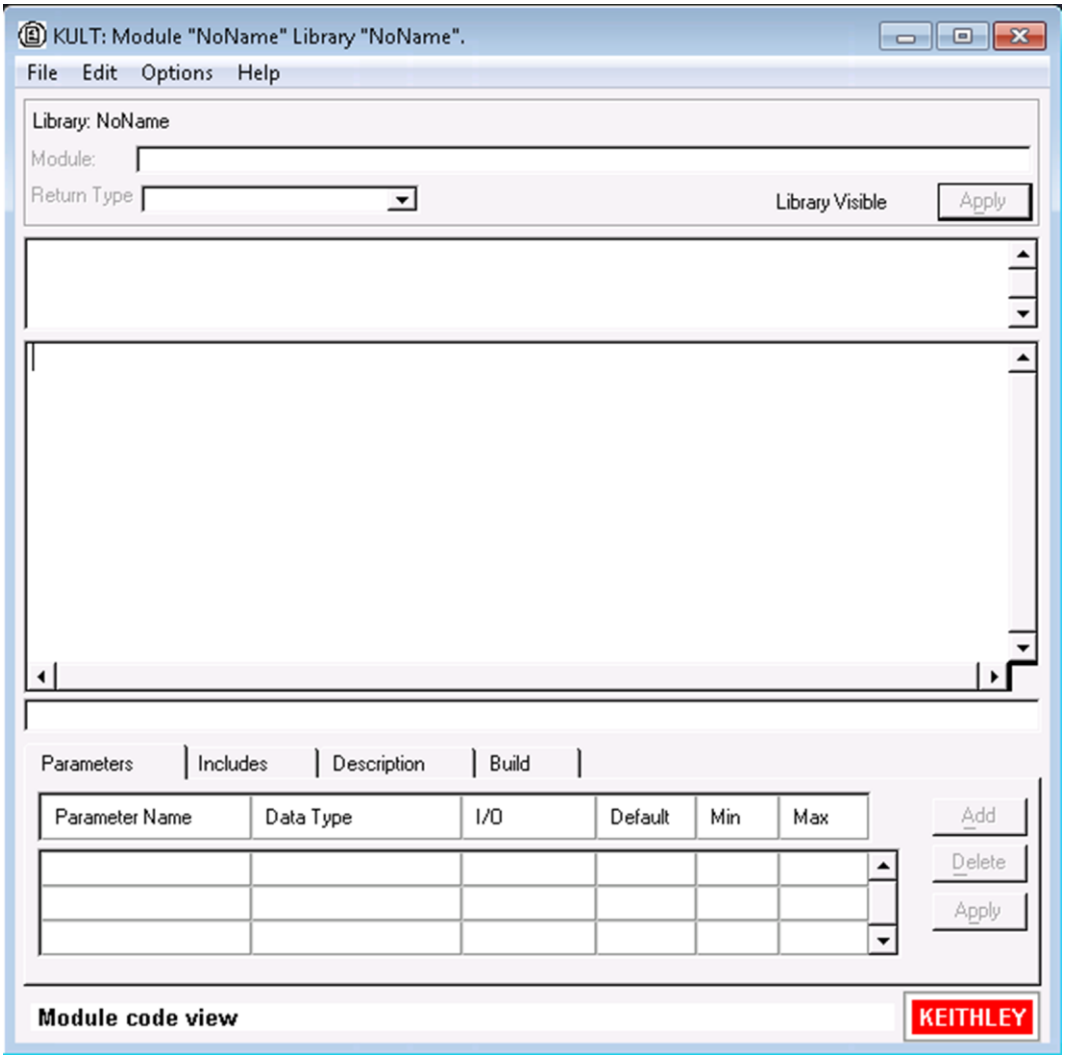
This section contains a tutorial that shows you how to create a new user library and new user module. A hands-on example is provided that illustrates how to create a user library that contains a user module that activates the internal beeper of the 4200A-SCS.

Starting KULT

To start KULT:

1. Select **KULT** in the Microsoft® Windows® **Start** menu (**Start > Programs > Keithley Instruments > KULT**).
2. A blank KULT window appears named **KULT:** Module **"NoName"** Library **"NoName"**, as shown below.

Figure 438: Blank KULT window



Continue with Naming a new user library (on page 8-20).

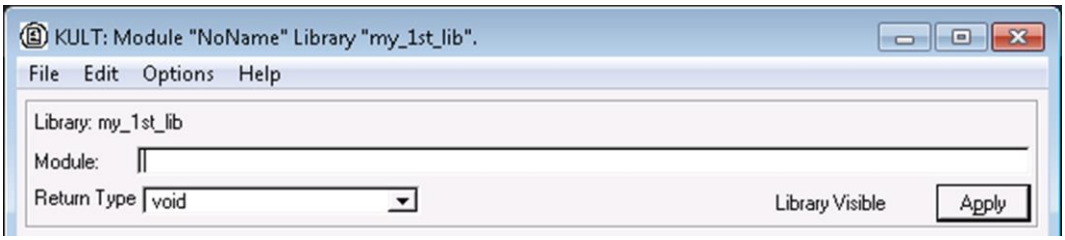
Naming a new user library

To name a new user library:

1. In the KULT **File** menu, click **New Library.**
2. In the Enter Library dialog box that opens, enter the new user library name. For this tutorial, enter `my_1st_lib` as the new user library name.
3. Click **OK**.

   The dialog box name changes to `KULT: Module "NoName" Library "my_1st_lib"`, and the name next to library in the top left of the dialog box is now `my_1st_lib`, as shown below.

Figure 439: KULT window after naming user library



Continue with Creating a new user module (on page 8-20).

Creating a new user module

NOTE

When naming a user module, conform to case-sensitive C programming language naming conventions. Do not duplicate names of existing user modules or user libraries.

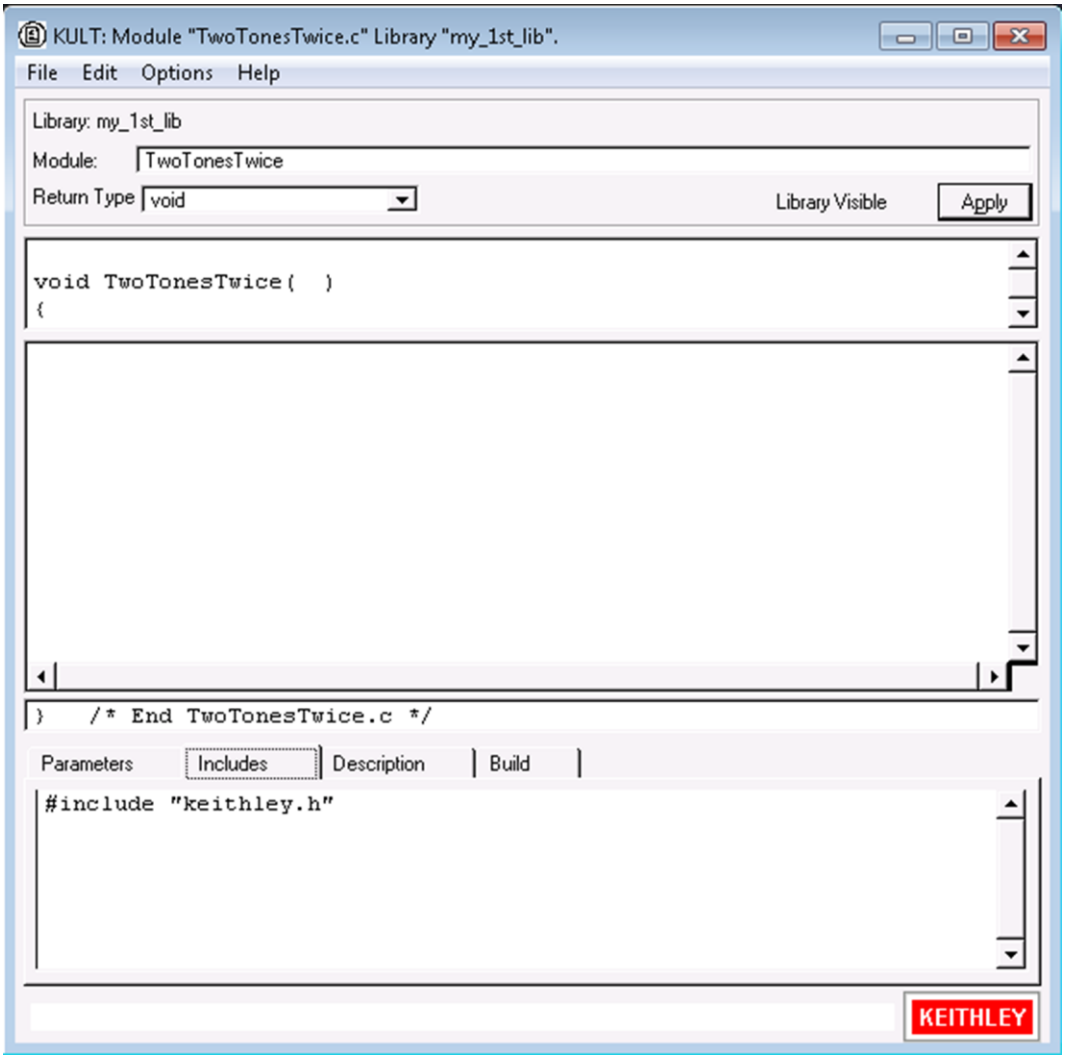
To create a new user module:

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.c" Library "my_1st_lib"`.
- You now see entries in the user-module parameters display area and in the terminating-brace display. If you click the **Includes** tab, there is also an entry there, as shown in the following figure.

Figure 440: KULT window after naming user module



NOTE

To view the entire module parameter display area, use the scroll bar.

Continue with Entering the return type (on page 8-22).

Entering the return type

If your user module generates a return value, select the data type for the return value in the **Return Type** box. However, the `TwoTonesTwice` user module will not produce a return value. Therefore, for the `TwoTonesTwice` module, keep the `void` default entry.

Continue with Entering user module code (on page 8-22).

Entering user module code

Enter the C code.

NOTE

Refer to LPT functions for SMUs and general operations (on page 14-62) for a complete list of supported I/O and SMU commands.

To enter the C code, enter the new C code into the module-code entry area.

For the `TwoTonesTwice` user module, enter the code listed below. The code deliberately contains a missing `;` 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 500 ms */
Beep(Freq2, 500);  /* Beep at second frequency */
Beep(Freq1, 500);
Beep(Freq2, 500);
Sleep(500)        /* NOTE deliberately forget semicolon initially */
```

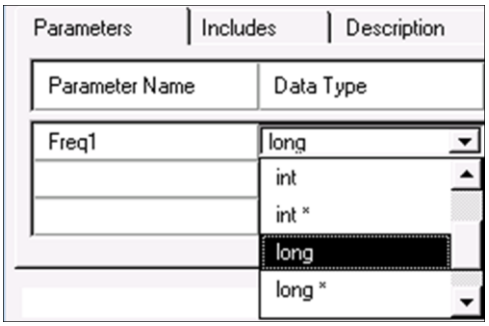
Continue with Entering parameters (on page 8-23).

Entering parameters

To enter the required parameters for the code:

1. Click the **Parameters** tab.
2. Click the **Add** button at the right side of the parameters tab area.
3. Under **Parameter Name**, enter `Freq1`.
4. Click the Data Type cell and select **long** under Data Type, as shown here. This is the C data type.

Figure 441: Data Type menu



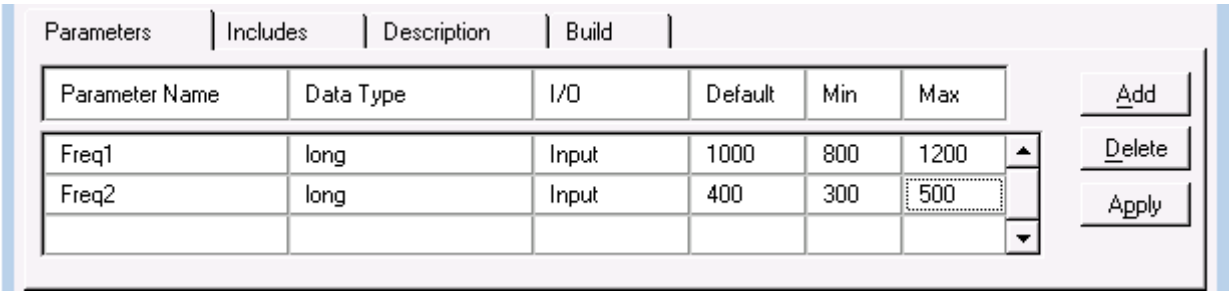
NOTE

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. For this user module, the `I/O` entry of `Input` is correct. If the Data Type is pointer or array, you could choose `Input` or `Output`.

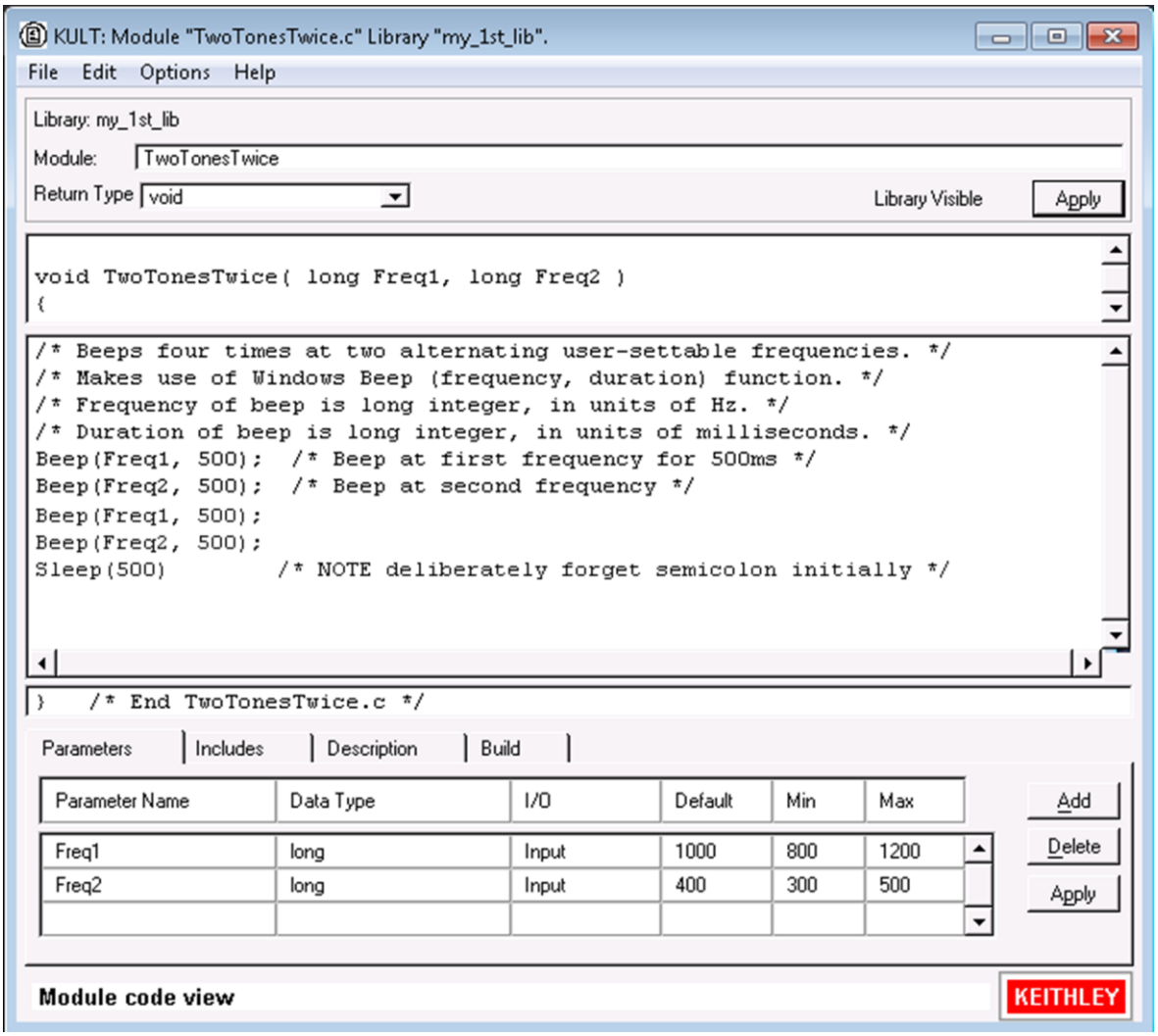
6. Under **Default**, **Min**, and **Max**, enter default, minimum, and maximum values. These values limit the choices the user sees. For the `TwoTonesTwice` user module, enter `1000`, `800`, and `1200`, respectively.
7. For the `TwoTonesTwice` module, add one more parameter with the values:
   - **Parameter name:** `Freq2`
   - **Data type:** `long`
   - **I/O:** `Input`
   - **Default:** `400`
   - **Min:** `300`
   - **Max:** `500`
8. Click **Apply**. (The Apply buttons at the top and bottom of the dialog box act identically.)

Figure 442: Parameter entries for the TwoTonesTwice user module



In the module-parameter display area, the function prototype now includes the declared parameters, as shown here.

Figure 443: KULT window - code and parameters applied



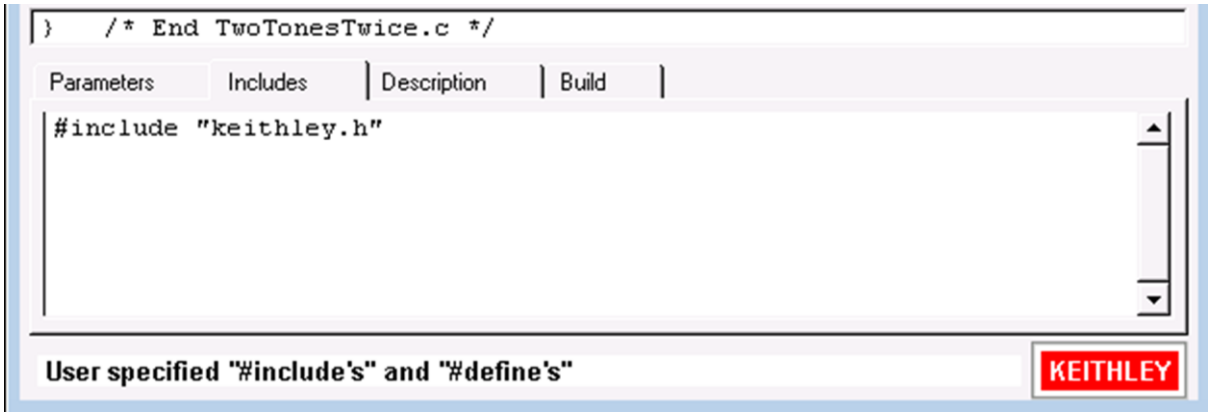
Continue with Entering header files (on page 8-26).

Entering header files

To enter the header files:

1. Select the **Includes** tab at the bottom of the dialog box to display the Includes area.

Figure 444: 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**.

Continue with Documenting the user module (on page 8-26).

Documenting the user module

To document the user module:

1. Click the **Description** tab at the bottom of the dialog box to open the Description tab area.
2. Enter any text needed to adequately document the user module to the Clarius user.

CAUTION

Do not use C-code comment designators (/*, */, or //) in the Description tab area. When the user-module code is built, 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 Clarius UTM definition document.

3. For the `TwoTonesTwice` user module, enter the following information in the **Description** tab area:

```
<!--MarkdownExtra-->
<link rel="stylesheet" type="text/css"
   href="http://clariusweb/HelpPane/stylesheet.css">

MODULE
======
TwoTonesTwice

DESCRIPTION
-----------
Execution results in sounding of four beeps at two alternating user-settable
   frequencies. Each beeps sounds for 500 ms.

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
```

Figure 445: Description tab area

Continue with Saving the user module (on page 8-27).

Saving the user module

Select the **File** menu, then select **Save Module**.

Continue with Building the user module.

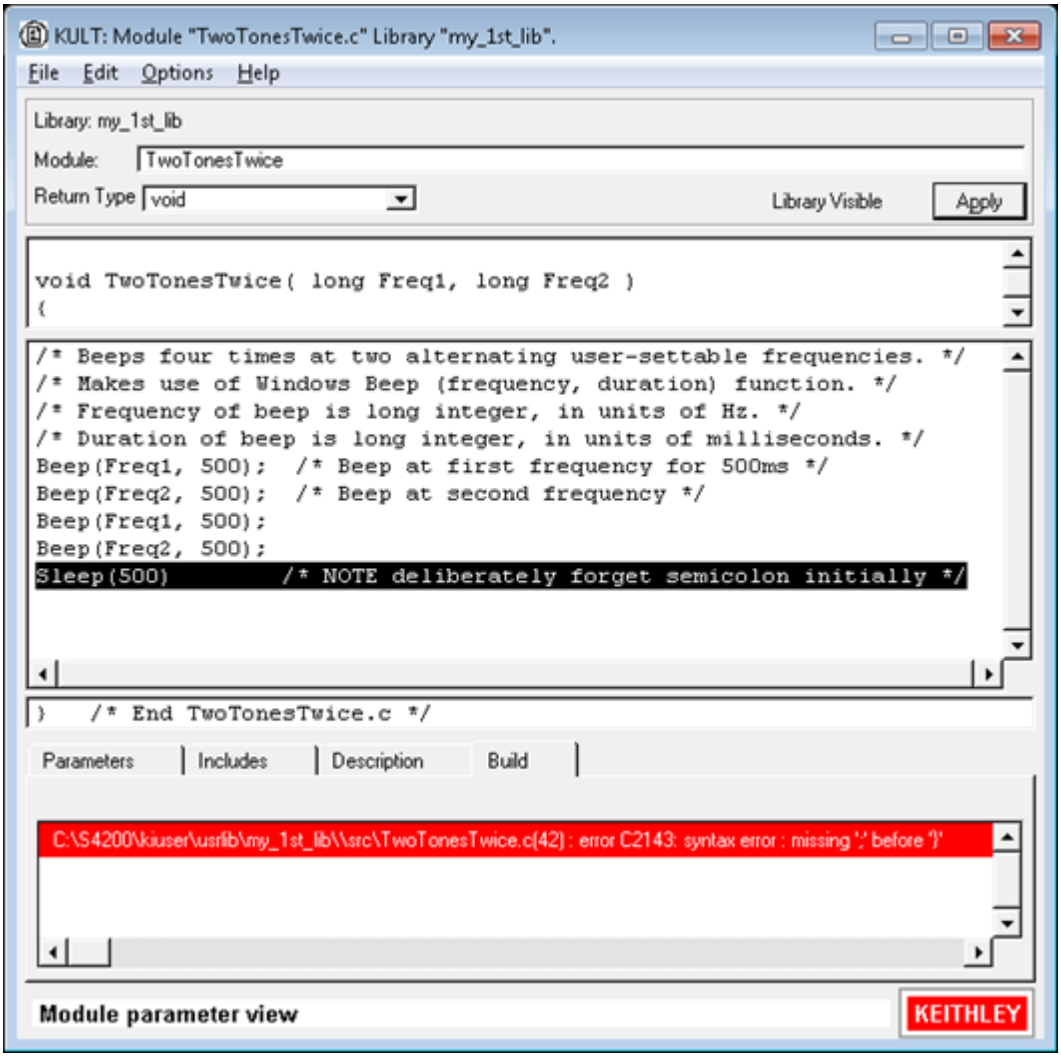
Finding code errors

When you click a build-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:

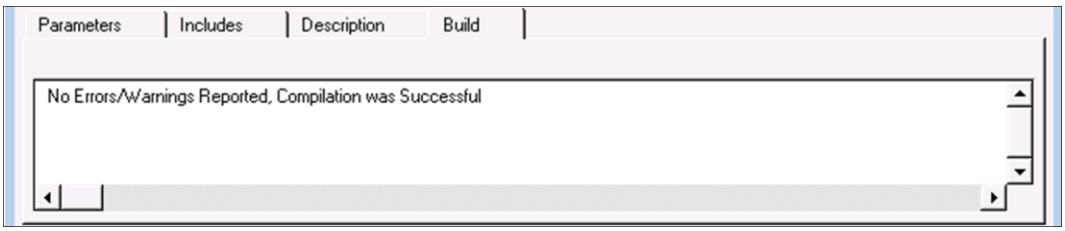
1. Click the error message. The last line of code is highlighted, as shown below.

Figure 446: Find a code error



2. Add the missing semicolon at the end of the code [`Sleep(500);`], and delete the comment about the missing semicolon.
3. Select **File > Save Module**.
4. Build the user module again.
   - The KULT Build message box should now display no error messages and disappears automatically.
   - The Build tab area should display the successful compilation message shown below.

Figure 447: Successful build message displayed in Build tab



Continue with Building the user library to include the new user module (on page 8-29).

Building the user library to include the new user module

After you have successfully built the user module, build the user library (or rebuild the user library) to include the module.

To build the user library:

1. Select the Build tab.
2. From the **Options** menu, select **Build Library**. The following occurs:
   - The user library is built. 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 created that is accessible using UTMs in Clarius.
   - The KULT Build Library message box indicates the build progress. If linker problems are encountered, this message box displays error messages. When you build the `TwoTonesTwice` user module, you should see no errors.

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 build was successful, this message appears: **No Errors/ Warnings Reported, Library Build was Successful**.
- If the build 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.

Continue with Finding build errors (on page 8-30).

Finding build errors

Find build errors using the information in the error message.

Continue with Checking the user module (on page 8-30).

Checking the user module

To check a user module, you need to create and execute a user test module (UTM) in Clarius. Create a simple Clarius project to check the user module.

To check the user module in Clarius:

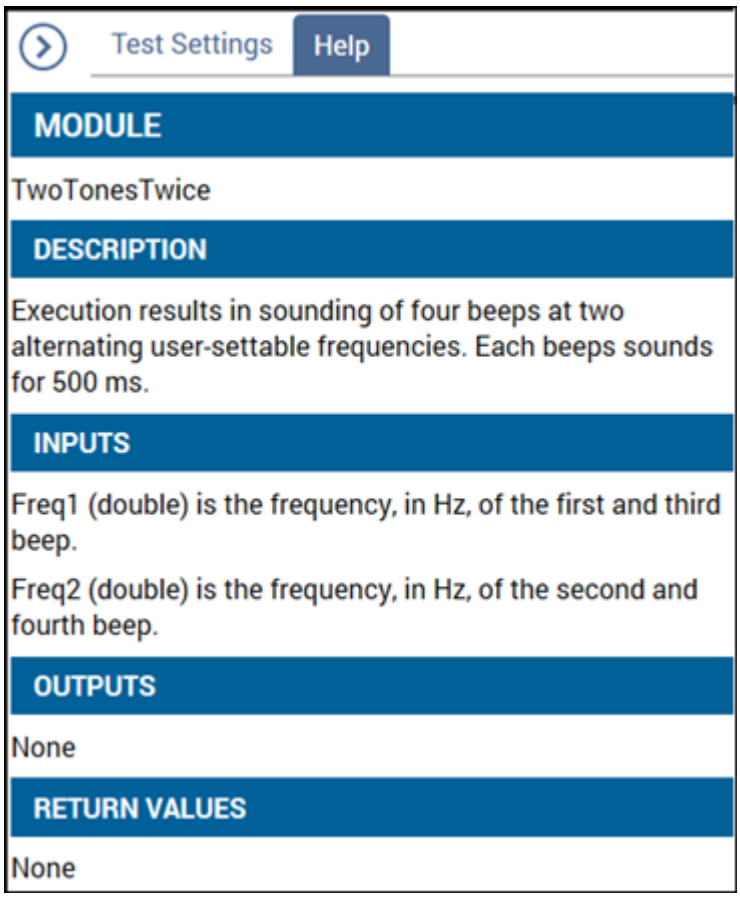
1. Start Clarius.
2. Choose the **Select** pane.
3. Select **Projects**.
4. Select **New Project**.
5. Select **Create**. You are prompted to replace the existing project.
6. Select **Yes**.
7. Select **Rename**.
8. Enter `UserModCheck` and press **Enter**.

9. Choose **Select**.
10. Select the **Actions** tab.
11. Drag **Custom Action** to the project tree. The action has a red triangle next to it to indicate that it is not configured.
12. Select **Rename**.
13. Enter `2tones_twice_chk` and press **Enter**.
14. Select **Configure**.
15. In the Test Settings pane, select the `my_1st_lib` user library.
16. From the User Modules list, select the `TwoTonesTwice` user module. A group of parameters are displayed for the UTM as shown in the following figure. Accept the default parameters for now. You can experiment later after you establish that the user module executes correctly.

Figure 448: Configured UTM

17. Select Help to verify that the HTML in the Description tab is correctly formatted. An example is shown in the following figure.

Figure 449: Example of help formatted as HTML for a user module



18. Select **Save**.
19. Execute the UTM by selecting **Run**. You should hear a sequence of four tones, sounded at alternating frequencies.

This tutorial generates no data. For an example of numerical data, see Tutorial 2: Creating a user module that returns data arrays (on page 8-33).

Tutorial 2: Creating a user module that returns data arrays

This section provides a tutorial that helps you 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).

To start Tutorial 2, go to Naming new user library and new VSweep user module (on page 8-33).

NOTE

Most of the basic steps that were detailed in Tutorial 1 are abbreviated in this tutorial. Before doing the following tutorial, complete Tutorial 1: Creating a new user library and user module (on page 8-18).

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. Click **OK**.
5. In the KULT **File** menu, click **New Module.**
6. In the **Module** text box at the top of the KULT dialog box, enter `VSweep` as the new module name.
7. Click **Apply**.

Continue with Entering the VSweep user-module return type (on page 8-33).

Entering the VSweep user-module return type

The VSweep user module generates an integer return value. Therefore, select **int** from the Return Type list.

Continue with Entering the VSweep user-module code (on page 8-34).

Entering the VSweep user-module code

In the module code-entry area, enter the C code below for the VSweep user module. Open the KULT dialog box to full screen view to simplify code entry.

```
/* 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 number 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 (on page 8-34).

Entering the VSweep user-module parameters

NOTE

This example uses the double-precision `D_ARRAY_T` array type. The `D_ARRAY_T`, `I_ARRAY_T`, and `F_ARRAY_T` are special array types that are unique to KULT. For each of these array types, 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 a hidden fourth line of incomplete parameter information for the array-size parameter specification.

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 Clarius 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.

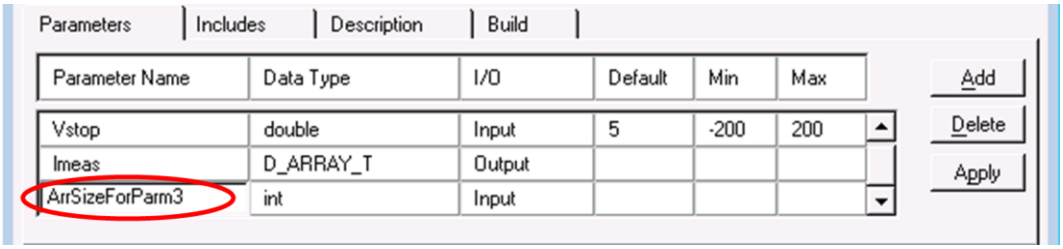
To enter the required parameters for the code:

1. Click the **Parameters** tab.
2. Enter the information for the two voltage input parameters, as shown in the following table. Click the **Add** button before adding each new parameter.

Parameter name	Data type	I/O	Default	Min	Max
Vstart	double	Input	0	-200	200
Vstop	double	Input	5	-200	200

3. Click **Add**.
4. On the third line, enter the following measured-current parameter information:
   - Parameter Name: `Imeas`
   - Data type: `D_ARRAY_T`
   - I/O: Output
5. Scroll down to display line 4 of the Parameters tab area. KULT enters the array size parameter in this line automatically for the array that is specified on line 3, as shown in the following figure.

Figure 450: KULT-entered array-size parameters



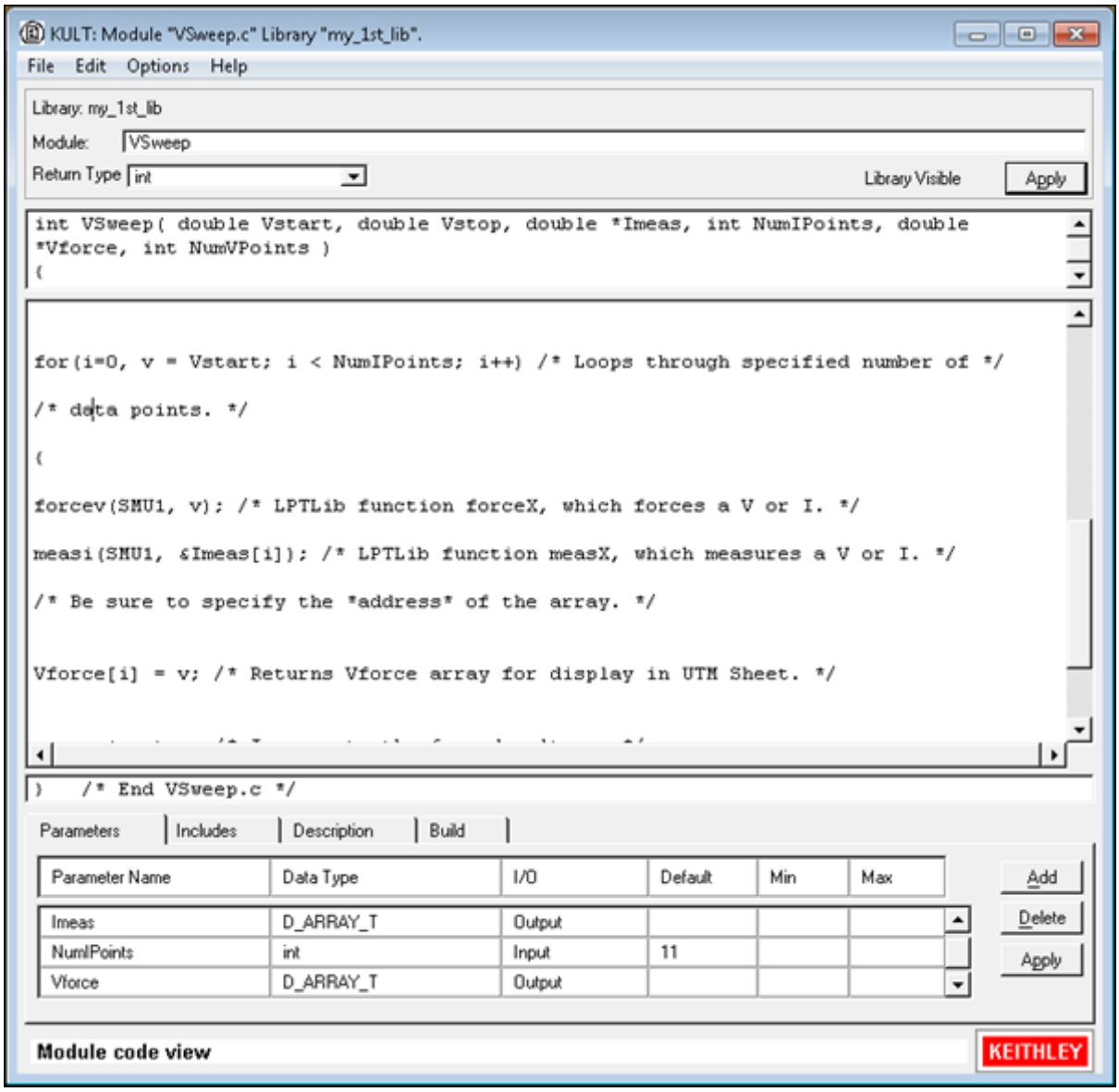
6. In the 4th line, under Parameter Name, change `ArrSizeForParm3` to `NumIPoints`. Note that the default Parameter Name entry is only a description of the required array size parameter. You must replace it with an appropriate array size parameter, as required by the user module code.
7. Leave the Data Type and I/O entries as is.
8. In the 4th line, under Default, enter the number `11` for the default current-array size. You can also add Min and Max array sizes if needed.
9. Click **Add**.
10. In the 5th line, enter the following forced-voltage parameter information:
    - Parameter Name: `Vforce`
    - Data type: `D_ARRAY_T`
    - I/O: Output
11. In the 6th line, under **Parameter Name**, change `ArrSizeForParm5` to `NumVPoints`.
12. In the 6th 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`. If the sizes do not match, the second return statement in the code halts execution and returns an error number (`-2`) in the `VSweep` column of the UTM Data worksheet.

13. Click **Apply.** In the module-parameter display area, the function prototype now includes the declared parameters, as shown below.

Figure 451: VSweep user-module dialog box after entering and applying code and parameters



Continue with Entering the VSweep user-module header files (on page 8-38).

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 (on page 8-38).

Documenting the VSweep user module

Select the Description tab and enter documentation for the user module, based on the comments provided in the code and other information about the module. Continue with Saving the VSweep user module (on page 8-38).

Saving the VSweep user module

From the **File** menu, select **Save Module**.

Continue with Building the VSweep user module (on page 8-38).

Building the VSweep user module

To build the user module:

1. Click the **Build** tab at the bottom of the dialog box to open the Build tab area.
2. In the **Options** menu, click **Build**. The user module is built. If the code and parameters were entered as specified, you should not see error messages.

NOTE

If you do see error messages, check for typographic errors; then fix and rebuild the user module. If necessary, review Finding code errors (on page 8-28).

3. In the **Options** menu, click **Build Library**. The user library builds. You should not see error messages.

Continue with Checking the VSweep user module (on page 8-39).

Checking the VSweep user module

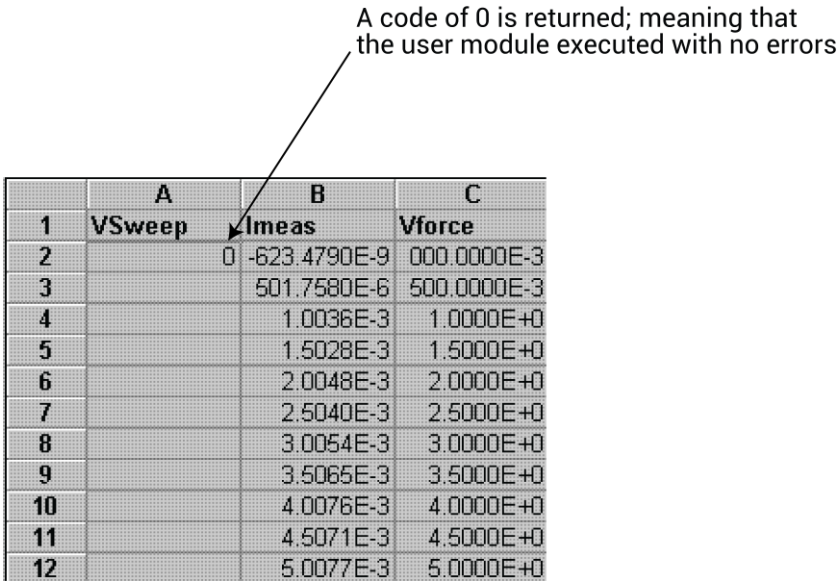
Check the user module by creating and executing a UTM in Clarius.

To check the user module:

1. Connect a 1 kΩ 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.
3. Choose **Select**.
4. Select the **Devices** tab.
5. Select the 2-wire-resistor.
6. Choose **Select**.
7. Select the **Tests** tab.
8. For the Custom Test, select **Choose a test from the pre-programmed library (UTM)**.
9. Drag **Custom Test** to the project tree. The test has a red triangle next to it to indicate that it is not configured.
10. Select **Rename**.
11. Enter the name `v_sweep_chk` for the UTM. You will use the `v_sweep_chk` UTM to execute the VSweep user module.
12. Select **Configure**.
13. In the right pane, from the User Libraries list, select the `my_2nd_lib` library.
14. From the User Modules list, select the `v_sweep_chk` user module. A default schematic and group of parameters are displayed for the UTM.
15. For Vforce, select **Enter Values** and enter the sweep values.
16. Select **Run**.
17. Select **Analyze**.

At the conclusion of execution, review the results in the Analyze sheet. If you connected a 1 kΩ resistor between SMU1 and GNDU, used the default UTM parameter values, and executed the UTM successfully, the results should be similar to the results in the following figure.The current/voltage ratio for each row of results should be approximately 1 mA / V.

Figure 452: Checking the VSweep user module



	A	B	C
1	VSweep	Imeas	Vforce
2	0	-623.4790E-9	000.0000E-3
3		501.7580E-6	500.0000E-3
4		1.0036E-3	1.0000E+0
5		1.5028E-3	1.5000E+0
6		2.0048E-3	2.0000E+0
7		2.5040E-3	2.5000E+0
8		3.0054E-3	3.0000E+0
9		3.5065E-3	3.5000E+0
10		4.0076E-3	4.0000E+0
11		4.5071E-3	4.5000E+0
12		5.0077E-3	5.0000E+0

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 in the same user library as the calling module or may be in another user library. This section 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 8-18) and Tutorial 2: Creating a user module that returns data arrays (on page 8-33):

- The VSweep user module in the `my_2nd_lib` user library, a copy of which is 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, called `VSweepBeep`, calls the `TwoTonesTwice` user module to signal the end of execution.

Creating the VSweepBeep user module by copying an existing user module

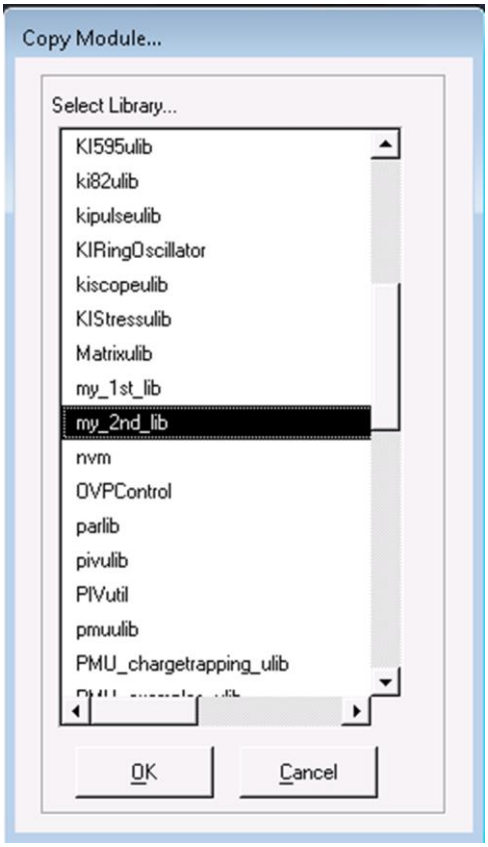
Open the Vsweep user module:

1. Start KULT.
2. Select **File > Open Library**.
3. Select `my_2nd_lib` from the list.
4. Click **OK**.
5. Click **File > Open Module**.
6. Select `VSweep.c` from the list.
7. Click **OK**.

Copy `VSweep.c` to the new user module `VSweepBeep`:

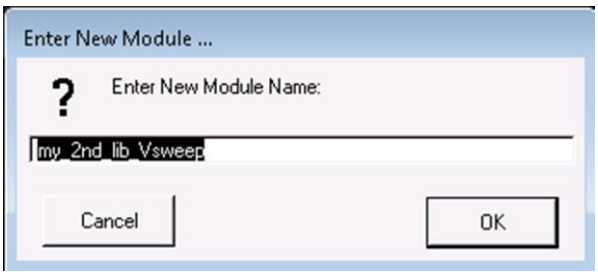
1. In the **File** menu, click **Copy Module**. The Copy Module list shown in the following figure opens.

Figure 453: Copy Module list box



2. Select `my_2nd_lib` (in this case, the user library for the copy is the same as the user library for the source).
3. Click **OK**. The Enter New Module dialog box opens, as shown here.

Figure 454: Enter New Module dialog box



4. Enter the name **VSweepBeep**.
5. Click **OK**.

NOTE

The name of the user module must not duplicate the name of any existing user module or user library in the entire collection of user 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 Managing user libraries (on page 8-56).

KULT creates a copy of the user module under the new name and displays a message indicating the need to rebuild the user library. You can skip the rebuild for now. Continue with the next step.

Open the new `VSweepBeep` user module:

1. Click **File > Open Module.**
2. Select **VSweepBeep.c** from the list. The KULT dialog box displays the `VSweepBeep` user module.

Continue with Calling independent user module from VSweepBeep user module (on page 8-43).

NOTE

You can also create a copy of the presently open user module in the same user library as follows:

- Enter a new name in the User Module text box.
- Click **Apply**. Before using the user module, you must save and 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`, just before the `return(0)` statement, add the following statement:

```
TwoTonesTwice(Freq1, Freq2); /* Beeps 4X at end of sweep. */
```

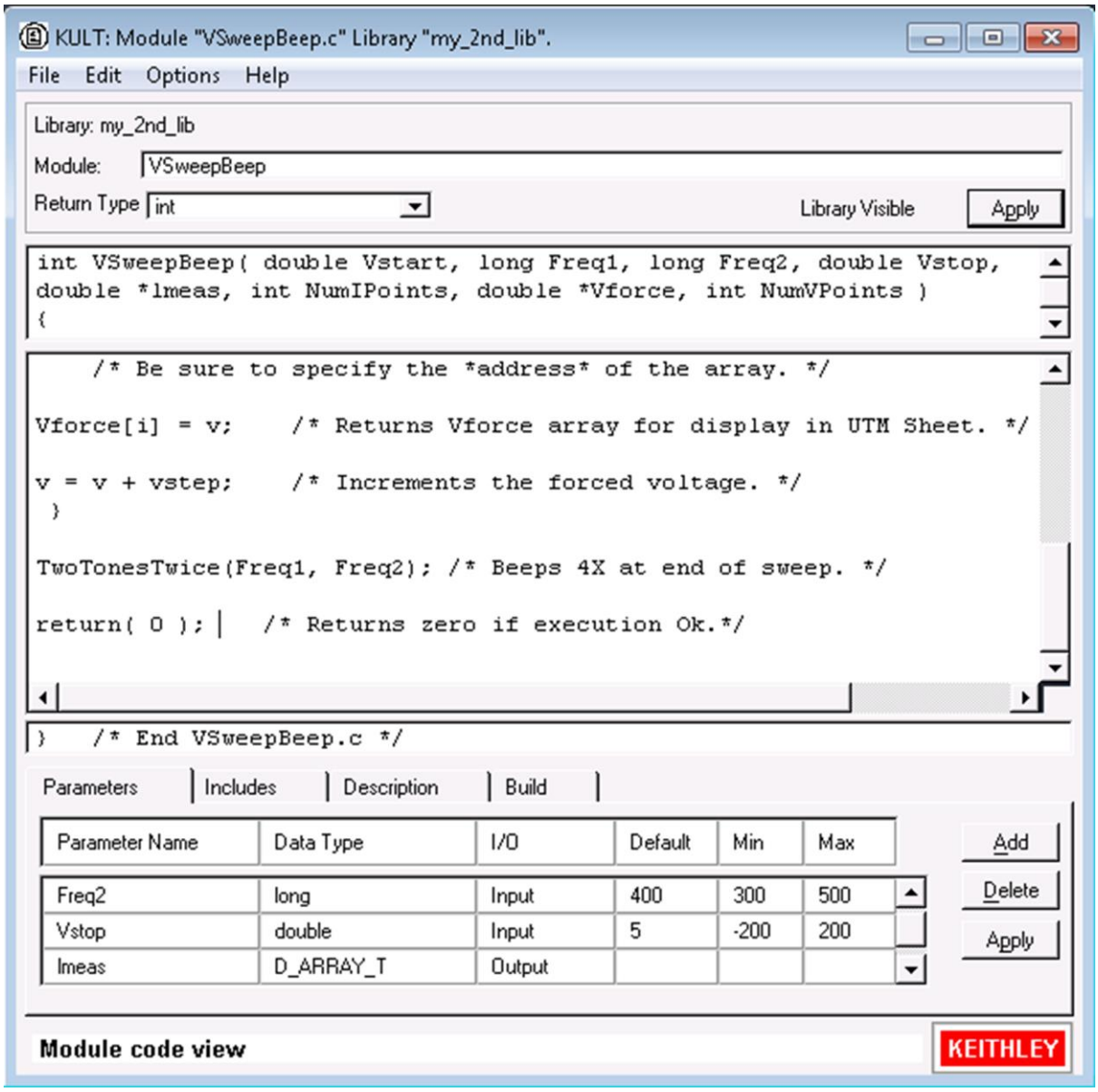
2. In the Parameters tab area, add the **Freq1** and **Freq2** parameters with the values shown in the following table, just as you did when you created the `TwoTonesTwice` user module, changing the Default, Min, and Max values as needed.

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

1. Click **Apply**. The **Freq1** and **Freq2** parameters are added to the function prototype as shown in the following figure.

Figure 455: Completed VSweepBeep user module



Continue with Specifying user library dependencies in VSweepBeep user module (on page 8-45).

Specifying user library dependencies in VSweepBeep user module

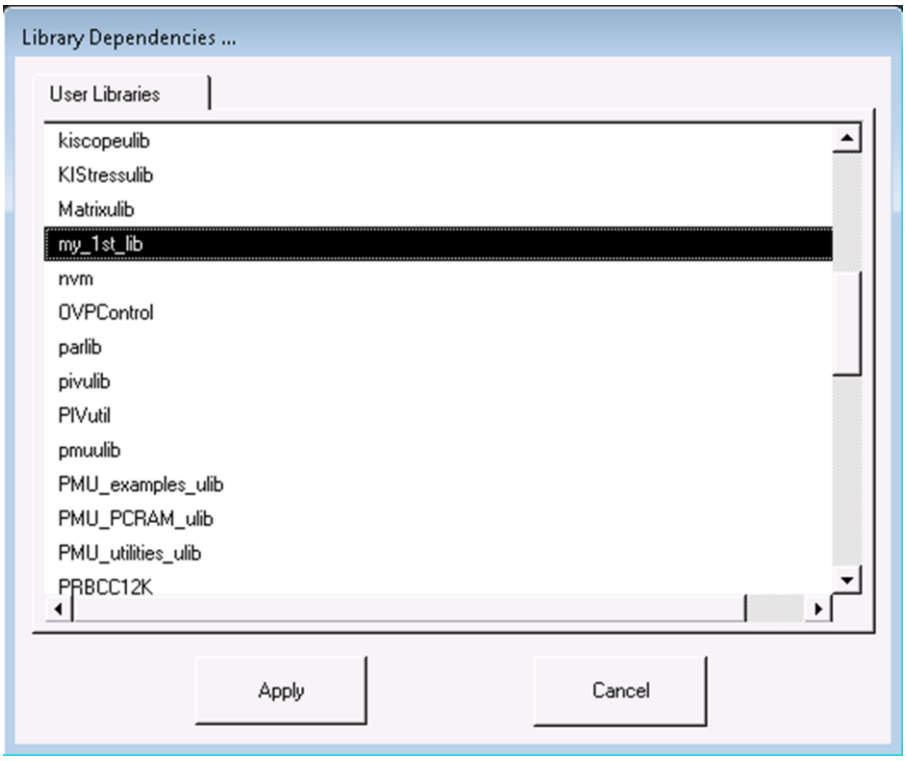
Before building the presently open user module, you must specify all user libraries on which the user module depends (the other user libraries that contain user modules that are called).

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 opens, as shown here.

Figure 456: Library Dependencies list



   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`.

2. Click **Apply**.

Continue with Building the VSweepBeep user module (on page 8-46).

Building the VSweep user module

To build the `VSweepBeep` user module:

1. Save the `VSweepBeep` user module.
2. Click the **Build** tab at the bottom of the dialog box to open the Build tab area.
3. In the **Options** menu, click **Build**. The user module is built. 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 rebuild the module. If necessary, review Finding code errors (on page 8-28).

4. In the **Options** menu, click **Build Library**. The user library builds. You should not see error messages.

Continue with Checking the VSweepBeep user module (on page 8-46).

Checking the VSweepBeep user module

Check the user module as you did in Tutorials 1 and 2, by creating and executing a user test module (UTM) in Clarius. Refer to Checking the user module (on page 8-30).

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, four beeps should sound at the end of execution.

Before proceeding:

1. Connect a 1 kΩ 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. Run 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 Analyze sheet (or the Graph document, if configured). If you connected a 1 kΩ resistor between SMU1 and GNDU, used the default UTM parameter values, and executed the UTM successfully, your results should be similar to the results shown in Checking the VSweep user module (on page 8-39). The current/voltage ratio for each row of results should be approximately 1 mA/V.

Tutorial 4: 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 (V_g) to saturate the MOSFET.
2. Applies 3 V to the drain (V_{d1}) and performs a current measurement (I_{d1}).
3. Applies 5 V to the drain (V_{d2}) and performs another current measurement (I_{d2}).

   Calculates the drain-to-source resistance `rdson` as follows:

   `rdson` = $(V_{d2}-V_{d1}) / (I_{d2}-I_{d1})$

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 is modified in KULT so that ten current measurements are made at V_{d1} and ten more at V_{d2}. The current readings at V_{d1} are averaged to yield I_{d1}, and the current readings at V_{d2} are averaged to yield I_{d2}. Using averaged current readings smooths out the noise.

The modified test, `rdsonAvg`, measures the drain-to-source resistance of a saturated MOSFET. The MOSFET is tested as follows when `rdsonAvg` is executed:

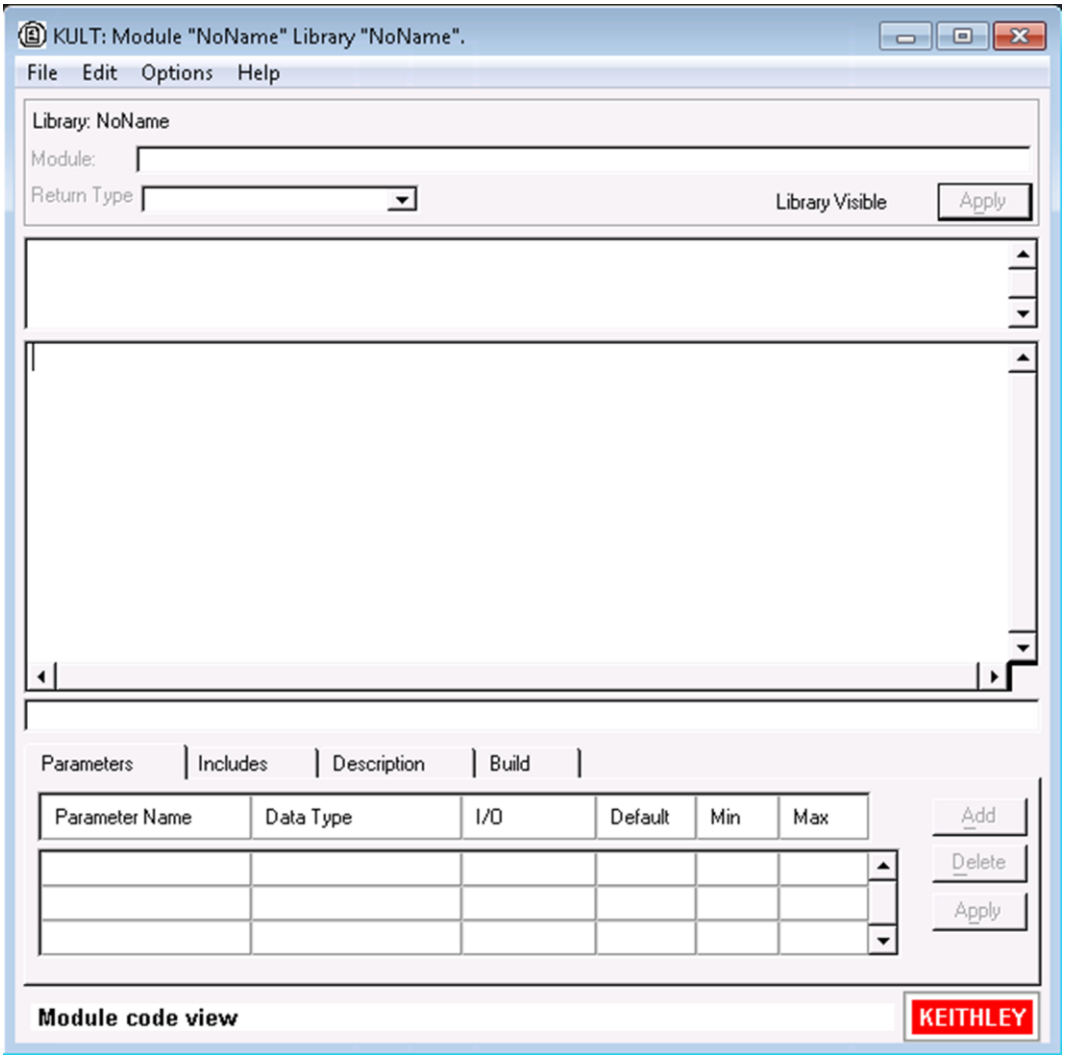
1. Applies 2 V to the gate (V_g) to saturate the MOSFET.
2. Applies 3 V to the drain (V_{d1}) and makes ten current measurements.
3. Averages the 10 current readings to yield a single reading (I_{d1}).
4. Applies 5 V to the drain (V_{d2}) and makes ten more current measurements.
5. Averages the ten current readings to yield a single reading (I_{d2}).
6. Calculates the drain-to-source resistance (`rdsonAvg`) as follows:

   `rdsonAvg` = $(V_{d2}-V_{d1}) / (I_{d2}-I_{d1})$

Open KULT

From the desktop, open the KULT tool by double-clicking the KULT icon. The KULT main window is shown below.

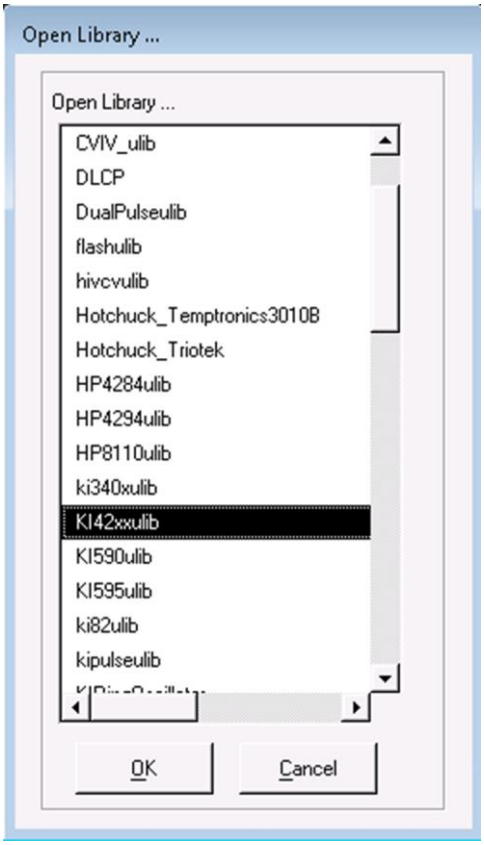
Figure 457: KULT main window



Open the KI42xxulib user library

1. From the **File** menu, select **Open Library**.
2. From the Open Library dialog box, select **KI42xxulib**.

Figure 458: KULT Open Library dialog box

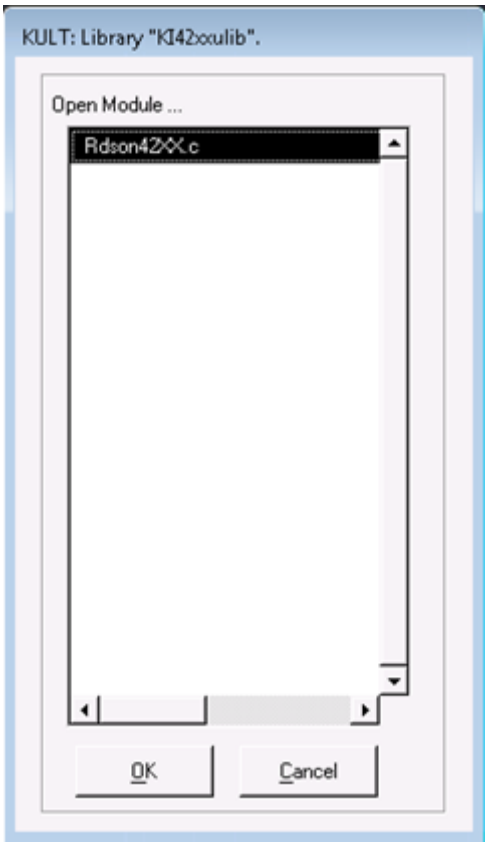


3. Click **OK**.

Open the Rdson42XX user module

1. From the **File** menu, select **Open Module**.
2. From the Open Module window, select **Rdson42XX.c**, as shown in the following figure.

Figure 459: KULT Open Module dialog box



3. Click **OK**. The Rdson42XX module opens.

Copy Rdson42XX to RdsonAvg

You create the new module by copying the `Rdson42XX` module to a module named `RdsonAvg` and then making the appropriate changes to the test module.

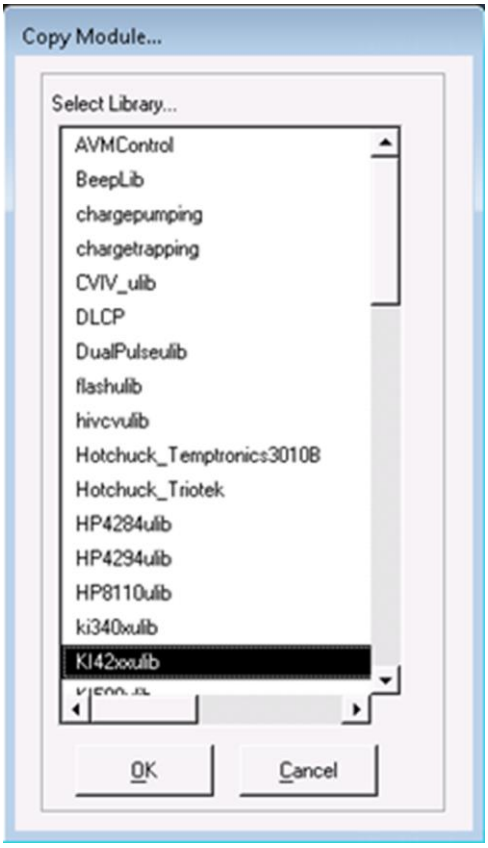
NOTE

When naming a user module, conform to case-sensitive C programming language naming conventions. Do not duplicate names of existing user modules or user libraries.

To create the new module:

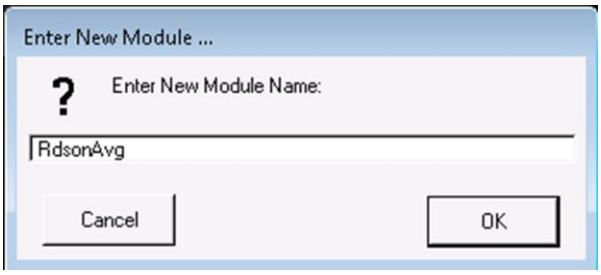
1. From the **File** menu, select **Copy Module**.
2. Select the library for the module. From the Copy Module dialog box, select **KI42xxulib**.

Figure 460: Copy Module dialog box



3. Click **OK**.
4. In the Enter New Module dialog box, type in `RdsonAvg`.

Figure 461: Enter New Module Name dialog box



5. Click **OK**. A reminder that the library using the new module needs to be built is displayed.
6. Click **OK**.

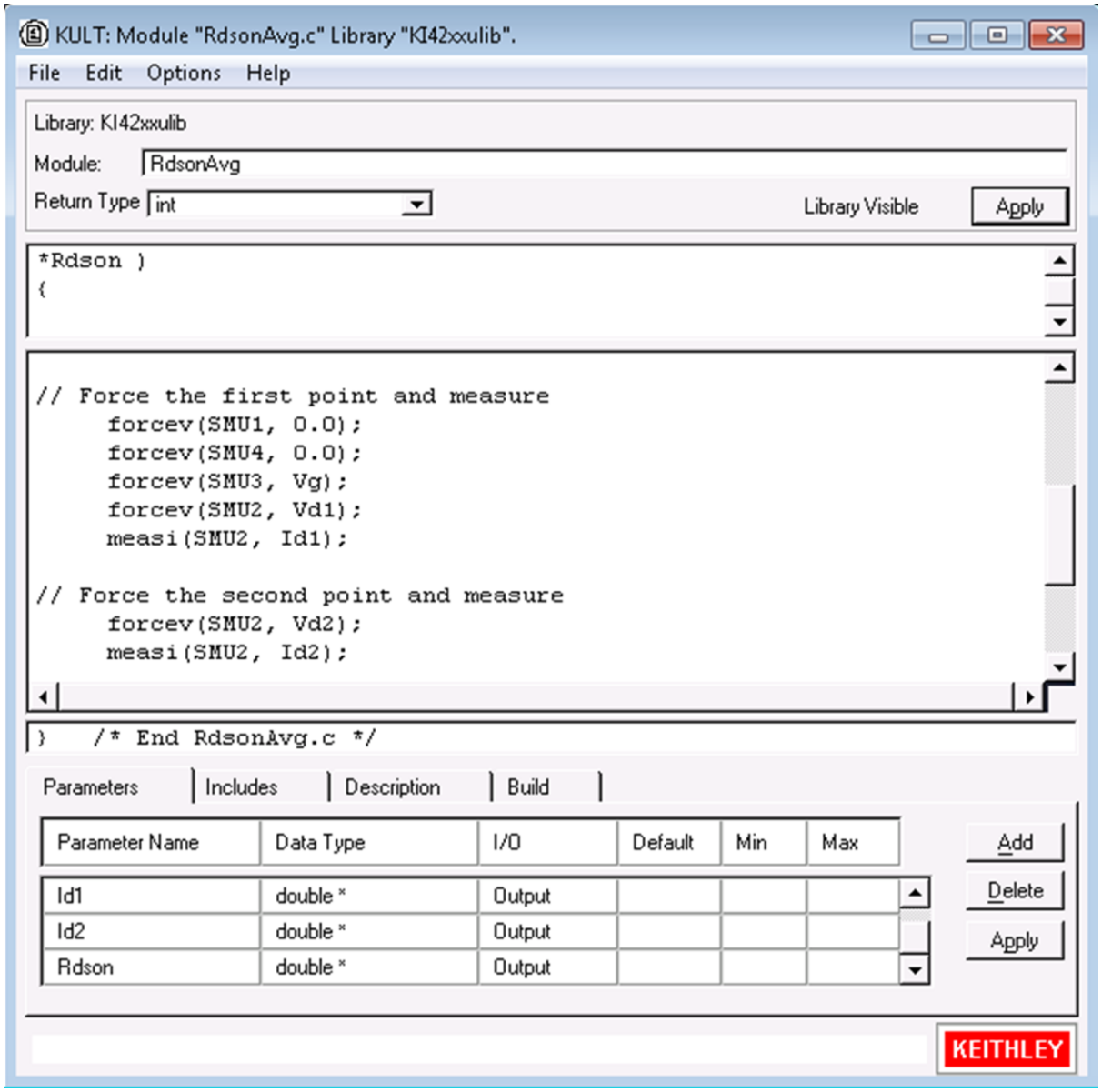
Open and modify the RdsonAvg user module

To open the user module:

1. From the **File** menu, select **Open Module**.
2. Select **RdsonAvg.c** from the Open Module dialog box.

   The `RdsonAvg` module is shown in the following figure.

Figure 462: KULT module window



Modify the user module code

In the user module code, you need to replace the `measi` commands with `avgi` commands. While a `measi` command makes a single measurement, an `avgi` command makes a specified number of measurements, and then calculates the average reading. For example:

```
avgi(SMU2, Id1, 10, 0.01);
```

For the above command, SMU2 makes 10 current measurements and then calculates the average reading (`Id1`). The 0.01 parameter is the delay between measurements (10 ms).

The source code for the module is in the module code area of the window. In this area, make the following changes.

Under `Force the first point and measure`, change the line:

```
measi(SMU2, Id1);
```

to

```
avgi(SMU2, Id1, 10, 0.01); // Make averaged I measurement
```

Under `Force the second point and measure`, change the line:

```
measi(SMU2, Id2);
```

to

```
avgi(SMU2, Id2, 10, 0.01); // Make averaged I measurement
```

Change the line:

```
*Rdson = (Vd2-Vd1)/(*Id2- *Id1);   // Calculate Rdson
```

to

```
*RdsonAvg = (Vd2-Vd1)/(*Id2- *Id1);   // Calculate RdsonAvg
```

Change a parameter name

Change the name of the `Rdson` parameter:

1. Select the Parameters tab.
2. Scroll down to the parameter `Rdson`.
3. Select the name and change it to `RdsonAvg`.
4. Select **Apply**.

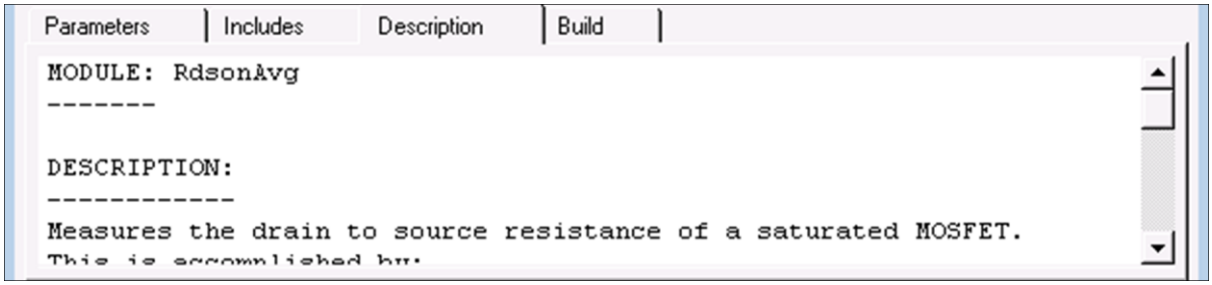
Change the module description

In Clarius, any user test modules (UTMs) that are connected to this user module show the text that is entered on the Description tab in KULT.

To change the module description:

1. Select the **Description** tab.
2. Above `DESCRIPTION`, change `MODULE: Rdson42xx` to `MODULE: RdsonAvg`, as shown in the following figure.
3. Replace all occurrences of `Rdson` with `RdsonAvg`.

Figure 463: User module description



Save and build the modified library

You must save and also rebuild the library to ensure that the new module is available for use by Clarius user test modules (UTMs).

To save and build the user module and library:

1. Select **File > Save Module**.
2. Select **Options > Build Library**. A dialog box is displayed that indicates the build is in process.

Add a new UTM to the ivswitch project

To add rdsonAvg to the ivswitch project:

1. Choose **Select**.
2. Select **Projects**.
3. In the Search box, enter **ivswitch** and select **Search**. The Library displays the I-V Switch Project (ivswitch).
4. Select **Create**. The `ivswitch` project replaces the previous project in the project tree.
5. Select the **Tests** tab.
6. For the Custom Test, select **Choose a test from the pre-programmed library (UTM)**.
7. Drag **Custom Test** to the project tree. The test has a red triangle next to it to indicate that it is not configured.
8. Select **Rename**.
9. Enter **rdsonAvg** and press **Enter**.
10. In the project tree, drag **rdsonAvg** to the `4terminal-n-fet` device, after the `rdson` test.
11. Choose **Configure**.
12. In the Test Settings pane, from the User Libraries list, select **KI42xxulib**.
13. From the User Modules list, select **Rdson42XX**.
14. Select **Save**.

The project tree for the `ivswitch` project with `rdsonAvg` added is shown in the following figure.

Figure 464: Project tree with rdsonAvg added to 4terminal-n-fet device

Advanced KULT features

The following text discusses the advanced features of KULT:

- Managing user libraries (on page 8-56)
- Working with interdependent user modules and user libraries (on page 8-66)
- Debugging user modules using Microsoft® Visual C++®
- Format user module help to display in the Clarius Help pane (on page 8-71)

Managing user libraries

This section addresses the following topics:

- Updating and copying user libraries using KULT command-line utilities (on page 8-56) 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 (on page 8-59) 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 user interface.

Updating and copying user libraries using KULT command-line utilities

This section describes two useful command-line utilities, `kultupdate` and `kultcopy`.

Updating user libraries using kultupdate

You must use the `kultupdate` utility 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 rebuilds each user module in the library and rebuilds the library as well.

Usage

```
kultupdate <library_name> [options]
```

Options

You can place any of the following options at the `[options]` position in the command:

- `-dep <library_dep_1>...[library_dep_6]`
  Specifies up to six libraries on which `library_name` depends.
- `-hide`
  Hides *`library_name`* so that it is not visible in Clarius.
- `+hide`
  Shows *`library_name`* so that it is visible in Clarius.

Example

Update the `KI590ulib` library in the active user-library directory, which depends on the `Winulib` library:

```
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. `kultcopy`:

- 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.

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.

- Performs `kultupdate` so that the user library is immediately ready for use (refer to Updating user libraries using kultupdate (on page 8-57)).

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 on which *library_name* depends.
- `-hide`
  Hides *library_name* so that it is not visible in Clarius.
- `+hide`
  Shows *library_name* so that it is visible in Clarius.

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 user interface. 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:
  - `gui`
  - `new_lib`
  - `bld_lib`
  - `del_lib`
  - `new_mod`
  - `add_mod`
  - `del_mod`
  - `bld_lib`
  - `zip`
  - `unzip`
  - `help`
- 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]`
  - `build_type`
- These options are described in the following descriptions of individual subcommands.
- If appropriate to the commanded action, [*module*] specifies the name of the involved user module.

The sections that follow describe the subcommands.

gui subcommand

The `gui` subcommand launches the KULT editor.

Usage

```
kult gui [option] [type]
```

The -`build_type` option may be placed at the [*`options`*] position in the command. The following *`[type]`* options are available:

- `Release`
  Default option. This option builds the library more efficiently than the `Debug` option.
- `Debug`
  Use this option if you want to use an integrated development environment, such as Visual Studio Code, to debug your source code.

Example

```
kult gui -build_type Release
```

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.

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.

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 in the active user-library directory. For more information about dependent libraries, refer to Working with interdependent user modules and user libraries (on page 8-66).

- `+hide`
  Hides *library_name* so that it is not visible in Clarius.
- `-hide`
  Shows *library_name* so that it is visible in Clarius.

NOTE

The `C:\s4200\kiuser\usrlib\<library name>\build` folder is created when you run the `bld_lib` subcommand or select the **Build Library** menu option. This folder can be safely deleted for debugging purposes.

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.

new_mod subcommand

The `new_mod` subcommand lets you create a new module in a user library. Its action is equivalent to the following steps in KULT:

- Starting KULT
- Selecting **File > Open Library > <library_name>**
- Clicking **OK**
- **Selecting File > New Module**
- **Enterin**g a new module name
- Clicking Apply
- Selecting **File > Exit**

Usage

```
kult new_mod -l<library_name> <module>
```

The *<module>* module is created in the *<library_name>* library.

Where:

- *<library_name>* is the target library into which *<module>* is to be created. It must be in the active user-library directory.
- *<module>* is the new module name.

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
- Selecting **File > Exit**

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*>.

Usage

```
kult add_mod -l<library_name> [-d<source_lib_path>\source_lib_name>\src] <module>
```

Where:

- <*library_name*> is the target library into which <*module*> is to be copied. It must be 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*>.

del_mod subcommand

The `del_mod` subcommand lets you delete a module from the command line. Its action is equivalent to the following steps in KULT:

- Starting KULT
- Selecting **File > Delete Module**
- Selecting a user module to be deleted
- Clicking **OK**
- Selecting **File > Exit**

Usage

```
kult del_mod -l<library_name> <module>
```

Where:

- *<library_name>* is the target library from where *<module>* will be deleted. It must be in the active user-library directory.
- *<module>* is the name of the module to be deleted.

zip subcommand

The *zip* subcommand creates a `.zip` file for a user library.

Usage

```
kult zip -l<library_name> [password] <zipfile_name>
```

The *<library_name>* user library is created in the active user-library directory.

The `[password]` parameter is optional.

unzip subcommand

The *unzip* subcommand unzips a file containing a KULT library.

Usage

```
kult unzip [-dest_path] [password] <zipfile_name>
```

Where:

- [-dest_path] is the target directory where the file will be unzipped.
- [password] is required if the file was compressed using the password parameter in the zip subcommand.

The <*zipfile_name*> archive is unzipped in the active user-library directory unless the [-dest_path] parameter is specified. The [-dest_path] parameter should not be used when you import a user library.

help subcommand

The help subcommand displays all usage information for subcommands and options.

Usage

```
kult help
```

Working with interdependent user modules and user libraries

KULT allows a user module to call other user modules. A called user module can be in the same user library as the calling module, or can be in another user library. When the module that you are creating calls a module in another user library, you must:

1. Select **Options > Library Dependencies**.
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 section provides guidance.

NOTE

The user libraries to be linked are saved so that future rebuilds do not require the dependencies to be reselected. This information is stored in the `<`*`library_name`*`>_modules.mak` file in the `%KI_KULT_PATH%\<`*`library_name`*`>\kitt_obj` directory.

- Structure dependencies hierarchically to avoid circular dependencies, and then build the dependent user libraries in the correct order. The next two sections provide guidance.

Structuring dependencies hierarchically

You can avoid user library circular dependency by calling user libraries in a hierarchical design, as illustrated in "Hierarchical design for user-library dependencies" below.

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 do 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 in `liba`. This avoids duplicate user module names and prevents confusion with similarly named modules that are in other user libraries and source files. When you execute the **File > Copy Library** 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 > Copy Module** command.

In the following table, the series of coded user modules amplifies the hierarchical dependencies shown in the following figure.

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();`
			`}`

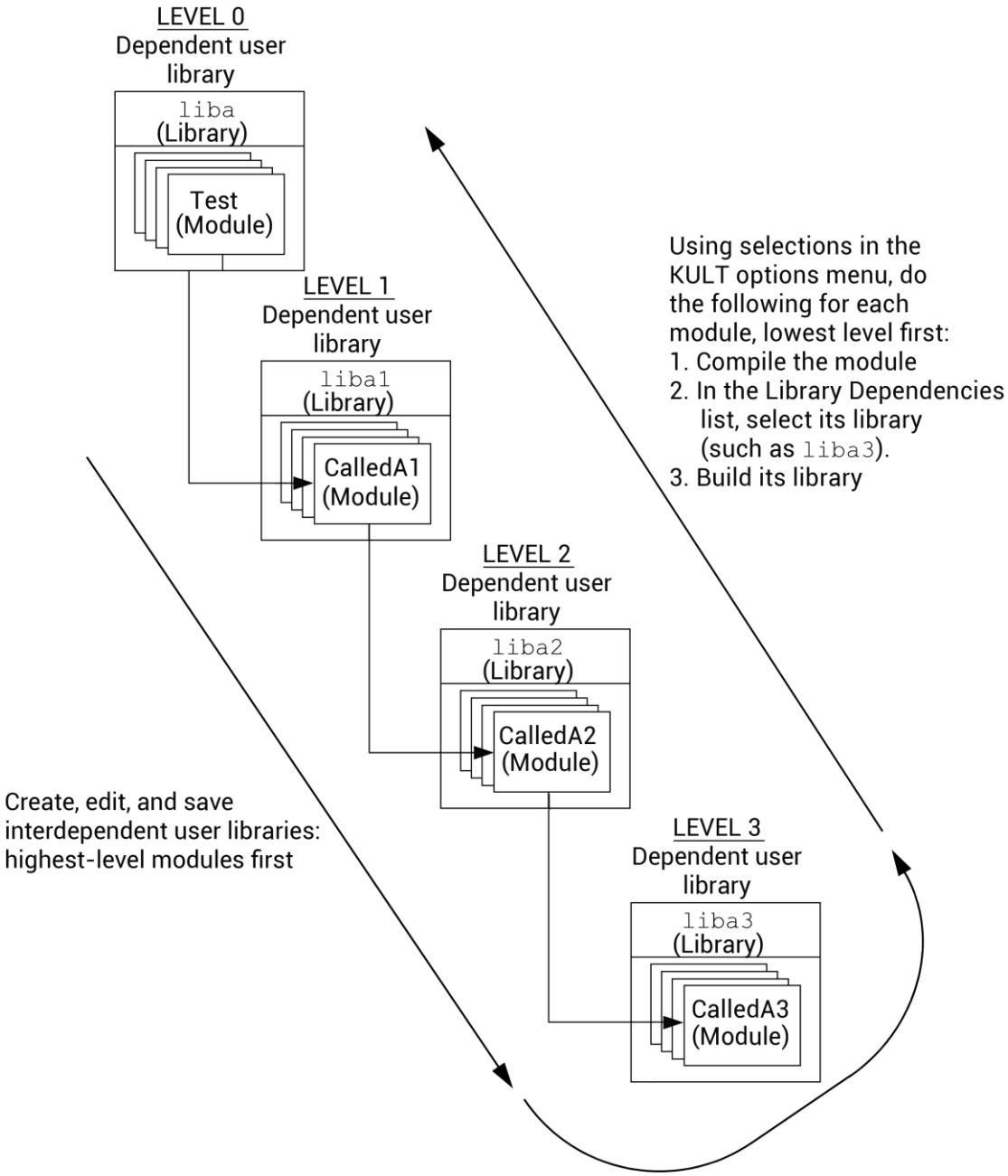
1	liba1	CalledA1	`void CalledA1(void)`
			`{`
			`  printf("In liba1, calling CalledA2()\n");`
			`  CalledA2();`
			`}`

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 in `libal`. In turn, a user module in `liba1` calls a user module in `liba2`. Finally, a user module in `liba2` calls a user module in `liba3`.

Figure 465: 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 illustrates the correct reverse build order for the dependencies shown in the table and figure in Structuring dependencies hierarchically (on page 8-67). 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 build. You do not need to repeat builds that are already complete up to a given level of dependency.

Build the Level 3 user module and user library:

1. Build the saved `CalledA3` user module, which is in the `liba3` user library (in the KULT **Options** menu, select **Build**).
2. Build the `liba3` user library (in the KULT **Options** menu, select **Build**).

Build and set dependencies for the Level 2 user module and user library:

1. Build the saved `CalledA2` user module, which is in the `liba2` user library.
2. Set the dependencies for the `CalledA2` user module:
    a. Select **Options > Library Dependencies**.
    b. Select `liba3` from the Library Dependencies list box.
    c. Click **Apply**.
3. Build the `liba2` user library.

Build and set dependencies for the Level 1 user module and user library:

1. Build the saved `CalledA1` user module, which is in the `liba1` user library.
2. Set the dependencies for the `CalledA1` user module:
    a. Select **Options > Library Dependencies**.
    b. Select **liba2** from the Library Dependencies list box.
    c. Click **Apply**.
3. Build the `liba1` user library.

Build and set dependencies for the Level 0 user module and user library:

1. Build the saved Test user module, which is in the `liba` user library.
2. Set the dependencies for the test user module:
    a. Select **Options > Library Dependencies**.
    b. Select **liba1** from the Library Dependencies list box.
    c. Click **Apply**.
3. Build the `liba` user library.

This reverse hierarchical build order results in a linking scheme that satisfies the dynamic linking requirements of Microsoft® Windows®.

Format user module help to display in the Clarius Help pane

If your user module includes a help description, but it is not set up for HTML, when you create a UTM in Clarius, the Help pane displays the Open UTM Comments button. If you click this button, text from the Description tab in KULT is displayed in an ASCII browser window.

You can set up this help to display as formatted HTML in the Help pane using PHP Markdown Extra tools. On the first line of the description, add the following stylesheet and MarkdownExtra code:

```
<!--MarkdownExtra-->

<link rel="stylesheet" type="text/css"
href="http://clariusweb/HelpPane/stylesheet.css">
```

In order to see the help in Clarius, you must build the UTM and rebuild the library after entering the Markdown code.

To format the text, you can use some of the following options:

- Create a first level heading: Place `=====` under a line to center and bold the line. (You can use any number of `=` characters.)
- Create a second level heading: Place `--------` under a line to bold the line. (You can use any number of `-` characters.)
- To create list: Insert a blank line before the start of the list, then use 1., 2., and so on to number each item in the list.
- Italicize text: Place `*` before and after the text to be italicized.
- Display text in a fixed-width font: Put 6 spaces before each line of the text or use four tilde characters (`~~~~`) before and after the lines of text.
- You can make changes to the `.c` file of the user module with KULT or a text editor. After saving changes, you can refresh the Help pane to view the changes. To refresh the help pane, select another project tree object and then return to the UTM.

An example of the code entered in the Description tab is shown in Documenting the user module (on page 8-26). An example of the result in the Help pane in Clarius is shown in Checking the user module (on page 8-30).

For information on additional formatting options, refer to the PHP Markdown Extra website of Michel Fortin (michelf.ca/projects/php-markdown/extra/).

Creating project prompts

KULT provides user modules that you can use to create dialog boxes to pause a test sequence with a prompt. These dialog boxes are available as user modules, shown in the table below.

You define the text message for the prompt. When one of these user modules is run, the test sequence pauses. The test sequence continues when a button on the dialog box is selected.

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 boxes

The `Winulib` user library has user modules for six action or decision dialog boxes and one input dialog box. The dialog box, with example prompts, are shown in Dialog box formats (on page 8-73).

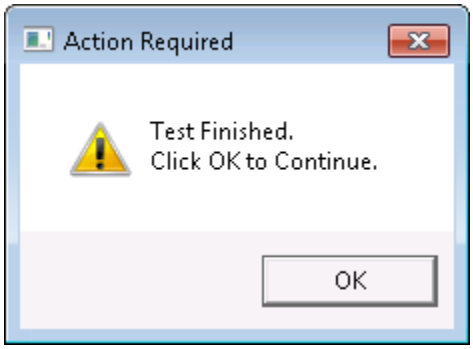
The text message for a prompt is entered by the user into the user module. See Winulib user-library reference (on page 6-399) for details on the user modules.

NOTE

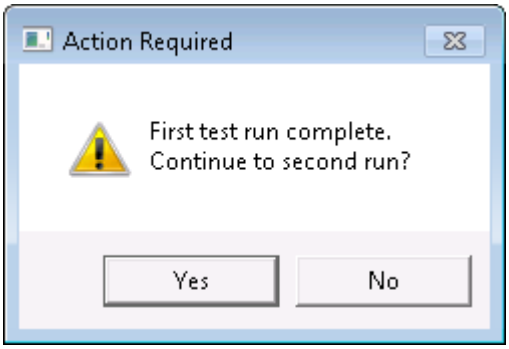
An example using the OK dialog box is provided in Dialog box test examples (on page 8-74).

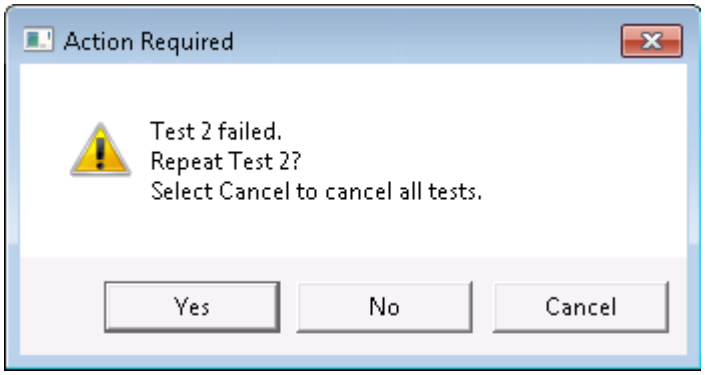
Dialog box formats

The OK dialog box in the following figures has only one button. You can use this dialog box 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.



The other dialog boxes have two or three buttons, as shown in the following examples. When a button on a dialog box is clicked, a status value that corresponds to that button is placed in the Analyze sheet for the action. If there are input parameters, the entries for the input parameters are placed in the Analyze sheet for the action. You can pass a parameter value into a user-created routine.





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 in the Analyze sheet 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 (on page 8-74).

Dialog box test examples

The following examples demonstrate how you can use dialog boxes in a test sequence.

Example: Announce end of test

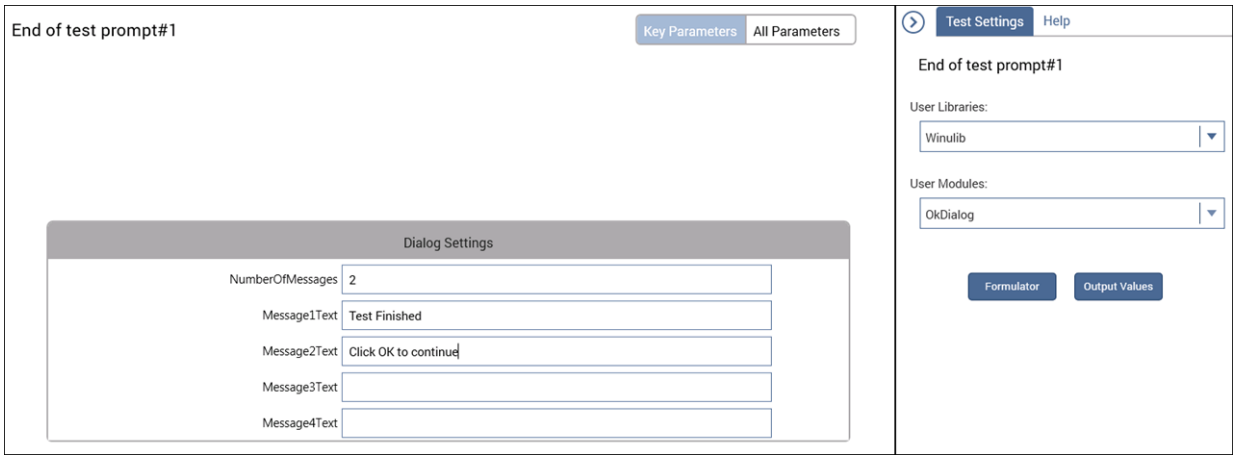
For this example, you will create a user test module (UTM) that uses the OK dialog user module. This dialog box announces the end of a test sequence. You can use this UTM in any project at the end of any test sequence.

To create an end-of-test announcement:

1. In the Clarius project tree, select the last test. The announcement will occur after this test.
2. Choose **Select**.
3. Select the **Tests** tab.
4. For the Custom Test, select **Choose a test from the pre-programmed library (UTM)**.
5. Drag **Custom Test** to the project tree. The test has a red triangle next to it to indicate that it is not configured.
6. Select **Rename**.
7. Enter the name **End of test prompt**.

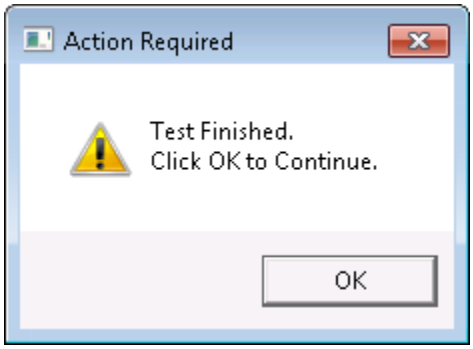
8. Select **Configure**.
9. In the Test Setting pane on the right, set the User Libraries to **Winulib**.
10. Set User Modules to **OkDialog**.
11. For NumberOfMessages, enter **2**.
12. For Message1Text, enter **Test Finished**.
13. For Message2Text, enter **Click OK to continue**. An example is shown in the graphic below.

Figure 466: New UTM using OkDialog user module



14. Select **Save**.

When you run the test sequence, the end of test dialog box is displayed, as shown in the graphic below. Select **OK** to continue.



Section 9

KULT Extension for Visual Studio Code

In this section:

Introduction

Keithley's KULT Extension for Visual Studio Code gives you the ability to write, compile, and debug user libraries outside of KULT. Combining the Visual Studio Code user-friendly editor with KULT creates an integrated development environment (IDE).

This guide details the download, installation, and setup process for Visual Studio Code and the KULT Extension. A connection to the internet on the unit is recommended but is not necessary for the installation process.

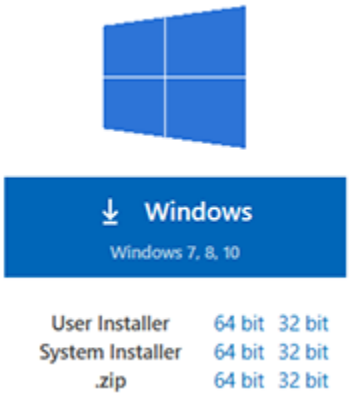
For units connected to the internet

Install Visual Studio Code

To install Visual Studio Code:

1. Go to the Visual Studio download site (code.visualstudio.com/download) from the instrument. Select either 32 or 64 bit version. Select **Download**.

Figure 467: Visual Studio download

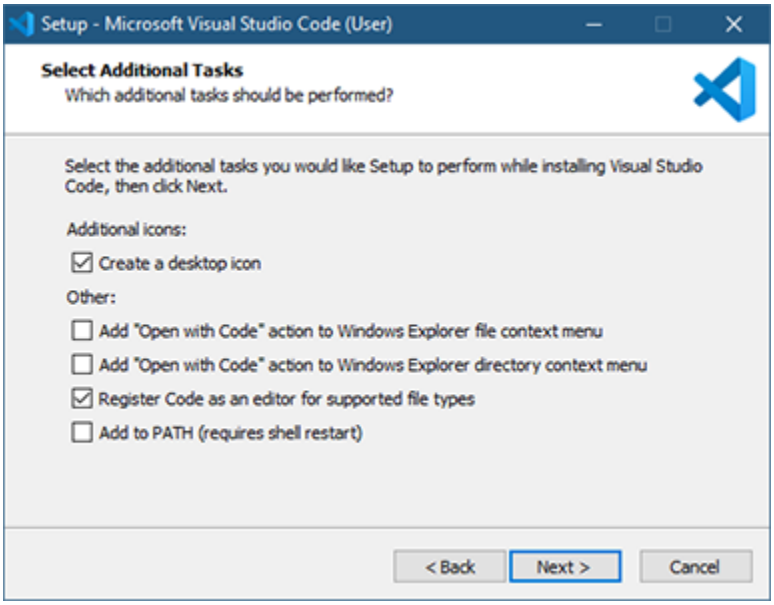


NOTE

The Visual Studio Code extension comes with the Clarius software suite.

2. Select the downloaded executable to open the installer. Select the box to accept the license agreement and select **Next**.
3. Select **Next** to install Visual Studio Code to the default location.
4. Select **Next** again to select the default Start Menu folder.
5. Under Select Additional Tasks, select the box next to **Add to PATH.** The other options can be selected if desired. You can check the boxes to:
   - **Create a desktop icon:** Creates an icon on the desktop to access Visual Studio Code easily.
   - **Register Code as an editor for supported file types:** Sets Visual Studio Code as an option to open files.
   - **Add Open with Code:** Adds an Open with Code option when you right click on any file or folder.

Figure 468: Select additional tasks



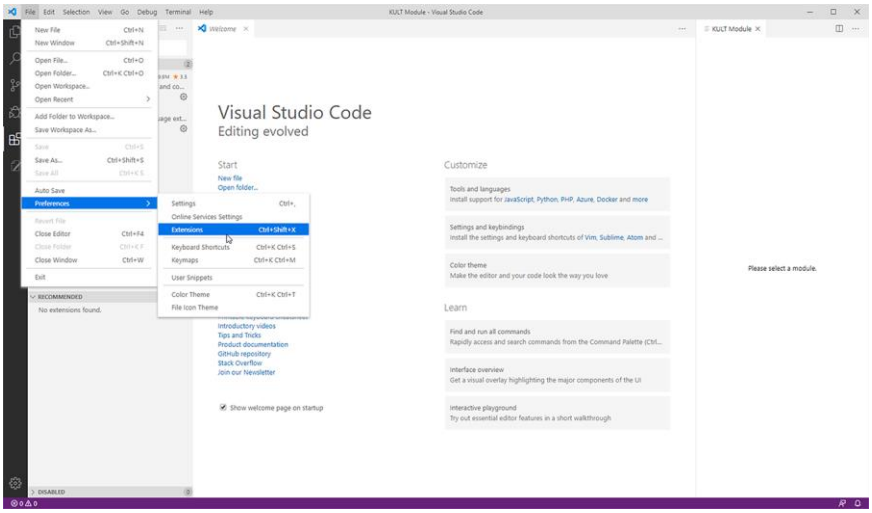
6. After selecting the desired additional tasks, select **Next**.
7. Select **Install.**
8. Visual Studio Code will install. When prompted, select **Finish** to complete the process.
9. The installer window will close and Visual Studio will open.

Install extensions for Visual Studio Code

To install Visual Studio Code extensions:

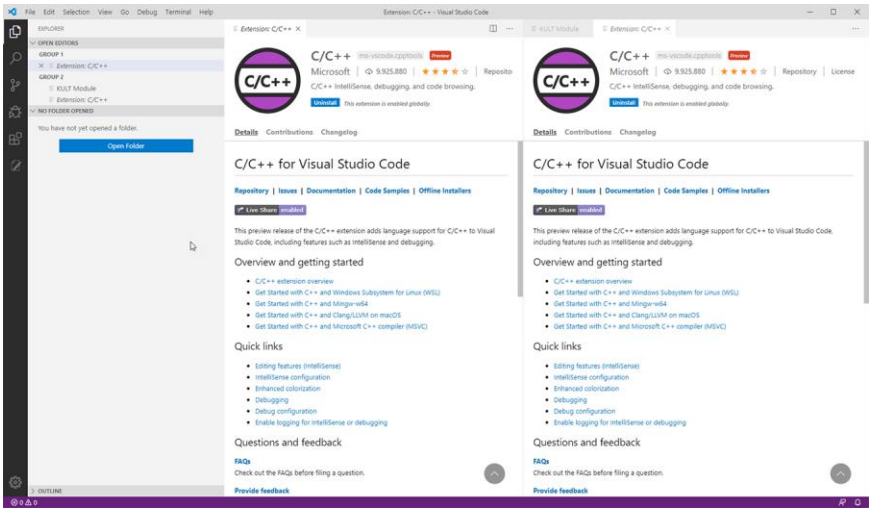
1. Select the extension icon on the left side of the Visual Studio Code window or go to **File > Preferences > Extensions** to open the Extension Marketplace.

Figure 469: Find Extensions



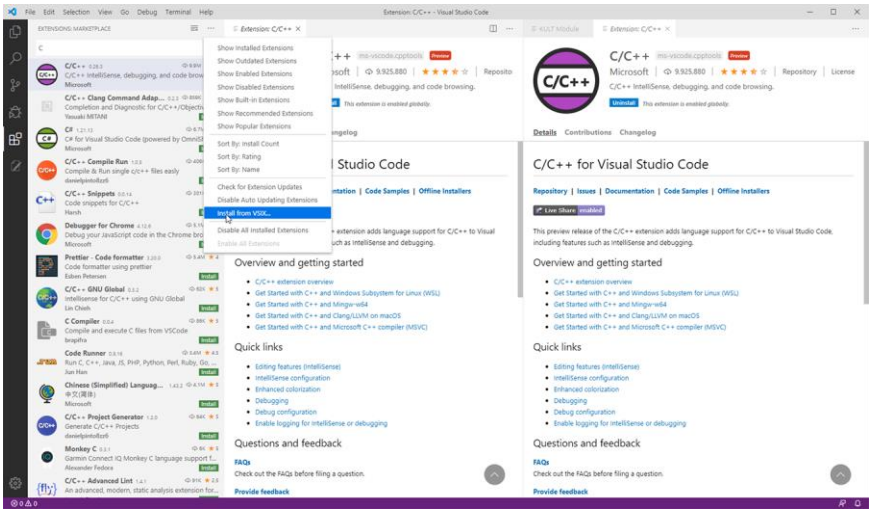
2. Search for “C++” in the Marketplace and select C/C++. The lowest icon on the left side of the GUI will also open the Marketplace.
3. Select **Install** to install the Microsoft C/C++ Extension

Figure 470: Install the C/C++ Extension



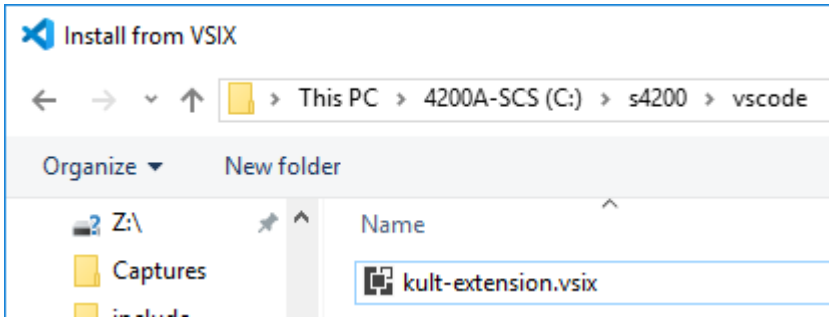
4. Select the **ellipsis button** at the top of the Extensions Marketplace to open more options for installing.
5. To open a file chooser window, select **Install** from VSIX.

Figure 471: Install from VSIX



6. Open `C:\s4200\vscode\kult-extension.vsix`. This will install the KULT Extension to Visual Studio Code.

Figure 472: Install the KULT Extension to Visual Studio Code



7. Close Visual Studio Code and reopen to complete the installation and enable all extensions.
8. Continue to the section Setting up Visual Studio Code for Library Development (on page 9-11).

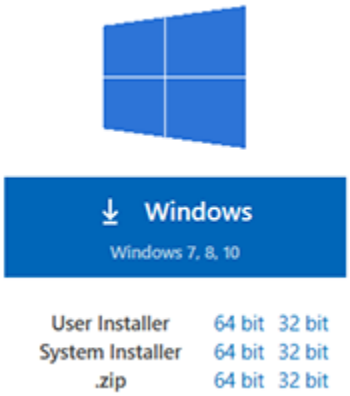
For units not connected to the internet

Install Visual Studio Code

To Install Visual Studio Code:

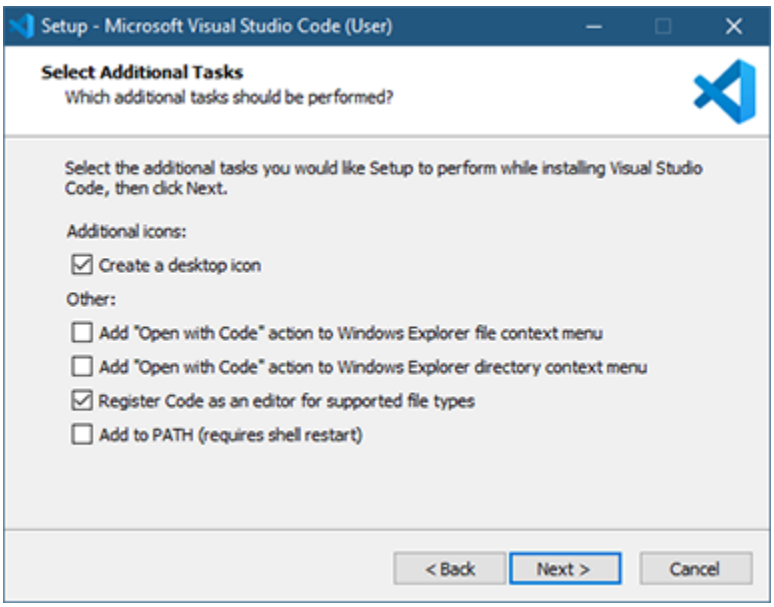
1. Go to the Visual Studio download site (code.visualstudio.com/download) using a separate PC. Select either the 32 or 64 bit version of the user installer. Click **Download**.

Figure 473: Visual Studio download



2. Copy the downloaded `.exe` file to a flash drive.
3. Use the flash drive to transfer the installer to the 4200A.
4. Select the executable to open the installer. Select the box to agree to the license and click **Next**.
5. Select **Next** to install Visual Studio to the default location.
6. Select **Next** again to select the default Start Menu folder.
7. Under Select Additional Tasks select the box next to **Add to PATH;** which will add it to the PATH**.** The other options can be selected if desired. You can check the boxes to:
   - **Create a desktop icon:** Creates an icon on the desktop to access Visual Studio Code easily.
   - **Register Code as an editor for supported file types:** Sets Visual Studio Code as an option to open files.
   - **Add Open with Code:** Adds an Open with Code option when you right click on any file or folder.

Figure 474: Select additional tasks



8. After selecting the desired additional tasks, select **Next**.
9. Select **Install.**
10. Visual Studio Code will install. When prompted, select **Finish** to complete the process.
11. The installer window will close and Visual Studio will open.

Install extensions for Visual Studio code

To install Visual Studio Code extensions:

1. To download the Microsoft C/C++ Extension, go to github.com and download the file named `cpptools-win32.vsix` under assets in the latest version, to your PC. This extension contains all of the outside dependencies that are usually downloaded the first time the extension is used.

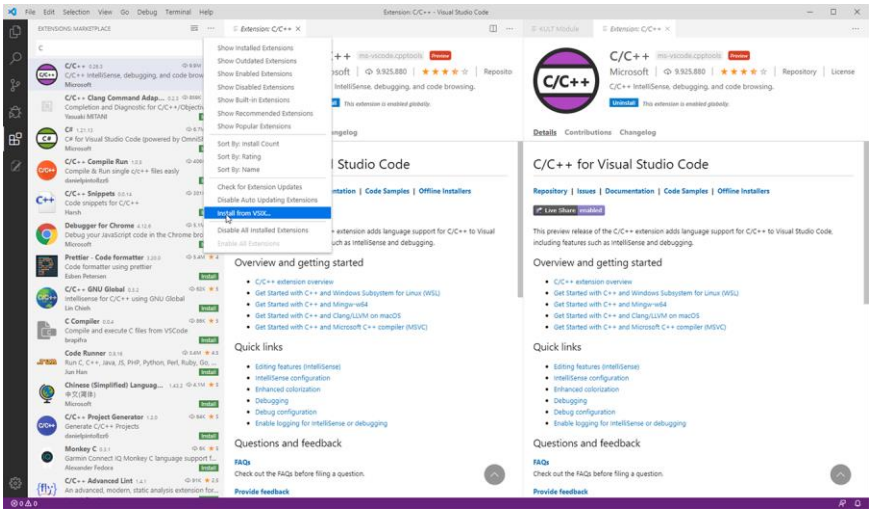
Figure 475: Microsoft C/C++ extension

2. Copy the *.vsix* file to a USB drive.
3. Move the `C:\s4200\vscode` folder on the unit.
4. Open the Extension Marketplace in Visual Studio Code by selecting the box icon.

Figure 476: Extensions marketplace

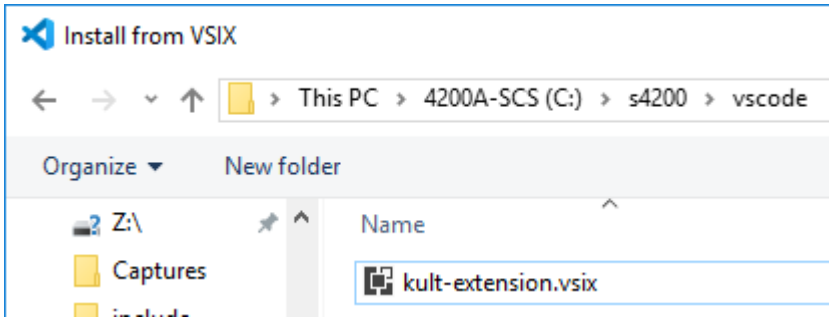
5. Select the ellipsis button at the top of the Extensions Marketplace to open more options for installing. Select **Install from VSIX…**
6. Select the `cpptools-win32.vsix` file. The extension will be installed.

Figure 477: Install from VSIX



7. Repeat the same process in Step 5 and Step 6 for the KULT Extension, located at `C:\s4200\vscode\kult-extension.vsix`. The installation will begin automatically.

Figure 478: Install the KULT Extension to Visual Studio Code



8. Close Visual Studio Code and reopen to complete the installation and enable all extensions.
9. Continue to the section Setting up Visual Studio Code for Library Development (on page 9-11).

Updating the KULT Extension after Installing Clarius

The KULT extension must be uninstalled and reinstalled after updating to a new version of Clarius. For installation of Visual Studio Code and first time installation of the extension, see the manual.

To uninstall the KULT extension:

1. Open Visual Studio Code and go to the Extension Marketplace.
2. Select the **KULT extension**.
3. Select the blue **Uninstall** button next to the extension icon in the editor window.
4. Close Visual Studio Code.

To reinstall the KULT-Extension:

1. Open Visual Studio Code and go to the Extension Marketplace.
2. Select the **More Actions** button at the top of the side bar to open a menu of extension actions.
3. Select **Install from VSIX.**
4. Select the KULT extension file, *`C:\s4200\vscode\kult-extension.vsix`*.
5. Once the installation is complete, close Visual Studio Code.
6. Reopen Visual Studio Code. The KULT extension will be fully enabled.

Setting up Visual Studio Code for library development

Opening the user library in Visual Studio Code

Open the entire user library so that you can access all of the all libraries without switching folders. Single user libraries can be opened by opening the folder of the desired user library. Visual Studio Code will reopen the last folder opened on startup.

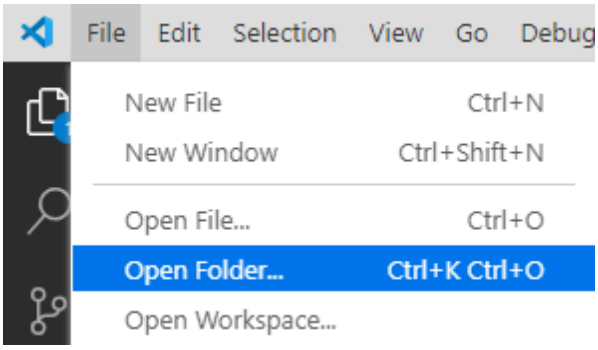
NOTE

The Visual Studio Configuration files will need to be created the first time you open a single library. See the section Creating the Visual Studio Code configuration files (on page 9-12) for more information.

To open the user library folder:

1. Go to **File > Open Folder** to select a folder to open in Visual Studio Code. Selecting a folder is required by Visual Studio Code to use the KULT extension.

Figure 479: Open folder dialogue box



2. Open the usrlib folder, `C:\s4200\kiuser\usrlib`.
3. Select `.vscode, then` **Select Folder.**

Creating the Visual Studio Code configuration files

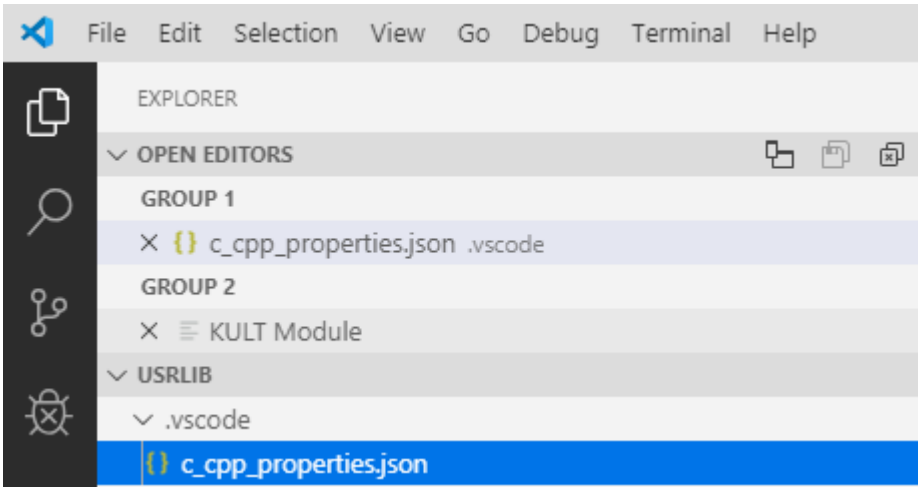
Visual Studio Code configuration files adopt the features of Visual Studio Code to be used with Keithley User Libraries. The `c_cpp_properties.json` configuration file controls the Intellisense features of the C/C++ Extension from Microsoft, such as compiler specific syntax checking and header file paths. Intellisense errors may occur if these features are not configured for Keithley user libraries. The errors will not affect compilation or code execution in Clarius, but extraneous intellisense errors may make code difficult to troubleshoot. The `launch.json` file has configuration settings for GNU Debugger (GDB), which is used when debugging. Debugging involves attaching GDB to Clarius's running process `UTMServer.exe`.

A `.vscode` folder is created with the configuration files in the folder that is open (workspace path). All files in the workspace path will reference these configuration files. If the `usrlib` folder is open, the configuration files can be created once, and all of the libraries will use them. If you are opening individual libraries, these files will need to be created for each library the first time it is opened. Created configuration files can be edited later. The file `settings.json` contains Visual Studio Code workspace level configuration settings.

To create the C/C++ Intellisense configuration file:

1. Open the Command Pallette by clicking **View > Command Pallete**.
2. Search KULT to filter for KULT Extension commands
3. Select the command `KULT: Create C/C++ Intellisense Configuration File`**.** This generates the `c_cpp_properties.json` configuration file and places it in the `.vscode` folder in the working directory. The command will not overwrite an existing configuration file.

Figure 480: Generate the c_cpp_properties configuration file



4. Select the `c_cpp_properties.json` file in the File Explorer side bar to open it in the editor.
5. Add any extra paths to the header files in the *includePath* settings. Paths are entered in quotes and separated by commas. This file can be updated with additional paths at any point.

NOTE

The paths to the include folder, usrlib folder and complier are necessary for most user libraries and are automatically entered. Deleting these paths will cause Intellisense errors in factory-user libraries.

Figure 481: c_cpp_properties

To create the launch configuration file:

1. Go to **Debug > Add Configuration** to add a launch configuration to the .vscode folder.
2. Select the **KULT Attach Process**. The `launch.json` file will be added with the settings to attach to the UTM Server. It does not need to be edited.

Figure 482: KULT Attach Process

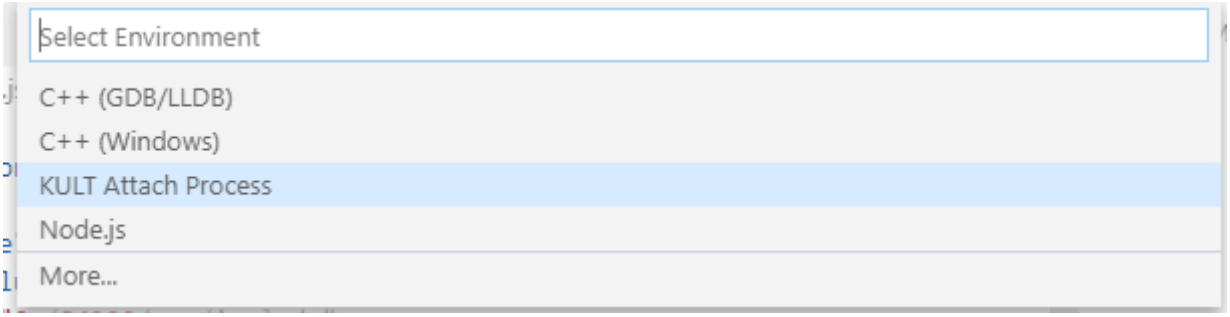
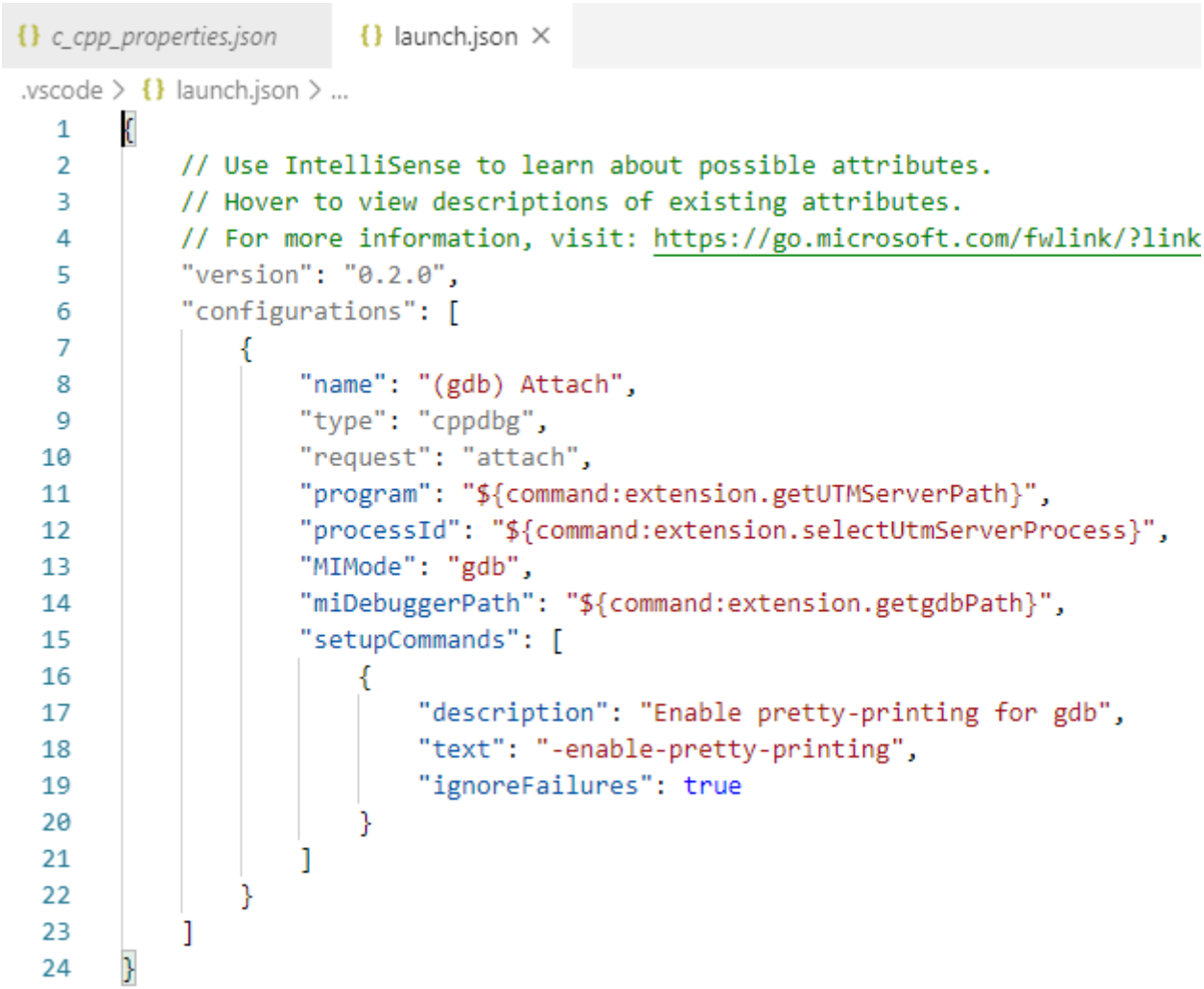


Figure 483: Launch. json



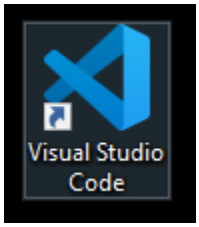
Visual Studio code overview

This section covers the features of Visual Studio Code. To learn more about Visual Studio Code as an editor, visit Visual Studio Code (code.visualstudio.com/).

Opening Visual Studio Code

Visual Studio Code can be opened using the desktop icon or by searching for Visual Studio Code in the Windows Start Menu

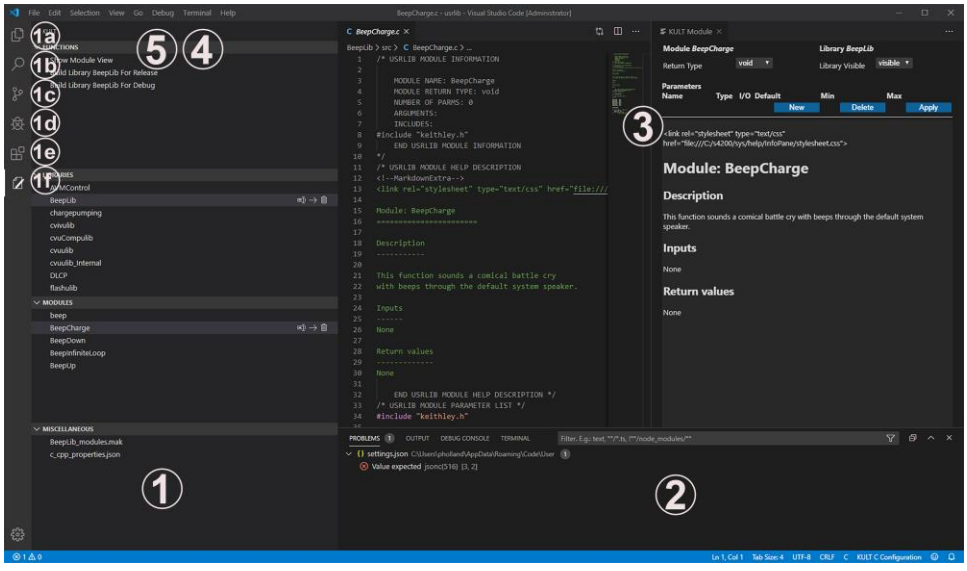
Figure 484: Visual Studio Code



The main GUI

Important parts of the main window are labeled below.

Figure 485: Main GUI



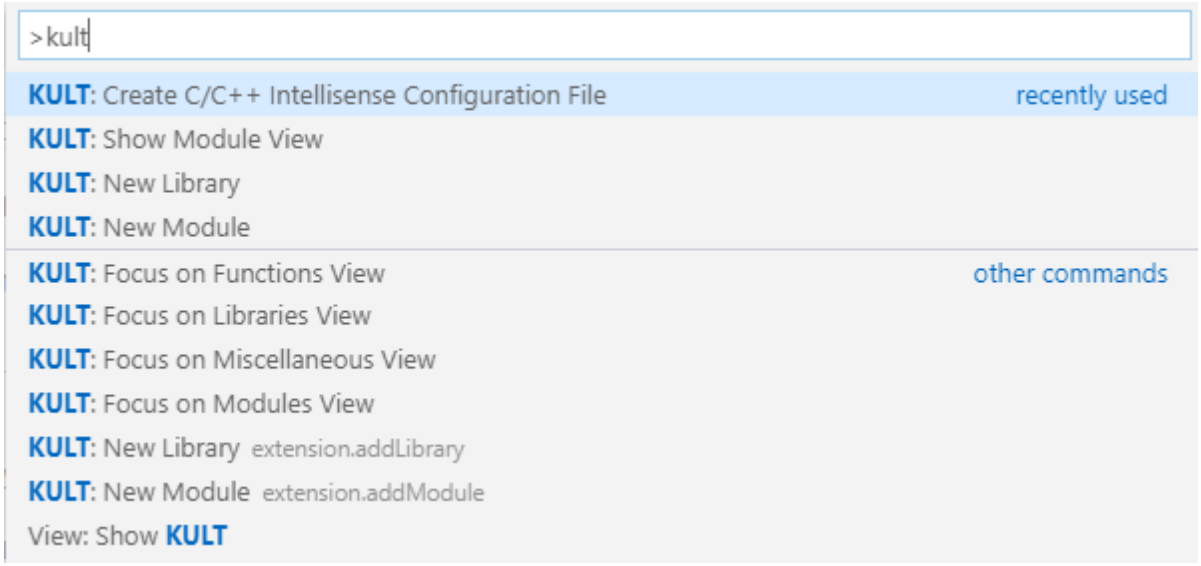
1	Side bar
2	Panels
3	Editor
4	Terminal menu
5	Debug menu

1. **Side Bar**: Displays different views to assist editing. You can switch views using the icons in the activity bar next to the side bar. The icons and their functions are described below from top to bottom.
   a. **Explorer**: Displays all files in the Visual Studio Code working directory.
   b. **Search**: Opens files for keywords and replace if desired.
   c. **Debug**: Allows users to monitor variables, threads and breakpoints during debug mode.
   d. **Source control**: Not used by the KULT Extension.
   e. **Extension marketplace**: Install and uninstall extensions to Visual Studio Code.
   f. **KULT**: Displays libraries and modules, build functions, and other useful tools for developing libraries. More information on the KULT side bar is in the section The KULT Side Bar (on page 9-19).
2. **Panels**: These four panels manage output to users.
   a. **Output**: Certain non-build KULT Extension functions (for example Clean Library) will output messages here.
   b. **Terminal**: Displays output from build tasks with same format as in KULT.
   c. **Debug console**: Used for expression evaluation and other tools during debugging.
   d. **Problems**: Displays various errors found before and during compilation. Clicking on the error message in the Problems panel will display the line of code in the editor. Error correction suggestions are offered by clicking the lightbulb over the offending line.
3. **Editor**: The center area is where the KULT Module and other files can be edited. Right clicking the tabs for the open files at the top allows you to split the area to view multiple files at once.
4. **Terminal menu**: Selecting **Run Task** on this menu tab shows all of the tasks that are available in Visual Studio Code. This list includes non-build tasks. Selecting **Run Build Task** displays a subset of the tasks specific to building. KULT build tasks are specific to a library which can be selected by opening a library module.
5. **Debug menu:** In this tab you can open or add configurations for debugging. You can also start a debugging session to debug a module. Additional options for working with breakpoints are also located here.

The Command palette

The command palette provides access to all Visual Studio Code commands. The command palette can be opened by pressing **Ctrl+Shift+P** or by going to **View > Command Palette** on the top menu bar. All of the important commands for the KULT extension can be accessed by the command palette; all available commands are displayed, but the KULT commands can be filtered out by searching KULT in the search bar.

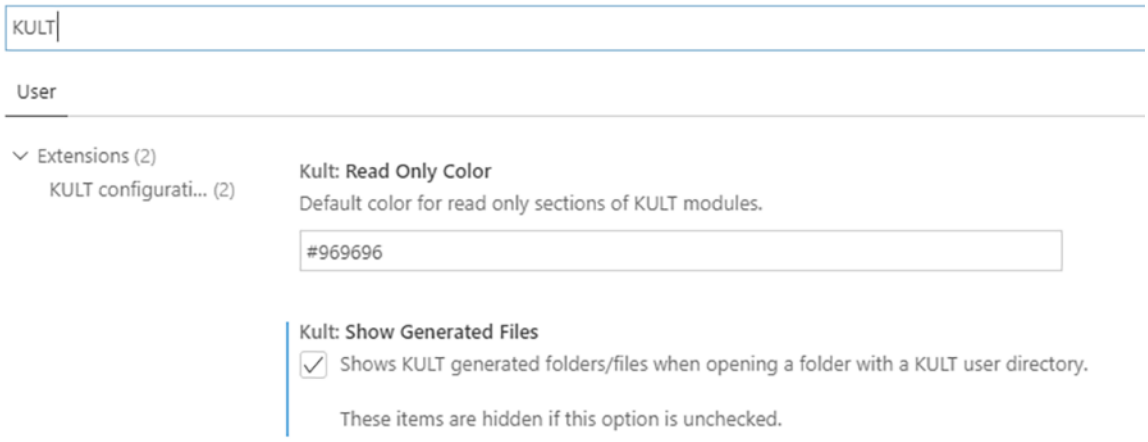
Figure 486: Command palette



Settings in Visual Studio Code

Other settings can be accessed by going to **File > Preferences > Settings**. Color scheme and general Visual Studio Code settings can be changed here. Some KULT extension features can also be modified, including color for description text and visibility of build generated files in the explorer. Search **KULT** in the settings to see these options.

Figure 487: Visual Studio Code extension settings



Working with user libraries in Visual Studio Code

This section covers basics on writing user libraries in Visual Studio Code.

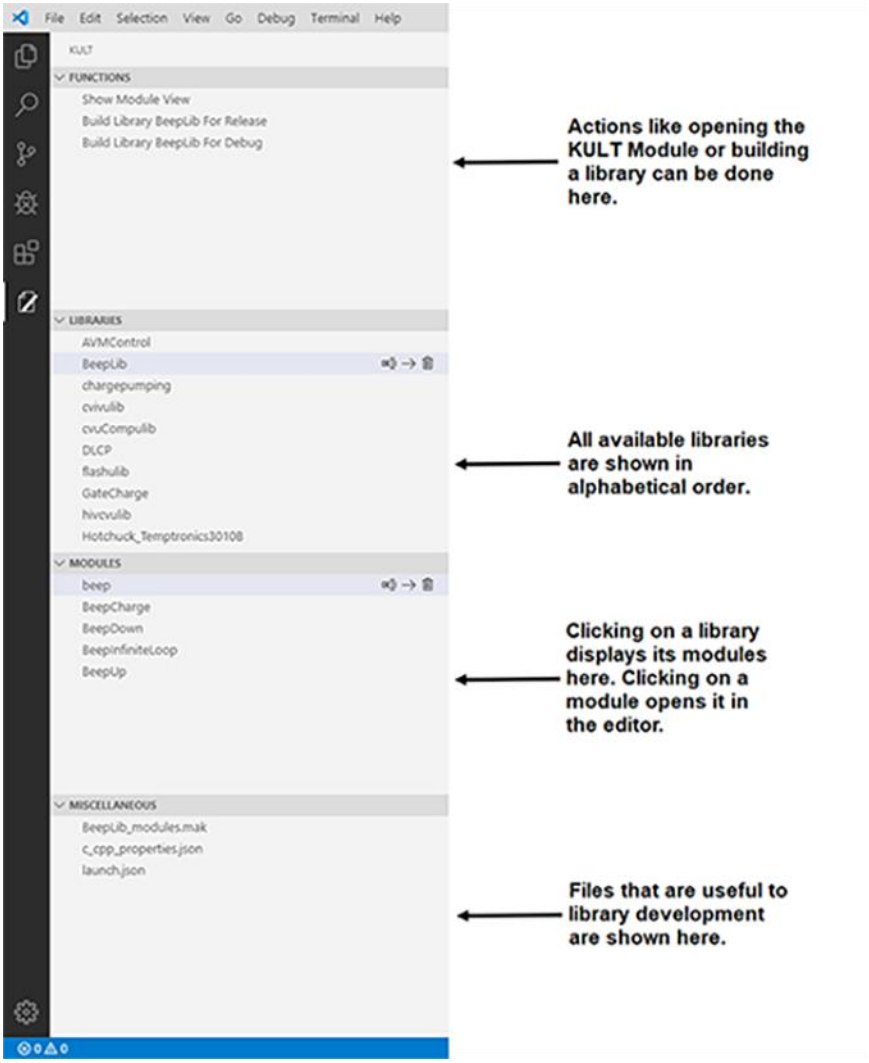
NOTE

A user library must be open in Visual Studio Code to view all KULT Extension features. See Opening the User Library Folder in Visual Studio Code (on page 9-11) for instructions on opening the user library folder.

The KULT side bar

The KULT Extension adds a side bar view to Visual Studio Code that enables simplified access to libraries and functions. Following is the side bar view.

Figure 488: Main GUI - KULT side bar



The KULT module

The KULT module displays the parameters and description of a module in the editor pane. Users can also set the return type for the module and make a library hidden in Clarius. The KULT module only shows information when a valid user module is open in the editor pane and active (cursor is in the module). This prevents unintentional editing of other modules. The KULT module will open on startup or by clicking the function **Show Module View** in the KULT side bar.

Modifying user libraries

Creating new libraries and modules

To create a new library:

1. Select the Add (+) icon on the library tab in the KULT side bar.

Figure 489: Create a new library

2. Name the library

To create a new module:

1. Select the desired library in the library tab of the KULT side bar.
2. Select the Add (+) icon on the module tab in the KULT side bar.

Figure 490: Create a new module

3. Name the new module.

NOTE

All modules must have unique names to avoid conflicts in library dependencies.

Deleting libraries and modules

To delete a library or module:

1. Hover over the name of the desired item in the KULT side bar and select the delete icon.

Figure 491: Delete a library or module

2. Select **Yes** on the prompt to permanently delete the library or module.

NOTE

Deleting a library will delete all of the modules in the library as well as all associated build files.

Copying a library or module

To copy a library or module:

1. Hover over the name of the desired item in the KULT side bar and select the copy icon.

Figure 492: Copy a library or module

2. The copied library must be built before it can be used in Clarius. See the section Building a library (on page 9-30).

Renaming a library or module

To rename a library or module:

1. Hover over the name of the desired item in the KULT side bar and select the rename icon.

Figure 493: Rename a library or module

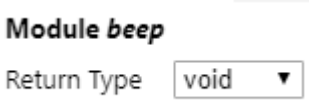
2. The renamed library must be built before it can used in Clarius. See the section Building a library (on page 9-30).

Setting the return type of module

To set the return type of a module:

1. Open the module in the editor by clicking the desired library and then the desired module in the KULT side bar.
2. Select the Return Type drop-down menu in the KULT module.

Figure 494: Select the type of module



3. Select **Apply** to add the changes to the read-only code.

Setting library visibility

To make a library hidden or visible in Clarius:

1. Open a module from the desired library in the editor by selecting the desired library and then a module in the KULT side bar.

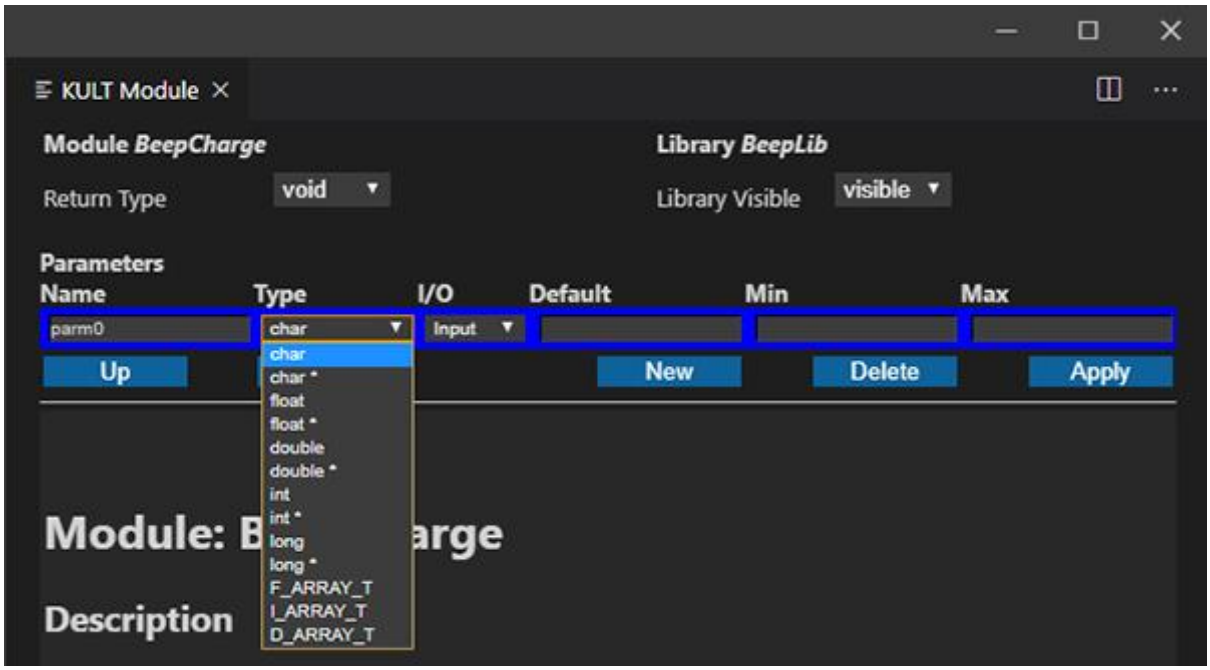
Figure 495: Select library visibility

2. Select the Library Visible drop-down menu to select visibility.
3. Select **Apply** to add the changes to the read-only code.

Editing module parameters

A module's parameters can be added, deleted, or modified in the KULT Module.

Figure 496: Edit parameters



To add a parameter:

1. Open the module in the editor by clicking the desired library and then the desired module in the KULT side bar.
2. Select **New** to add a new parameter to the top of the list.
3. Select an existing parameter to highlight it and then select **New** to add a parameter after the selected one.
4. Enter a name for the new parameter, as a type, input or output (must be a pointer type to be output), and maximum and minimum default values for the parameter.
5. Select **Apply** to add the changes to the code. Changes will appear in the gray read-only code at the top of the module.

To edit a module's parameters:

1. Open the module in the editor by selecting the desired library and then the desired module in the KULT side bar.
2. Select the desired parameter to highlight it.
3. Modify the parameter.
4. Select **Apply** to add the changes to the code. Changes are reflected in the gray read-only code at the top of the module.

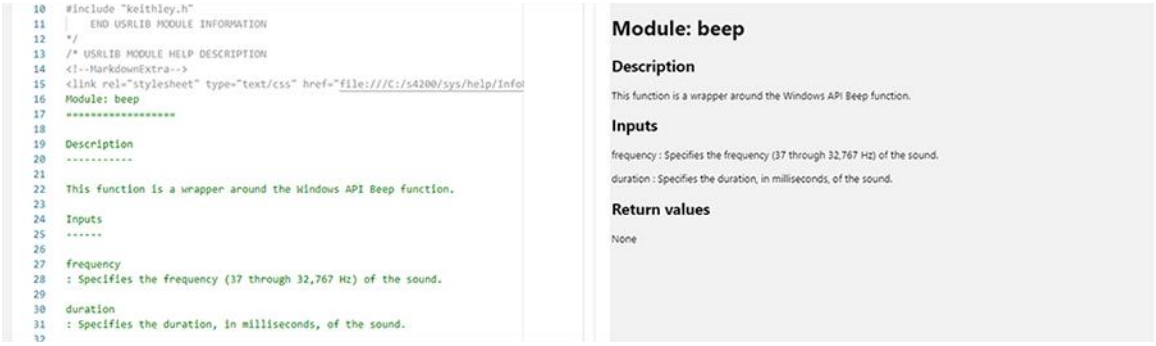
To delete a module's parameter:

1. Open the module in the editor by clicking the desired library and then the desired module in the KULT side bar.
2. Select the desired parameter to highlight it.
3. Select **Delete** to remove the highlighted parameter.
4. Select **Apply** to add the changes to the code. Changes will reflect in the gray read-only code at the top of the module.

Editing the module description

The module description appears in the help pane in Clarius when using the module. You can see a sample view of these descriptions in the KULT module in Visual Studio Code, beneath the parameters. This view is automatically updated when the description code is edited.

Figure 497: Module description



To edit the module description:

1. Open the module in the editor by clicking on the desired library and then the desired module in the KULT side bar.
2. Edit the description code below the read only gray code at the top of the module, inside the comments for **USRLIB MODULE HELP DESCRIPTION**. The code uses Markdown syntax. For more information see markdownguide.org/.

Including header files

Header files are included in the code before the main module function. Intellisense errors may appear in the Problems pane because paths to header files are not listed in the Intellisense configuration file, `c_cpp_properties.json`. These errors do not affect compilation and can be ignored if desired.

To add a header file to a module:

1. Open the module in the editor by clicking on the desired library and then the desired module in the KULT side bar.
2. Add the header file directly below the comment **USRLIB MODULE PARAMETER LIST** using the format **#include "headerName.h"**.

Figure 498: Add a header file to a module

```
37
38      END USRLIB MODULE HELP DESCRIPTION */
39  /* USRLIB MODULE PARAMETER LIST */
40  #include "keithley.h"
41
```

To remove Intellisense header file errors:

1. Create the `c_cpp_properties.json` file if it does not already exist in the `.vscode` folder. See Creating the Visual Studio Code configuration (on page 9-12) files for instructions.
2. If the file already exists, or creating the file did not remove the errors, open the file in the KULT side bar under Miscellaneous.
3. Add the desired header file in the `includePath` setting. File paths should be enclosed in quotes and separated by commas.

Figure 499: Remove Intellisense header file errors

```
"name": "KULT C Configuration",
"includePath": [
    "C:/S4200/sys/include",
    "c:/s4200/kiuser/usrlib",
    "C:/Program Files (x86)/MinGW/**"
],
```

NOTE

You may need to add header files located in other places on the system to the system environment variables. See the section Entering Library Dependencies and Environment Variables (on page 9-26) for more information.

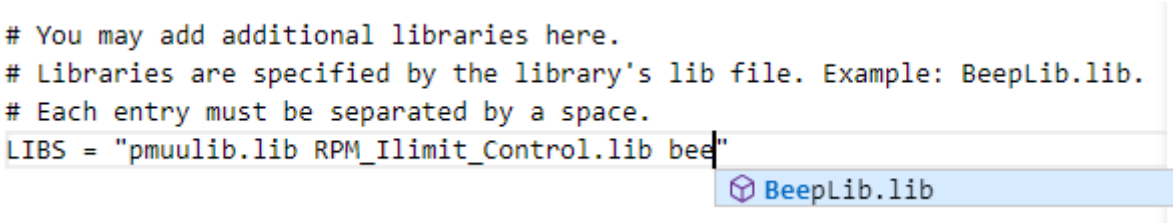
Entering library dependencies and environment variables

Library dependencies allow a user library to call other libraries. Library dependencies are directly edited in the library's `.mak` file. Library files that are not located in the workspace directory, such as third party libraries, can be added in the `.mak` file, as well. The system's LIB environment variable must be updated with the path to this library. Users can also update the INCLUDE path environment variable for header files located outside of the workspace directory.

To add a library dependency:

1. Select the library to edit in the KULT side bar by clicking on it.
2. Click the `.mak` file `libName_modules.mak` under the Miscellaneous tab of the KULT side bar.
3. Under the variable LIBS, type the name of the desired library. Name should be entered in the quotes and separated by spaces. Pressing **Ctrl+Space** opens a drop-down menu of all available libraries and typing will filter the results. Pressing **Enter** populates the result.

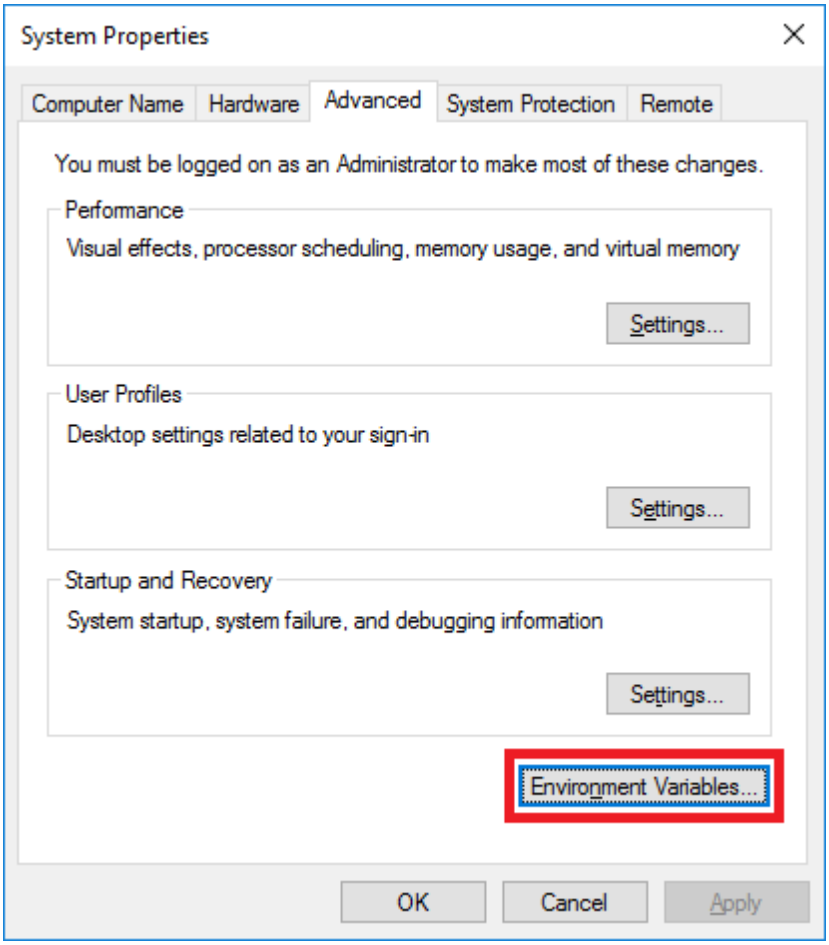
Figure 500: Add a library dependency



To update the system environment variables for external libraries and headers:

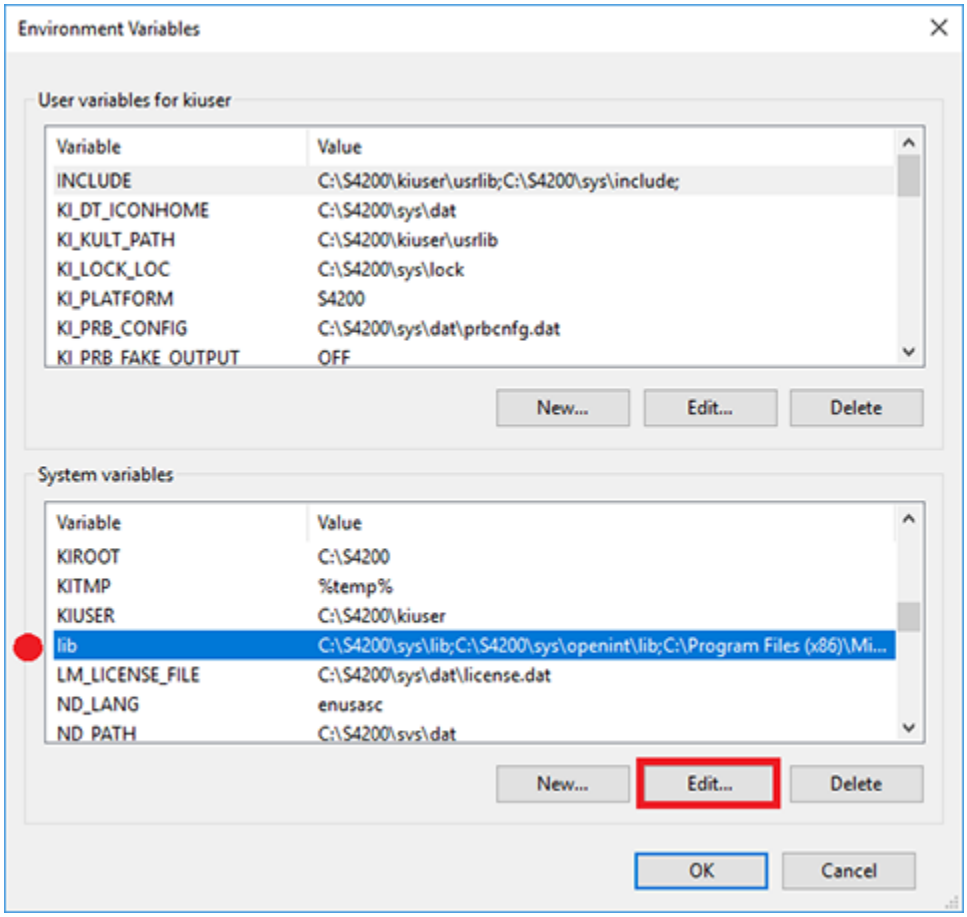
1. Search **Environment Variables** in the Windows search bar.
2. Open System Properties and selecct **Environment Variables.**

Figure 501: Set environmental variables



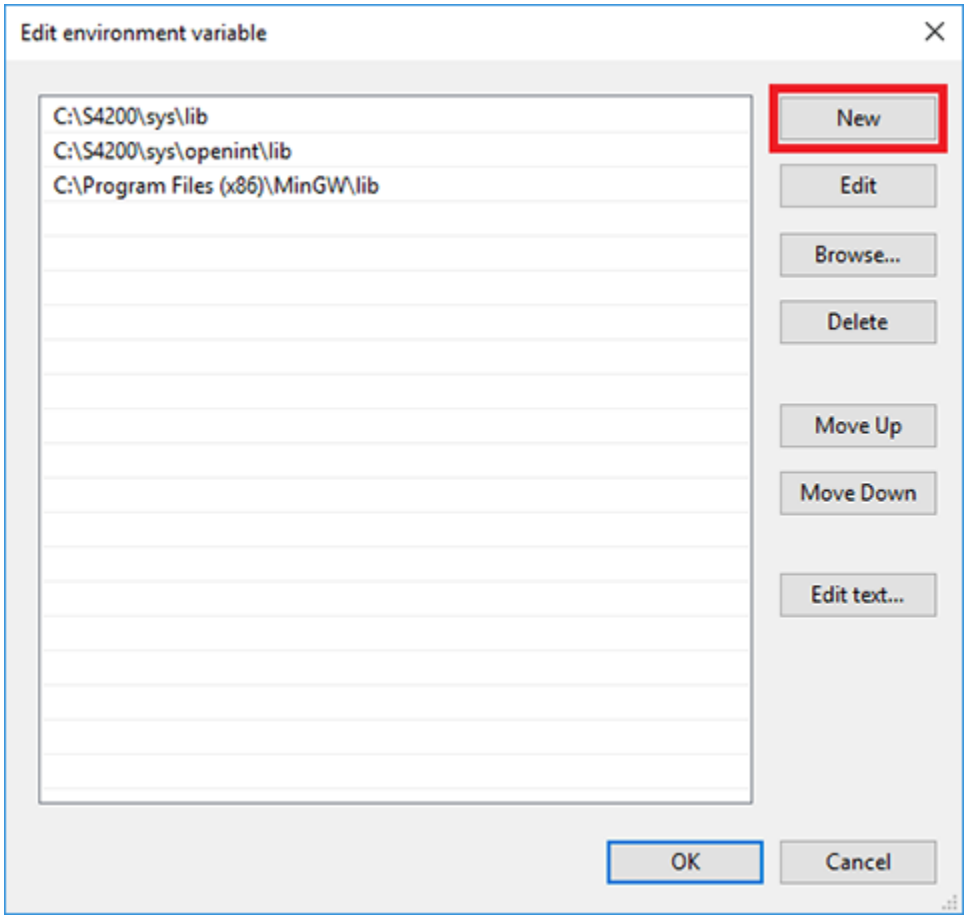
3. Scroll to the LIB variable under system variables, highlight it and select **Edit**.

Figure 502: Enter environmental variables



4. Select **New** and enter the path to the external library. You can repeate this process for any header files by selecting the **INCLUDES** variable and entering in the path to the header file.

Figure 503: Edit environment variable



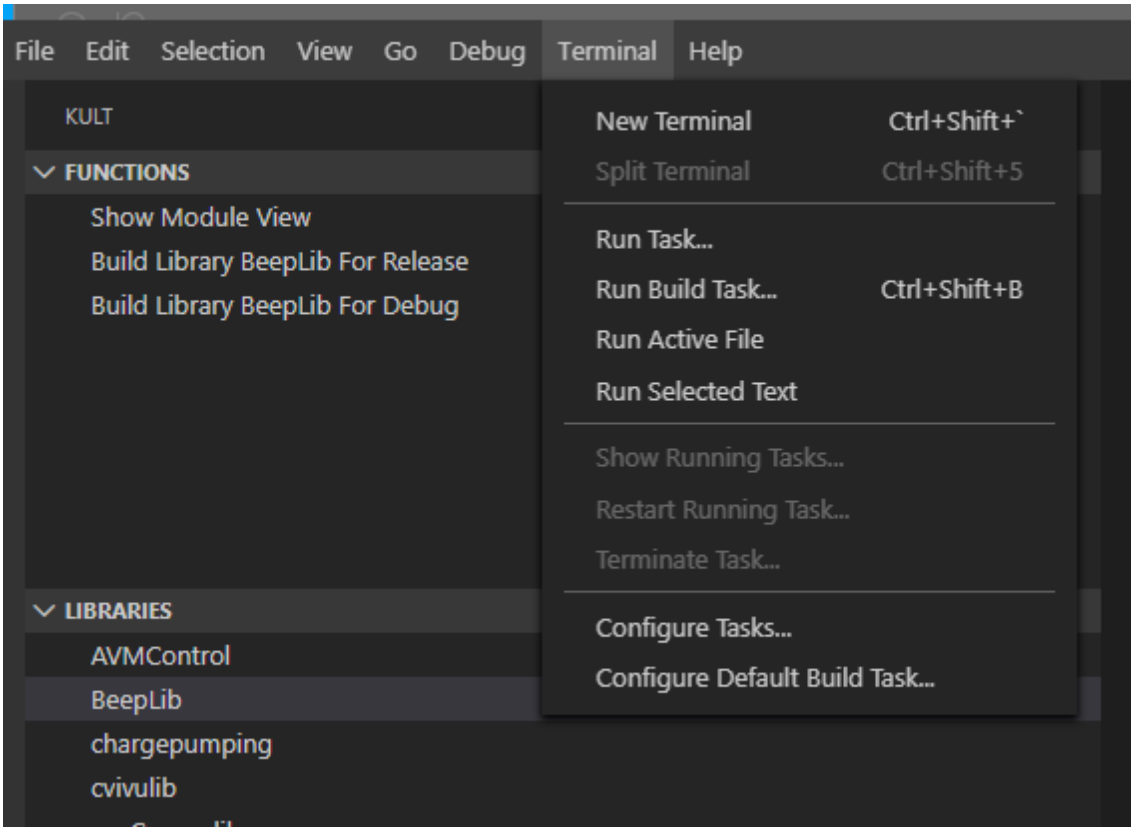
Building a library

There are two options to build a library; building for debug and building for release. Building a library for debug creates symbols that the debugger requires to watch variables. Building for release is faster since these symbols are not created and should be used if not using the debugger.

To build a library:

1. Select the desired library in the KULT side bar..
2. Select the run icon next to `Build Library LibName for Release` or `Build Library LibName for Debug` in the Functions tab of the KULT side bar.

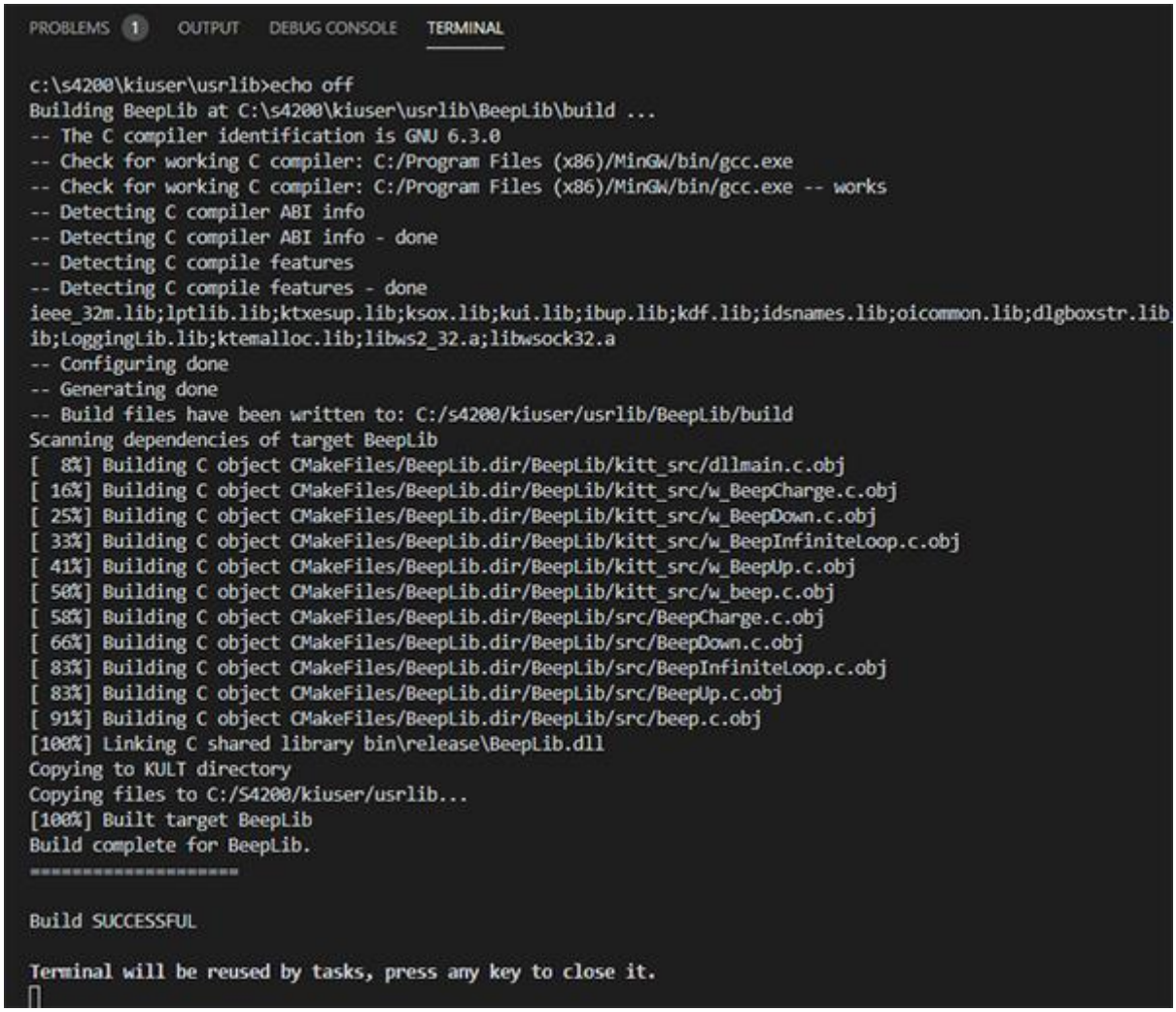
Figure 504: Build a library



a. You can also build a \Library from the Terminal menu: **Terminal -> Run Task or Terminal -> Run Build Task**.

3. Build output will appear in the Terminal Pane at the bottom of the screen. Any problems with the build will appear in the Problems pane.

Figure 505: Build output



Users can also build by going to **Terminal > Run Build Task** and selecting either **KULT: Build Library "LibName" For Release** or **KULT: Build Library "LibName" for Debug.** A library must be selected in the KULT side bar to view these build tasks.

Cleaning a library

Cleaning a library deletes all build generated files, leaving the source code. This is useful if a build file gets corrupted.

To clean a library:

1. Select the desired library in the KULT side bar.
2. Go to **Terminal > Run Task.**
3. Search KULT to filter out KULT tasks if there are other tasks.
4. Select **KULT: Clean Library "LibName**" to clean the library.
5. Output will show in the output tab.

Debugging libraries

Visual Studio Code has the capability to attach a debugger to an execution process to monitor code execution for debugging purposes. In the case of Keithley User Libraries, Visual Studio Code uses the GNU debugger (GDB) and attaches it to the UTMServer. Running the code as a UTM in Clarius allows the debugger to watch and control execution in the UTMServer.

To run and debug modules in Visual Studio Code, launch configuration (`launch.json`) must exist in the `.vscode` folder. For instructions to set these up, see the section Setting up Visual Studio Code for Library Development (on page 9-11).

Debug limitation notes:

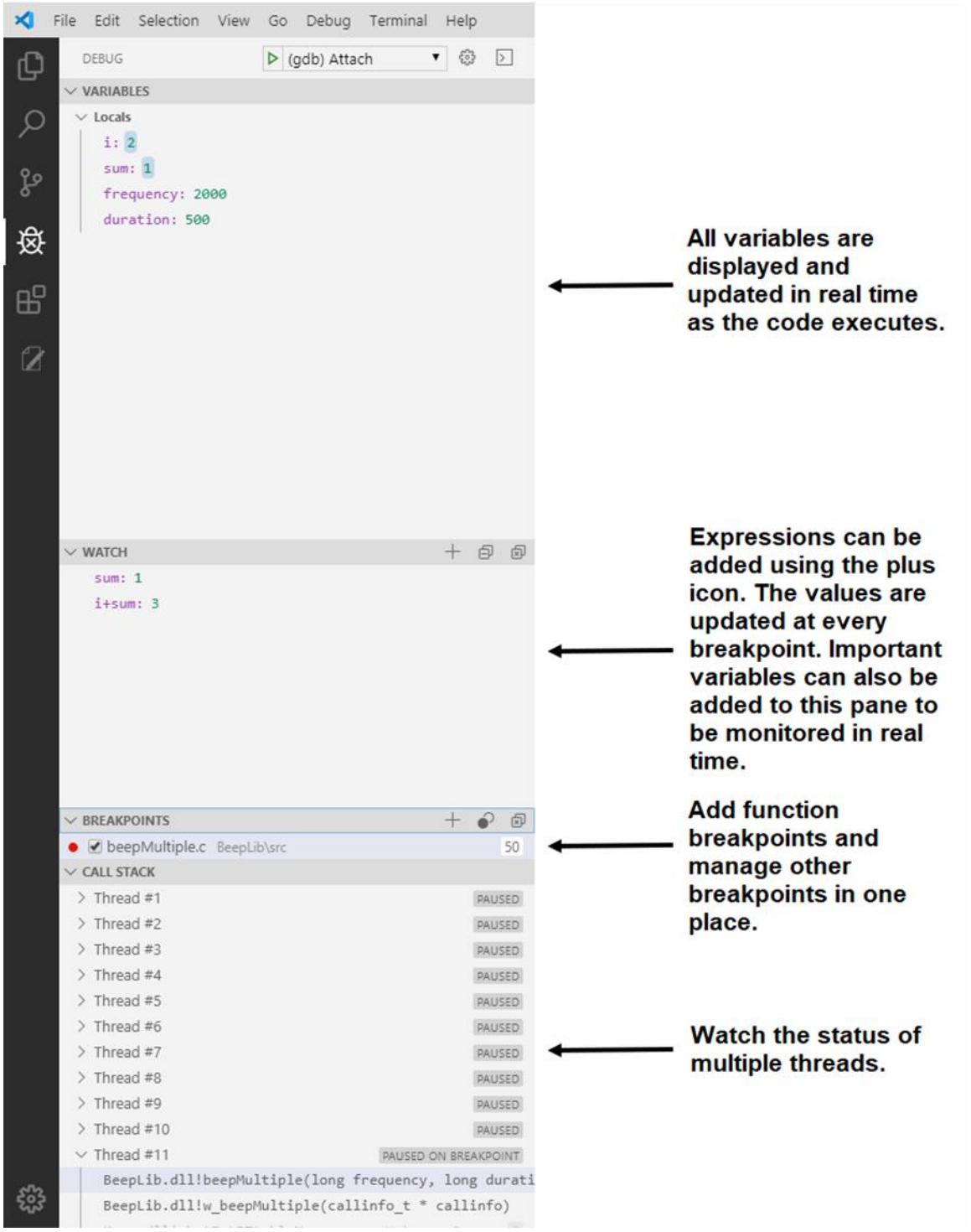
1. You may not be able to change breakpoints while attached. To change breakpoints, stop the debugger, detaching from `UTMServer.exe`. Update your breakpoints and reattach
2. While attached in debugging, do not press the stop button in Clarius or a Clarius process may hang. Should this happen, open windows task manager and kill the processes: `Clarius.exe`, `KiteServer.exe`, and `UTMServer.exe`.

Running the debugger

Running the debugger allows users to step through code, monitor variables, evaluate expressions and manipulate values. During debugging,both the debug side bar and debug toolbar will be visible.

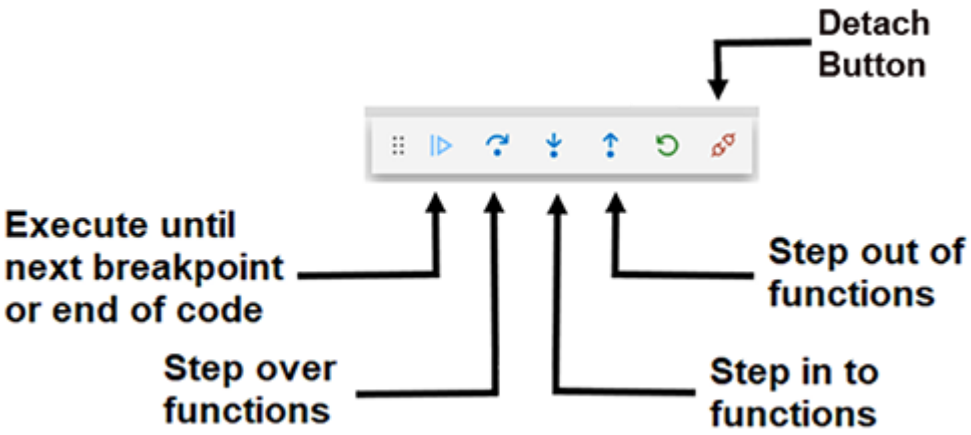
The side bar gives users access to variable values, expressions, breakpoints, and threads for multi-threaded debugging.

Figure 506: Debugging side bar



The debug toolbar gives users control over the code execution to step over or into lines of code.

Figure 507: Debug toolbar



Setup the debugger

To set up the debugger:

1. Open the module to be debugged by selecting the desired library and then the desired module in the KULT side bar.
2. Build the library for debug by clicking the run icon next to the function **Build Library LibName for Debug** in the KULT side bar.

NOTE

The library must be compiled for debug before using the debugger. Compiling for debug creates extra symbols that the debugger requires to use breakpoints and watch variables.

3. Open Clarius if not already running.
4. Configure a new test to run the desired module or open an existing test that uses the module.

NOTE

Libraries must be rebuilt after every change. Clarius can remain open but, the UTM must be reloaded for changes to take effect. Reload the UTM by clicking to a different test and navigating back, opening a different project and back or closing and reopening Clarius. Clarius must be fully closed to load a new library for the first time. You don't need to change to a different test if you are changing module content and not parameters of a module.

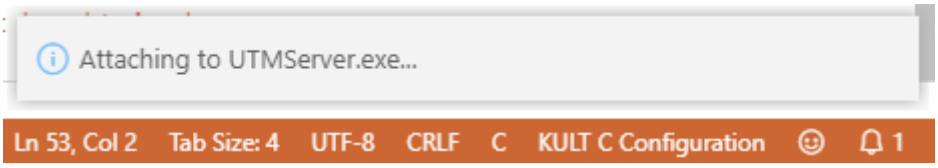
5. Set any breakpoints. See Setting breakpoints in modules for more information (on page 9-37).

NOTE

Due to the properties of the GNU Debugger, breakpoints will be unbound when the debugger starts and must be reapplied. However, you must reapply breakpoints before starting debugging for proper functionality.

6. In Visual Studio Code, press **F5** or go to **Debug > Start Debugging** to start the debugger.
7. Wait for the attach process to complete. This may take several seconds. The message, "Attaching to UTMServer.exe," in the lower right hand corner will disappear when the process is complete.

Figure 508: Attaching to UTMServer



Run code with the debugger

To run code with the debugger:

1. See the previous section to set up the debugger and the module in Clarius.
2. Run the UTM from Clarius.
3. When code execution is paused, debugging tools can be used or the code can either be stepped through line by line or run to the next breakpoint using the debug toolbar.

End a debugging session

To end a debugging session:

1. Open the terminal panel at the bottom of the screen.
2. Type **Ctrl+C** into the terminal. This stops a running GDB process.

CAUTION

Do not abort a UTM in Clarius when execution is paused in Visual Studio Code. This causes a conflict between the debugger and Clarius and will freeze Clarius. Type Ctrl+C in the terminal and select Enter **or allow the UTM to run without breaking, then abort in Clarius.**

Debugging tools

Several tools are available in Visual Studio Code to assist debugging.

Setting breakpoints in modules

Setting a breakpoint stops code execution and gives users control to step through code line by line. Breakpoints must be set on lines of code; they will not work on comments or blank lines. The GDB environment allows three different kinds of breakpoints:

- **Unconditional breakpoint:** Pauses execution on a specific line
- **Conditional breakpoint:** Pauses execution on a specific line if a given statement is true
- **Function breakpoint:** Pauses execution at the first line of a function

To set an unconditional breakpoint:

1. Set up the debugger as indicated in Running the Debugger. (on page 9-33)
2. Select the space to the left of the line number. A red dot indicates the placed breakpoint.

NOTE

If the red circle immediately turns gray, wait for the GDB attach process to finish before entering breakpoints to ensure binding. GDB does not allow breakpoints to bind during the attach process. Any grayed-out breakpoints must be removed and set again to be functional.

3. The breakpoint will also be logged in the debugging side bar under Breakpoints.

To set a conditional breakpoint:

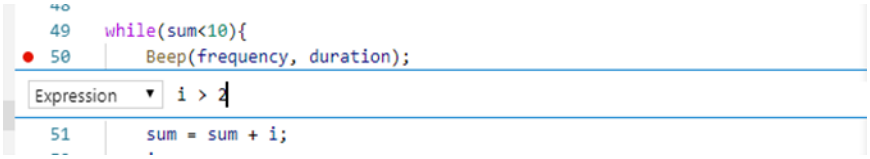
1. Set up the debugger as indicated in Running the Debugger (on page 9-33).

NOTE

GDB will only bind breakpoints when it is not running any attach process. Any breakpoints set before starting the attach process will be unbound during the attach process and must be reset. Setting up the debugger before setting breakpoints prevents this.

2. Select the space directly to the left of the line number. A red dot indicates the placed breakpoint.
3. Right click on the red dot and select **Edit Breakpoint.**
4. Type in an expression to be evaluated in the popup editor. This expression will be evaluated before the line is executed and will pause execution if true.

Figure 509: Edit breakpoint



5. The breakpoint will also be logged in the debugging side bar under Breakpoints.

To set a function breakpoint:

1. Set up the debugger as indicated in Running the Debugger. (on page 9-33)

NOTE

GDB will only bind breakpoints when it is not running any attach process. Any breakpoints set before starting the attach process will be unbound during the attach process and will need to be reset. Setting up the debugger before setting breakpoints prevents this.

2. In the debug side bar under the breakpoints section, click the plus icon to add a function breakpoint.
3. Type the name of the desired function. The breakpoint will be verified the first time the code is executed and will stop execution at the first line of the function.

Expression evaluation

Visual Studio Code allows expressions to be watched and evaluated while code is executed. Variables can also be modified, however, any modifications will remain for the rest of the execution.

To evaluate an expression once:

1. Set up the debugger as indicated in Running the Debugger (on page 9-33).
2. Set a breakpoint to pause the code.
3. Run the code in Clarius.
4. While the code is paused on a breakpoint, enter the expression to be evaluated into the Debug Console pane at the bottom.
5. The value will be outputted to the Debug Console.

To evaluate an expression at every breakpoint:

1. Set up the debugger as indicated in Running the Debugger. (on page 9-33)
2. Select the plus icon in the Watch pane of the Debug side bar to add a new expression.
3. Enter the desired expression.
4. Set any desired breakpoints.
5. Run the code. The expression will be evaluated every time the code is paused on a breakpoint. Additional expressions can be added at any time.

To edit a variable value:

1. Set up the debugger as indicated in Running the Debugger. (on page 9-33)
2. Set a breakpoint to pause the code.
3. Run the code in Clarius.
4. When the code is paused on a breakpoint, enter an expression to change the value of a variable (varName = newValue) in the Watch pane (click the plus (+) icon) or in the Debug Console. Edited values will be used for the remainder of the code execution.

Watching variables

All variables are visible in the Variables pane of the Debug side bar. Specific variables can be singled out by adding them to the Watch pane. Values are updated in real time.

To add a variable to the Watch pane:

1. Set up the debugger as indicated in Running the Debugger (on page 9-33).
2. Set a breakpoint to pause the code.
3. Run the code in Clarius to initialize the variables in the Variable pane.
4. When the code is paused on a breakpoint, right click the desired variable in the Variables pane and click **Add to Watch**.
5. The variable will now appear in the Watch pane and will be updated in real time.

Tutorials

Creating a new user library and user module

The KULT Extension is a tool for Visual Studio Code that helps you develop user libraries. Each user library is comprised of one or more user modules. Each user module is created using the C programming language.

This section contains a tutorial that shows you how to create a new user library and a new user module. A hands-on example is provided that illustrates how to create a user library that contains a user module that activates the internal beeper of the 4200A-SCS. You will then build and run the module in Clarius. This tutorial also explores some of the features of VS Code to assist with writing code.

Starting Visual Studio Code

To start Visual Studio Code:

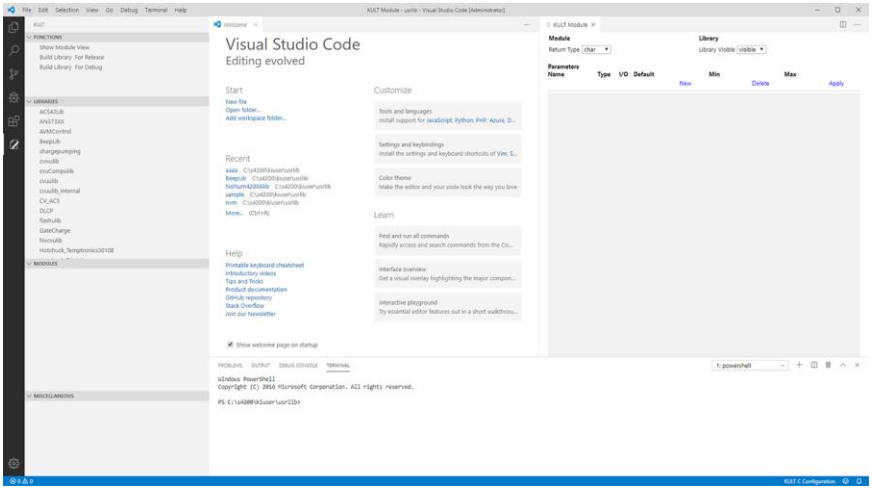
1. Double click the desktop icon or search **Visual Studio Code** in the Windows Start Menu.

NOTE

See the section on installation for step by step instructions to install and set up Visual Studio Code for user library development.

2. Open the KULT side bar by clicking the KULT Icon.

Figure 510: Opening KULT sidebar in Visual Studio Code



Creating a new user library

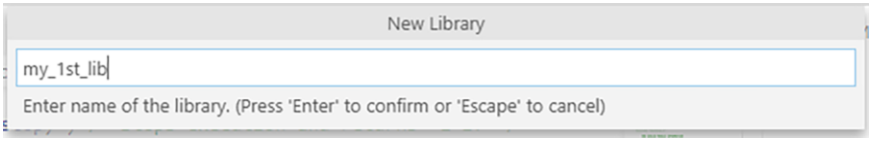
To create a new user library:

1. Select the plus icon on the library pane of the KULT side bar.

Figure 511: Add a new user library

2. Enter a name for the new library in the dialog box that appears. For this tutorial, enter `my_1st_lib` as the new user library name.

Figure 512: New library name



3. Select **Enter** to enter the new name. The library now appears in the list of libraries. It will automatically create the necessary build files.

Creating a new user module

NOTE

When naming a user module, conform to case-sensitive C programming language naming conventions. Do not duplicate names of existing user modules or user libraries.

To create a new user module:

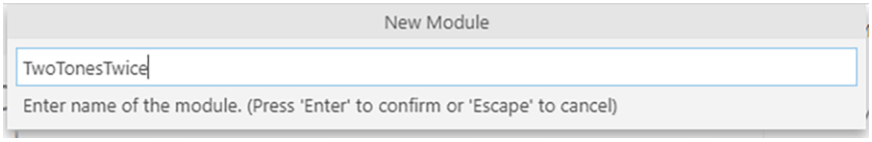
To create a new user module:

1. Select the library `my_1st_lib` in the library pane of the KULT side bar. This displays the modules in the library in the module pane.
2. Select the plus icon at the top of the module pane in the KULT side bar to add a module to the selected library.

Figure 513: Create a new user module

3. Enter a name in the dialog box at the top. For this tutorial, enter `TwoTonesTwice` as the new user module.

Figure 514: Naming a new module



4. Press Enter to apply the name. The module now appears in the list of modules for the library.
5. Open the module by selecting it in the KULT side bar. It will appear in the editor window with the KULT module.

Entering a return type

If your user module generates a return value, select the data type for the return value in the **Return Type** drop down menu on the KULT module then select **Apply**. The `TwoTonesTwice` module will not produce a return value. Therefore, keep the `void` default entry.

Entering user-module code

NOTE

Refer to the section LPT library function reference for a complete list of supported commands.

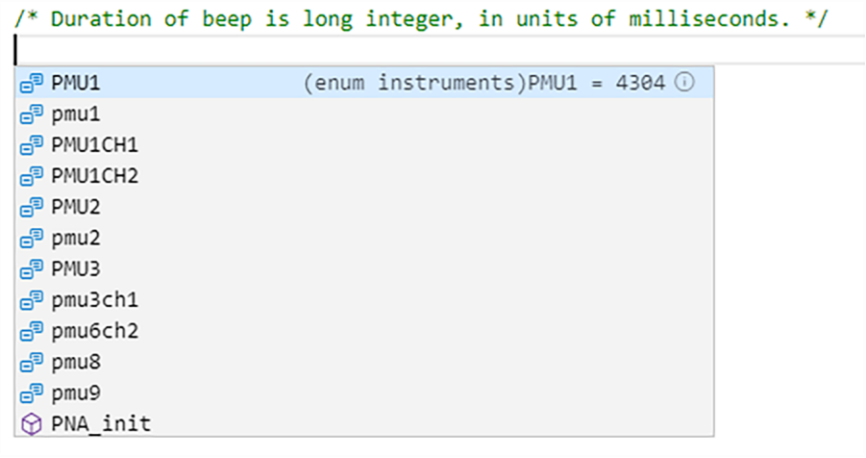
To add code to the module:

1. Enter the following four comments describing the purpose of the user module in the area for user module code indicated by the gray comment lines `USRLIB MODULE CODE` and `USRLIB MODULE END`.

```
/* 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. */
```

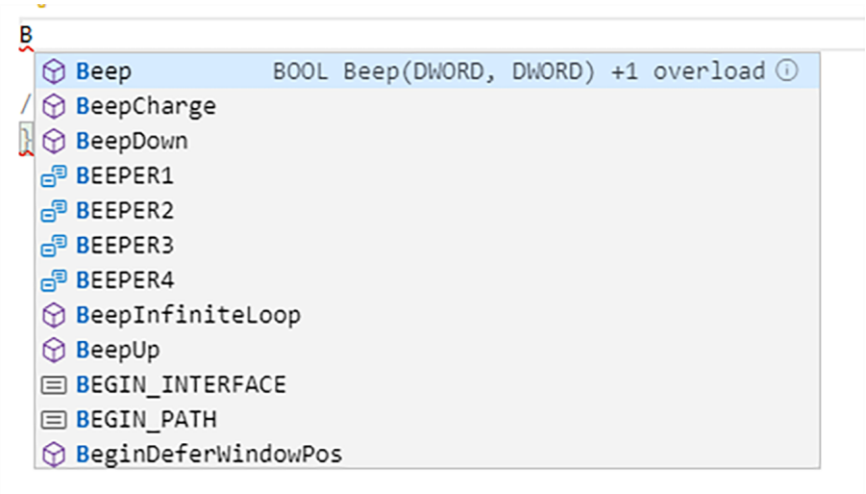
2. Go to the next line and press **Ctrl+Space** to open a drop down menu of all code suggestions.

Figure 515: Add code to the module



3. Filter these suggestions by typing `B`. The list will keep filtering as you type.

Figure 516: Add code to the module - filtered list



4. Select the code suggestion for the function `Beep` or select **Enter**. It will auto fill in the function name.
5. Continue the line by typing and a function prototype model appears. The bold underlined parameter is the next parameter that needs to be entered.

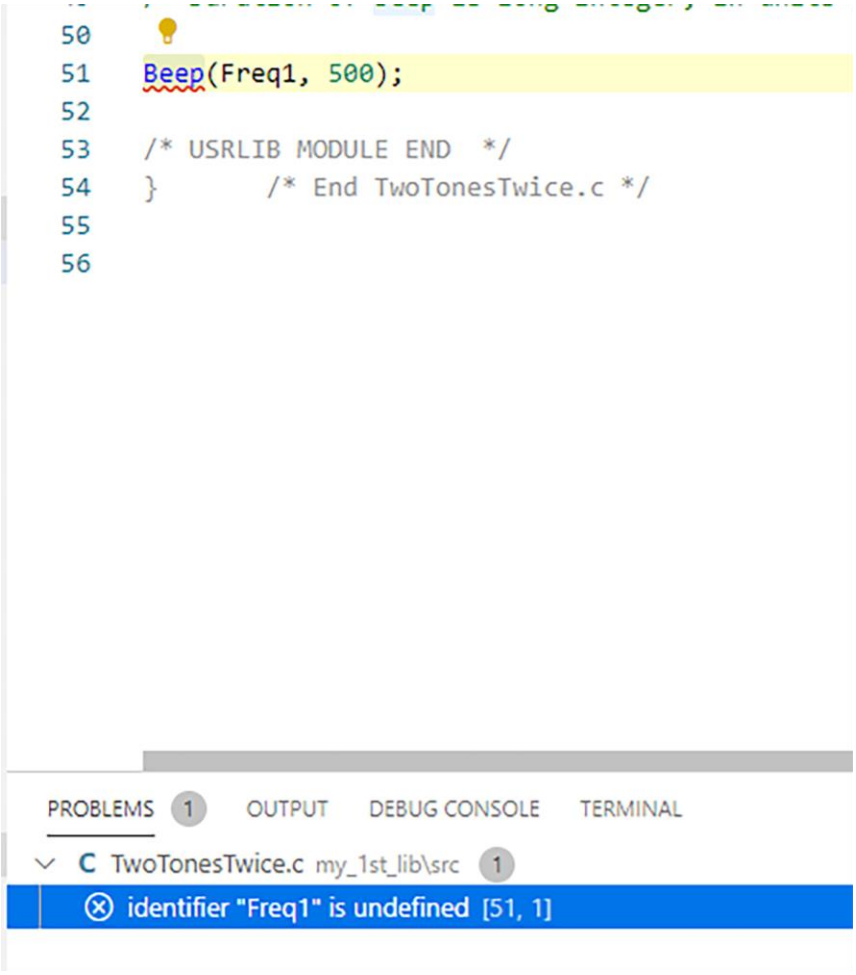
Figure 517: Entering line of code

6. Type the parameters `Freq1` and `500` for a and b. End the function with a closing parenthesis and a semicolon. Add the comment as shown below:

```
Beep(Freq1, 500); /* Beep at first frequency for 500 ms */
```

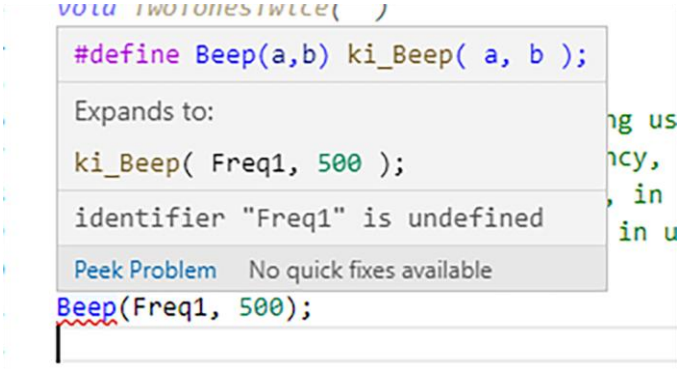
7. Note that there is now a problem in the Problems tab at the bottom. Open the tab and select the problem.
8. The line of code we just wrote is now highlighted. According to the problem, the identifier “Freq1” is undefined. This is because we have not added Freq1 as a parameter yet. This will be defined later in the tutorial.

Figure 518: Identifier



9. Hover the cursor over the Beep function you just wrote. Note that it expands to show details about the function.

Figure 519: Function description details



10. Continue entering in the rest of the C code below. Note that the code deliberately contains a missing ; error to demonstrate a build error.

```
Beep(Freq2, 500); /* Beep at second frequency */
Beep(Freq1, 500);
Beep(Freq2, 500);
Sleep(500) /* NOTE deliberately forget semicolon initially */
```

Entering parameters

To enter the required parameters for the code:

1. In the KULT module, select **New**.
2. In the parameter name, enter `Freq1`.
3. Select **long** in the drop down menu for Data Type as shown here. This is a C data type.

NOTE

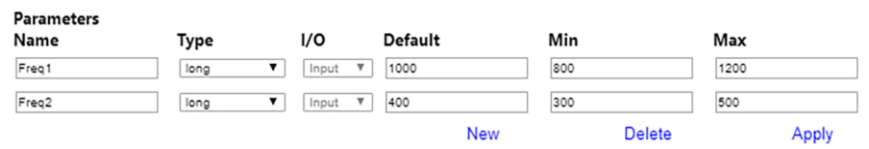
For an output parameter, only the following data types are acceptable: Pointers (char*, float*, double* and so on) and arrays `(I_ARRAY_T, F_ARRAY_T, or D_ARRAY_T)`.

4. For this user module, **Input** is the correct I/O state.
5. Under **Default, Min,** and **Max**, enter default, minimum and maximum values. These values limit the choices the user sees. For the `TwoTonesTwice` user module, enter 1000, 800 and 1200 respectively.

Figure 520: Entering parameters

6. For the `TwoTonesTwice` module, add one more parameter with the values:
   - **Parameter name:** `Freq2`
   - **Data type:** `long`
   - **I/O:** `Input`
   - **Default:** `400`
   - **Min:** `300`
   - **Max:** `500`

Figure 521: Reviewing both parameters



7. Select **Apply** in the KULT Module. This will add the changes to the read-only code at the top of the module. Note that this removes many errors from the problems tab at the bottom.

Entering header files

Any header files that are required are entered below the gray comment line `USRLIB MODULE PARAMETER LIST`. The header file `keithley.h` is added automatically when the module is created, since it is most commonly used. No additional header files are needed for this tutorial.

Documenting the user module

Module descriptions are entered in between the comment lines `USRLIB MODULE HELP DESCRIPTION` and `END USRLIB MODULE HELP DESCRIPTION`. Code entered here in markdown format will appear in the Clarius help pane.

CAUTION

Do not use C-code comment designators (/*, */ or //) in the Description area. When the user module code is built, KULT also evaluates the text in this area. C-code comment designators in the Description area can be misinterpreted, causing errors.

For the `TwoTonesTwice` user module, enter the following information in the Description area:

```
<link rel="stylesheet" type="text/css"
href="http://clariusweb/HelpPane/stylesheet.css">

MODULE
======
TwoTonesTwice

DESCRIPTION
-----------
Execution results in sounding of four beeps at two alternating user-settable
frequencies. Each beeps sounds for 500 ms.

INPUTS
------
Freq1 (long) is the frequency, in Hz, of the first and third beep.
Freq2 (long) is the frequency, in Hz, of the second and fourth beep.

OUTPUTS
-------
None

RETURN VALUES
-------------
None
```

Saving the user module

Go to the **File** menu then select **Save** or press **Ctrl+S.**

Building the library

To build the library:

1. In the functions pane of the KULT side bar, click the run icon next to the function **Build Library my_1st_lib for Release**.

Figure 522: Building a library

2. Observe the build output in the terminal tab at the bottom. Note that it was unsuccessful.

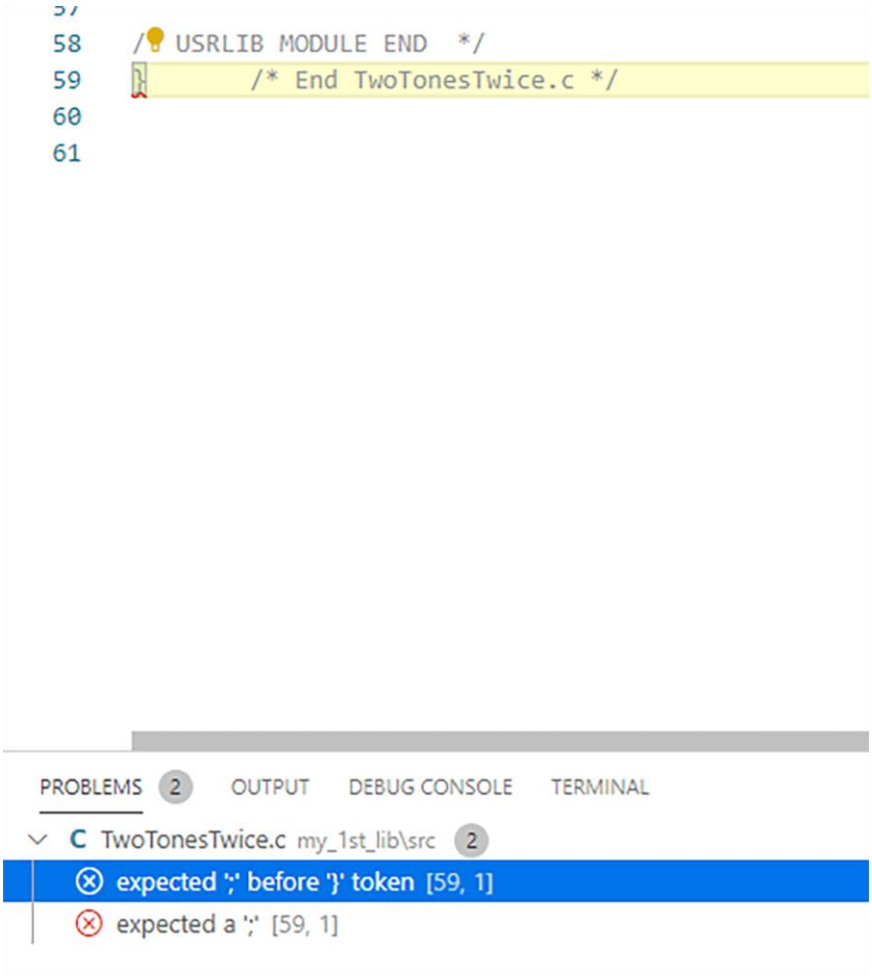
Finding code errors

Code errors appear in the **Problems** tab at the bottom of the window and indicate the line with the error along with a description.

To find code errors in the TwoTonesTwice user module:

1. Select the **Problems** tab at the bottom of the screen. There will be two errors, one generated by the Intellisense feature and one generated by the build.
2. Select either of the problems. The line of code will be highlighted in the code editor.

Figure 523: Find code errors



3. The error description indicates the problem. In this case, there is a missing semicolon before the closing brace. Correct the error by adding the missing semicolon (;) and delete error message. This will remove the Intellisense error. The build error will be removed at the next successful build.
4. Build the user library again.

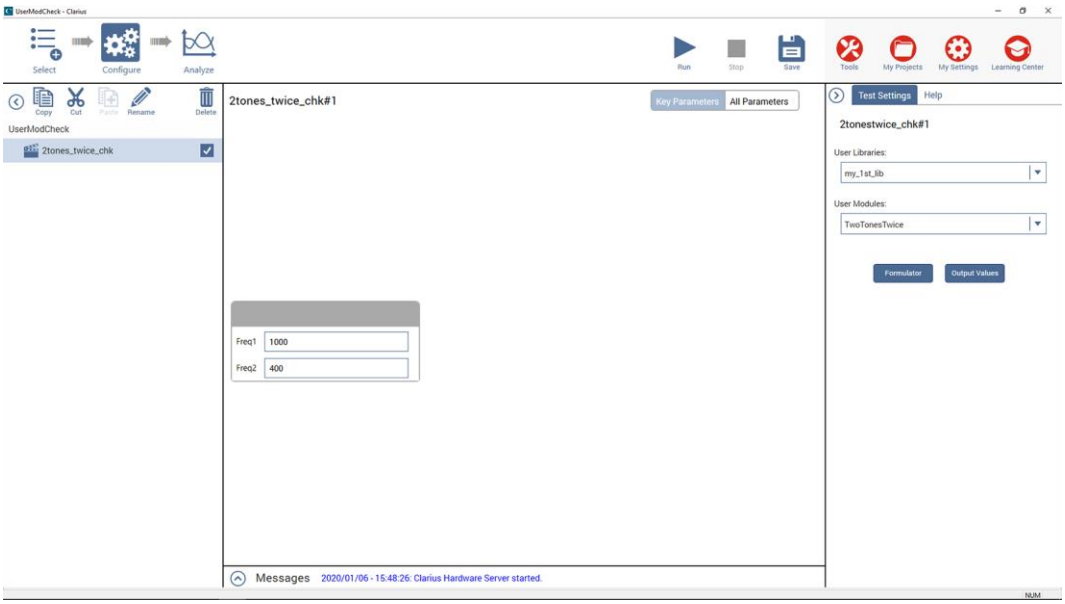
Checking the user module in in Clarius

Check the user module in Clarius by setting up a UTM.

To check the module in Clarius:

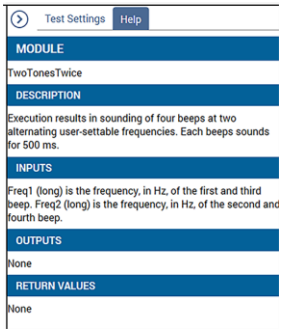
1. Start Clarius.
2. Choose the **Select** pane.
3. Select **Projects**.
4. Select **New Project**.
5. Select **Create**. You are prompted to replace the existing project.
6. Select **Yes**.
7. Select **Rename**.
8. Enter `UserModCheck` and press **Enter**.
9. Select the **Actions** tab.
10. Drag **Custom Action** to the project tree. The action has a red triangle next to it to indicate that it is not configured.
11. Select **Rename**.
12. Enter `2tones_twice_chk` and press **Enter**.
13. Select **Configure**.
14. In the Test Settings pane, select the `my_1st_lib` user library.
15. From the User Modules list, select the `TwoTonesTwice` user module. A group of parameters are displayed for the UTM as shown in the following figure. Accept the default parameters for now. You can experiment later after you establish that the user module executes correctly.

Figure 524: User modules list



16. Select Help to verify that the HTML in the Description tab is correctly formatted. An example is shown in the following figure.

Figure 525: Verify that the HTML in the description tab is correctly formatted



17. Select **Save**.
18. Execute the UTM by selecting **Run**. You should hear a sequence of four tones, sounded at alternating frequencies.

This tutorial generates no data. For an example of numerical data, see Tutorial 2: Creating a user module that returns data arrays.

Creating a user module at returns data arrays

This section provides a tutorial that the use of array variables in the KULT Extension. It also illustrates the use of return types (or codes), and the use of two functions from the Keithley Linear Parametric Test Library (LPTlib).

NOTE

Most of the basic steps that were detailed in Tutorial 1 are abbreviated in this tutorial. Before doing the following tutorial, complete Tutorial 1: Creating a new user library and user module).

Creating a new user library and a new VSweep user module

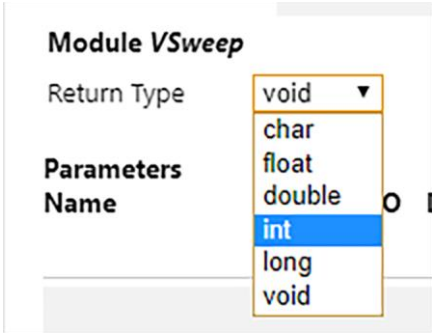
To name a new user library and new VSweep user module:

1. Open Visual Studio Code and go to the KULT side bar.
2. Click the plus icon in the library pane to create a new user library.
3. Name the library `my_2nd_lib` and press **Enter**.
4. Click the library name in the library pane to select it.
5. Click the plus icon in the module pane to create a new user module in the library.
6. Name the module VSweep and hit **Enter**.
7. Click the module in the module pane to open it in the editor.

Entering the return type for VSweep

The VSweep user module generates an integer return value. Therefore, select **int** in the Return Type drop down menu. Click **Apply** to apply the changes.

Figure 526: VSweep return user module



Entering the VSweep user module code

In the editor, inside the gray comments for `USRLIB MODULE CODE`, enter in the following C code for the VSweep user module.

NOTE

When returning data using arrays, it is good practice to add a check to make sure that the number of points returned from a sweep is less than the size of the array to avoid memory errors. This is not necessary here, since the array size is used as the number of points to calculate the step size. For modules that specify step size, the number of measurement points is always one greater than the number of steps.

```
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 number 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.*/
```

Entering the VSweep user module parameters

NOTE

This example uses the double-precision `D_ARRAY_T` array type. The `D_ARRAY_T`, `I_ARRAY_T`, and `F_ARRAY_T` are special array types that are unique to Keithley User Libraries. For each of these array types, you cannot enter values in the Default, Min, and Max fields. An extra parameter will be created to indicate the array size.

NOTE

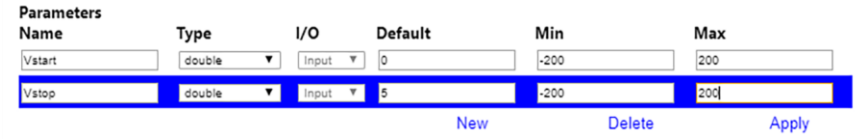
When executing the Vsweep user module in a UTM, the start and stop voltages (Vstart and Vstop) must differ. Otherwise, the first return statement in the code halts execution and returns an error number (-1). When a user module is executed using a Clarius UTM, this return code is stored in the UTM Data worksheet. The return code is stored in a column that is labeled with the user-module name.

To enter the required parameters for the code:

1. Select **New** on the KULT module to create a new parameter for Vstart. Enter the following information and repeat for Vstop.

Parameter Name	Data Type	I/O	Default	Min	Max
Vstart	Double	Input	0	-200	200
Vstop	Double	Input	5	-200	200

Figure 527: Enter required code parameter



2. Select **New** to add a new parameter.
3. Enter the following measured-current parameter information:
   - Parameter Name: `Imeas`
   - Data type: `D_ARRAY_T`
   - I/O: Output

Figure 528: Code parameters

4. In the array size variable that was automatically added, change the name to `NumIPoints`. Note that the default name entry is only a description of the required array size parameter. It must be renamed with the name used in the user module code.
5. Under default, enter 11 for the default current array size. You can also add `min` and `max` sizes if needed.

Figure 529: Specifying Min and Max parameters

6. Select **New**.
7. Enter the following forced-voltage parameter information:
   - Parameter Name: `Vforce`
   - Data type: `D_ARRAY_T`
   - I/O: Output
8. For the size parameter, change the name to `NumVPoints`.
9. Under default for `NumVPoints`, enter 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`. If the sizes do not match, 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. Select **Apply** to save the changes. The function prototype will now contain the declared parameters.

Figure 530: All parameters applied.

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.

Documenting the VSweep user module

Enter a markdown documentation for the module in the userlib module help description comments. The description should be based on the comments provided in the code and other information about the module.

A sample description is shown below:

```
<link rel="stylesheet" type="text/css"
   href="http://clariusweb/HelpPane/stylesheet.css">

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).
```

Saving the VSweep user module

From the **File** menu, select **Save** or press **Ctrl+S.**

Building the VSweep user module

To build the user module:

1. Select the run icon next to **Build Library my_2nd_lib for Release.**

Figure 531: Build the user library

2. Check the build output in the terminal tab at the bottom for status.
3. If there are any errors, correct them build the user module again.

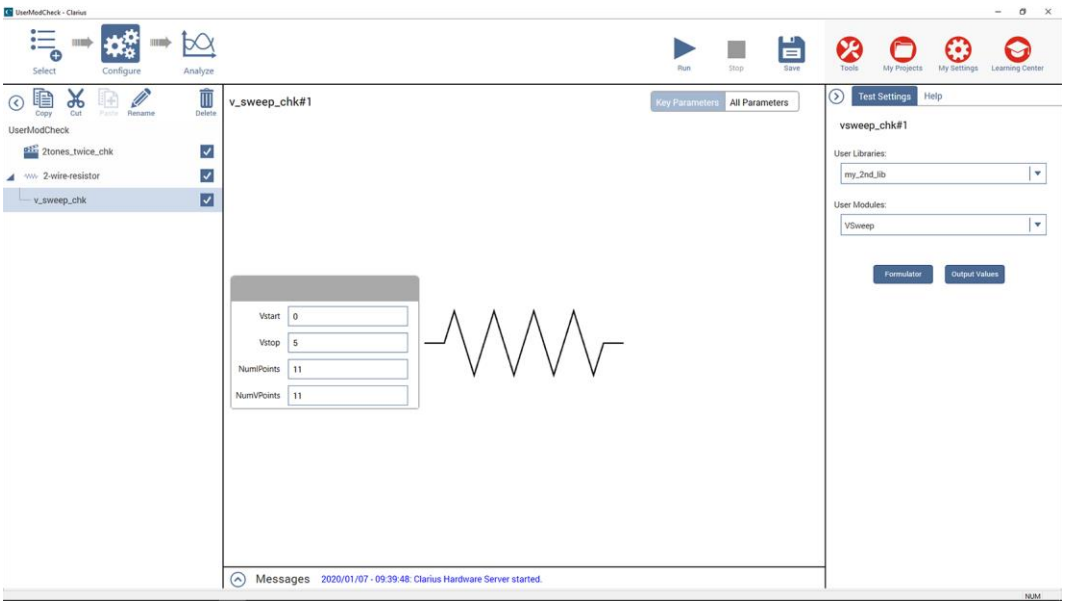
Checking the VSweep user module in Clarius

You can check the user module by creating and executing a UTM in Clarius.

To check the user module:

1. Connect a 1 kΩ 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.
3. Choose **Select**.
4. Select the **Devices** tab.
5. Select the 2-wire resistor.
6. Choose **Select**.
7. Select the **Tests** tab.
8. For the Custom Test, select **Choose a test from the pre-programmed library (UTM)**.
9. Drag **Custom Test** to the project tree. The test has a red triangle next to it to indicate that it is not configured.
10. Select **Rename**.
11. Enter the name `v_sweep_chk` for the UTM. You will use the `v_sweep_chk` **UTM** to execute the VSweep user module.
12. Select **Configure**.
13. In the right pane, from the User Libraries list, select the `my_2nd_lib` library.
14. From the User Modules list, select the `VSweep` user module. A default schematic and group of parameters are displayed for the UTM.

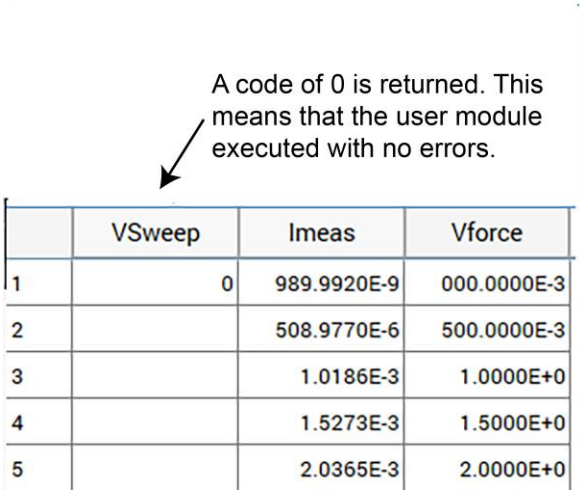
Figure 532: Schematic and parameters for the UTM



15. Select Run.
16. Select Analyze.

After execution, review the results in the Analyze sheet. If you connected a 1 kΩ resistor between SMU1 and GNDU, used the default UTM parameter values, and executed the UTM successfully, the results should be similar to the results in the following figure. The current-to-voltage ratio for each row of results should be approximately 1 mA / V.

Figure 533: Analyze sheet



	VSweep	Imeas	Vforce
1	0	989.9920E-9	000.0000E-3
2		508.9770E-6	500.0000E-3
3		1.0186E-3	1.0000E+0
4		1.5273E-3	1.5000E+0
5		2.0365E-3	2.0000E+0

Calling one user module from within another

User modules can have the capability to call other user modules. A called user module can be in any user library. This section provides a brief tutorial that illustrates application of such dependencies. It also describes how to copy a module.

In this tutorial, a new user module is created using the two user modules created in the previous tutorials:

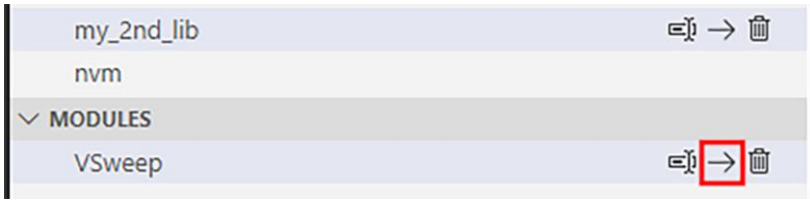
- **Tutorial 1**: Creating a new user library and user module.
- **Tutorial 2**: Creating a user module that returns data arrays:
    a. The `VSweep` user module in the `my_2nd_lib` user library, a copy of which is used as the dependent user library.
    b. The `TwoTonesTwice` user module, in the `my_1st_lib` user library, which is the independent user library that will be called by the `VSweep` user module.

A copy of the VSweep user module, called `VSweepBeep`, calls the `TwoTonesTwice` user module to signal the end of execution.

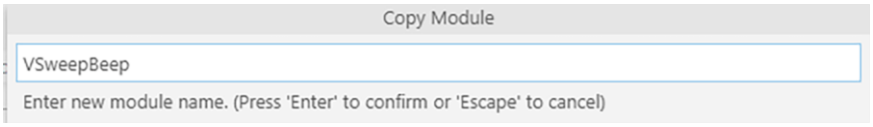
Creating the VSweepBeep user module by copying an existing user module

To copy the VSweep user module:

1. Start Visual Studio Code and open the KULT side bar.
2. Select `my_2nd_lib` from the library pane on the side bar.
3. Select the **VSweep module** from the module pane on the side bar.
4. Select the copy icon next to the module to make a copy.



5. Name the copied module **VSweepBeep**.



6. Select **Enter**.
7. Select VSweepBeep in the side bar to open it in the editor.

Calling an independent user module from the VSweepBeep user module

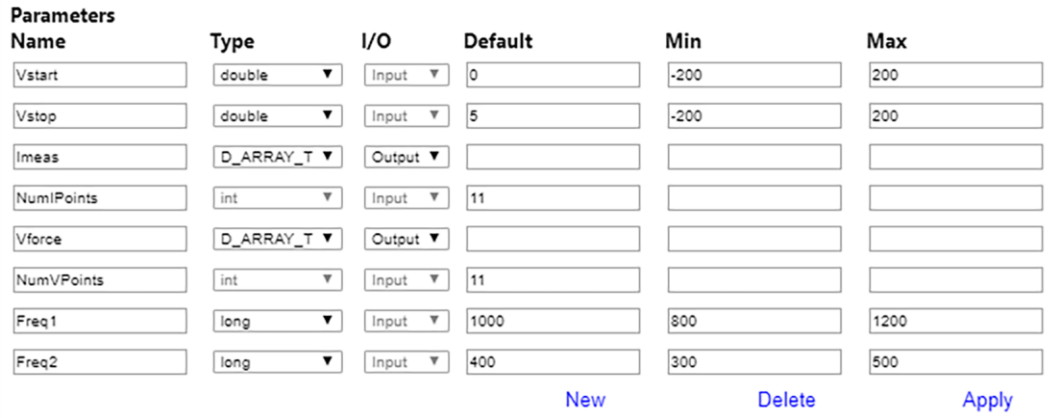
To call the TwoTonesTwice user module at the end of the VSweepBeep user module:

1. At the end of VSweepBeep, before the return(0)statement, add the following statement:

```
TwoTonesTwice(Freq1, Freq2); /* Beeps 4X at end of sweep. */
```

2. On the KULT module, add the `Freq1` and `Freq2` parameters in the following table, just as you did during the first Tutorial, changing the default, min and max values as needed.

Parameter Name	Data Type	I/O	Default	Min	Max
Freq1	Long	Input	1000	800	1200
Freq2	Long	Input	400	300	500



3. Select **Apply** to add the new parameters to the function prototype.

Specifying user library dependencies

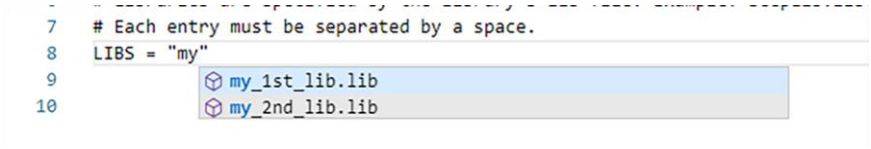
Before building the presently open user module, you must specify all of the user libraries on which the user module depends.

The `VSweepBeep` user module depends on the `my_1st_lib` user library.

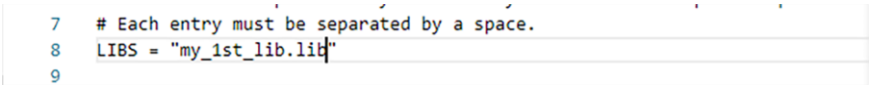
To specify the library dependency:

1. Select the `my_2nd_lib_modules.mak` file in the Miscellaneous pane at the bottom of the KULT side bar to open it in the editor.

2. Select inside the quote next to the `LIBS=` variable to put the cursor there.
3. Press **Ctrl+Space** to display all libraries or type **my** to automatically filter.



4. Select `my_1st_lib`.
5. Save the file by going to **File**, then **Save** or by pressing Ctrl+S. You can close the file from the editor if desired.



Building the user library

To build the user library:

1. Save the `VSweepBeep` module by going to **File**, then **Save** or by pressing **Ctrl+S**.
2. Select the run icon next to the function **Build library my_2nd_lib** for release on the KULT side bar.

Figure 534: Build the user library

3. Check the build output for any errors.

Checking the VSweepBeep user

Check the user module as you did in Tutorials 1 and 2, by creating and executing a user test module (UTM) in Clarius. Refer to Checking the user module. (on page 8-30)

The text of the tutorial-specific guidelines below is almost identical to the text of the Tutorial 2 guidelines. The data produced should be the same as the Tutorial 2 data. However, four beeps should sound at the end of execution.

Before proceeding:

1. Connect a 1 kΩ 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.
3. Add to this project a UTM called `v_sweep_bp_chk`.
4. Configure the `v_sweep_bp_chk` UTM to execute the `VSweepBeep` user module, which is found in the `my_2nd_lib` user library.
5. Run 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.
6. At the conclusion of execution, review the results in the Analyze sheet (or the Graph document, if configured). If you connected a 1 kΩ resistor between SMU1 and GNDU, used the default UTM parameter values, and executed the UTM successfully, your results should be similar to the results shown in Checking the VSweep user module. The current/voltage ratio for each row of results should be approximately 1 mA/V.

Customizing a user test module (UTM)

This tutorial demonstrates how to modify a user module using the KULT Extension. 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.

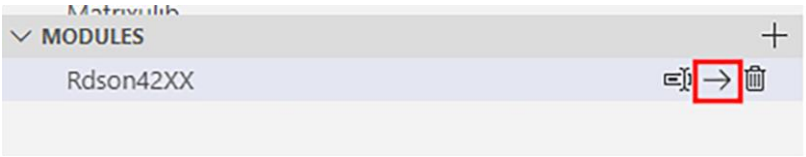
In this example, the user module is modified in VS Code so that ten current measurements are made at Vd1 and ten more at Vd2. The current readings at Vd1 are averaged to yield Id1, and the current readings at Vd2 are averaged to yield Id2. Using averaged current readings smooths out the noise. The modified test, `rdsonAvg`, measures the drain-to-source resistance of a saturated MOSFET. The MOSFET is tested as follows when `rdsonAvg` is executed:

1. Applies 2 V to the gate (Vg) to saturate the MOSFET.
2. Applies 3 V to the drain (Vd1) and makes ten current measurements.
3. Averages the 10 current readings to yield a single reading (Id1).
4. Applies 5 V to the drain (Vd2) and makes ten more current measurements.
5. Averages the ten current readings to yield a single reading (Id2).
6. Calculates the drain-to-source resistance (`rdsonAvg`) as follows:

```
rdsonAvg = (Vd2-Vd1) / (Id2-Id1)
```

Copy the Rdson42XX user module

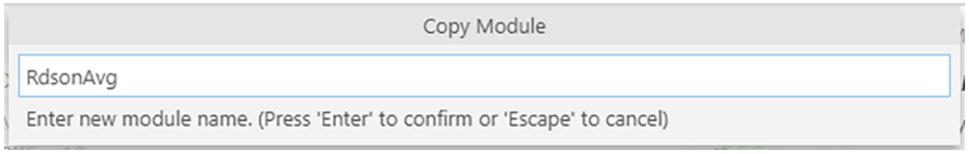
1. Open Visual Studio Code and the KULT sidebar
2. On the side bar under Libraries, select the KI42xxulib.
3. Under Modules, select the Rdson42XX user module.
4. Select the copy button to make a copy of this module.



5. Rename the copied module.

NOTE

When naming a user module, conform to case-sensitive C programming language naming conventions. Do not duplicate names of existing user modules or user libraries.



6. Press **Enter** to confirm the name.
7. Select on the new module `RdsonAvg` to open it in the editor.

Modify the RdsonAvg user module

In the user module code, you need to replace the `measi` commands with `avgi` commands. While a `measi` command makes a single measurement, an `avgi` command makes a specified number of measurements, and then calculates the average reading. For example:

```
avgi(SMU2, Id1, 10, 0.01);
```

For the above command, SMU2 makes 10 current measurements and then calculates the average reading (Id1). The 0.01 parameter is the delay between measurements (10 ms).

The source code for the module is in the module code area of the window. In this area, make the following changes.

Under `Force the first point and measure`, change the line:

```
measi(SMU2, Id1);
```

to

```
avgi(SMU2, Id1, 10, 0.01); // Make averaged I measurement
```

Under `Force the second point and measure`, change the line:

```
measi(SMU2, Id2);
```

to

```
avgi(SMU2, Id2, 10, 0.01); // Make averaged I measurement
```

Change the line:

```
*Rdson = (Vd2-Vd1)/(*Id2- *Id1); // Calculate Rdson
```

to

```
*RdsonAvg = (Vd2-Vd1)/(*Id2- *Id1); // Calculate RdsonAvg
```

Change a parameter name

To change the name of the Rdson parameter:

1. Open the KULT module using the function **Show Module View** on the side bar if not currently open.
2. Select the name of the `Rdson` parameter and change it to `RdsonAvg`

Figure 535: Change the name of the Rdson parameter

3. Select **Apply**.

Change the module description

In Clarius, any user test modules (UTMs) that are connected to this user module show the text that is entered in the Description section.

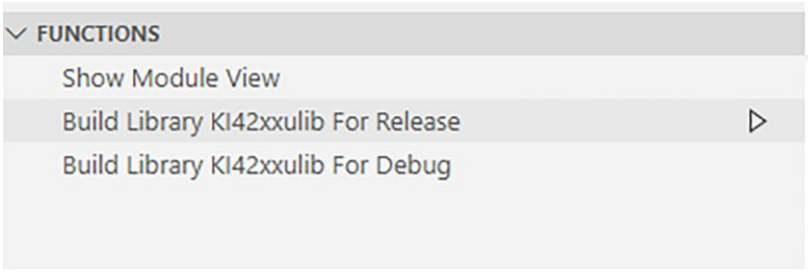
To change the module description:

1. Between the gray comments for `MODULE HELP DESCRIPTION` change `MODULE: Rdson42xx` to `MODULE: RdsonAvg`.
2. Replace all instances of `Rdson` with `RdsonAvg`.

Save and build the modified library

1. Select **File**, then **Save** or **Ctrl+S** to save the modified module.
2. Select the run icon next to the function **Build library KI42xxulib for Release** in the KULT side bar.

Figure 536: Build the modified library



3. Check the build output for errors.

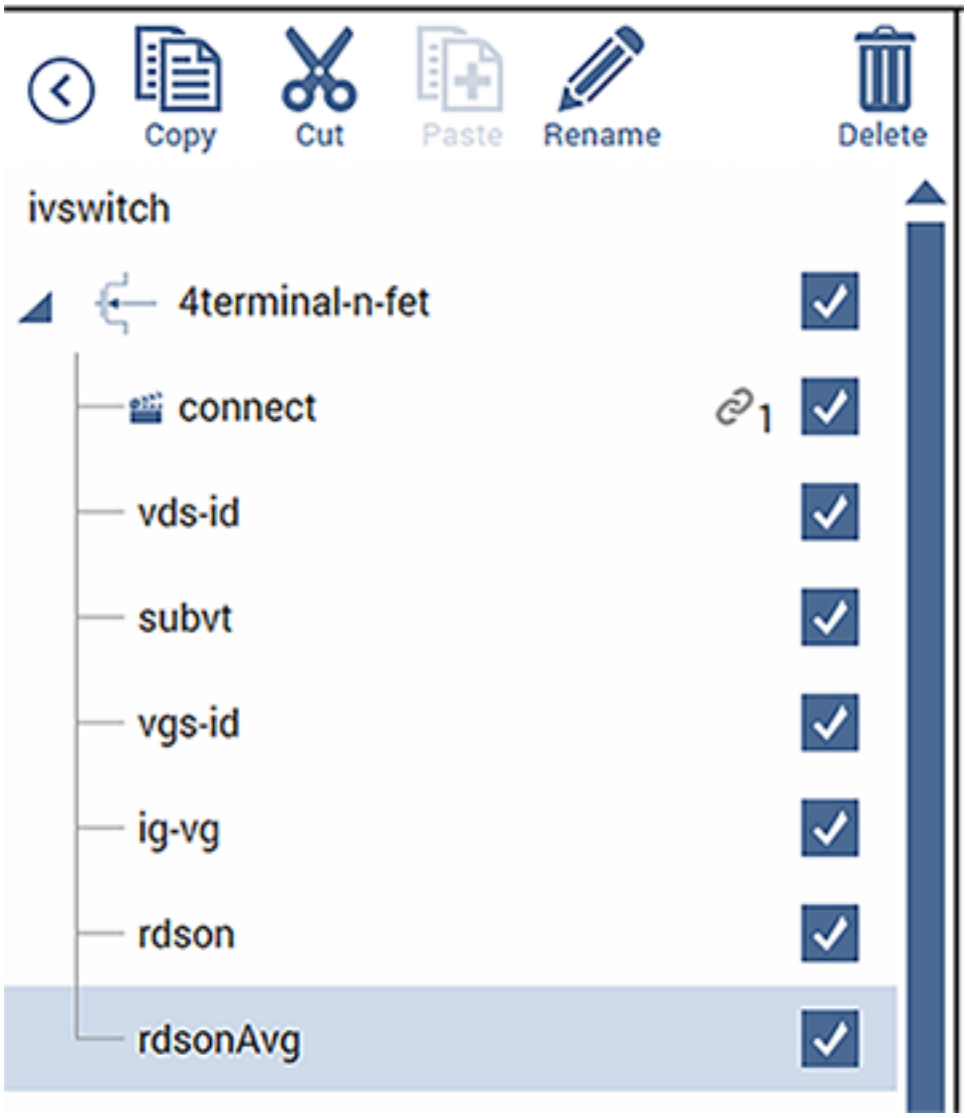
Add a new UTM to the ivswitch project

To add rdsonAvg to the ivswitch project:

1. Choose **Select**.
2. Select **Projects**.
3. In the Search box, enter **ivswitch** and select **Search**. The Library displays the I-V Switch Project (`ivswitch`).
4. Select **Create**. The `ivswitch` project replaces the previous project in the project tree.
5. Select the **Tests** tab.
6. For the Custom Test, select **Choose a test from the pre-programmed library (UTM)**.
7. Drag **Custom Test** to the project tree. The test has a red triangle next to it to indicate that it is not configured.
8. Select **Rename**.
9. Enter **rdsonAvg** and press **Enter**.
10. In the project tree, drag **rdsonAvg** to the `4terminal-n-fet device`, after the `rdson` test.
11. Choose **Configure**.
12. In the Test Settings pane, from the User Libraries list, select **KI42xxulib**.
13. From the User Modules list, select **Rdson42XX**.
14. Select **Save**.

The project tree for the `ivswitch` project with `rdsonAvg` added is shown in the following figure.

Figure 537: Add a new UTM to the ivswitch project



Tutorial five: Debugging a user module

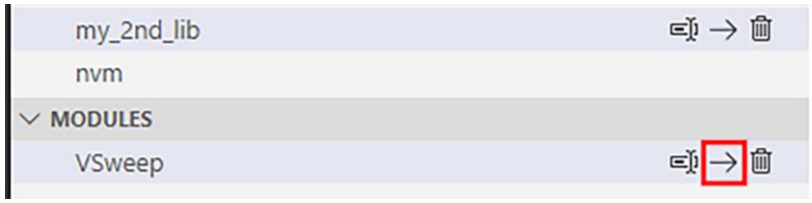
One of the most useful features of the KULT Extension and Visual Studio code is the ability to debug code with the GNU Debugger (GDB). Users can pause execution, monitor variables and expressions, and step through code one line at a time. This tutorial explores these features using the new module VSweepRes.

Creating the VSweepRes user module by copying an existing user module

To copy the VSweep user module:

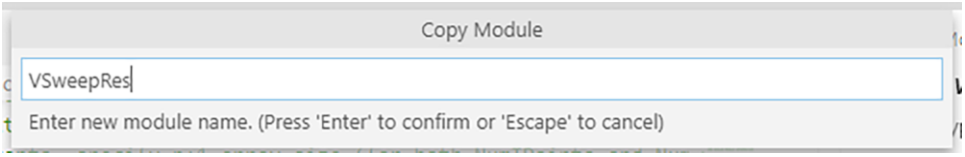
1. Start Visual Studio Code and open the KULT side bar.
2. Select `my_2nd_lib` from the library pane on the side bar.
3. Select on the `VSweep` module from the module pane on the side bar.
4. Select the copy icon next to the module to make a copy of it.

Figure 538: Copy the VSweep user module



5. Name the copied module `VSweepRes`.

Figure 539: Name the copied module VSwepRes



6. Select **Enter**.
7. Select `VSweepRes` in the side bar to open it in the editor.

Adding an average resistance calculation to VSweepRes

To add a calculation for the average resistance:

1. At the beginning of `VSweepRes`, after the line defining `int i`, add the following statement:

```
double sum = 0; /*Sum of all resistance measurements*/
```

2. Inside the for loop, after the line `v = v + vstep; /* Increments the forced voltage. */` add the following statement:

```
sum =(Vforce[i]/Imeas[i]); /*Intentionally incorrect line*/
```

   - That line is intentionally incorrect. We will find the error using the debugger later.

3. After the for loop, before the return statement, add the following statement:

```
*AvgRes = sum/NumIPoints; /*Divide by the number of measurements to get average. */
```

Adding another parameter to VSweepRes

On the KULT module, add a new parameter, AvgRes, as shown in the following table, just as you did during the first Tutorial.

Parameter Name	Data Type	I/O	Default	Min	Max
AvgRes	Double	Output			

Select **Apply** to add the new parameter to the function prototype.

Figure 540: Add a new parameter

Building the user library

To build the user library:

1. Save the `VSweepRes` module by going to **File**, then **Save** or by pressing **Ctrl+S**.
2. Select the run icon next to the function "Build library my_2nd_lib for release" on the KULT side bar.

Figure 541: Build library

3. Check the build output for any errors. The build should be successful.

Checking the VSweepRes user module

Check the user module as you did in the previous tutorials, by creating and executing a user test module (UTM) in Clarius. Refer to Checking the user module.

Before proceeding:

1. Connect a 1 kΩ 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.
3. Add to this project a UTM called `v_sweep_res_chk`.
4. Configure the `v_sweep_res_chk` UTM to execute the `VSweepRes` user module, which is found in the `my_2nd_lib` user library.
5. Run the `v_sweep_res_chk` UTM.
6. At the conclusion of execution, review the results in the Analyze sheet (or the Graph document, if configured). Notice that there is a new value returned in the sheet, `AvgRes`, which is the average calculated resistance. However, the value is incorrect. If you connected a 1 kΩ resistor, the value will be closer to 100 Ω. There is something wrong in the user module, so the debugger will be useful in finding the error.

Figure 542: Debugger

	VSweepRes	Imeas	Vforce	AvgRes
1	0	1.0000E-6	000.0000E-3	89.2412E+0
2		509.3060E-6	500.0000E-3	
3		1.0190E-3	1.0000E+0	
4		1.5281E-3	1.5000E+0	
5		2.0377E-3	2.0000E+0	

Starting the debugger

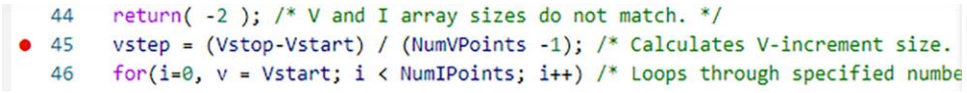
To start the debugger:

1. Return to Visual Studio Code. Build the `VSweepRes` user module for Debug by clicking the function **Build Library my_2nd_lib For Debug**. This is necessary to create the symbols needed for breakpoints.

Figure 543: Starting the debugger

2. Return to Clarius. Click away from the v_sweep_res_chk test and back to reload the module.
3. Leave Clarius running and return to Visual Studio Code.
4. Select the white space to the left of the line that calculates the V-increment size to place an unconditional breakpoint. Code execution will pause at this line.

Figure 544: Breakpoint

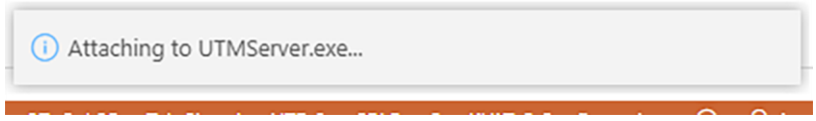


NOTE

The attach process will remove the breakpoint automatically but this step is necessary for breakpoints to function during code execution.

5. **F5** or go to **Debug** then **Start Debugging** to start the debugger. A status message will appear in the bottom right hand corner.

Figure 545: Attach to UTMServer



6. Wait for the attach process to complete. The message will disappear when it is done.

Debugging the code

Once the attach process is complete, breakpoints can be set and the code can be executed.

To start debugging the code:

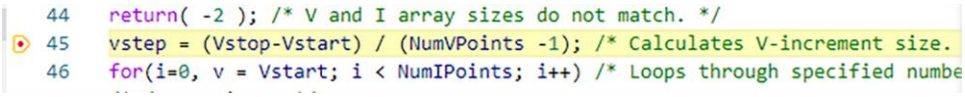
1. Select on the breakpoint to remove it and click again to reenable it.

NOTE

If the circle turns from red to hollow gray, the attach process is still completing. Click the breakpoint to remove it and then click again to replace it. The circle will stay red when the breakpoint is fully bound.

2. Select the Run button in Clarius to start the code.
3. Return to Visual Studio Code. The code will pause on the breakpoint.

Figure 546: Code paused on breakpoint



4. On the variables pane in the debug side bar, find the sum variable. Right click the variable and click **Add to watch.** The variable will appear by itself in the watch pane on the side bar.

Figure 547: Add to watch, variables pane

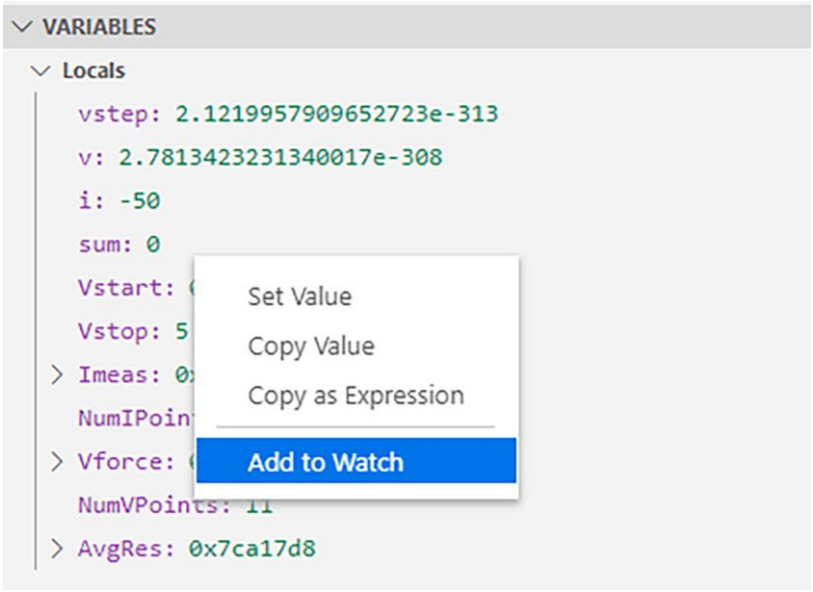


Figure 548: Sum added to watch

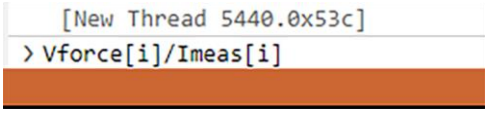
5. Select the **Step over** button on the debug toolbar or press **F10** to step through the code line by line. When you get to line `sum =(Vforce[i]/Imeas[i]);`, the code loops back to the top of the for loop. Continue until you get to the sum line again.

Figure 549: Step over

6. Select the **Step into** button or press **F10** one more time. This line has now executed and the sum value has changed.
7. Select the **Debug Console** at the bottom of the screen. Enter the formula below:

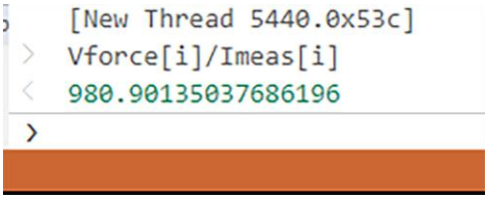
```
Vforce[i]/Imeas[i]
```

Figure 550: Enter formula in Debug Console



8. Press **Enter**. Notice that the value returned is the same as the value in sum and is approximately the value of the resistor. We now know that we are correctly calculating the resistance from the current measurement.

Figure 551: Returned value



9. Continue stepping through the code until you get to the top of the `for` loop again. The value for sum is not changing. Therefore, our sum formula must be incorrect.
10. Press **F5** the **Continue** button on the debug toolbar. This will run the code until completion.
11. Select to the **Terminal** tab at the bottom.
12. Type **Ctrl+C.** This terminates the debug session.

NOTE

Pressing the disconnect button on the tool bar will prevent Visual Studio Code from reconnecting to the UTM Server unless it is closed and reopened.

13. Correct the sum line from

```
sum =(Vforce[i]/Imeas[i]); /*Intentionally incorrect line*/
to
sum = sum + (Vforce[i]/Imeas[i]); /*Sum Resistances*/
```

14. Rebuild the library for release by selecting the command, Build Library `my_2nd_lib` For Release on the KULT side bar.

Retest the VSweepRes user module in Clarius

1. Return to Clarius. Click away from the `v_sweep_res_chk` test and back to reload the module.
2. Rerun the module with the same settings as before, 0 to 5 V with number of V and I points as 11.
3. Select the **Analyze** view. Note that the resistance value is now correct.

Figure 552: Analyze view

	VSweepRes	Imeas	Vforce	AvgRes
1	0	972.3850E-9	000.0000E-3	892.5630E+0
2		509.4740E-6	500.0000E-3	
3		1.0186E-3	1.0000E+0	
4		1.5280E-3	1.5000E+0	
5		2.0371E-3	2.0000E+0	
6		2.5458E-3	2.5000E+0	

Section 10

Keithley External Control Interface (KXCI)

In this section:

Introduction

The Keithley External Control Interface (KXCI) allows you to use an external computer to remotely control the SMUs and pulse generator cards in the 4200A-SCS. You can use remote control over the general-purpose instrument bus (GPIB) or ethernet. You define the type of communications in KCon (in the KXCI Settings options). When controlled by an external computer, the 4200A-SCS functions like any other GPIB instrument.

This section contains:

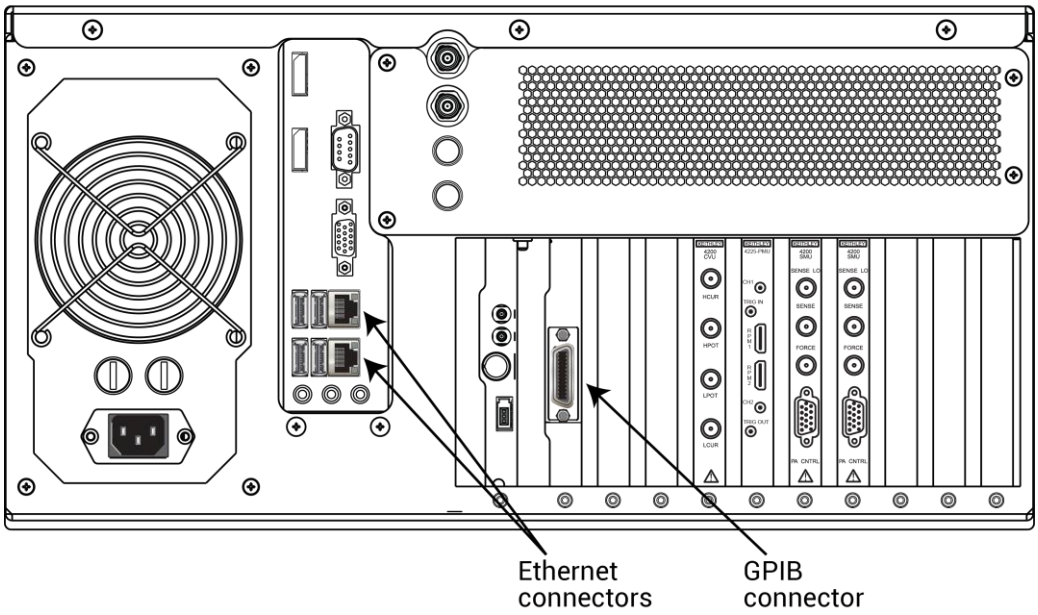
- KXCI user interface: Describes how to use the KXCI user interface, which is used to control GPIB operation.
- GPIB command set: The command strings to control the source-measure operations of the 4200A-SCS are summarized in three tables.
- GPIB command reference: Provides detailed reference information on the command set.
- Ethernet command reference: Describes the command set that can be used for Ethernet communications.
- 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 4200A-SCS.
- Sample programs: Provides 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 commands: Provides reference information for the commands to control the pulse generator 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.

KXCI communications connections

You can communicate with the 4200A-SCS using GPIB or ethernet. The connections for each are described in the following topics.

The locations of the connections are shown in the following figure.

Figure 553: GPIB and ethernet connectors on the 4200A-SCS



GPIB connections

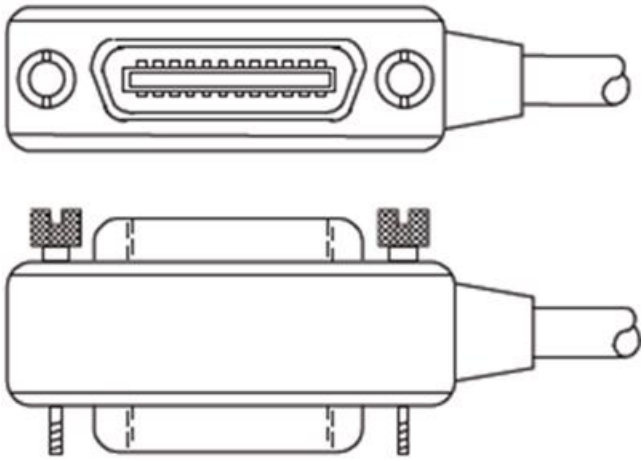
To connect the 4200A-SCS, use a GPIB cable equipped with standard GPIB connectors, as shown in the figure below. Either end of this cable mates to the GPIB connector on the rear panel of the 4200A-SCS. Connect the other end of the cable to the GPIB connector on the computer.

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 GPIB cables, such as the Keithley Instruments Models 7007-1 and 7007-2.

Figure 554: Standard IEEE-488 connectors



GPIB standards

The GPIB cable 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 4200A-SCS conforms to these standards:

- IEEE-488.1-1987
- IEEE-488.2-1992

Ethernet connections

Use a standard cable (CAT-5, RJ-45 terminated) to connect to the 4200A-SCS, as shown in the figure in Communications connections (on page 10-3).

Using KCon to configure KXCI

You use the Keithley Configuration Utility (KCon) to configure KXCI. Use KCon to select and configure communications (GPIB or ethernet) if you are using 4200A-SCS as a remote instrument. If you are using 4145 emulation, you use KCon to assign source-measure functions to the installed SMUs.

For details on KXCI that are set through KCon, refer to Keithley Configuration Utility (KCon) (on page 7-1).

NOTE

Before opening KCon to change the present KXCI configuration, you must close KXCI.

The presently selected communications interface (GPIB or ethernet) and its settings are displayed in the KXCI console. The command and message area below the KXCI settings displays sent commands, KXCI error messages, and numerical test results (refer to Using KXCI (on page 10-56)).

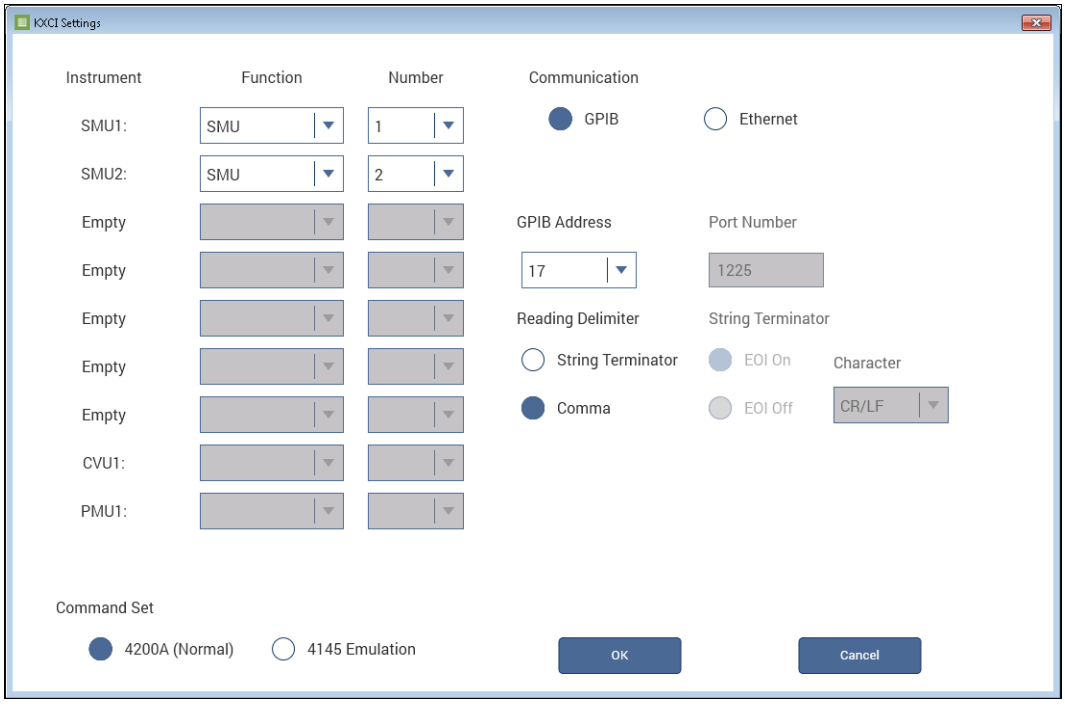
Setting the remote control mode (GPIB or ethernet)

Set the remote control mode on the 4200A-SCS through KCon. By default, the 4200A-SCS is set up for GPIB remote control.

To set up the remote control mode on the 4200A-SCS:

1. Open KCon.
2. Select **KXCI Settings**. The **KXCI Settings** dialog box is displayed, as shown in the following figure.
3. Select the **Command Set**. You can select the Keithley 4200A (Normal) (the default) or Keysight 4145 Emulation. In many cases, test programs developed for use with a 4145B Parameter Analyzer run without modification when they are used with a 4200A-SCS running KXCI.

Figure 555: KXCI Settings dialog box



To set up GPIB communications:

1. Select **GPIB**.
2. Set the **GPIB Address**.
3. Select a **Reading Delimiter**. The delimiter determines if the delimiter for output data is a string terminator or a comma.
4. If you selected String Terminator, select the **String Terminator**. The string terminator determines if an EOI (End or Identify) signal is sent after data transfer.

To set up ethernet communications:

1. Select **Ethernet**.
2. Set the **Port Number** (default is 1225).

GPIB communications

Starting KXCI and the GPIB command interpreter

NOTE

You cannot run Clarius and KXCI at the same time. When Clarius is running, the 4200A-SCS is the controller and controls all internal and external instruments. When KXCI is running, the 4200A-SCS is a subordinate to a controlling computer over GPIB or ethernet.

To start KXCI:

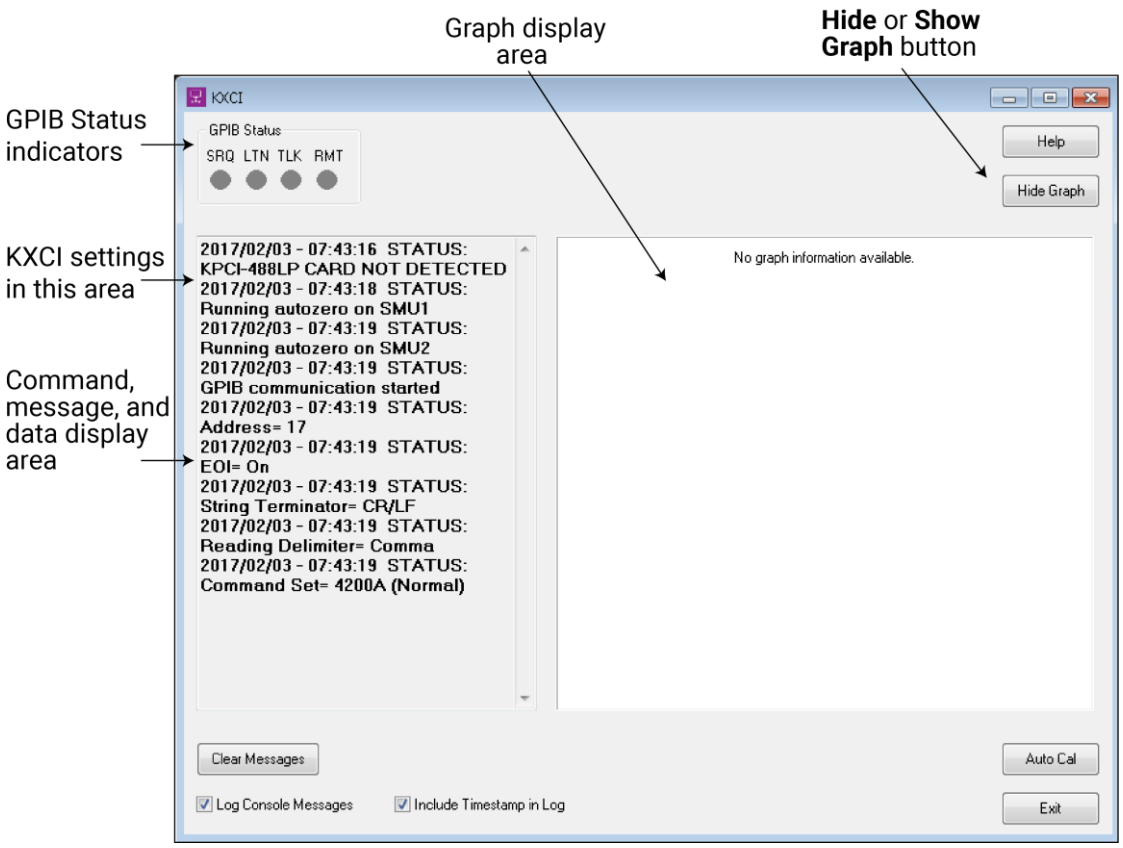
1. Close Clarius.
2. Select the KXCI icon on the desktop.

The KXCI user interface opens.

NOTE

The **GPIB Status Indicators** only apply if communications is set to GPIB.

Figure 556: KXCI user interface



NOTE

For further information about controlling the 4200A-SCS through ethernet, refer to Communications connections (on page 10-3) and KXCI ethernet client driver (on page 10-60).

GPIB status indicators (GPIB communications only)

After KXCI is started, indicators in the GPIB Status box report interface status. A green indicator signals the present status:

- **SRQ:** Turns on when an error or operating condition occurs.
- **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. To hide the graph, select the Hide Graph button, which provides a larger display area for commands, error messages, and test results.

GPIB command set

GPIB commands to control instrument operation are grouped as follows:

- **Page commands:** KXCI locates the GPIB instrument control routines in pages that are similar to the Keysight Model 4145B command pages. Therefore, before sending an instrument control command string, you must send the appropriate page command. The page commands are summarized in Page commands (on page 10-9).
- **System mode commands**: This comprehensive set of commands uses all of the source-measure capabilities of up to eight SMUs installed in the 4200A-SCS. These commands are summarized in System mode commands (on page 10-9).
- **User mode commands**: This limited set of commands perform basic source-measure operation. These commands are summarized in User mode commands (on page 10-10).
- **Common commands**: These commands are valid in any operating mode. These commands are summarized in Commands common to system and user modes (on page 10-11).
- **4200A only commands:** These commands are specific to the 4200A mode (cannot be used if 4145 Emulation is selected in the KXCI Settings in KCon). They are summarized in 4200A only command set (on page 10-11).

The KXCI GPIB command sets for the SMUs and PMUS and PGUs are similar to the command set used by the Keysight Model 4145B. This similarity allows many programs already developed for the Keysight Model 4145B to run on the 4200A-SCS.

NOTE

Detailed information on the command set is presented after the tables.

Page commands

Page command	Function
DE (on page 10-13)	Accesses SMU channel definition page.
SS (on page 10-17)	Accesses source setup page.
SM (on page 10-30)	Accesses measurement setup page.
MD (on page 10-36)	Accesses measurement control page.
US (on page 10-42)	Accesses user-mode page.
UL (on page 10-103)	Allows for use of direct user library module calls – see Calling KULT user libraries remotely (on page 10-103).

System mode commands

The system mode commands are listed in the following topics. Full descriptions are provided in System mode commands (on page 10-12).

DE page command strings

The DE page command strings include:

- CH: SMU channel definition (on page 10-14)
- VS: VS1...VSn channel definition (on page 10-16)
- VM: VM1...VMn channel definition (on page 10-17)

SS page command strings

The SS page command strings include:

- VR and IR: VAR1 setup (on page 10-18)
- VP and IP: VAR2 setup (on page 10-20)
- ST command: Auto standby (on page 10-22)
- FS command: VAR1' setup (on page 10-23)
- RT command: VAR1' setup (on page 10-25)
- VL and IL commands: List sweep setup (on page 10-27)
- VC and IC commands: SMU constant voltage or current setup (on page 10-28)
- SC command: Constant VS setup (on page 10-29)
- HT command: Set sweep hold time (on page 10-29)
- DT command: Set sweep delay time (on page 10-30)

SM page command strings

The measurement setup (SM) page command strings include:

- WT command: Set wait time (on page 10-31)
- IN command: Set interval (on page 10-31)
- NR command: Select number of readings (on page 10-32)
- DM command: Select display mode (on page 10-32)
- LI command: List mode, enables V and I functions for test sequence (on page 10-33)
- XN command: Configure graph X-axis for electrical parameter (on page 10-34)
- XT command: Configure graph X-axis for time domain (on page 10-34)
- YA command: Configure graph Y1 axis (on page 10-35)
- YB command: Configure graph Y2 axis (on page 10-36)

MD page command strings

The measurement control (MD) page command string includes:

- ME command: Control measurements (on page 10-37)

Data output and file commands

The data output and file commands command strings are valid in any system mode page. They are:

- DO command: Obtain output data (on page 10-38)
- SR command: Set fixed source range (on page 10-42)
- SV command: Save file (on page 10-39)
- GT command: Get file (on page 10-40)
- MP command: Map channel (on page 10-41)

User mode commands

The user mode commands are listed in the following. Full descriptions are provided in User mode commands (on page 10-42).

- DV and DI commands: SMU setup (on page 10-43)
- DS Command: VS1...VS9 setup (on page 10-44)
- TV and TI commands: Triggering (on page 10-45)

Commands common to system and user modes

Some commands are common to the system and user modes, as listed below. Descriptions of these commands are provided in Commands common to system and user modes (on page 10-47).

4200A command set only

The 4200A command set only command is:

GPIB command reference

GPIB commands to control instrument operation are grouped as follows:

- **System mode commands**: This comprehensive set of commands allows you to use all the source-measure capabilities of the SMUs installed in 4200A-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.

These commands include 4200A only commands, which are valid only while using the 4200A command set instead of the 4145 Emulation command set.

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 2.

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 mapping command
- Fixed source ranging command

Channel definition page (DE)

Use the command strings for the DE page for the following operations:

- SMU channel definition
- `VS1`...`VS`*`n`* channel definition
- `VM1`...`VM`*`n`* channel definition

To send the channel definition command strings to the 4200A-SCS, you must select the channel definition page by sending the command:

```
DE
```

CH

This command defines a SMU channel.

Usage

```
CHA, 'BBBBBB', 'CCCCCC', D, E
```

A	1, 2, ... or *n* SMU channel number; the largest value of *n* is the number of channels in the system (9 maximum); if no parameters are specified after the channel number, the channel is turned off
BBBBBB	Voltage name
CCCCCC	Current name
D	Source mode or common: ■ 1: Voltage source mode ■ 2: Current source mode ■ 3: Common (output high connected to common)
E	Source function: ■ 1: VAR1 sweep source function ■ 2: VAR2 step source function ■ 3: Constant (fixed) source function ■ 4: VAR1' source function

Details

For every used channel that is configured as a SMU, you must specify names for voltage and current, select the source mode (voltage, current, or common), and select the source function (VAR1, VAR2, constant, or VAR1').

When the source mode (*D*) is set to common (3), the source function (*E*) must be set to constant (3).

The VAR1 source function performs 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 +1 V to +3 V using 1 V 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 = (1 V × 2) + 1 = 3 V

VAR1' step 2 = (2 V × 2) + 1 = 5 V

VAR1' step 3 = (3 V × 2) + 1 = 7 V

To disable a channel, send:

```
CHx
```

Where *x* = 1 to 9

Example 1

```
CH3, 'V1', 'I1', 1, 3
```

This command string sets up the SMU assigned to channel 3 to source a fixed voltage (1 V). The specified names for voltage and current are `V1` and `I1`, respectively.

Example 2

```
CH1; CH2; CH3; CH4
```

This command string disables channels 1 through 4.

Also see

KXCI Settings tab (on page 7-19)
System mode commands (on page 10-9)

VS

This command specifies the name and selects the source function for each voltage-source channel.

Usage

```
VSA, 'BBBBBB', C
```

A	Voltage source (1 to 9); the assigned value depends on how instruments are mapped in KCon
BBBBBB	User-specified name (up to 6 characters)
C	Source function: ■ 1: VAR1 ■ 2: VAR2 ■ 3: Constant ■ 4: VAR1'

Details

KXCI allows up to eight source-measure units to function solely as voltage sources. You can use any channel for any voltage-source function between VS1 and VS9. For example, in a system containing four SMUs, you can use SMU2 as VS5.

For each voltage source that is used, you must specify a name and select the source function. The VAR1 function performs a voltage sweep that is synchronized to the steps of VAR2. The VAR1 source function performs a linear or logarithmic voltage sweep. The VAR1 sweep is done 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.

If nothing is specified after the prefix and channel number, the channel is not used.

To source voltage using the 4145B VS1...VS*n* function, define one of the 4200A-SCS SMUs to emulate the VS.

Example

```
VS1, 'VS1', 1
```

This command string sets up the channel used by VS1 to perform a voltage sweep.

Also see

None

VM

This command defines channels that are used as voltmeters.

Usage

`VMA, 'BBBBBB'`

A	Voltmeter channel: 1 to 9; the assigned value for a voltmeter depends on how instruments are mapped in KCon
BBBBBB	User-specified name (up to 6 characters)

Details

KXCI allows up to eight source-measure units (SMUs) to function solely as voltmeters. You can use any channel for any voltmeter function between VM1 and VM9. For example, in a system containing four SMUs, you can use `SMU3` as `VM7`.

If you do not define one of the 4200A-SCS SMUs to emulate a VM, attempts to measure voltages through the nonexistent VM result in data values of `9.000e+37`.

If nothing is specified after the prefix and channel number, the channel is not used.

To measure voltage using the 4145B `VM1...VMn` function, define one of the 4200A-SCS SMUs to emulate `VM`.

Example

```
VM1, 'VM1'
```

This command string defines the channel used for VM1 (specifies the name `VM1`).

Also see

KXCI Settings tab (on page 7-19)

Source setup page (SS)

Use the command strings for the SS page 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 4200A-SCS, you must first select the source setup page by sending the command:

```
SS
```

VR and IR

This command sets up the VAR1 source function.

Usage

AAB, ±CCC.CCCC, ±DDD.DDDD, ±EEE.EEEE, ±FFF.FFFF

AA	The source mode: ■ VR: Voltage source (SMU or VS1...VS9) ■ IR: Current source (SMU only)
B	The sweep type: ■ 1: Linear sweep ■ 2: Log10 sweep ■ 3: Log25 sweep ■ 4: Log50 sweep
CCC.CCCC	Start value: ■ Voltage source: –210.00 V to +210.00 V ■ Current source, 4200-SMU: –0.1050 A to +0.1050 A ■ Current source, 4210-SMU: –1.0500 A to +1.0500 A
DDD.DDDD	Stop value: ■ Voltage source: –210.00 V to +210.00 V ■ Current source, 4200-SMU: –0.1050 A to +0.1050 A ■ Current source, 4210-SMU: –1.0500 A to +1.0500 A
EEE.EEEE	Step value (linear sweep only): ■ Voltage source: –210.00 V to +210.00 V ■ Current source, 4200-SMU: –0.1050 A to +0.1050 A ■ Current source, 4210-SMU: –1.0500 A to +1.0500 A For a log sweep, do not set a step value (steps are determined by the setting for *B*)
FFF.FFFF	Compliance value (also see **Details**): ■ Voltage source: –210.00 V to +210.00 V ■ Current source, 4200-SMU: –0.1050 A to +0.1050 A ■ Current source, 4210-SMU: –1.0500 A to +1.0500 A For a log sweep, do not set a compliance value

Details

If you specify a voltage start or step value below 0.001 V, KXCI automatically sets the value to zero.

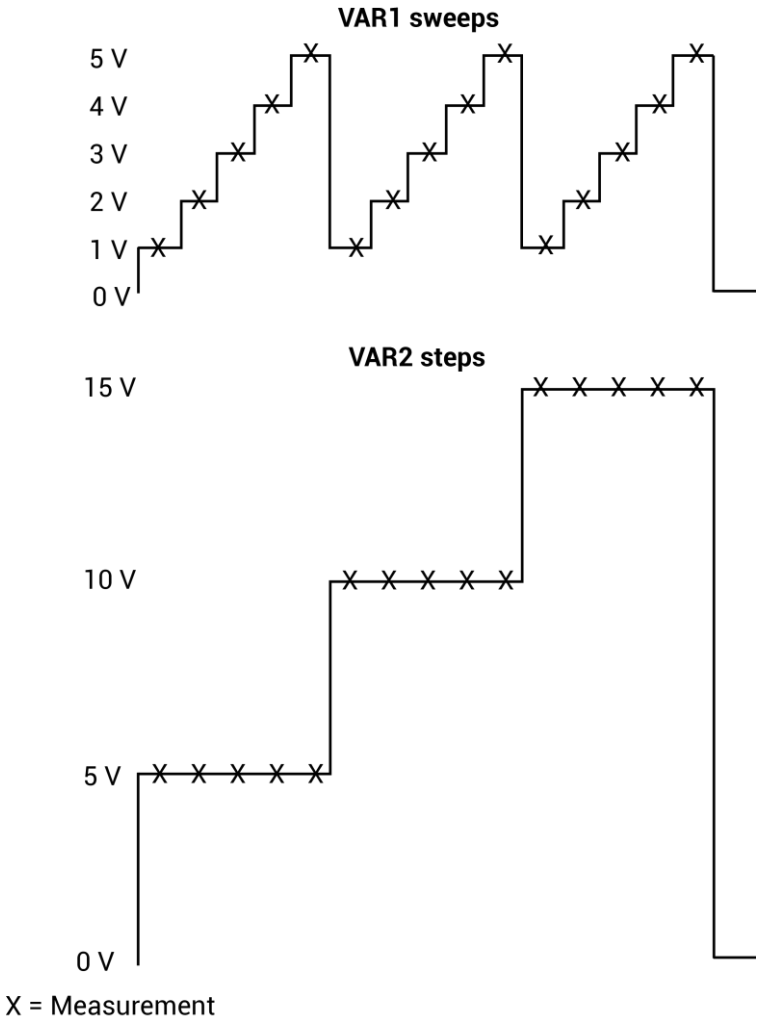
When setting the step value, be aware that the maximum number of points for VAR1 is 1024:

Number of points = (int)(Abs((Stop Value - Start Value) / Step Value) + 1.5)

When you are setting a voltage source mode (VR), you are setting a current compliance value. If you are setting a current source mode (IR), you are 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 preamplifier installed and 100 nA without a preamplifier), KXCI sets it to the minimum allowable value.

When VAR1 is a selected source function, it does 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, as shown in the following figure.

Figure 557: Synchronized VAR1 sweeps and VAR2 steps



If the source is a SMU, the source mode for the sweep can be voltage or current. If, however, a voltage source (`VS1...Sn`) is used, the source mode must be voltage.

You can do the sweep 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 the figure above sweeps from 1 V (start) to 5 V (stop) in 1 V steps.

With a logarithmic sweep mode selected (log base 10, 25, or 50), only specify the start and stop values. Step size is automatically set to provide a symmetrical sweep on the logarithmic scale.

NOTE

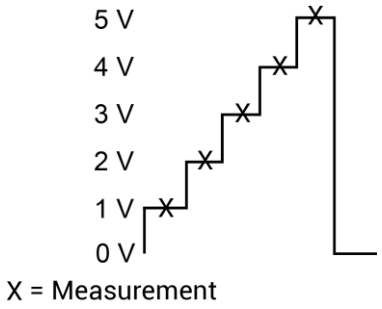
The time spent on each sweep step depends on the user-set delay time and the time it takes to perform the measurement.

You can delay the start of the sweep by setting a hold time.

Example

```
VR1, 1, 5, 1, 0.01
```

This command string sets up a VAR1 linear sweep with a start = 1 V, stop = 5 V, step = 1 V, and compliance = 10 mA.
The following figure shows the sweep that results from this setup. The figure in the **Details** shows the results of the same setup when used with a VAR2 step command.



Also see

None

VP and IP

This command sets up the VAR2 step sweep.

Usage

AA ±*BBB.BBBB*, ±*CCC.CCCC*, *DD*, ±*EEE.EEEE*
AA ±*BBB.BBBB*, ±*CCC.CCCC*, *DD*, ±*EEE.EEEE*, *FF*

AA	The source mode: ■ VP: Voltage source (SMU or VS1...VS8) ■ IP: Current source (SMU only)
BBB.BBBB	Start value: ■ Voltage source: –210.00 V to +210.00 V ■ Current source, 4200-SMU: –0.1050 A to +0.1050 A ■ Current source, 4210-SMU: –1.0500 A to +1.0500 A
CCC.CCCC	Step value: ■ Voltage source: –210.00 V to +210.00 V ■ Current source, 4200-SMU: –0.1050 A to +0.1050 A ■ Current source, 4210-SMU: –1.0500 A to +1.0500 A
DD	Number of steps: 1 to 32
EEE.EEEE	Compliance value (see **Details**): ■ Voltage source: –210.00 V to +210.00 V ■ Current source, 4200-SMU: –0.1050 A to +0.1050 A ■ Current source, 4210-SMU: –1.0500 A to +1.0500 A
FF	VAR2 source stepper index: 1 to 4 (see **Details**)

Details

With the voltage source mode (`VP`), the output value is in volts. For the current source mode (`IP`), the output value is in amps.

If you specify a voltage start or step value below 0.001 V, KXCI automatically sets the value to zero.

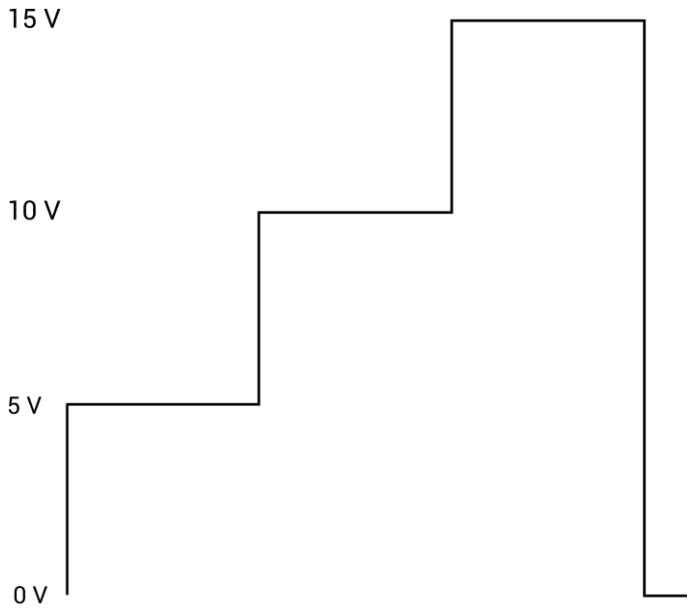
With the voltage source mode (`VP`), you are setting the current compliance. For the current source mode (`IP`), you are setting the voltage compliance. When sourcing voltage, note that if you specify a compliance current that is below the minimum allowable value (100 pA with a preamplifier installed and 100 nA without a preamplifier), KXCI sets it to the minimum allowable value.

If you are using is a SMU as the source, the source mode for the VAR2 steps can be voltage or current. If, however, a voltage source (`VS1...VS9`) is being used, the source mode must be voltage (for configuration details, refer to Keithley Configuration Utility (on page 7-1)).

If the source stepper index (*FF*) is omitted, the default is `1` (first stepper). Use the `CH` command to define one or more VAR2 sources. The first defined VAR2 is index = `1`. Second channel defined as VAR2 is index = `2`, and so on up to a maximum of four VAR2 sources. Note that this is an extension to the traditional VAR2 capability.

Parameters for the VAR2 step function include the start value, step value, and the number of steps. In the following figure, the VAR2 step function starts at 5 V, steps in 5 V increments and has 3 steps.

Figure 558: Steps resulting from the VP 5, 5, 3, 0.01 command string



Example

```
VP 5, 5, 3, 0.01
```

This command string sets up a VAR2 voltage sweep with start = 5 V, step = 5 V, number of steps = 3, and compliance = 10 mA. This command string performs the steps shown in the figure in the **Details**. These are the same steps that are shown synchronized with sweeps in VR and IR: VAR1 setup (on page 10-18).

Also see

CH: SMU channel definition (on page 10-14)

ST

This command controls auto standby.

Usage

```
ST A, B
```

A	SMU channel number (1 to 9)
B	Enable or disable auto standby: ■ 0: Disable auto standby ■ 1: Enable auto standby

Details

For a SMU, the ST command controls auto standby. When auto standby is enabled, the SMU automatically goes into standby when the test completes. When disabled, the output stays on when the test completes.

Also see

None

FS

This command sets the offset value when VAR1' is a selected source function.

Usage

```
FS ±AAA.A
FS ±AAA.A,B
```

AAA.A	Offset value: –210 to 210
B	Available SMU channel (1 to 9); if this parameter is not included, offset applies to all channels that are configured to VAR1'.

Details

When VAR1' is a selected source function, it does the VAR1 sweep with each step scaled by the ratio (RT) value and offset (FS) value as follows:

VAR1' sweep step = (VAR1 sweep step x RT) + FS

The VAR1 sweep shown in the example in VR and IR (on page 10-18) has five steps: 1 V, 2 V, 3 V, 4 V, and 5 V. The corresponding VAR' sweep has the following steps when RT=3 and FS = 2:

VAR1' step 1 = (1 V × 3) + 2 = 5 V

VAR1' step 2 = (2 V × 3) + 2 = 8 V

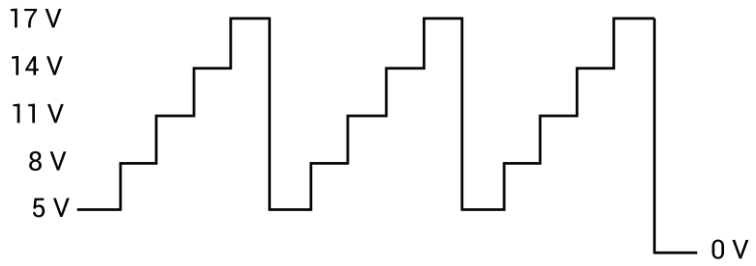
VAR1' step 3 = (3 V × 3) + 2 = 11 V

VAR1' step 4 = (4 V × 3) + 2 = 14 V

VAR1' step 5 = (5 V × 3) + 2 = 17 V

This VAR1' sweep (not drawn to scale) is shown in the following figure.

Figure 559: VAR' sweep when ratio = 3 and full-scale = 2



You can assign a unique offset value to any available SMU channel in the system.

Example

```
RT +3,2              Ratio
FS +2,2              Offset
```

These command strings set up the `VAR1'` sweep shown in the figure in the Details (ratio = 3, offset = 2). The above commands set up `VAR1'` for SMU Channel 2. When Channel 2 is defined for a `VAR1'` sweep, ratio is set to 3 and offset is set to 2.

Also see

None

RT

This command sets the ratio value when VAR1' is a selected source function.

Usage

```
RT ±AA.A
RT ±AA.A,B
```

AAA.A	Ratio value: –10 to 10
B	Available SMU channel (1 to 9); if this parameter is not included, ratio applies to all channels that are configured to VAR1'

Details

When VAR1' is a selected source function, it does the VAR1 sweep with each step scaled by the ratio (RT) value and offset (FS) value as follows:

VAR1' sweep step = (VAR1 sweep step x RT) + FS

The VAR1 sweep shown in the example in VR and IR (on page 10-18) has five steps: 1 V, 2 V, 3 V, 4 V, and 5 V. The corresponding VAR' sweep has the following steps when RT=3 and FS = 2:

VAR1' step 1 = (1 V × 3) + 2 = 5 V

VAR1' step 2 = (2 V × 3) + 2 = 8 V

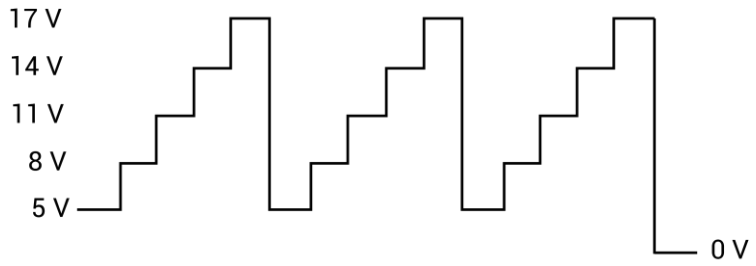
VAR1' step 3 = (3 V × 3) + 2 = 11 V

VAR1' step 4 = (4 V × 3) + 2 = 14 V

VAR1' step 5 = (5 V × 3) + 2 = 17 V

This VAR1' sweep (not drawn to scale) is shown in the following figure.

Figure 560: VAR' sweep when ratio = 3 and full-scale = 2



You can assign a unique ratio value to any available SMU channel in the system.

Example

```
RT +3,2              Ratio
FS +2,2              Offset
```

These command strings set up the `VAR1'` sweep shown in the figure in the **Details** (ratio = 3, offset = 2). The above commands set up `VAR1'` for SMU Channel 2. When Channel 2 is defined for a `VAR1'` sweep, ratio is set to 3 and offset is set to 2.

Also see

None

VL and IL

This command sets up a list sweep.

Usage

AAB, C, ±DDD.DDDD, ±EE.EEEE, ... ±EE.EEEE

AA	The source mode: ■ Voltage source (SMU or VS1...VS9): VL ■ Current source (SMU only): IL
B	Channel number: 1 to 9
C	Master or subordinate mode: ■ Subordinate mode: 0 ■ Master mode: 1
DDD.DDDD	Compliance value: ■ Voltage source: –210.00 V to +210.00 V ■ Current source, 4200-SMU: –0.1050 A to +0.1050 A ■ Current source, 4210-SMU: –1.0500 A to +1.0500 A
EE.EEEE	List values: ■ Voltage source: –210.00 V to +210.00 V ■ Current source, 4200-SMU: –0.1050 A to +0.1050 A ■ Current source, 4210-SMU: –1.0500 A to +1.0500 A

Details

When list sweep is a selected source function, it does a list sweep with arbitrary sweep points in order of the defined sweep parameters. If the source that is being used is a 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.

With the voltage source mode (VR) specified in the command string, you are setting the current compliance. For the current source mode (IR), you are 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 preamplifier installed and 100 nA without a preamplifier, KXCI sets it to the minimum allowable value.

The maximum number of points for the list is 4096. Sweep points must be delimited by commas.

Example

```
VL1,1, 0.01, 1, 5, 2
```

This command string sets up a channel 1 list sweep (1 V, 5 V, 2 V arbitrary steps, and compliance = 10 mA).

Also see

None

VC and IC

This command configures the SMU to output a fixed (constant) voltage or current level.

Usage

AAB, ±CCC.CCCC, ±DDD.DDDD

AA	Source mode: ■ Voltage: VC ■ Current: IC
B	SMU channel number: 1 to 9
CCC.CCCC	Output value: ■ Voltage source: –210.00 V to +210.00 V ■ Current source, 4200-SMU: –0.1050 A to +0.1050 A ■ Current source, 4210-SMU: –1.0500 A to +1.0500 A
DDD.DDDD	Compliance value: ■ Voltage source: –210.00 V to +210.00 V ■ Current source, 4200-SMU: –0.1050 A to +0.1050 A ■ Current source, 4210-SMU: –1.0500 A to +1.0500 A

Details

For any channel configured as a SMU, this command configures the SMU to output a fixed (constant) voltage or current level.

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.

Also see

Keithley Configuration Utility (KCon) (on page 7-1)

SC

This command configures the source to output a fixed voltage level for channels that are configured to be voltage source only.

Usage

```
SCA, ±BBB.BBBB
```

A	Voltage source channel: 1 to 9; see **Details**
BBB.BBBB	Output voltage value: –210.00 to +210.00

Details

This command is for use with any channel configured solely as a VS1...VS*n* voltage source. It configures the source to output a fixed (constant) voltage level.

If nothing is specified after the channel number, the channel is turned off.

The range of possible values for *A* depends on how instruments are mapped in KCon. The parameter *A* represents the *n* in voltage source VS*n*.

Example

```
SC1, 20
```

This command string sets up VS1 to output a constant 20 V level.

Also see

Keithley Configuration Utility (KCon) (on page 7-1)

HT

This command sets a hold time that delays the start of a sweep.

Usage

```
HT AAA.A
```

AAA.A	Hold time in seconds: 0 to 655.3

Details

You can delay the start of a sweep by setting a hold time. When the sweep is triggered, it starts after the hold time period expires.

Example

```
HT 1
```

This command string sets the hold time to one second.

Also see

None

DT

This command sets the time duration spent on each step of the sweep for a VAR1 sweep.

Usage

```
DT A.AAA
```

A.AAA	Delay time in seconds: 0 to 6.553

Details

For a VAR1 sweep, the time spent on each step of the sweep is determined by the user-set delay time and the time it takes to make a measurement.

You typically use the delay time to allow the source to settle before making a measurement. For example, assume a delay time of 1 s. At each step of the sweep, the source is allowed to settle for 1 s before the measurement is made.

Example

```
DT 1
```

This command string sets delay time to 1 s.

Also see

None

Measurement setup page (SM)

Use the command strings for the SM page for the following operations:

- Set wait time
- Set interval
- Select number of readings
- Select list display mode

To send the following command strings to the 4200A-SCS, select the `SM` page by sending the command:

```
SM
```

WT

This command delays the start of a test sequence for time domain measurements.

Usage

```
WT AAA.AAA
```

AAA.AAA	Wait time in seconds: 0 to 100

Details

For time domain measurements, you can delay the start of a test sequence by setting a wait time. The test sequence starts after the wait time period expires.

Example

```
WT 0.1
```

This command string sets the wait time to 100 ms.

Also see

None

IN

This command sets the time interval between sample measurements.

Usage

```
IN AA.AA
```

AA.AA	Interval in seconds: 0.01 to 10

Details

For time domain measurements, you can set the time interval between sample measurements. After a sample measurement is made, the next measurement starts after the time interval expires.

Example

```
IN 0.1
```

This command string sets the interval to 100 ms.

Also see

None

NR

This command sets the number of readings that can be made for time domain measurements.

Usage

```
NR AAAA
```

AAAA	Number of measurements to make: ■ 4200A command set: 1 to 4096 ■ 4145 Emulation command set: 1 to 1024

Details

For time domain measurements, you can set the number of sample measurements that can be made. The readings are stored in the buffer.

Example

```
NR 200
```

This command string sets up the 4200A-SCS to make 200 sample measurements.

Also see

None

DM

This command selects the Keysight 4145B display mode.

Usage

```
DMA
```

A	Display modes: ■ Graphics display mode: `1` ■ List display mode: `2`

Details

The 4200A-SCS supports the Keysight 4145B graphics display mode and the Keysight 4145B list display mode command.

The 4200A-SCS does not accept the matrix mode and schmoo mode commands (DM3 and DM4).

Example

```
DM1
```

This command string prepares the 4200A-SCS to receive graphics commands.

Also see

None

LI

This command enables voltage and current functions to be measured when the 4200A-SCS is in list display mode.

Usage

```
LI 'AAAAAA'
LI 'AAAAAA', 'AAAAAA'
LI 'AAAAAA', 'AAAAAA', 'AAAAAA'
LI 'AAAAAA', 'AAAAAA', 'AAAAAA', 'AAAAAA'
LI 'AAAAAA', 'AAAAAA', 'AAAAAA', 'AAAAAA', 'AAAAAA'
LI 'AAAAAA', 'AAAAAA', 'AAAAAA', 'AAAAAA', 'AAAAAA', 'AAAAAA'
```

AAAAAA	A name assigned for CH, VS, or VM; see channel definition page (DE)

Details

When the 4200A-SCS is in the list display mode (DM2 asserted), the LI command enables functions to be measured in a test sequence. To enable a function, include the SMU channel name (as assigned by the CH command), voltage source name (as assigned by the VS command), or the voltmeter name (as assigned by the VM command) in the command string. The DE page is used to assign names to voltage and current functions.

Only functions that are specified (enabled) are measured. Data sheet columns for disabled functions are not shown.

Example

```
L1 'V1', 'I1', 'VS1'
```

Assume the following names have been assigned using the DE page for SMU channel 1 (CH1):

- Voltage is named V1
- Current is named I1
- Voltage source is named VS1

The command string enables the above functions for a test sequence.

Also see

Channel definition page (DE) (on page 10-13)
DM (on page 10-32)

XN

This command configures the X-axis of the graph to plot an electrical parameter.

Usage

```
XN 'AAAAAA', B, ±CCCC.CCC, ±DDDD.DDD
```

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
B	X-axis scale type ■ Linear scale: 1 ■ Log scale: 2
CCCC.CCC	X-axis minimum value ■ Volts: ±9999 ■ Amps: ±999
DDDD.DDD	X-axis maximum value ■ Volts: ±9999 ■ Amps: ±999

Example

```
XN 'V1', 1, -5, 5
```

This command string:

- 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.

Also see

Channel definition page (DE) (on page 10-13)

XT

This command configures the X-axis of the graph to plot time domain values in seconds.

Usage

```
XT AAAA.AA, BBBB.BB
```

AAAA.AA	X-axis minimum time value (in seconds): 0.01 to 9999
BBBB.BB	X-axis maximum time value (in seconds): 0.01 to 9999

Example

```
XT 0, 10
```

This command string:

- 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 s.

Also see

None

YA

This command configures the Y1-axis of the graph.

Usage

```
YA 'AAAAAA', B, ±CCCC.CCC, ±DDDD.DDD
```

AAAAAA	The SMU channel name for the Y1-axis, up to 6 characters; must be one of the SMU channel names that you specify on the channel definition (DE) page
B	Y1-axis scale type: ■ Linear scale: 1 ■ Log scale: 2 ■ Log scale absolute value: 3
CCCC.CCC	Y1-axis minimum value: ■ Volts: ±9999 ■ Amps: ±999
DDDD.DDD	Y1-axis maximum value: ■ Volts: ±9999 ■ Amps: ±999

Example

```
YA 'I1', 1, -5E-9, 5E-9
```

This command string:

- 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.

Also see

Channel definition page (DE) (on page 10-13)

YB

This command configures the Y2-axis of the graph.

Usage

```
YB 'AAAAAA', B, ±CCCC.CCC, ±DDDD.DDD
```

AAAAAA	The SMU channel name for the Y2-axis, up to 6 characters; must be one of the SMU channel names that you specify on the channel definition (DE) page
B	Y2 axis scale type: ■ Linear scale: 1 ■ Log scale: 2 ■ Log scale absolute value: 3
CCCC.CCC	Y2-axis minimum value ■ Volts: ±9999 ■ Amps: ±999
DDDD.DDD	Y2-axis maximum value ■ Volts: ±9999 ■ Amps: ±999

Example

```
YB 'I2', 2, 100E-9, 1E-3
```

The command string:

- 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 100 nA and 1 mA.

Also see

Channel definition page (DE) (on page 10-13)

Measurement control page (MD)

To send the command string to control measurements, select the measurement control page by sending the command:

```
MD
```

ME

This command controls measurements.

Usage

```
MEA
```

A	Measurement action: ■ SingleTrigger test, store readings in cleared buffer: 1 ■ RepeatTrigger test, store readings in cleared buffer: 2 ■ AppendTrigger test, append readings to buffer: 3 ■ StopAbort test: 4

Details

The ME1 or ME2 command triggers the start of the test and makes 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.

Example

```
ME1
```

This command string triggers the start of the test and stores the readings in the cleared buffer.

Also see

None

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

DO

This command requests readings.

Usage

```
DO 'AAAAAA'
```

AAAAAA	User-specified name of the channel that made the measurement, up to 6 characters

Details

After making measurements, use this command string to request the readings. After the 4200A-SCS is addressed to talk, the readings are sent to the computer.

NOTE

To access the timestamp data that was acquired along with voltage or current measurements or both, use the command:

```
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 is sent to the computer in the format:

X±*N.NNNN* E±*NN*, *X*±*N.NNNN* E±*NN*, ... *X*±*N.NNNN* E±*NN*

Where *X* is the status of the data (where *X* = *N* for a normal reading), followed by the reading mantissa and exponent. You can set the delimiter to CR/LF or Comma, and enable or disable EOI.

Example

```
DO 'Volt'
```

This command string requests the reading string for a SMU channel that is named `Volt`.

Also see

None

SV

This command saves a program file or data file.

Usage

```
SV 'A BBBBBB'
SV 'A BBBBBB CCCCCCCC'
```

A	File type: ■ Program file: `P` ■ Data/Program file: `D`
BBBBB	Name of file (up to 6 characters)
CCCCCCCC	Comment (up to 8 characters)

Details

You must separate the file name and the comment from each other by a space.

When saving a program file, the present instrument settings are stored in a file at the directory path:

```
C:\s4200\sys\KXCI
```

You specify the name for the file.

NOTE

The `get` command string is used to acquire the saved file.

Example

```
SV 'P Setup1'
```

This command string saves the command sequence as a program file named `Setup1`.

Also see

GT command: Get file (on page 10-40)

GT

This command acquires (loads) the saved data file or program file.

Usage

```
GT 'A BBBBBB'
```

A	File type: ■ Program file: `P` ■ Data/program file: `D`
BBBBB	Name of file (up to 6 characters)

Details

The file type and file name must be separated by a space.

For a program file, this command launches the program. For a data file, it 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 to store data acquired in a test.

Example

```
GT 'P Setup1'
```

This command string gets the program file named `Setup1`.

Also see

SV command: Save file (on page 10-39)

Channel mapping command

You can use the following command with any system mode page. It maps channel *n* to a given `VS`, `SMU`, or `VM` function.

MP

This command maps channel *n* to a given VS, SMU, or VM function.

Usage

```
MP A, BBBC
```

A	The channel to be mapped: a value between 1 and the number of channels in the system (9 maximum)
BBB	SMU, VS, or VM
C	The number of the SMU, VS, or VM

Details

If *BBB* and *C* are not included in the command, the function defaults to SMU<A>, where <A> is the number of the channel to be mapped.

Example

```
MP 3, VM5
```

This command string maps channel 3 to VM5.

Also see

None

Fixed source ranging command

You can use the following command with any system mode page.

SR

This command sets a fixed source range on channel *n*.

Usage

```
SR A, B
```

A	The channel to be controlled; a value between 1 and the number of channels in the system (9 maximum)
B	Range: ■ Auto: 0 ■ Best fixed range (determined by maximum sweep parameters: 2 ■ Fixed range: > 0 to 1.0

Details

The default setting is autorange for backward 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

```
SR 1, 2
```

This command string selects best fixed range on channel 1.

Also see

None

User mode commands (US)

You can use the user mode (US) command strings for the following operations:

- SMU setup
- VS1...VS9 setup
- Triggering

To send these command strings to the 4200A-SCS, select the user mode by sending the command:

```
US
```

DV and DI

These commands set up each SMU channel.

Usage

AAB, CC, ±DDD.DDDD, ±EEE.EEEE

AA	Source mode: ■ Voltage source: `DV` ■ Current source: `DI`
B	SMU channel: 1 to 9
CC	Voltage source range: ■ Autorange: `0` ■ 20 V range: `1` ■ 200 V range: `2` ■ 200 V range: `3` ■ 200 mV range: `4` (only with a preamplifier) ■ 2 V range: `5` (only with a preamplifier) Current source range: ■ Autorange: `0` ■ 1 nA range (only with a preamplifier): `1` ■ 10 nA range (only with a preamplifier): `2` ■ 100 nA range: `3` ■ 1 µA range: `4` ■ 10 µA range: `5` ■ 100 µA range: `6` ■ 1 mA range: `7` ■ 10 mA range: `8` ■ 100 mA range: `9` ■ 1 A range (only with a 4210-SMU): `10` ■ 1 pA range (only with a preamplifier): `11` ■ 10 pA range (only with a preamplifier): `12` ■ 100 pA range (only with a preamplifier): `13`
DDD.DDDD	Output value: ■ Volts: -210.00 to +210.00 ■ Amps, 4200-SMU: -0.1050 to +0.1050 ■ Amps, 4210-SMU: -1.0500 to +1.0500
EEE.EEEE	Compliance value: ■ Volts: -210.00 to 210.00 ■ Amps, 4200-SMU: -0.1050 to +0.1050 ■ Amps, 4210-SMU: -1.0500 to +1.0500

Details

For every channel that is configured as a SMU, you must select the source mode (voltage or current) and source output range, and set the output and compliance values.

With the voltage source mode (`DV`) specified in the command string, the output value is in volts. For the current source mode (`DI`), the output value is in amps.

With the voltage source mode (`DV`) specified in the command string, you are setting the current compliance. For the current source mode (`DI`), you are setting the voltage compliance. When sourcing voltage, note that if you specify a compliance current that is below the minimum-allowable value (1 pA with a preamplifier installed and 100 nA without a preamplifier), KXCI sets it to the minimum allowable value.

Example

```
DV1, 1, 10, 10E-3
```

This command string configures SMU1 to source 10 V on the 20 V source range and sets current compliance to 10 mA.

Also see

None

DS

This command specifies the channel number and the voltage output value for each voltage source.

Usage

`DS`*`A`*`, ±`*`BBB.BBBB`*

A	*n*, for voltage source VS*n*; range 1 to 9
BBB.BBBB	Output voltage value: –200.00 to +200.00

Details

KXCI allows up to nine source-measure units to function solely as voltage sources. You can use any channel for any voltage-source function between `VS1` and `VS9`. For example, in a system containing four SMUs, you can use SMU2 as `VS5`.

The assigned *n* value for a voltage source (`VS`*n*) or voltmeter (`VM`*n*) depends on how instruments are mapped in KCon.

Example

```
DS1, 20
```

This command string sets `VS1` to output 20 V.

Also see

KCon (on page 7-1)

TV and TI

These commands trigger a measurement.

Usage

AABB

AA	Type of measurement: ■ Voltage: `TV` ■ Current: `TI`
BB	Measure channel for voltage measurements: ■ SMU1: `1` ■ SMU2: `2` ■ SMU3: `3` ■ SMU4: `4` ■ VM1: `5` ■ VM2: `6` ■ SMU5: `7` ■ SMU6: `8` ■ SMU7: `9` ■ SMU8: `10` ■ VM3: `11` ■ VM4: `12` ■ VM5: `13` ■ VM6: `14` ■ VM7: `15` ■ VM8: `16`
BB	Measure channel for current measurements: ■ SMU1: `1` ■ SMU2: `2` ■ SMU3: `3` ■ SMU4: `4` ■ SMU5: `5` ■ SMU6: `6` ■ SMU7: `7` ■ SMU8: `8`

Details

Use the trigger command string to start the testing process. In the command string, you specify the type of measurement (voltage or current), and the SMU or voltmeter (VM1...VM*n*) that makes the measurement.

A SMU that is used to measure current is always specified in the trigger command string by its mapped function number (1...9). The mapped function number normally corresponds to the 4200A-SCS channel number — the physical SMU number. You can map non-corresponding SMU function numbers, but it is not recommended. However, because you can use a SMU to measure voltage either directly (as mapped SMU1...SMU9) or as a mapped voltmeter (VM1...VM9), 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.

After sending the command string to trigger a measurement and addressing the 4200A-SCS to talk, the output data string is sent to the computer in the following format:

X Y Z ±N.NNNN E±NN

Where:

- *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)

When channels are mapped to different functions (VM or VS), KXCI tries to trigger measurements on the specified channels. However, if the mapped function for a channel does not match the requested measurement, KXCI reports an error. For example, if the mapped function for a channel is VS2, but the requested measurement is TI2, KXCI reports an error, because a VS cannot measure current.

Also see

None

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
- Autocalibration
- Exit on compliance
- Switch between 4200A and 4145 modes

IT

This command sets the integration time.

Usage

```
ITA
IT4, X, Y, Z
```

A	Integration time: ■ Short (0.1 PLC): 1 ■ Medium (1.0 PLC): 2 ■ Long (10 PLC): 3 ■ Custom (4200A command set only): 4
X	Delay factor for custom setting: 0.0 to 100
Y	Filter factor for custom setting: 0.0 to 100
Z	A/D converter integration time in number of PLCs for custom setting: 0.01 to 10.0

Details

The integration time is the time to convert a measurement. In general, 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.

KXCI provides three preconfigured integration time settings that are equivalent to the Fast, Normal, and Quiet settings available in Clarius. Short is equivalent to the Clarius Fast setting, Medium to Normal, and Long to Quiet. Integration time is based on power line cycles (PLC). Assuming 60 Hz line power, the integration time for a 1 PLC setting is 16.67 ms (1/60).

The custom setting 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. For additional information on the custom settings, see the Clarius timing window information.

Example 1

```
IT2
```

This command string sets the integration time to 1.0 PLC.

Example 2

```
IT4,2.5,0.6,1.3
```

This 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.

Also see

Speed (on page 6-109)
Test settings (on page 6-106)

ID

This command places the ID of the instrument in a particular buffer.

Usage

```
ID
```

Details

The instrument ID depends on whether you are in 4200A mode or whether you are in the 4145 mode. Refer to the table below.

Hardware comparisons

Instrument type	Keithley Instruments 4200A-SCS	Keysight 4145B
Source measure units (SMUs)	2 to 9	4 (fixed)
Voltage monitor (VM)	You can configure any SMU to function as a VM. Up to 9 VMs are possible.	2 (fixed)
Voltage source (VS)	You can configure any SMU to function as a VS. Up to 9 VSs are possible.	2 (fixed)

Also see

None

*IDN?

This command (identification query) generates an identification query and reads identification information.

Usage

```
*IDN?
```

Details

The identification code includes the manufacturer, model number, serial number, and firmware revision of the instrument. The string is formatted as follows:

```
KEITHLEY INSTRUMENTS,MODEL nnnn,xxxxxxx,yyyyy
```

Where:

- *nnnn* = the model number
- *xxxxxxx* = the serial number
- *yyyyy* = the firmware revision level

Also see

None

DR

This command enables or disables service request for Data Ready.

Usage

```
DRA
```

A	Set service request for Data Ready: ■ Disable service request for data ready: `0` ■ Enable service request for data ready: `1`

Details

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.

Also see

None

BC

This command clears all readings from the buffer.

Usage

```
BC
```

Details

The GPIB data buffer can hold up to 4096 readings.

This command string clears all readings from the buffer. It also clears bit B0 (Data Ready) of the status byte.

Also see

None

RS

This command sets the measurement resolution for all channels.

Usage

```
RS A
```

A	Resolution, in number of digits: ■ 4200A command set: 3 to 7 ■ 4145 Emulation command set: 3 to 5

Example

```
RS 7
```

This command string provides full SMU resolution when the 4200A command set is selected.

Also see

None

*RST

This command resets the instrument settings to default settings.

Usage

```
*RST
```

Example

Returns the instrument to default settings, cancels all pending commands, and cancels the response to any previously received `*OPC` and `*OPC?` commands.

Also see

Reset default values

RI

This command instructs a SMU to go a specified range immediately without waiting until the initiation of a test.

Usage

```
RI channel,range,compliance
```

channel	SMU number (1 to 9)
range	Range, permitted values: ■ 4200-SMU without a preamplifier: 100e-9 A to 100e-3 A ■ 4200-SMU with a preamplifier: 1e-12 A to 100e-3 A
compliance	Compliance, permitted values: 10% to 100% of range

Example

```
RI 1, 1E-3, 1E-3
```

This command string instructs SMU1 to go to the 1 mA range and set compliance to 1 mA.

Also see

None

RG

This command sets the lowest current range to be used when measuring.

Usage

```
RG A,B
```

A	SMU number (1 to 9)
B	The lowest autoranged range: ■ 4200-SMU without a preamplifier: –100e-9 to 100e-3 A ■ 4200-SMU with a preamplifier: –1e-12 to 100e-3 A ■ 4210-SMU without a preamplifier: –100e-9 to 1 A ■ 4210-SMU with a preamplifier: –1e-12 to 1 A

Details

The default autoranged ranges are 100 nA without a preamplifier and 1 nA with a preamplifier.

Example

```
RG 2, 10E-12
```

This command string sets the lowest range of SMU2 with a preamplifier to 10 pA.

Also see

None

AC

This command autocalibrates a SMU channel.

Usage

```
AC A
```

A	SMU number (1 to 9)

Details

When this command is sent, the selected SMU channel is calibrated automatically. The busy bit in the status register is set so that you can detect when autocalibration is finished. The 4200A-SCS will not respond to any commands while autocalibration is executing.

Example

```
AC 1
```

This command string performs autocalibration on SMU channel number 1.

Also see

None

EC

This command sets the condition to exit the test if compliance is reached.

Usage

```
EC A
```

A	Action on compliance: ■ Off (do not exit if compliance is reached): `0` ■ On (exit if compliance is reached): `1`

Example

```
EC 1
```

This command enables exit on compliance.

Also see

None

EM

This command switches between 4145 Emulation and 4200A command sets.

Usage

```
EM A,B
```

A	Mode: ■ 4145 Emulation: `0` ■ 4200A: `1`
B	Which sessions: ■ This session only: `0` ■ This and all subsequent sessions (writes to KCon file): `1`

Example

```
EM 0,1
```

This selects the 4145 Emulation command set permanently.

Also see

None

RD

This command requests real_time readings.

Usage

```
RD 'AAAAAA', N
```

AAAAAA	User-specified name of the channel that made the measurement, up to 6 characters.
N	Index of the data point to retrieve back. Valid range from 1 to. TotalNumberOfReading`1`

Details

While making measurements, use this command string to request desired real time reading.

NOTE

To access the timestamp data that was acquired along with voltage or current measurements or both, use the command:

```
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 is sent to the computer in the format: ±N.NNNN E±NN

Output data equal '0' indicates this data point is not measured yet.

```
RD 'Volt', 1E
```

This command string requests the first reading for a SMU channel that is named `Volt`.

Also see

None

4200A command set only commands

The `OPT?` command string returns a result string that indicates the 4200A-SCS slot configuration for KXCI.

*OPT?

This command returns a result string that contains the 4200A-SCS slot configuration for KXCI.

Usage

```
*OPT?
```

Details

When the 4200A-SCS receives this command, it returns the following configuration string:

```
xxx,xxx,xxx,xxx,xxx,xxx,xxx,xxx
```

This string contains eight sets of characters, each represented by *xxx*. Each character set represents the configuration of a slot in the 4200A-SCS. If the corresponding slot contains a channel, *xxx* is one of the following: `SMU`*n*, `HPSMU`*n*, `SMUPA`*n*, `HPSMUPA`*n*, `VS`*n*, or `VM`*n*, where *n* is the channel number (1 to 9) and:

- `SMU` = Medium-power SMU without a preamplifier
- `HPSMU` = High-power SMU without a preamplifier
- `SMUPA` = Medium-power SMU with a preamplifier
- `HPSMUPA` = High-power SMU with a preamplifier
- `VS` = Voltage source only (20 V is presently supported).
- `VM` = Voltage meter

If the corresponding slot does not contain a channel, then *xxx* = "".

Also see

None

GPIB error messages

KXCI error messages and numbers are shown in the following table.

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."

Using KXCI

To start GPIB operation, start KXCI. The 4200A-SCS is ready to accept GPIB commands immediately after you start KXCI.

For command information, refer to GPIB command set (on page 10-8), GPIB command reference (on page 10-12), Ethernet command reference (on page 10-60), and Pulse generator commands (on page 10-71).

Logging commands, errors, and test results

When you send GPIB commands, KXCI logs the commands, error messages, and test results as described in the following topics.

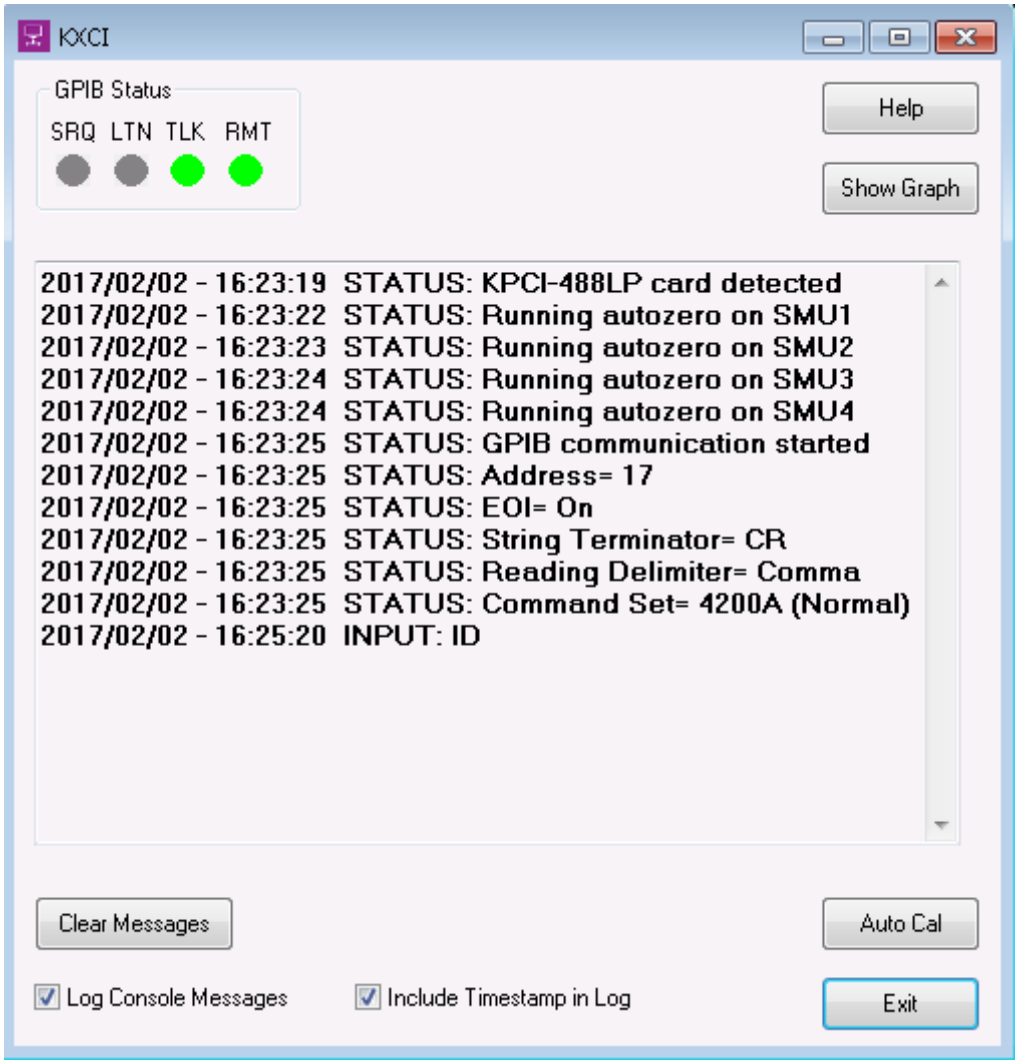
To stop GPIB operation, close KXCI by selecting **Exit**.

Logging commands

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, as shown in the figure below.

Figure 561: Command and message display area



  - If Console Logging Enabled is selected, KXCI also logs each command into the KXCI log file (`C:\s4200\sys\KXCI\KXCILogfile.txt`).
  - If Include Timestamp in Log is selected, KXCI logs a timestamp for each command.

Logging errors

The command and message display area displays error messages as they occur.

Logging test results

The command and message display area also displays the numerical test results for both the 4200A and 4145 Emulation commands sets (refer to Command Set (on page 7-23)).

You can turn off the graph display to better display a long sequence of test results. Select **Hide Graph** in the upper right to hide the graph.

NOTE

The test results will be 0.0000 if the interlock is disconnected.

If the sent commands include the graphics commands, the graph display area graphs the test data. Refer to Graphing the test results (on page 10-59) section.

Understanding the log file

If the Log Console Messages check box is selected (lower left of the user interface), KXCI logs all GPIB commands to a file named `KXCILogfile.txt`. You can open the text file after KXCI is closed. Note that whenever you open KXCI, the log file clears.

The log file is in the directory:

```
C:\s4200\sys\KXCI\
```

You can use any text editor to open the file.

Note that SMUs may return a line such as:

```
DATA:NAI 22.329E-09
```

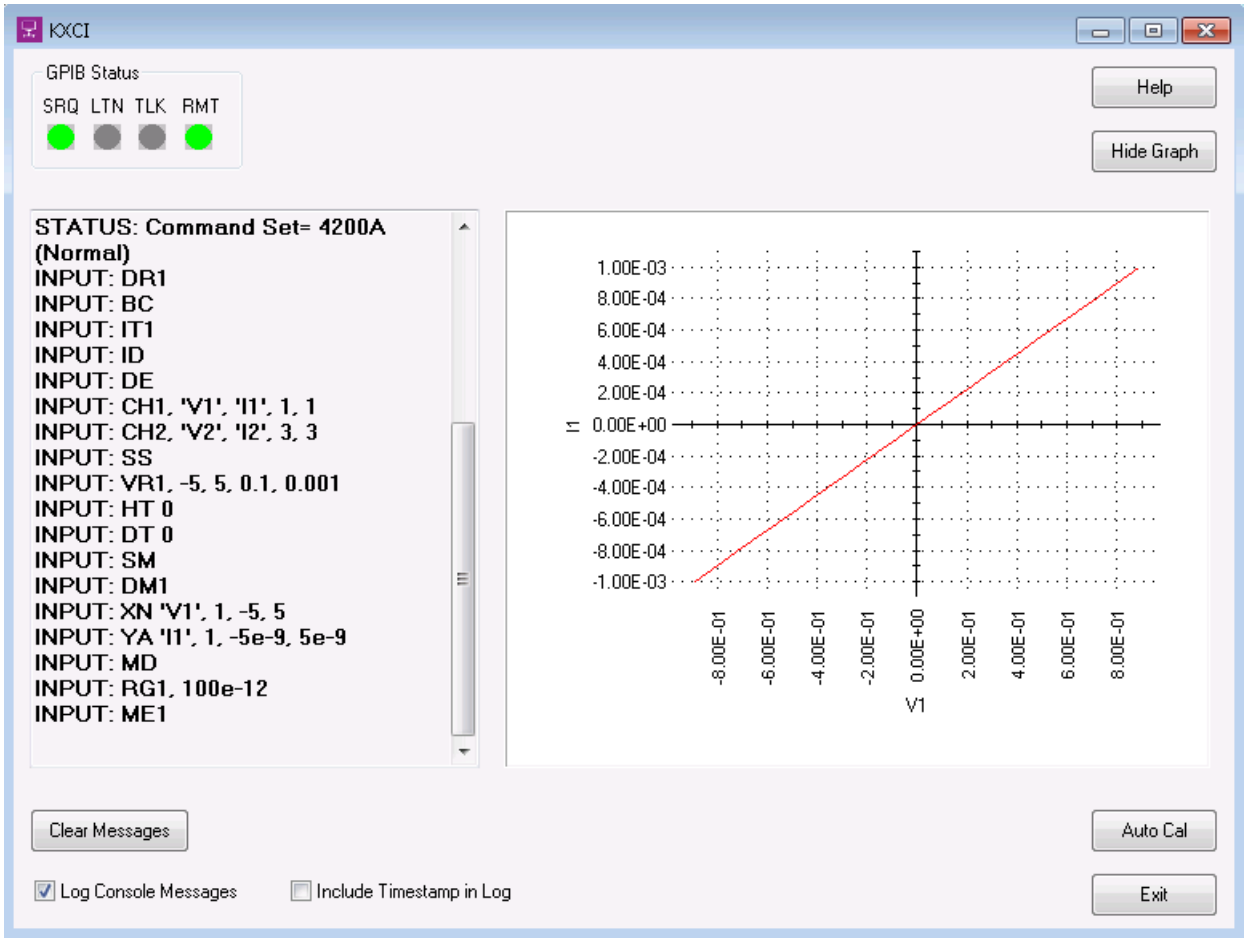
The three characters after `DATA:` represent:

- The operating condition: `N` for normal or `C` if the SMU is in compliance.
- The instrument number, converted to a letter: `A` for SMU1, `B` for SMU2 and so on to letter `I`.
- The measurement function: `V` for voltage or `I` for current.

Graphing the test results

If you have sent the 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 graphics commands in the following figure.

Figure 562: KXCI data graph



NOTE

KXCI plots all Y1 axis curves in red and all Y2 axis curves in blue. To hide the graph, select **Hide Graph**.

Ethernet communications

Ethernet command reference

The Ethernet command set includes all the GPIB commands documented in the GPIB command reference (on page 10-12), plus the `SP` command. The `SP` command allows you to acquire the GPIB spoll byte when ethernet communications is selected.

The `SP` command is not available for GPIB communications.

KXCI ethernet client driver

A KXCI client driver (32- or 64-bit) controls KXCI through the ethernet. You can copy this driver to your controlling computer. The DLL is standalone. It does not depend on any other DLLs, so it can be easily moved or copied.

The driver DLLs are named `KXCIclient.dll` (32-bit) and `KXCIclient64.dll` (64-bit) and are at the command path:

```
C:\s4200\sys\bin
```

The `KXCIclient.lib` (32-bit) and `KXCIclient64.lib` (64-bit) files are at the command path:

```
C:\s4200\sys\lib
```

For convenience, a C header file (`KXCIclient.h`) is included and has the above prototypes. It is at the 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 string format nnn.nn.nn.nn (for example, 129.22.35.17).
- *PortNum*: IP Port assigned in KXCI tab of KCon.
- *err*: Socket error code returned by WSAGetLastError().

```
int SendKXCICommand_C(char *cmdstr, char *ReturnString, int *err);
```

- *cmdstr*: KXCI command string, for example, "DE;CH1;CH2".
- *ReturnString*: Data returned by command, if any. If input command results in data to be returned, it is placed here.
- *err*: Socket error code returned by WSAGetLastError().

```
int GetKXCISpollByte_C(unsigned short *spbyte, int *err);
```

- *spbyte*: KXCI spoll byte (same as GPIB byte).
- *err*: Socket error code returned by WSAGetLastError().

```
void CloseKXCIConnection_C(void);
```

SMU default settings

You can return SMUs to power-on default settings by transmitting the DCL (device clear) or SDC (selected device clear) general bus command to the 4200A-SCS. The power-on default settings for the User Mode are listed in the table below.

The DCL command returns all instruments (including the SMUs) connected to the bus to their default settings. The 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 4200A 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.

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 Keysight Model 4145B, as shown in the channel definition and source setup tables below. This means that each SMU is active and part of the test, whether or not the test used all the SMUs in the 4200A-SCS chassis.

This may be undesirable, as many tests use a different number of SMUs or SMUs in a different state from the Keysight 4145 default. To avoid the complication of building a KXCI test starting with the default setup, 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
```

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.					

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	2 V

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, use the `DO` command to get one or more triggered readings. After sending the `DO` command string and addressing the 4200A-SCS to talk, the output data string is sent to the computer in the following format:

```
X±N.NNNNE±NN,X±N.NNNNE±NN,...X±N.NNNNE±NN
```

Data status

The hierarchy for data status is `L`, `V`, `X`, `C`, `T`, `N`, where *X* is:

- `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)

±*N.NNNN* E±*NN*

NOTE

Scientific notation for the reading exponent:
E+00 = 0
E-03 = 10^{-3} (m)
E-06 = 10^{-6} (µ)
E-09 = 10^{-9} (n)
E-12 = 10^{-12} (p)

Data format for user mode readings

For user mode operation, use the `TI` or `TV` command string to trigger and make a reading. After sending the `TI` or `TV` command string and addressing the 4200A-SCS to talk, the output data string is sent to the computer in the following format:

```
XYZ ±N.NNNN E±NN
```

Data status

The hierarchy for data status is `L`, `V`, `X`, `C`, `T`, `N`, where *X* is:

- `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
	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)

±*N.NNNN* E±*NN*

NOTE

Scientific notation for the reading exponent:
E+00 = 0
E-03 = 10^{-3} (m)
E-06 = 10^{-6} (µ)
E-09 = 10^{-9} (n)
E-12 = 10^{-12} (p)

Status byte and serial polling

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.

You can use serial polling to read the status byte.

The following sections describe the 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 the following table.

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 Weight	—	64	—	16	—	—	2	1

The four used bits of the status byte register are described in the following topics.

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 clear (0) bit B0:

- Clears (0) when the data transfer starts.
- Clears (0) when the BC (buffer clear) command is sent to the 4200A-SCS.
- Clears (0) when the 4200A-SCS is serial polled.

Bit B1, Syntax Error

This bit sets (1) when an invalid command string is sent to the 4200A-SCS. Any of the following actions will clear (0) bit B1:

- When the 4200A-SCS is serial polled.
- When a device clear command (DCL or SDC command) is sent to the 4200A-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 4200A-SCS is serial polled.
- Clears (0) when a device clear command (DCL or SDC command) is sent to the 4200A-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 4200A-SCS. Serial polling reads the status byte. Generally, the controller uses the serial polling sequence to determine which of several instruments has requested service with the SRQ line. However, the status byte of the 4200A-SCS may be read to determine when an operating condition has occurred.

If you try to get 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 4200A-SCS to detect an SRQ, you can monitor the service request line. When an SRQ occurs, the SRQ line goes true. You can use the following C language programming routine 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 system control using an external computer communicating through GPIB. For these programs, configure KXCI 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 4200A-SCS to perform a VAR1 and VAR2 sweep. It assumes that channels 1 through 4 of the KXCI are configured for the SMU function.

```
MAXLEN = 2048;
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 4200A-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.

```
MAXLEN = 2048;
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);

// Get 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.

```
MAXLEN = 2048;
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);
```

KXCI pulse generator commands

The KXCI commands to control pulse generators are documented in the following topics.

NOTE

The following documentation includes the corresponding LPT library functions for each KXCI command string. Refer to LPT functions for PGUs and PMUs (on page 14-98) for details on pulsing functions.

PD

This command sets the output impedance (pulse load).

Usage

```
PD A, BBB.BBBB
```

A	Pulse card channel number: 1 to 8; the largest value is the number of channels in the system
BBB.BBBB	Pulse load, in ohms: 1.0 to 10e6; default 50.0

Corresponding LPT library function

pulse_load (on page 14-164)

Details

You can use this command to set the DUT impedance of the load (DUT) independently for each channel from 1 Ω to 10 MΩ.

Example

```
PD 1, 1e3
```

This command string sets channel 1 of pulse card 1 for an output impedance of 1 kΩ.

Also see

None

PE

This command is used to start the execution on a configured pulse test.

Usage

```
PE
```

Corresponding LPT library functions

pulse_exec (on page 14-107)

Details

You can use this command to start test execution.

Also see

None

PG

This command sets the trigger mode and initiates the start of pulse output or arms the pulse card.

Usage

```
PG A, B, C
```

A	Pulse card channel: 1 to 8; the largest value is the number of channels in the system
B	Trigger mode: ■ Burst mode: `0` ■ Continuous: `1` (default) ■ Trigger burst: `2`
C	Burst or trigger burst only: Pulse count in number of pulses: 1 to 2^{32}-1; set to 1 for continuous (default `1`)

Corresponding LPT library functions

pulse_trig (on page 14-172)
pulse_burst_count (on page 14-156)

Details

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.

Example

```
PG 1, 2, 50
```

This command string sets pulse card 1 for the Trigger Burst trigger mode and a burst count of 50.

Also see

None

PH

This command stops all pulse output from the selected pulse card.

Usage

```
PH A
```

A	Pulse card channel: 1 to 8; the largest value is the number of channels in the system

Corresponding LPT library functions

pulse_halt (on page 14-162)

Example

```
PH 3
```

This command string stops pulse output from pulse card 2.

Also see

None

PN

This command configures the pulse of the pulse channel.

Usage

PN A,BBB.BBBB,CCC.CCCC,DDD.DDDD,EEE.EEEE,FFF.FFFF,GGG.GGGG,HHH.HHHH,<JJJ.JJJJ>

A	Pulse card channel number: 1 to 8; the largest value is the number of channels in the system.
B	Voltage Low Level
C	Voltage High Level
D	Pulse Trigger Count
E	Pulse Period
F	Pulse Width
G	Pulse Rise Time
H	Pulse Fall Time
J	Pulse Delay Time (optional). Default is 0.

Corresponding LPT library function

pulse_train (on page 14-138)
pulse_source_timing (on page 14-130)

Details

You can use this command to set up a pulse train to output on the pulse channel.

Example

```
PN 1, 0.0, 5.0, 10, 100e-6, 20e-6, 200e-9, 200e-9
```

This command string sets Channel 1 of the pulse card for a pulse train with a 5 V amplitude, 100 µs period, 20 µs width, and 200 ns rise and fall times. This pulse will output 10 times.

Also see

None

PO

This command sets the output state (on or off) and mode (normal or complement) independently for each channel.

Usage

```
PO A, B, C
```

A	Pulse card channel: 1 to 8; the largest value is the number of channels in the system
B	Output state: ■ Off: 0 (default) ■ On: 1
C	Output mode: ■ Normal: 0 (default) ■ Complement: 1

Corresponding LPT library functions

pulse_output (on page 14-165)
pulse_output_mode (on page 14-166)

Example

```
PO 1, 1, 1
```

This command string turns on channel 1 of pulse card 1 and selects complement pulse output mode.

Also see

None

PS

This command resets both channels of the selected pulse card to the default settings.

Usage

```
PS A
```

A	Pulse card channel: 1 to 8; the largest value is the number of channels in the system

Corresponding LPT library functions

pulse_init (on page 14-163)

Example

```
PS 3
```

This command string resets pulse card 2.

Also see

None

PT

This command sets the pulse period, pulse width, pulse rise time, and pulse fall time.

Usage

```
PT A, BBB.BBBB, CCC.CCCC, DDD.DDDD, EEE.EEEE
```

A	Pulse card channel: 1 to 8; the largest value is the number of channels in the system
BBB.BBBB	Pulse period in seconds (floating point number): ■ 5 V range: 10e-9 to 1 ■ 20 V range: 500e-9 to 1 ■ Default: 1e-6
CCC.CCCC	Pulse width in seconds (floating point number): ■ High speed: 10e-9 to (Period – 10e-9) ■ High voltage: 250e-9 to (Period – 10e-9) ■ Default: 500e-9
DDD.DDDD	Rise time in seconds (floating point number): ■ High speed: 10e-9 to 33e-3 ■ High voltage: 50e-9 to 33e-3 ■ Default: 100e-9
EEE.EEEE	Fall time in seconds (floating point number): ■ High speed: 10e-9 to 33e-3 ■ High voltage: 50e-9 to 33e-3 ■ Default: 100e-9

Corresponding LPT library functions

pulse_period (on page 14-167)
pulse_width (on page 14-182)
pulse_rise (on page 14-169)
pulse_fall (on page 14-161)

Details

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 setting.

The minimum transition time for pulse source only (no measurement) on the rise time high voltage for the 40 V range is 50 ns for the 4225-PMU and 4220-PGU.

Example

```
PT 1, 100e-6, 20e-6, 200e-9, 200e-9
```

This command string sets the pulse period of pulse card 1 to 100 µs. It also sets channel 1 for a pulse width of 20 µs and rise/fall times for 200 ns.

Also see

None

PV

This command sets pulse high, pulse low, range, and current limit independently for each channel of the selected pulse card.

Usage

```
PV A, BBB.BBBB, CCC.CCCC, DDD.DDDD, EEE.EEEE
```

A	Pulse card channel: 1 to 8; the largest value is the number of channels in the system
BBB.BBBB	Pulse high in volts: ■ 5 V range (high speed): -5.0 to +5.0 ■ 20 V range (high voltage): -20.0 to +20.0 ■ Default: 0.0
CCC.CCCC	Pulse low in volts ■ 5 V range (high speed): -5.0 to +5.0 ■ 20 V range (high voltage): -20.0 to +20.0 ■ Default: 0.0
DDD.DDDD	Pulse range in volts ■ 5 V range: 5 ■ 20 V range: 20 ■ Default: 5
EEE.EEEE	Current limit in amps (range and load dependent) ■ 5 V range (high speed): -0.2 to +0.2 ■ 20 V range (high voltage): -0.8 to +0.8 ■ Default: 0.105 (5 V range)

Corresponding LPT library functions

pulse_vhigh (on page 14-178)
pulse_vlow (on page 14-180)
pulse_range (on page 14-168)
pulse_current_limit (on page 14-157)

Example

```
PV 1, 2, -2, 5, 200e-3
```

For pulse card 1 channel 1, this command string sets pulse high to +2 V, pulse low to -2 V, pulse range to 5 V and current limit to 200 mA.

Also see

None

TO

This command sets the trigger output parameters for pulse delay and trigger polarity.

Usage

```
TO A, BBB.BBBB, C
```

A	Pulse card channel: 1 to 8; the largest value is the number of channels in the system
BBB.BBBB	Pulse delay in seconds: ■ 0.0 to (Period – 10e-9) ■ Default: 0.0
C	Trigger polarity ■ Negative: `0` ■ Positive: `1` ■ Default: `1` (rising edge)

Corresponding LPT library functions

pulse_delay (on page 14-159)
pulse_trig_polarity (on page 14-175)
pulse_output (on page 14-165)

Details

This command sets 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.

Example

```
TO 1, 2e-6, 1
```

This command string sets the pulse delay for channel 1 of pulse card 1 to 2 µs and selects positive trigger polarity.

Also see

None

TS

This command sets the trigger source that is used to trigger the pulse card to start its output.

Usage

```
TS A, B
```

A	Pulse card channel: 1 to 8; the largest value is the number of channels in the system
B	Trigger source: ■ Software triggered: `0` ■ External — Initial Trigger Only — Rising Edge: `1` ■ External — Initial Trigger Only — Falling Edge: `2` ■ External — Trigger Every Pulse — Rising Edge: `3` ■ External — Trigger Every Pulse — Falling Edge: `4`

Corresponding LPT library functions

pulse_trig_source (on page 14-176)

Details

With the Software trigger source selected, the `PG` command selects the trigger mode (Continuous, Burst, or Trig Burst), and initiates the start of pulse output.

NOTE

Since trigger source is a card level setting and NOT a channel setting, using channel 1 or 2 sets the card to the specified source card 1. Similarly, channel 3 or 4 sets the source for card 2.

With an external trigger source selected, the `PG` command 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 a "Trigger per pulse" mode.

Example

```
TS 1, 1
```

This command string sets the trigger source to External – Initial Trigger Only – Rising Edge.

Also see

PG (on page 10-73)

PC

This command combines four channels from two 4220-PGU or 4225-PMU instruments.

Usage

```
PC A,B,C,D
```

Corresponding LPT library function

NONE

Details

This command is to be used with the 4205-PCU to combine four channels from two 4220-PGU or 4225-PMU instruments..

Example

```
PC 1, 2, 3, 4
```

This command string combines the first two pulser cards' channels in the system.

Also see

None

VF

This command sets the state of the floating relay for the given pulse channel.

Usage

```
VF A,B
A Pulse card channel number: 1 to 8; the largest value is the number of channels in the
    system.
B Floating relay state: 0: disabled; 1: enabled.
```

Corresponding LPT library function

pulse_float (on page 14-117)

Details

You can use this command to allow the pulse channel to float.

Example

```
VF 1, 1,
```

This command string sets channel 1 of pulse card to enable the floating relay.

Also see

None

KXCI CVU commands

The user and system command modes are used to control the CVU using KXCI. This mimics the paradigm used to control the SMUs in I/V tests.

User mode is similar to LPT library (when running locally). All the commands are processed sequentially, and you have 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 Clarius tests. When you send the run command, the system buffers and executes all CVU test parameters automatically. The CVU card executes full sweeps. The output sequencing, settling, and sweeping are all handled internally. You return the entire buffer of test data.

Many of the CVU commands are valid in both user and system mode. 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 firmware acts on the settings and executes the test sequence.

NOTE

All the commands act on the selected card, which is set with the `:CVU:CHANNEL` command.

User mode

:CVU:MEASZ?

This command triggers and returns single Z-measurement using current CVU settings.

Usage

```
:CVU:MEASZ?
```

Details

When the command is complete, the single reading is available over GPIB or ethernet.

Also see

None

:CVU:CABLE:COMP:OPEN

This command performs open compensation and collects the open compensation cable data for the CVU.

Usage

```
:CVU:CABLE:COMP:OPEN
```

Also see

Connection compensation (on page 4-14)

:CVU:CABLE:COMP:SHORT

This command performs short compensation and collects the short compensation cable data for the CVU.

Usage

```
:CVU:CABLE:COMP:SHORT
```

Also see

Connection compensation (on page 4-14)

:CVU:CABLE:COMP:LOAD

This command performs load compensation and collects the load compensation cable data for the CVU.

Usage

```
:CVU:CABLE:COMP:LOAD
```

Also see

Connection compensation (on page 4-14)

:CVU:CABLE:COMP:MEASCUSTOM

This command performs custom cable length compensation and collects the compensation cable data for the CVU.

Usage

```
:CVU:CABLE:COMP:MEASCUSTOM
```

Also see

Connection compensation (on page 4-14)

System mode

:CVU:SOAK:DCV

This command sets the presoak DC voltage for CVU card for all sweeps for the selected CVU card.

Usage

```
:CVU:SOAK:DCV voltage
```

voltage	Voltage to bias before test sequence begins; voltages range from -30 V to +30 V

Details

All sweeps include frequency sweep, DC voltage sweep, AC voltage sweep, and so on.

Also see

None

:CVU:BIAS:DCV:SAMPLE

This command configures the CVU to bias a DC voltage and sample *n* Z-measurements for the CVU card.

Usage

```
:CVU:BIAS:DCV:SAMPLE biasv, samples
```

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

Details

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.

Also see

None

:CVU:STEP:FREQ

This command configures the CVU to step frequency and sample Z-measurements for the selected CVU card.

Usage

```
:CVU:STEP:FREQ fstart, fstop
```

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

Details

Only the start frequency and stop frequency are set here for the step order. The other parameters are set by their respective commands.

NOTE

Values are coerced to one of the 37 discrete frequencies for the Model 4210-CVU.

Also see

None

:CVU:SWEEP:FREQ

This command configures the CVU to sweep frequency and sample Z-measurements for the selected CVU card.

Usage

```
:CVU:SWEEP:FREQ fstart, fstop
:CVU:SWEEP:FREQ fstart, fstop, order
```

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	Determines whether the test will bias DC voltage and sweep frequency or step voltage and sweep frequency: ■ 1: The voltage bias the voltage set by the :CVU:DCV command; the default ■ 2: The voltage will step based on the *vstart*, *vstop*, *vstep* parameters set by the :CVU:SWEEP:DCV command.

Details

Only the start frequency, stop frequency, and sweep order are set here. The other parameters are set by their respective commands.

NOTE

Values are coerced to one of the 37 discrete frequencies for the Model 4210-CVU.

Also see

None

:CVU:SWEEP:DCV

This command configures the CVU to sweep DC voltage and sample Z-measurements for the selected CVU card.

Usage

```
:CVU:SWEEP:DCV dcvstart, dcvstop, dcvstep
```

dcvstart	Start voltage
dcvstop	Stop voltage
dcvstep	Voltage step size

Details

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.

Also see

None

:CVU:SWEEP:ACV

This command configures the CVU to sweep AC voltage and sample Z-measurements for the selected CVU card.

Usage

```
:CVU:SWEEP:ACV acvstart, acvstop, acvstep
```

acvstart	Start AC voltage
acvstop	Stop AC voltage
acvstep	AC voltage step size

Details

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.

Also see

None

:CVU:DELAY:STEP

This command sets the hold time for the CVU test on the selected card for the `:CVU:SWEEP:FREQ` and `:CVU:SWEEP:DCV` operations.

Usage

`:CVU:DELAY:STEP` *stepd*

stepd	Hold time to apply the pre-soak value (set by the `:CVU:SOAK:DCV` command); valid values range from 0 to 999 s

Also see

:CVU:SWEEP:DCV (on page 10-86)
:CVU:SWEEP:FREQ (on page 10-85)

:CVU:DELAY:SWEEP

This command sets the sweep delay for the CVU test on the selected card.

Usage

`:CVU:DELAY:SWEEP` *sweepd*

sweepd	Delay (0 to 999 s)

Details

This command is used when in sweeping mode (which is all sweep types except `:CVU:BIAS:DCV:SAMPLE`).

Also see

None

:CVU:SAMPLE:HOLDT

This command sets the hold time for a sampling mode test on the selected card.

Usage

`:CVU:SAMPLE:HOLDT` *holdt*

holdt	Hold time (0 to 999 s)

Details

This is only used when executing the `:CVU:BIAS:DCV:SAMPLE` command.

Also see

:CVU:BIAS:DCV:SAMPLE (on page 10-84)

:CVU:SAMPLE:INTERVAL

This command sets the delay between samples for the selected card.

Usage

```
:CVU:SAMPLE:INTERVAL interval
```

interval	Delay (0 to 999 s)

Details

This is only used when executing the :CVU:BIAS:DCV:SAMPLE command.

Also see

:CVU:BIAS:DCV:SAMPLE (on page 10-84)

:CVU:SWEEP:LISTDCV

This command configures the selected CVU card to sweep arbitrary DC voltage points and sample Z-measurements.

Usage

```
:CVU:SWEEP:LISTDCV pt1, pt2, pt3 ... ptn
```

pt1, *pt2*, *pt3* ... *ptn*	Sweep points (1 to 4096)

Details

Only starting voltage sweep points are set with this command.

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.

Also see

None

:CVU:TEST:RUN

This command starts a CVU test on the specified card.

Usage

```
:CVU:TEST:RUN
```

Details

Use the serial poll byte to determine when not busy or data ready.

Also see

None

:CVU:TEST:ABORT

This command stops a running KXCI CVU test.

Usage

```
:CVU:TEST:ABORT
```

Details

This command terminates all ongoing processes and returns the 4200A-SCS to the idle state. Data that results 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 10-103)).

This command is not valid in User Mode.

Also see

None

:CVU:STANDBY

This command configures the selected CVU card to disable DC bias at the end of a test or leave it active.

Usage

```
:CVU:STANDBY state
```

state	The state of the CVU card at the end of the test: ■ Disable the DCV output: `1` ■ Leave the state active: `0`

Also see

None

:CVU:DATA:Z?

This command queries the Z measurement of the CVU when a test is complete.

Usage

```
:CVU:DATA:Z?
```

Details

Z measurements are returned as semi-colon delimiter pairs. Each reading is then delimited with commas.

Also see

None

:CVU:DATA:VOLT?

This command queries the voltage measurement of the CVU once a test is complete.

Usage

```
:CVU:DATA:VOLT?
```

Also see

None

:CVU:DATA:FREQ?

This command queries the frequency measurement of the CVU once a test is complete.

Usage

```
:CVU:DATA:FREQ?
```

Also see

None

:CVU:DATA:STATUS?

This command queries the status of the CVU when a test is complete.

Usage

```
:CVU:DATA:STATUS?
```

Also see

None

:CVU:DATA:TSTAMP?

This command queries the timestamp of the measurements of the CVU once a test is complete.

Usage

```
:CVU:DATA:TSTAMP?
```

Also see

None

Modeless commands

:CVU:CHANNEL chan

This command selects the CVU card on which subsequent CVU commands will act.

Usage

```
:CVU:CHANNEL chan
```

chan	CVU card on which KXCI CVU commands act (default 1)

Details

The majority of systems only have one CVU card. You can use this command to configure multiple cards.

Also see

None

:CVU:MODEL

This command sets the measurement model for the selected CVU card.

Usage

```
:CVU:MODEL model
```

model	The model: ■ 0: Z, theta ■ 1: R + jx ■ 2: Cp, Gp ■ 3: Cs, Rs ■ 4: Cp, D ■ 5: Cs, D

Also see

None

:CVU:MODE

This command sets User or System mode.

Usage

```
:CVU:MODE mode
```

mode	**The mode:** ■ 0: User Mode ■ 1: System Mode

Also see

None

:CVU:RESET

This command sends a soft reset to the specified card.

Usage

```
:CVU:RESET
```

Details

This command places the card in default state.

Parameters are:

- DC voltage: 0 V
- AC voltage: 30 mV
- Frequency: 100 kHz
- Model: R + jx
- Range: 1 mA

Also see

None

:CVU:SPEED

This command sets the measurement speed for the selected CVU card.

Usage

```
:CVU:SPEED speed
:CVU:SPEED 3, delay_factor, filter_factor, aperture
```

speed	Applies one of four defined speed selections: ■ 0: Fast ■ 1: Normal ■ 2: Quiet ■ 3: Custom
delay_factor	Custom only: 0 to 100
filter_factor	Custom only: 0 to 707
aperture	Custom only: 0.006 to 10.002 PLCs

Details

The parameters *delay_factor*, *filter_factor*, and *aperture* are only used if speed is set to custom.

delay_factor

After applying a voltage or current, the instrument waits for a delay time before making a measurement. The delay time allows for source settling. The Delay Factor adjusts the delay time for the selected speed. It is fixed for Fast, Normal, and Quiet speeds. If you select Custom Speed, you can type a factor from 0 to 100.

The applied delay time is a multiple of the default delay time. The Delay Factor specifies this multiple. That is:

Applied delay time = (Default delay time) x (Delay Factor)

For example, if the default delay time is 1 ms and the Delay Factor is 0.7, the actual applied delay time is 0.7 ms (1 ms x 0.7).

The following table summarizes the Delay Factor settings.

Speed Mode	Delay Factor
Fast	0.7
Normal	1.0
Quiet	1.3
Custom	0 to 100

When typing a custom Delay Factor setting, consider the following:

- A Delay Factor of 1 allows for 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.

NOTE

If you set the Filter Factor and Delay Factor to `0`, the internal pre-programmed values are ignored.

In general, cables and matrices increase the settling time. You may need to experiment to find the ideal time. However, for a good quality switch, such as the Keithley Instruments 7174A Ultra-Low Current matrix, you should not need to increase the Delay Factor by more than two times.

filter_factor

To reduce measurement noise, each 4200A-SCS applies filtering, which averages multiple readings to make one measurement. The Filter Factor is fixed for Fast, Normal, and Quiet speeds. If you select the Custom Speed mode, you can type a Filter Factor of 0 to 100.

The default Filter Factor settings are listed in the following table.

Speed Mode	Filter Factor
Fast	0.2
Normal	1
Quiet	3
Custom	0 to 100

When typing 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.

aperture

You can specify the A/D converter integration time that is used to measure a signal. Each measured reading 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 with increased noise. A long integration time results in a relatively low-noise reading at a relatively low speed.

The integration time setting is based on the number of power line cycles (NPLCs). For 60 Hz line power, 1.0 PLC = 16.67 ms (1/60). For 50 Hz line power, 1.0 PLC = 20 ms (1/50).

The applied A/D conversion time also depends on the Filter Factor setting:

- If the Filter Factor is not zero, the CVU 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 Aperture Time.
- If the Filter Factor setting is zero, the CVU applies a fixed A/D converter time that equals the specified A/D Aperture Time.

The Auto A/D Aperture and A/D Aperture Time settings for each Speed Mode are shown in the following table.

Speed Mode	A/D Aperture Time
Fast	Auto
Normal	Auto
Quiet	Auto
Custom	0.01 to 10 PLC

Also see

None

:CVU:ACV

This command sets the AC drive level for specified CVU card.

Usage

```
:CVU:ACV aclevel
```

aclevel	10 mV to 100 mV

Details

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 (which is set with the `:CVU:SWEEP:ACV` command).

Also see

:CVU:SWEEP:ACV (on page 10-86)

:CVU:DCV

This command sets the DC bias voltage for specified CVU card.

Usage

```
:CVU:DCV dclevel
```

dclevel	-30 V to +30 V

Details

In user mode, this takes immediate effect.

In system mode, the value is buffered for use in Frequency and AC Voltage sweeps and DC Bias/Sample.

Also see

None

:CVU:ACZ:RANGE

This command sets the AC measurement range for the specified CVU card.

Usage

```
:CVU:ACZ:RANGE range
```

range	Range: ■ `0`: Auto ■ `1e-6`: 1 µA ■ `30e-6`: 30 µA ■ `1e-3`: 1 mA

Also see

None

:CVU:FREQ

This command sets the frequency for the AC source for the specified CVU card.

Usage

```
:CVU:FREQ freq
```

freq	10 mV to 100 mV; values that fall between supported frequencies are coerced to closest supported frequency

Details

In user mode, this takes effect immediately.

In system mode, the value is buffered for use in Voltage Bias/Sample tests, List and linear Voltage sweeps, and AC Voltage sweeps.

Also see

None

:CVU:LENGTH

This command selects the cable length for the specified CVU card.

Usage

```
:CVU:LENGTH len
```

len	The cable length: ■ `0`: 0 m (no cable compensation) ■ `1.5`: 1.5 m CVU cable ■ `3.0`: 3.0 m CVU cable ■ `4.0`: Custom ■ `5.0`: 1.5 m CVIV cable; 2-wire mode ■ `6.0`: 1.5 m CVU to CVIV cable with 0.75 m CVIV to DUT cable; 4-wire mode ■ `7.0`: Blue 1.5 m CVU to CVIV cable with 0.61 m CVIV to DUT cable; 4-wire mode

Also see

None

:CVU:CORRECT

This command enables or disables open, short, and load correction for the specified CVU card.

Usage

```
:CVU:CORRECT open, short, load
```

open	off: 0 on: 1
short	off: 0 on: 1
load	off: 0 on: 1

Details

Each of the parameters is a Boolean (0 = off, 1 = on).

Also see

None

:CVU:DCV:OFFSET

This command applies an offset value to the DC low terminal.

Usage

```
:CVU:DCV:OFFSET offsetv
```

offsetv	-30 V to +30 V

Details

Voltage offset to apply while sampling Z-measurements.

Also see

None

:CVU:CONFIG:ACVHI

This command allows you to define the source terminal (AC only) for the CVU test.

Usage

```
:CVU:CONFIG:ACVHI source
```

source	The source terminal to be used: ■ HCUR/HPOT (default): 1 ■ LCUR/LPOT: 2

Also see

None

:CVU:CONFIG:DCVHI

This command allows you to define the source terminal (DC only) for the CVU test.

Usage

```
:CVU:CONFIG:DCVHI source
```

source	The source terminal to be used: ■ HCUR/HPOT (default): 1 ■ LCUR/LPOT: 2

Also see

None

Code examples

Example 1

The following code segment sets CVU1 to perform a system mode sweep of DC voltage from 5 V to 10 V in 1 V steps. After the test completes, the Z, DCV, F, timestamps, and status values are queried. The operation mode is then changed to perform a list sweep of DC voltage (2 V, 4 V, 3 V, 5 V, 7 V) and the test is run again with all other test conditions as previous.

```
// Soft Reset card
send(addr, ":CVU:RESET", &status);

// Set CVU to System Mode
send(addr, ":CVU:MODE 1", &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

This example makes 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 a 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 4200A-SCS from a remote interface. Refer to Keithley User Library Tool (on page 8-1) for details on using KULT.

The commands for calling user modules are:

- UL: usrlib (on page 10-103): Mode Switch: Switches KXCI to the usrlib mode.
- EX: execute (on page 10-104): Execute: Executes the specified module in a KULT user library.
- GN: get parameter (on page 10-105): Returns the parameter value (or values) for the specified input or output parameter. The parameter is specified by name.
- GP: get parameter (by number) (on page 10-106): 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 description (on page 10-107): Get Parameter Description: Returns the description of the specified function. The function is specified by User Library and User Module.

UL

This command switches KXCI operation to the `usrlib` mode.

Usage

```
UL
```

Details

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.

Also see

None

AB

This command aborts the execution of a module of the user library.

Usage

```
AB
```

Also see

EX (on page 10-104)

EX

This command executes a user module using specified parameter values.

Usage

```
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)

Details

Separate parameter values by commas. Do not use quotation marks to enclose strings or names.

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 is used.

For an output parameter, leave the space empty (see example below).

NOTE

When used before the `GN` or `GP` commands, you may need to add a delay to allow the execution of the module to finish.

Example

```
EX my_2nd_lib VSweep(0, 5, , 11, , 11)
```

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`

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 a user function can be run from the KXCI console using parameters that perform an 11-point sweep that starts at 0 V and stops at 5 V.

When the KXCI is in the `usrlib` mode, the example command runs the KULT function.

After execution of the module completes, you can query input or output parameters or both to return the values. Use GN: get parameter (by name) (on page 10-105) or GP: get parameter (by number) (on page 10-106) to query parameters.

Also see

AB (on page 10-103)

GN

This command queries input or output parameter values, or both, by name for the last user module run in KXCI.

Usage

```
GN ParameterName
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 **Details**

Details

NumValues is only used for an output parameter that is an array. If *NumValues* is not used, one value is returned.

Arrays are returned as a list of values separated by semicolons.

The value returned for an input parameter is the given value. The values returned for an output parameter is the outcome of the test (for example, measured readings).

NOTE

You can also query a parameter by specifying the corresponding number for the parameter in the KULT module (see GP: get parameter (by number) (on page 10-106)).

Example

```
GN Vforce 11
```

This command queries all 11 test voltages (*Vforce* parameter) for the function that was run in the EX (on page 10-104) example.
The following array of test voltages are 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
```

Also see

None

GP

This command queries input or output parameter values, or both, by number for the last user module run in KXCI.

Usage

```
GP ParameterPosition
GP ParameterPosition NumValues
```

ParameterPosition	The position of the parameter in the KULT module
NumValues	The number of values in an output array to be returned; see **Details**

Details

NumValues is only used for an output parameter that is an array. If *NumValues* is not used, one value is returned.

Arrays are returned as a list of values separated by semicolons.

The value returned for an input parameter is the given value. The values returned for an output parameter is the outcome of the test (for example, measured readings).

NOTE

You can also query a parameter by specifying the name of the parameter in the user module. See GN (on page 10-105).

Example

```
GP 5 11
```

This command queries all test voltages for the *Vforce* parameter for the user module that was run in the EX example. *Vforce* is the fifth parameter in the function.
The following array of test voltages is 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
```

Also see

None

GD

This command queries and returns the description for a KULT module.

Usage

```
GD UserLibrary UserModule
```

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

Details

Example

```
GD my_2nd_lib VSweep
```

This command queries the description of the user module in the example in EX: execute (on page 10-104). The description is displayed in the KXCI console.

Also see

None

SystemUtil User Library

This user library permits KXCI using the `UL` mode (see Calling KULT user libraries remotely (on page 10-103)) to retrieve information about the 4200A-SCS instrument and the system.

NOTE

This user library is not compatible with Clarius UTMs. It only works with the `UL` mode of KXCI.

instrumentinfo

This command retrieves all information on the instrument cards in the 4200A-SCS and the 4200A-CVIV Multi-Switch, if attached.

Usage

```
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 store in the buffer
len	Number of bytes returned by the function

Details

This function returns all of the instrument-level information for the cards in the 4200A-SCS and the 4200A-CVIV, if attached. The results contain the following information for each instrument card in the 4200A-SCS chassis:

- Slot number (for the 4200A-CVIV, `1` is returned)
- Instrument card ID
- Model number
- Serial number
- Hardware version
- Firmware version
- Calibration date
- Calibration due date

All of the information is comma-separated.

Example

Output for a 4200A-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,calda
    te:Dec 19, 2009,caldue:Dec 19, 2010,
slotno:5,name:PMU1,model:KIPMU4225,serialno:1276563,hwver:1.0:544911,fwver:1.5
    0,caldate:Nov 28, 2011,caldue:Nov 27, 2012,
slotno:7,name:CVU1,model:KICVU4210,serialno:Z005712,hwver:3.0:493083,fwver:2.0
    3,caldate:Aug 09, 2010,caldue:Aug 09, 2011
```

Also see

remotemoduleinfo (on page 10-109)

remotemoduleinfo

This command retrieves all information on the preamplifiers and RPMs that are attached to the 4200A-SCS.

Usage

```
int remotemoduleinfo(char *result, int maxlen, int *len);
```

result	String of results for the remote modules
maxlen	The maximum number of bytes that the result can store in the buffer
len	Number of bytes returned by the function

Details

This function returns all the on the preamplifiers and RPMs that are attached to the 4200A-SCS. The results contain the following information for each preamplifier and RPM:

- 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.

Example

Output for a 4200A-SCS with two preamplifiers (attached to SMUs in slot 1 and slot 2) and one RPM attached to channel 1 of the PMU in slot 6:

```
slotno:1,name:PA1,model:KI4200,serialno:0734120,hwver:D7481,fwver:E02.1,caldat
    e:Feb 29, 2016,caldue:Mar 01, 2017,
slotno:2,name:PA2,model:KI4200,serialno:0901553,hwver:D08288,fwver:E03,caldate
    :Feb 28, 2016,caldue:Feb 28, 2017,
slotno:6,name:RPM1-1,model:KIRPM4225,serialno:1314347,hwver:1.4,fwver:2.00,cal
    date:Jan 31, 2017,caldue:Jan 31, 2018
```

Also see

instrumentinfo (on page 10-108)

systeminfo

This command retrieves 4200A-SCS system information.

Usage

```
int systeminfo(char *result, int maxlen, int *len);
```

result	String of results for the system
maxlen	The maximum number of bytes that the result can be stored in the buffer
len	Number of bytes returned by the function

Details

This function returns system level information for the 4200A-SCS. The *result* string contains the following information:

- 4200A-SCS serial number
- System software version
- System platform version
- Clarius+ application version

The results are comma-separated.

Example

Example system info output:

```
serialno:1209478,swver:4200A-852-1.0,platformver:4200A-300-1, clariusver:V1.2
```

Also see

None

Section 11

KPulse (for Keithley Pulse Cards)

In this section:

Keithley Pulse Application

Keithley Pulse Application (KPulse) is a non-programming alternative that you can use to configure and control the installed Keithley pulse cards. You can use it for quick tests that require minimal interaction with other 4200A-SCS test resources.

The KPulse application supports the source-only configuration of 4225-PMU and 4220-PGU 2-channel pulse cards. The 4225-PMU is identified as PMU on the card tab. All other card types are identified as VPU.

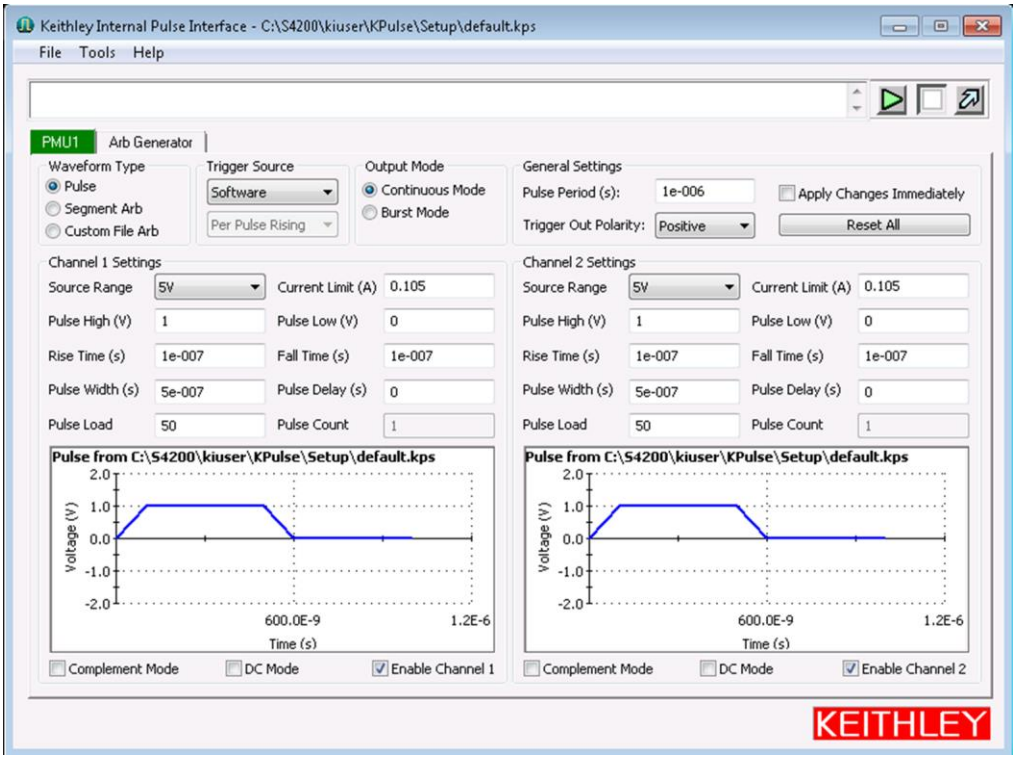
Starting KPulse

To open KPulse, double-click **KPulse** on the desktop. The following example shows one PMU installed in the system.

From the user interface, each pulse generator can be used to configure and control the following waveform types:

- Standard pulse waveforms (on page 11-5) and Segment Arb waveforms (on page 11-7): Pulses are configured and run from the VPU or PMU tabs of KPulse. There is a tab for every Keithley pulse card installed in the 4200A-SCS.
- Custom file arb waveforms (full-arb) (on page 11-10): Pulses are configured and saved as a `.kaf` file using the Arb Generator tab of KPulse. You can then use the VPU or PMU tab to load the saved `.kaf` file into the pulse generator and run it.

Figure 563: KPulse GUI

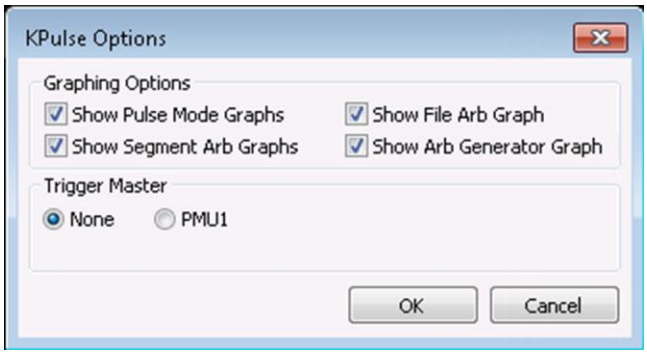


KPulse setup and help

The KPulse menus include:

- **File:** Use this menu to load and save KPulse setups and exit KPulse. By default, setup files are saved at the command path location:
  `C:\s4200\kiuser\KPulse\Setup`.
- **Tools:** From this menu, select **Options** to open the **KPulse Options** dialog box, which includes the following options:
  - **Show Pulse Mode Graphs:** When enabled, shows the Standard Pulse waveform previewers for each pulse card tab.
  - **Show Segment Arb Graphs:** When enabled, shows the Segment Arb pulse waveform previewers for each pulse card tab.
  - **Show File Arb Graph:** When enabled, shows the Custom File Arb pulse waveform previewer for each pulse card tab.
  - **Show Arb Generator Graph:** When enabled, shows the Arb Generator pulse waveform previewer.
  - **Trigger Master:** Select the pulse card that will serve as the trigger master. Select **None** if you are not using a trigger master. See Triggering (on page 11-4) for more information.

Figure 564: KPulse Options dialog box



  - **Help:** Use this menu to access the Learning Center information and to open the About KPulse dialog box.

Triggering

With a Keithley pulse card selected as the trigger master, its Trigger Out can be used to start (trigger) itself or other pulse cards in the system.

NOTE

For the master pulse card, the polarity of the pulse trigger source (`pulse_trig_source`) and pulse trigger polarity (`pulse_trig_polarity`) function must be the same. If you are using a rising-edge trigger source, the pulse trigger polarity must be positive. If you are using a falling-edge trigger source, the pulse trigger polarity must be negative. This requirement applies to all pulse modes, including Standard Pulse, Segment Arb, and Full Arb.

NOTE

When triggering multiple pulse cards in a master/subordinate configuration, changing the trigger output polarity of the master card results in a transition in the trigger output levels that may be interpreted as a trigger pulse by the other cards.

Standard pulse waveforms

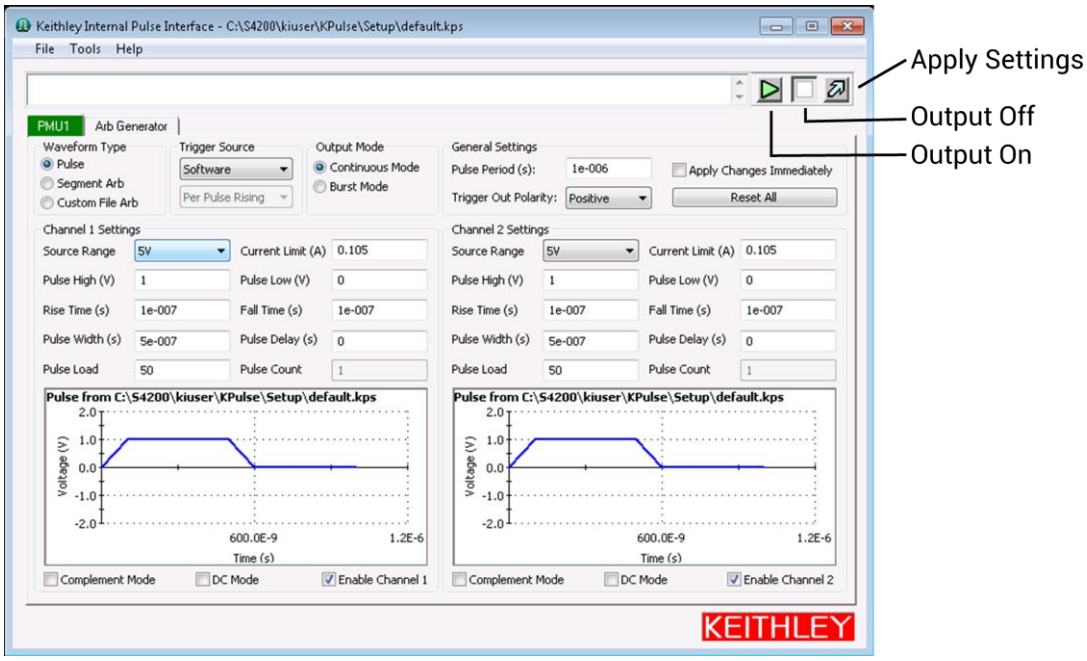
Standard pulse waveforms are configured and controlled from the pulse card tabs in the KPulse window. The following figure explains how to use KPulse for a standard pulse output.

Standard pulse waveform previewers: KPulse provides a preview of configured standard pulse waveforms for each enabled channel. Each waveform previewer shows the high and low levels, and timing for the waveform.

In the next figure, the configuration shown in the waveform previewer for Channel 1 uses the default settings for KPulse (pulse high = 1 V and pulse low = 0 V). Channel 2 uses the same settings, but the complement mode is enabled. Pulse high goes to pulse low level (0 V) and pulse low goes to pulse high level (1 V).

The following instructions describe how to use KPulse for a standard pulse output. Repeat these steps for any installed cards that you want to set to standard pulse.

Figure 565: Standard pulse operation settings



Set waveform, trigger, output, and general settings:

1. Select the pulse card tab (in this example, **PMU1**).
2. For the Waveform Type, select **Pulse**.
3. For the Trigger Source, select **External**.
4. Select the Trigger Source type - **Per Pulse Rising**, **Initial Falling**, **Initial Rising**, or **Per Pulse Falling**.
5. For the Output Mode, select **Continuous Mode** or **Burst Mode**.
6. Under General Settings, set the **Pulse Period** in seconds.
7. Set the Trigger Polarity to **Positive** or **Negative**.
8. Select how to apply the changes by doing one of the following:
   - Select **Apply Changes Immediately** to enable automatic update for pulse output. After the outputs are turned on, the pulse output updates immediately when the settings are changed.
   - Select **Apply Settings** (the button with the light blue arrow) to manually apply settings.

NOTE

Select **Reset All** to return the pulse card to the Standard Pulse waveform type and its default settings. It also updates the previewer.

Configure the Channel 1 and Channel 2 settings:

1. Configure the channel settings as needed. Note that the Pulse Count field is only available if the Output Mode is set to Burst Mode.
2. At the bottom of the window, select **Enable Channel 1** or **Enable Channel 2**. A channel must be enabled to preview its waveform and turn on the output for the channel.
3. If needed, select **DC mode**. This sets the output to fixed DC at the pulse high level. Clear this to return the output to the defined pulse.
4. If needed, select **Complement Mode**. This sets pulse high to the low level and pulse low to the high level.
5. Select the **Output On** button to turn on all enabled channels for all pulse cards installed on the 4200A-SCS. Output Off turns red.
6. To turn off the output, select **Output Off**.

Segment Arb waveforms

Segment Arb® waveforms are configured and controlled from the PGU or PMU tab in KPulse. The below figure explains how to use KPulse for segment Arb output.

NOTE

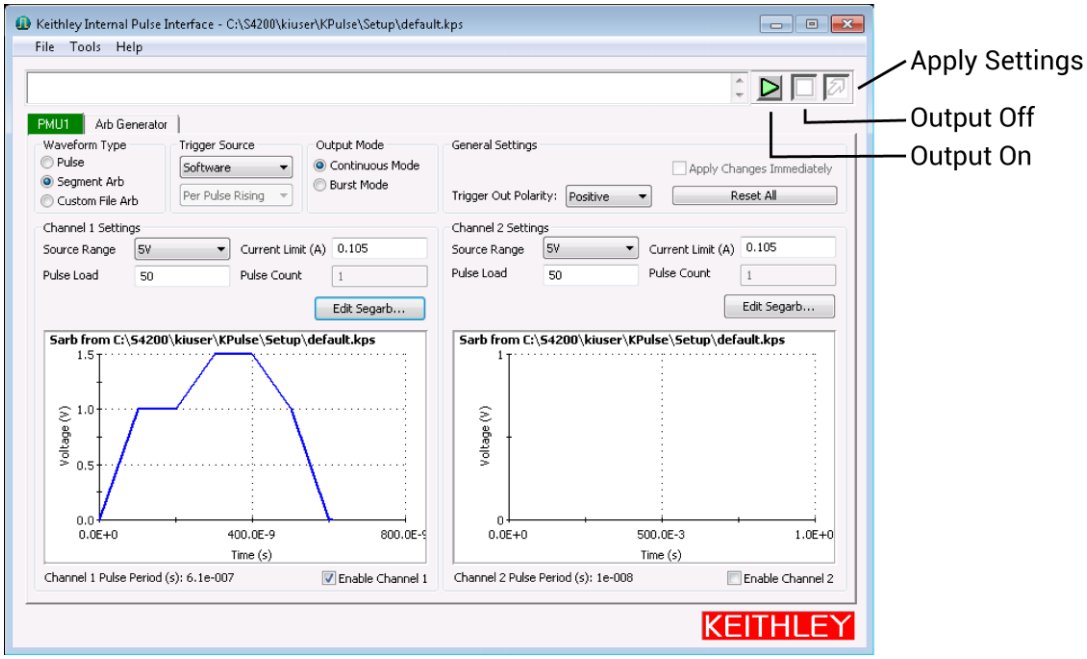
Due to the Segment Arb engine overhead, there is an additional 10 ns interval added to the end of the last segment of a Segment Arb waveform. During this interval, the output voltage and trigger output values remain the same as the final value reached in the last segment.

Start, stop, and time restrictions:

- The start level of the first segment and the stop level of the last segment must be the same. In the following figure, Segment 1 start and Segment 7 stop are both set for 0.0 V.
- The stop level for a segment must be the same as the start level for the next segment. In the following figure, the stop level for Segment 1 is 1.0 V, which is the same as start level for Segment 2 (no discontinuities are allowed).
- Time values are in 10 ns increments, with a minimum of 20 ns.

Segment Arb pulse waveform previewers: KPulse provides a preview of configured Segment Arb waveforms for each enabled channel. Each waveform previewer shows the segment levels and timing for the waveform.

Figure 566: Segment Arb waveform settings



NOTE

The output trigger levels are not shown in the waveform previewers.

The following instructions describe how to use KPulse for a standard pulse output. Repeat these steps for any installed cards that you want to set to standard pulse.

To set waveform, trigger, output, and general settings:

1. Select the pulse card tab (in this example, **PMU1**).
2. For the Waveform Type, select **Segment Arb**.
3. For the Trigger Source, select **External**.
4. Select the Trigger Source type - **Per Pulse Rising**, **Initial Falling**, **Initial Rising**, or **Per Pulse Falling**.
5. For the Output Mode, select **Continuous Mode** or **Burst Mode**.
6. Under General Settings, set the **Pulse Period** in seconds.
7. Set the Trigger Polarity to **Positive** or **Negative**.
8. Select how to apply the changes by doing one of the following:
   - Select **Apply Changes Immediately** to enable automatic update for pulse output. After the outputs are turned on, the pulse output updates immediately when the settings are changed.
   - Select **Apply Settings** (the button with the light blue arrow) to manually apply settings.

NOTE

Select **Reset All** to return the pulse card to the Standard Pulse waveform type and its default settings. It also updates the previewer.

To configure the Channel 1 and Channel 2 settings:

1. Configure the channel settings as needed. Note that the Pulse Count field is only available if the Output Mode is set to Burst Mode.
2. At the bottom of the window, select **Enable Channel 1** or **Enable Channel 2**. A channel must be enabled to preview its waveform and turn on the output for the channel.
3. Select **Edit Segarb**.
4. For each segment, enter the **Start** voltage, **Stop** voltage, **Time** in seconds, the TTL output **Trig** level (`0` = low; `1` = high), and the state of the **SSR** (solid-state relay, `0` = open, `1` = closed).
5. Select **OK**.
6. Select the **Output On** button to turn on all enabled channels for all pulse cards installed on the 4200A-SCS. Output Off turns red.
7. To turn off the output, select **Output Off**.

Exporting Segment Arb waveform files

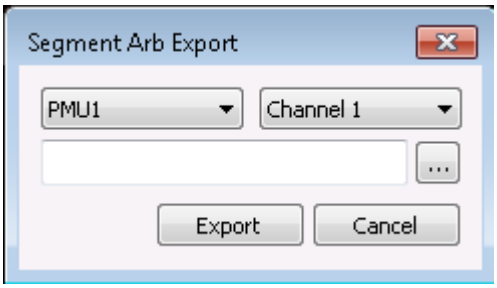
After configuring a Segment Arb® waveform in KPulse, you can save it as a `.ksf` file. The Segment Arb `.ksf` files should be exported into the `SarbFiles` folder at the following location:

```
C:\s4200\kiuser\KPulse\SarbFiles
```

To export a Segment Arb waveform file:

1. Select **Tools**.
2. Select **Export Segment Arb** to open the Segment Arb Export dialog box.

Figure 567: Segment Arb Export dialog box



3. Select the pulse card and channel for the waveform to be exported.
4. Select **...**.
5. Locate the target folder.
6. Type a name for the file. The `.ksf` extension is added automatically.
7. Select **Open**.
8. Select **Export**.

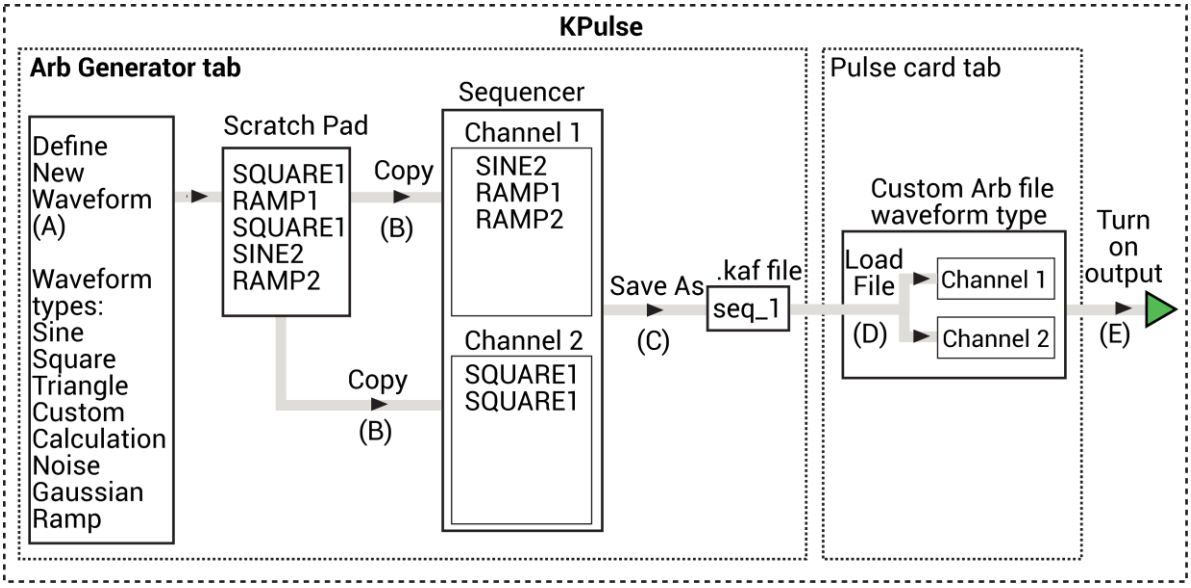
You can import a saved `seg_arb.ksf` waveform file into the pulse card using the `seg_arb_file` function.

For segment stress/measure testing, you can import the `.ksf` file through the Clarius Configure pane. For details about segment stress/measure testing, refer to Segment stress/measure mode (on page 6-210).

Custom file arb waveforms (full arb)

The following figure summarizes the basic processes to create a custom full arb waveform file, to load the file into a pulse card, and to output the pulse waveforms.

Figure 568: Basic process to create and output custom file Arb waveforms



To create custom file arb waveforms (full arb):

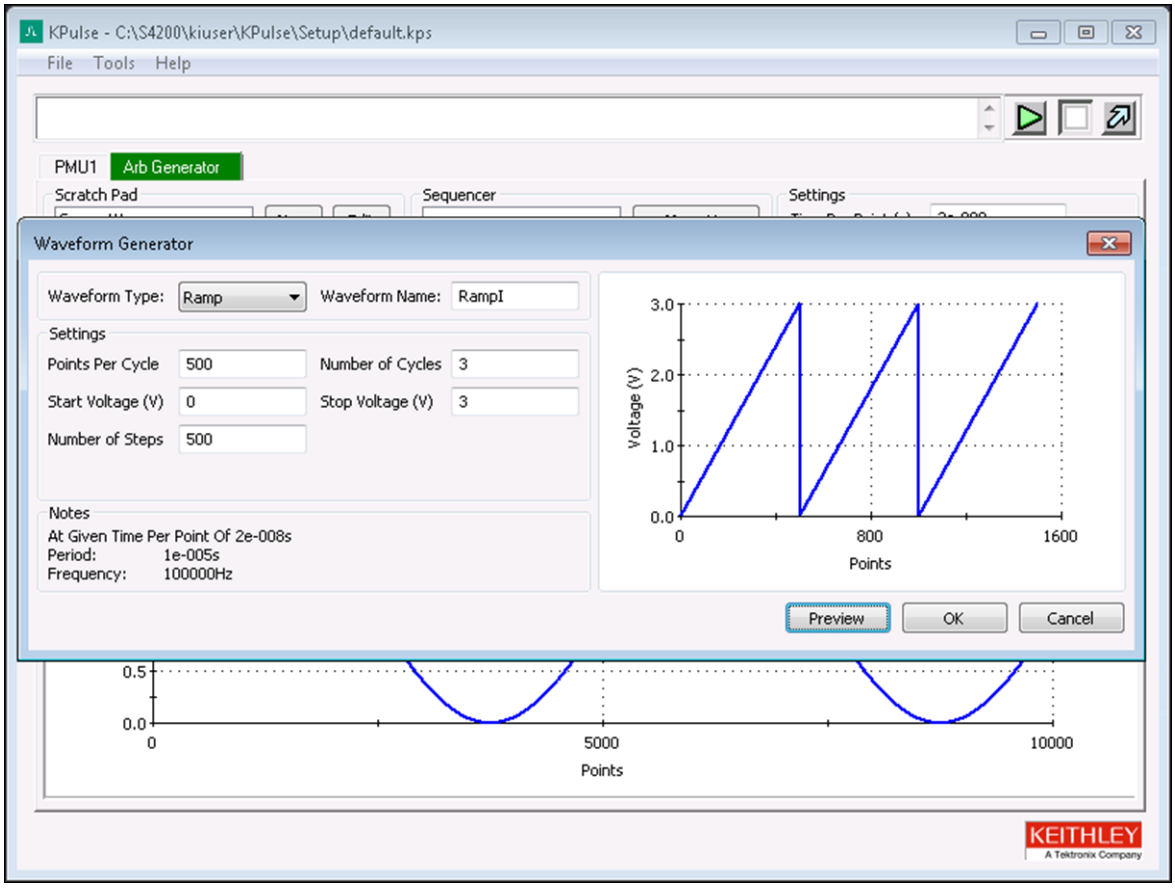
1. Select and configure waveforms.
   - After you select an available waveform type, configure its settings, and name it, the waveform is placed in the Scratch Pad.
   - Waveforms remain in the Scratch Pad until they are deleted by the user.
   - Refer to Waveform types (on page 11-16) for information about the waveform types available for custom file arb. Refer to Custom Arb file operation: Select and configure waveforms (on page 11-12) for details.
2. Copy the waveforms into the Sequencer for Channel 1, Channel 2, or both:
   - The order that two or more waveforms appear in a channel sequencer is the order that the waveforms are output by that channel.
   - Refer to Custom Arb file operation: Copy waveforms into Sequencer (on page 11-13) for details.
3. Save the waveforms in the Sequencer as a `.kaf` file. See Custom Arb file operation: Copy waveforms into Sequencer (on page 11-13).
4. Load the `.kaf` waveform file into a pulse generator (using the appropriate pulse card tab): Refer to Custom Arb file operation: Load waveform and turn on output (on page 11-14) for details.
5. Turn on the output for enabled channels. See Custom Arb file operation: Load waveform and turn on output (on page 11-14).

Custom Arb file operation: Select and configure waveforms

To select and configure waveforms:

1. Select the **Arb Generator** tab.
2. Select **New Waveform** to open the Waveform Generator dialog box, shown in the following figure.

Figure 569: Custom Arb file operation: Select and configure waveforms



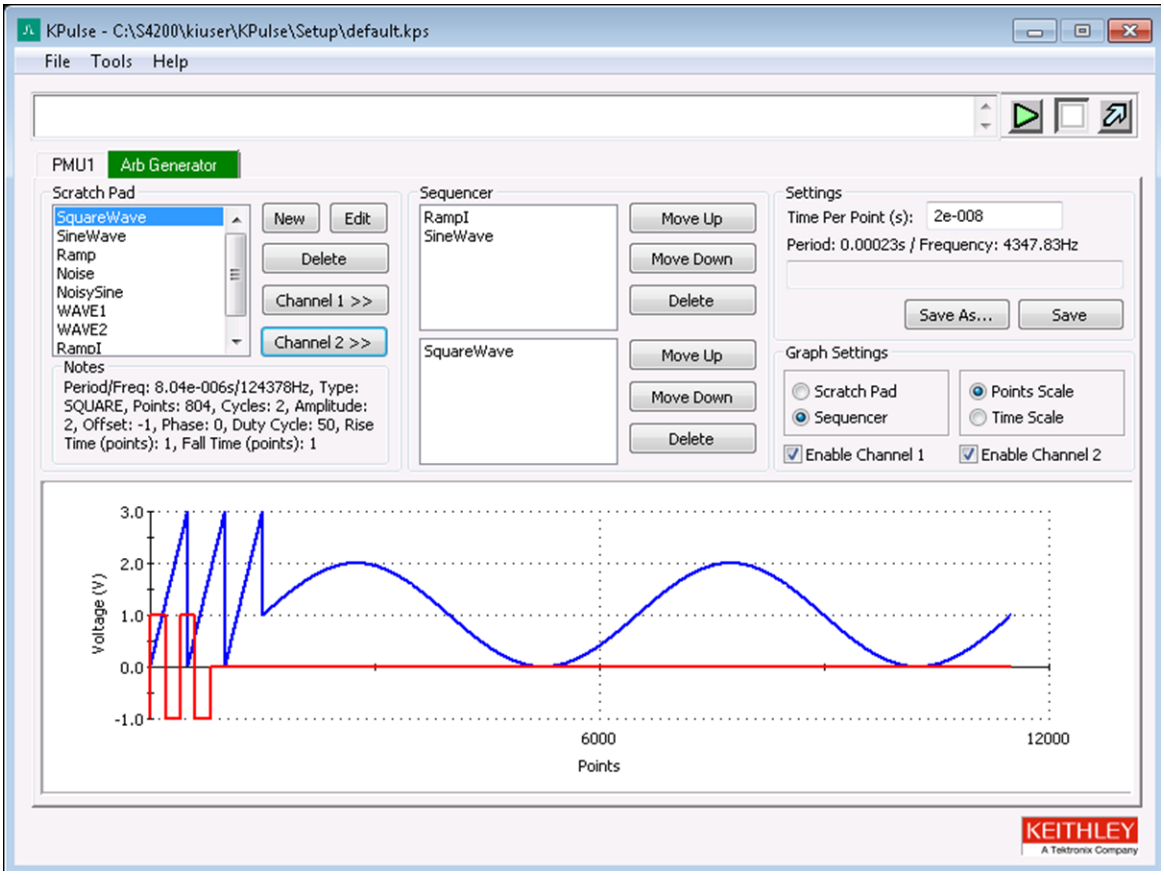
3. Select the **Waveform Type** to be created.
4. Configure the **Settings** for the selected waveform type.
5. Select **Preview** to update the preview of the waveform.
6. In the **Waveform Name** field, type in a name for the waveform. You cannot use a name that is already used in the Scratch Pad.
7. Select **OK** to create the waveform. The new waveform is added to the Scratch Pad.
8. Repeat these steps to create another waveform in the Scratch Pad.

Custom Arb file operation: Copy waveforms into Sequencer

To copy waveforms into the Sequencer:

1. Select the **Arb Generator** tab.
2. Select **Scratch Pad** or **Sequencer**:
   - Scratch Pad previews the waveform that is selected in the Scratch Pad.
   - Sequencer previews enabled waveform sequences. To preview the waveforms in the Sequencer, select **Enable Channel 1** or **Enable Channel 2**.
3. Select the scale for the graph, **Points Scale** or **Time Scale**.
4. In the Scratch Pad, select a waveform to be copied into the Sequencer.
5. Select **Channel 1** to copy the selected waveform onto the Sequencer for Channel 1 or select **Channel 2** to copy the waveform into the Sequencer for Channel 2. You can copy the same waveform to both channels.

Figure 570: Custom File Arb operation: Copy waveforms into Sequencer



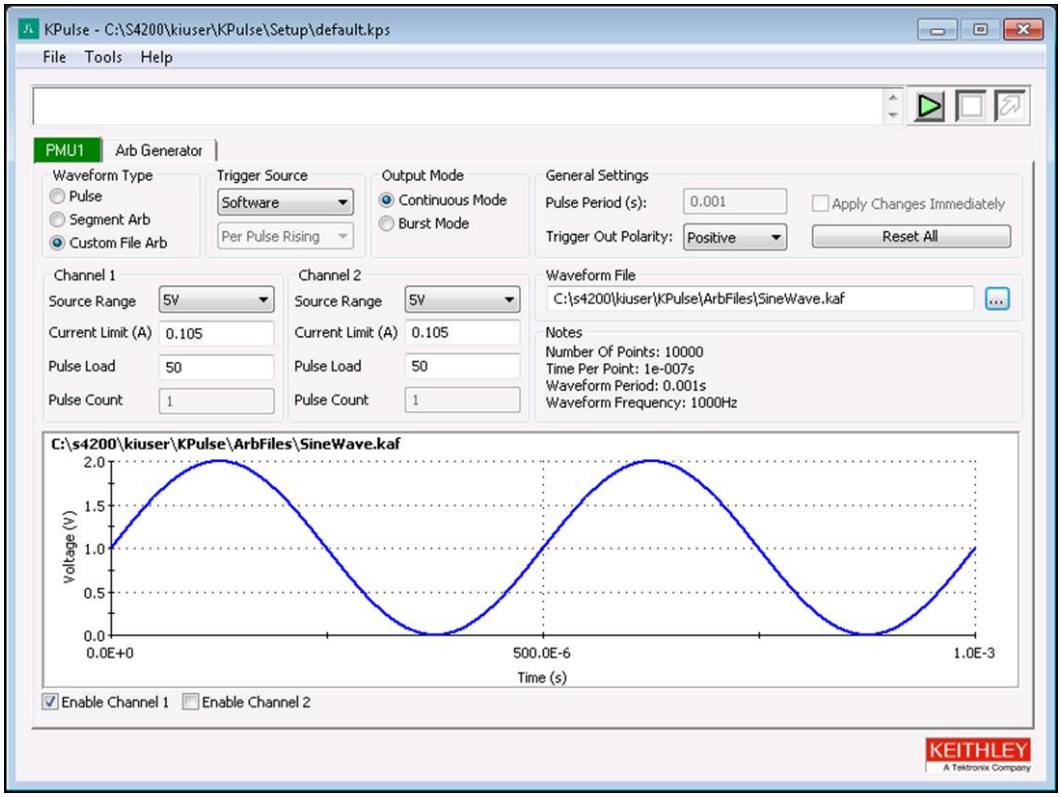
6. To change a waveform sequence, select a waveform in the Sequencer and select **Move Up** or **Move Down**.
7. **Delete** buttons - After selecting a waveform in the Scratch Pad or Sequencer, select the appropriate Delete button to remove it. Note that deleting a waveform from the Scratch Pad also removes it from the Sequencer.
8. Set the **Time per Point** (in seconds). This is the time interval between each point in the waveforms.
9. Save the waveforms as a Keithley Arb File (.kaf). By default, `.kaf` files are saved in the folder: `C:\s4200\kiuser\KPuIse\ArbFiles`
10. Use **Save As** to name the file and save it.
11. After any subsequent changes, select **Save** to overwrite the `.kaf` file.

Custom Arb file operation: Load waveform and turn on output

To load the waveform and turn on the output:

1. Select a pulse card tab.
2. Select **Custom File Arb**.
3. Browse for a **Waveform File** and then select and load the `.kaf` file.

Figure 571: Custom File Arbe operation: Load waveform and turn on output



4. **Enable Channel 1** and/or **Enable Channel 2**. The loaded `.kaf` file consists of a waveform for one or both of the channels. If the .kaf file was saved with one or both channels enabled, the `.kat` tile loads into this tab with the same channels enabled. A channel must be enabled in order to preview and output its waveform. The waveform for Channel 1 is blue and the waveform for Channel 2 is red.
5. Configure triggers for both channels of the pulse card:
   - **Trigger Source** - **Software**, **External**, or **Internal Bus**. With **External** enabled, select the trigger source: **Initial Falling**, **Initial Rising**, **Per Pulse Falling**, or **Per Pulse Rising**.
   - **Output Mode** - Select the output trigger mode: **Continuous Mode** or **Burst Mode**.
6. Configure the **Channel 1 Settings** and/or **Channel 2 Settings**. The Pulse Count field is active if the Burst Mode is the selected trigger mode.

NOTE

To configure other installed pulse cards for Custom File Arb, repeat steps 1 through 6.

7. Turn on all enabled channels - Select the green triangle to turn on enabled channels for all installed pulse cards in the 4200A-SCS. With the output on, the square box turns red. Select the red box to turn off the outputs of all pulse cards.

Waveform types

KPulse provides the following fundamental waveform types to use as the building blocks for custom file arb:

- Sine waveform (on page 11-17)
- Square waveform (on page 11-18)
- Triangle waveform (on page 11-19)
- Custom waveform (on page 11-20)
- Calculation waveform (on page 11-22)
- Noise waveform (on page 11-23)
- Gaussian waveform (on page 11-24)
- Ramp waveform (on page 11-25)
- Sequences waveform (on page 11-26)

As explained in Custom Arb file operation: Select and configure waveforms (on page 11-12), a waveform is created using the Waveform Generator. After selecting and configuring one of the above waveform types, the waveform is placed into the Scratch Pad.

NOTE

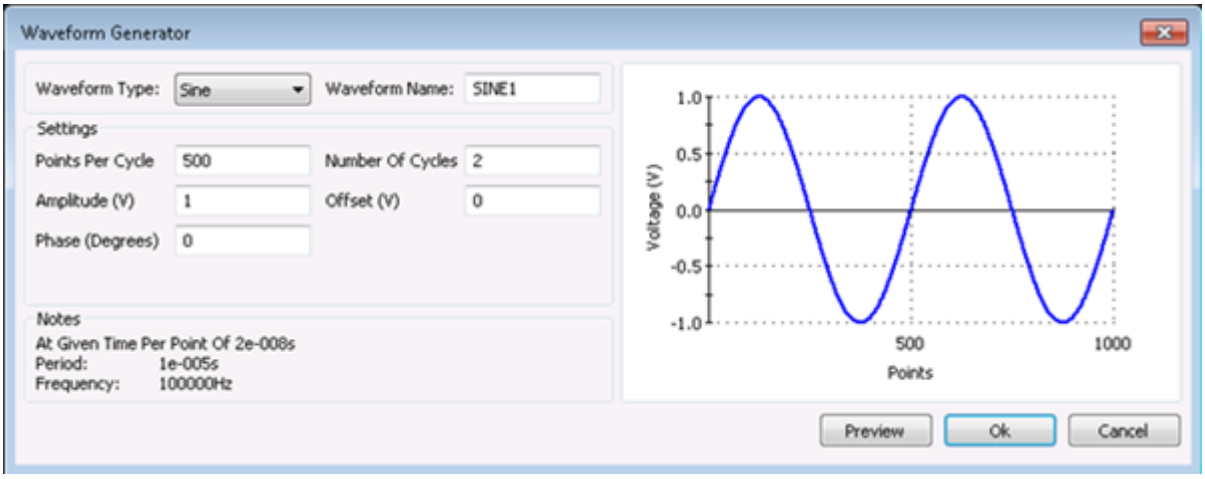
The period for the waveforms is determined by the **Time Per Point** setting in the Arb Generator tab (refer to Custom Arb file operation: Copy waveforms into Sequencer (on page 11-13)).

Sine waveform

An example of a sine waveform is shown in the next figure. The waveform for this example is named SINE1, but can be any name that is not already used in the Scratch Pad.

After changing one or more settings, select **Preview** to display the waveform. Select **OK** to place the waveform in the Scratch Pad.

Figure 572: Sine waveform

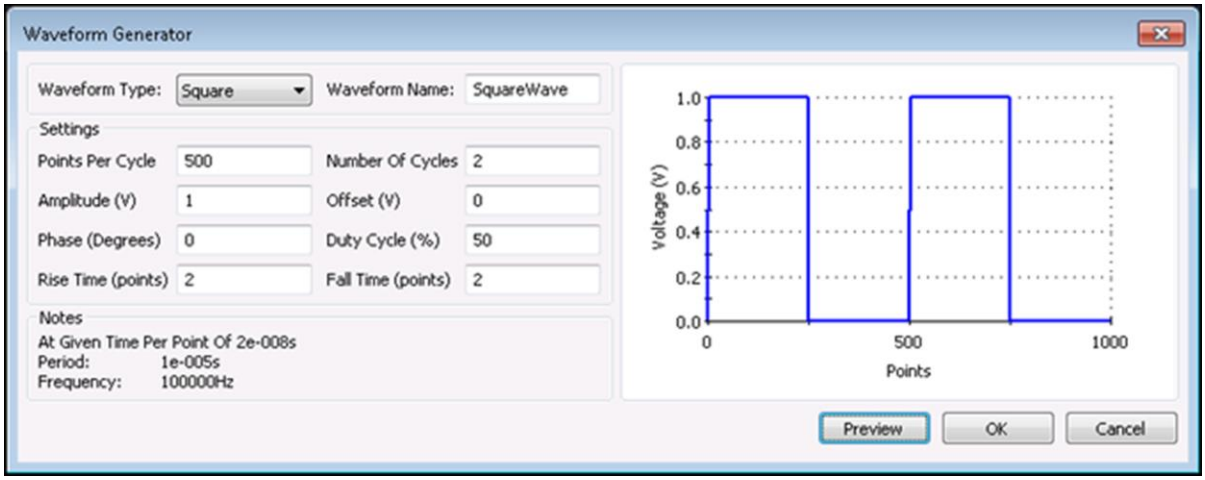


Square waveform

An example of a square waveform is shown in the next figure. The waveform for this example is named WAVE3, but can be any name that is not already used in the Scratch Pad.

After changing one or more settings, select **Preview** to display the waveform. Select **OK** to place the waveform in the Scratch Pad.

Figure 573: Square waveform

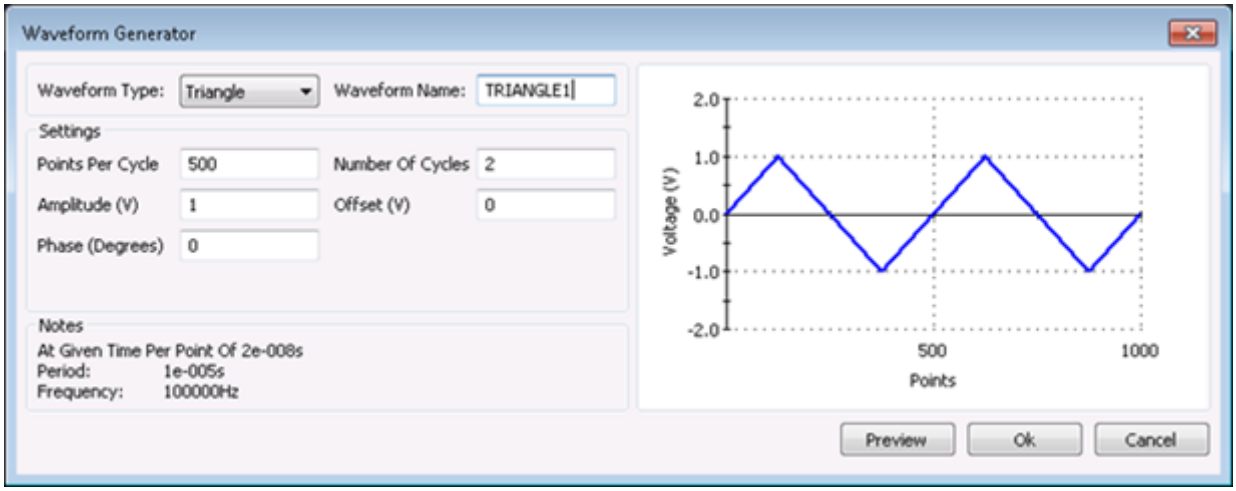


Triangle waveform

An example of a triangle waveform is shown in the next figure. The waveform for this example is named TRIANGLE1, but can be any name that is not already used in the Scratch Pad.

After changing one or more settings, select **Preview** to display the waveform. Select **OK** to place the waveform in the Scratch Pad.

Figure 574: Triangle waveform



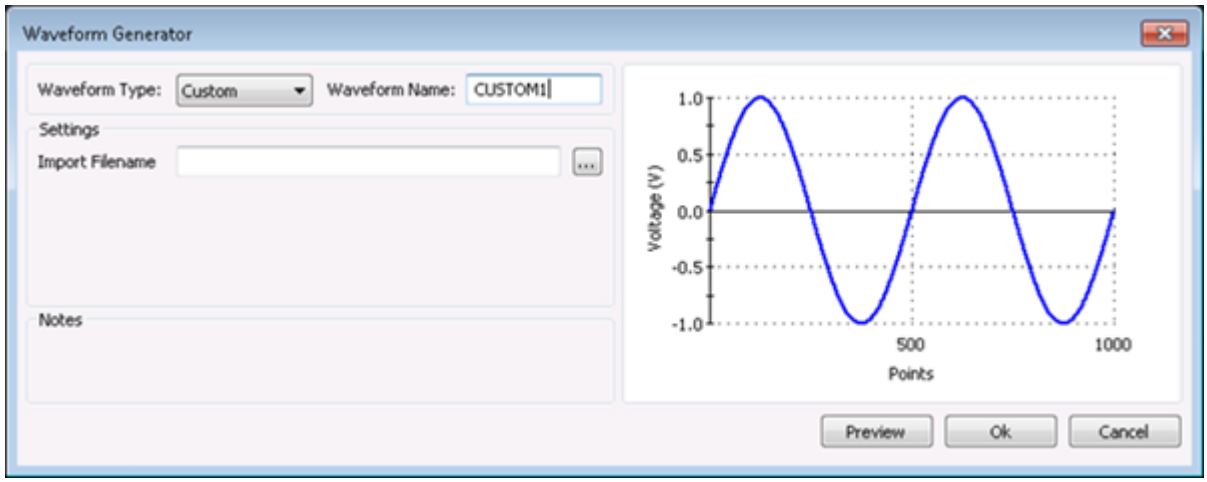
Custom waveform

An example of a custom waveform is shown in the following figure. The waveform for this example is named CUSTOM1, but can be any name that is not already used in the Scratch Pad.

The voltage values for the waveform are retrieved from an imported file (.txt or .csv). After creating a file (.txt or .csv) for the custom waveform, use **Import Filename** to import the file into the Waveform Generator.

After importing the file, select **Preview** to show the waveform. Select **OK** to place the waveform in the Scratch Pad.

Figure 575: Custom waveform



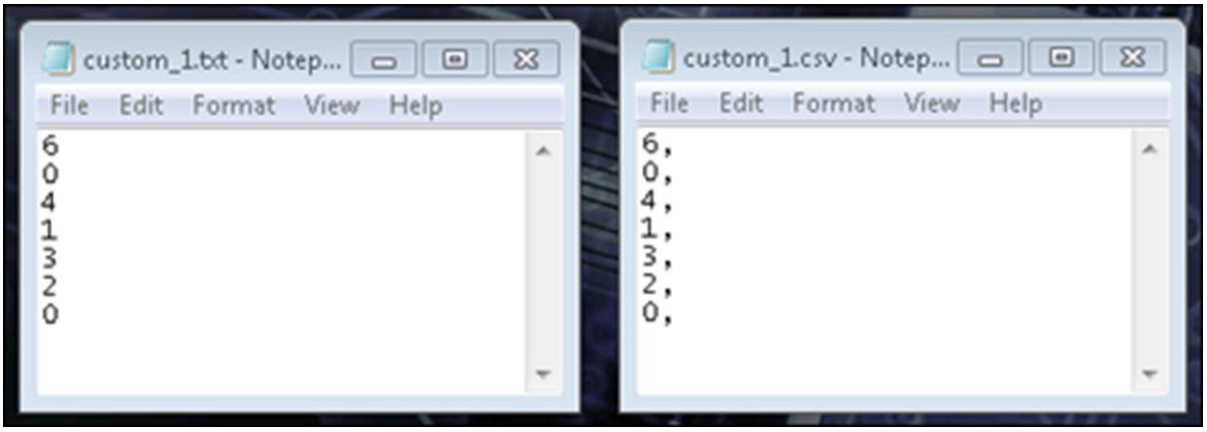
Creating a file (.txt or .csv) for custom waveform

The waveform file is created using a text editor, such as Notepad.

To create the list of voltage points:

1. Open a text editor.
2. On the first line, type the number of voltage points in the waveform, and then type the list (one per line) of values for the waveform:
   - .**txt file format:** As shown in the below figure, commas are not used to separate values.
   - **.csv file format:** As shown in the below figure, commas are used to separate values. Only the first column of data is used for the waveform. Additional columns are ignored.

Figure 576: Creating a .txt or .csv file for a custom waveform



3. The custom waveform is a simple 6-point waveform made up of these voltage values: 0 V, 4 V, 1 V, 3 V, 2 V, 0 V. Those seven entries are shown in the text editors in the above graphic. The time at each point is determined by the Time Per Point setting in the Arb Generator tab.
4. Save as a waveform file (`.txt` or `.csv`) in the ArbFiles folder at the location:

   `C:\s4200\kiuser\KPulse\ArbFiles`

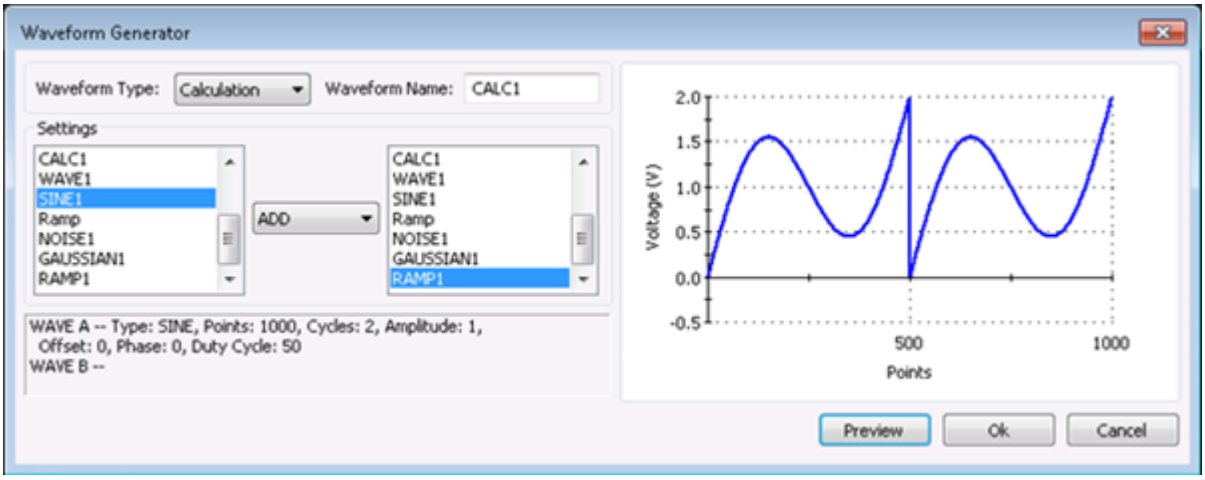
Calculation waveform

An example of a calculation waveform is shown in the next figure. The waveform for this example is named CALC1, but can be any name that is not already used in the Scratch Pad.

The calculation (add, subtract, multiple or divide) performs the selected math operation on two selected Scratch Pad waveforms. In the below example, SINE1 is added to Ramp.

After selecting the two waveforms and the math operation, select **Preview** to display the result of the calculation. Select **OK** to place the waveform in the Scratch Pad.

Figure 577: Calculation waveform

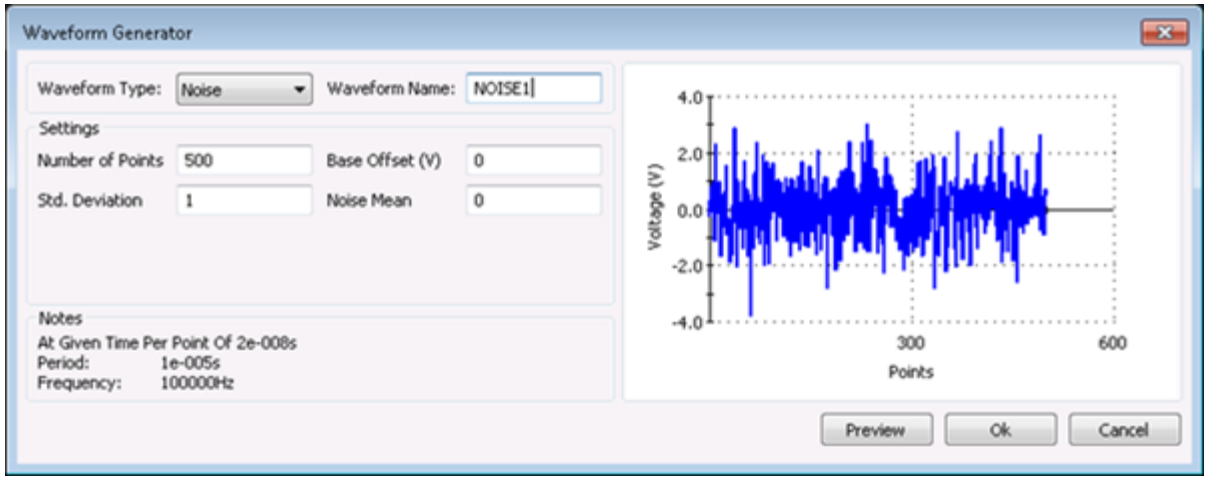


Noise waveform

An example of a noise waveform is shown in the next figure. The waveform for this example is named NOISE1, but can be any name that is not already used in the Scratch Pad.

After changing one or more settings, select **Preview** to display the waveform. Select **OK** to place the waveform in the Scratch Pad.

Figure 578: Noise waveform

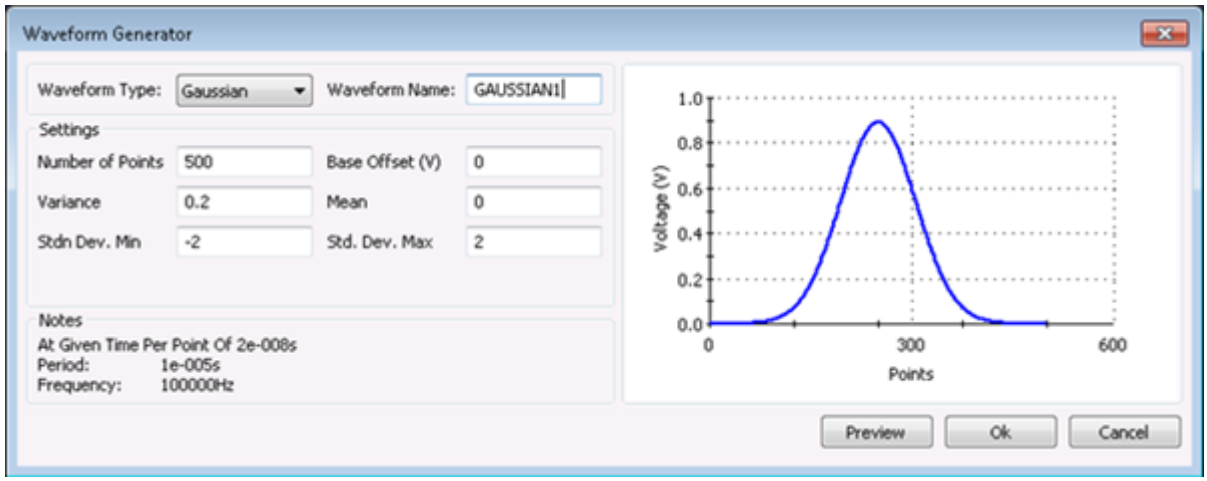


Gaussian waveform

An example of a Gaussian waveform is shown in the next figure. The waveform for this example is named GAUSSIAN1, but can be any name that is not already used in the Scratch Pad.

After changing one or more settings, select **Preview** to display the waveform. Select **OK** to place the waveform in the Scratch Pad.

Figure 579: Gaussian waveform

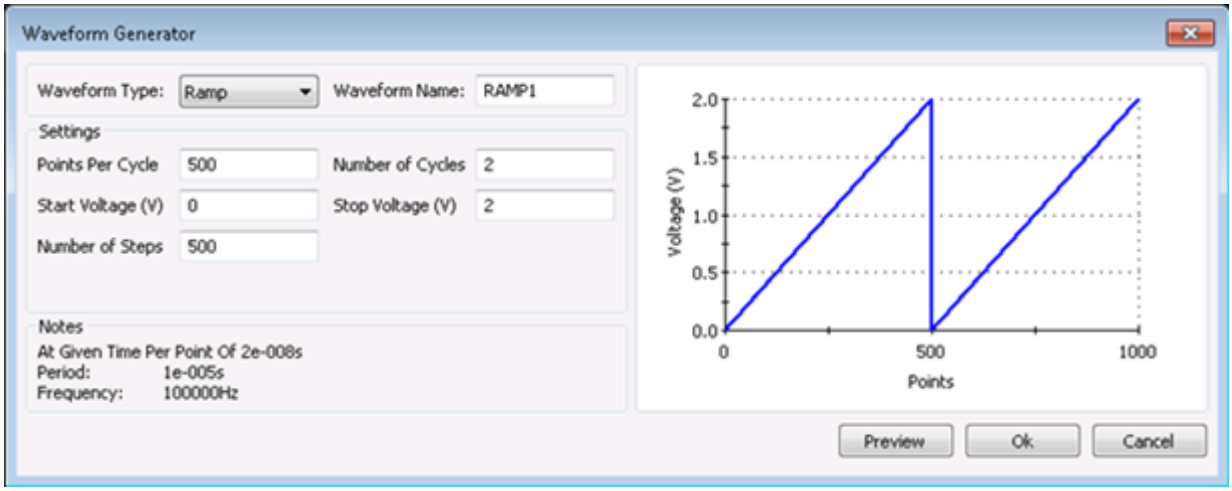


Ramp waveform

An example of a ramp waveform is shown in the next graphic. The waveform for this example is named RAMP1, but can be any name that is not already used in the Scratch Pad.

After changing one or more settings, select **Preview** to display the waveform. Select **OK** to place the waveform in the Scratch Pad.

Figure 580: Ramp waveform



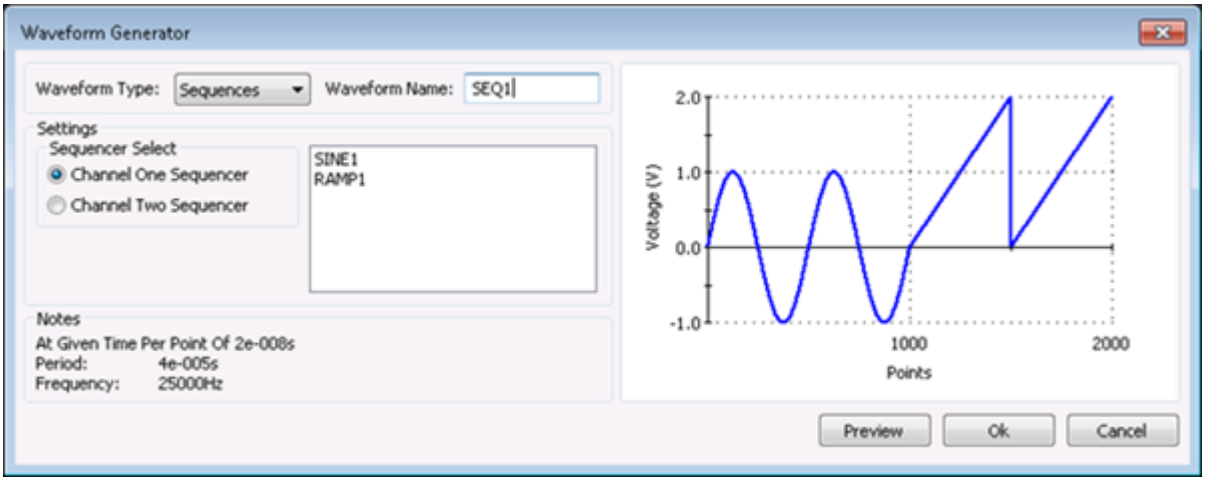
Sequences waveform

An example of a sequences waveform is shown in the below graphic. The waveform for this example is named SEQ1, but can be any name that is not already used in the Scratch Pad.

A sequence waveform consists of the waveforms that are present in the Channel 1 or Channel 2 Sequencer.

After selecting either **Channel One Sequencer** or **Channel Two Sequencer**, select **Preview** to show the waveform. Select **OK** to place the waveform in the Scratch Pad.

Figure 581: Sequences waveform



Section 12

System administration

In this section:

System Administration introduction

This section provides information on system maintenance tasks for the 4200A-SCS computer.

NOTE

Most of the procedures discussed in this section should be done by a knowledgeable Microsoft® Windows® system administrator.

Embedded computer policy

CAUTION

If you install software that is not part of the standard application software for the 4200A-SCS, the non-standard software may be removed if the instrument is sent in for service. Back up the applications and any data related to them before sending the instrument in for service.

CAUTION

Do not reinstall or upgrade the Microsoft® Windows® operating system (OS) on any 4200A-SCS unless the installation is performed as part of authorized service by Keithley Instruments. Violation of this precaution will void the 4200A-SCS warranty and may render the 4200A-SCS unusable. Any attempt to reinstall or upgrade the 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 you must not attempt to reinstall or upgrade the operating system, you can restore the hard drive image (complete with the operating system) using the Acronis True Image OEM software tool, described in System-level backup and restore software (on page 12-8).

Default user accounts

There are two preconfigured Windows® user accounts on the 4200A-SCS. You can also create a unique user account.

You can use any user account to access all Clarius+ applications and applicable non-Keithley software tools.

The `kiuser` account is the default account. The logon password is `kiuser1`. You do not need a password by default. However, if you log out of the `kiuser` account and log in again, you are prompted for the password.

The `kiadmin` account logon password is `kiadmin1`.

NOTE

You need to use the `kiadmin` account to install software and additional application tools.

Windows 10 compatibility

Clarius+ versions 1.4 and newer require Microsoft Windows 10 for full compatibility.

Create new user accounts

You can create unique Windows® user accounts for each 4200A-SCS user. This enables you to customize the behavior of a 4200A-SCS without affecting its behavior for other users. It also provides additional data protection and privacy, because each user can log onto the 4200A-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:

1. Make sure that you are logged into the Microsoft® Windows® administrative account. The default administrative account is `kiadmin`.
2. From the Windows start menu, type **settings** in the search box. The results display the Windows **Settings** utility.
3. Press <Enter> to open the application.
4. Select **Accounts**.
5. Select **Other people**, then **Add someone else to this PC**.
6. Select **I don't have this person's sign-in information**.
7. Select **Add a user without a Microsoft account**.
8. Enter a user name.
9. Enter a password, confirm the password, and provide a password hint.
10. Select **Next**.
11. Log on using the new account and set up the desktop and other elements of the new profile.
12. Select **Start > All Programs > Keithley Instruments > Initialize New User** to configure Clarius+ for the new user.

NOTE

Initialize New User must be run each time a new user account is created.

System directories and files

All Clarius+ files are stored in the `C:\s4200` folder.

User files and the sample projects and standard libraries that are included with Clarius+ are stored in the directory `C:\s4200\kiuser`. All 4200A-SCS users can access these libraries.

The subdirectories in the user folder contain:

- `Devices`: Default Clarius device library.
- `Projects`: Default Clarius project library.
- `Tests`: Default Clarius test library.
- `usrlib`: Default collection of KULT user libraries.

The system directory stores all of the binary and executable files that Clarius+ needs to control the 4200A-SCS. Clarius+ system files are stored in `C:\s4200\sys`.

CAUTION

The files stored in the `C:\s4200\sys` and `C:\4200A-SCS` folders must not be modified by 4200A-SCS users or by system administrators. This folder (directory) must reside on the 4200A-SCS hard disk.

Manage projects for multiple users

You cannot use multiple directories for the 4200A-SCS.

If you have multiple users that are using one 4200A-SCS, you can use options in the My Project dialog box and in the Library Information Editor to assign unique keywords to each project. These keywords can be used in the library and project search fields to locate your projects. For information on adding keywords through My Projects, refer to Edit project information (on page 6-20). For information on adding keywords to projects that are added to the library, refer to Edit information for a library object (on page 6-373).

When adding projects, you can also assign project names that help you identify the project.

You can also use the import, export, and delete features in My Projects to manage multiple users.

To use import, export, and delete to manage projects, each user will:

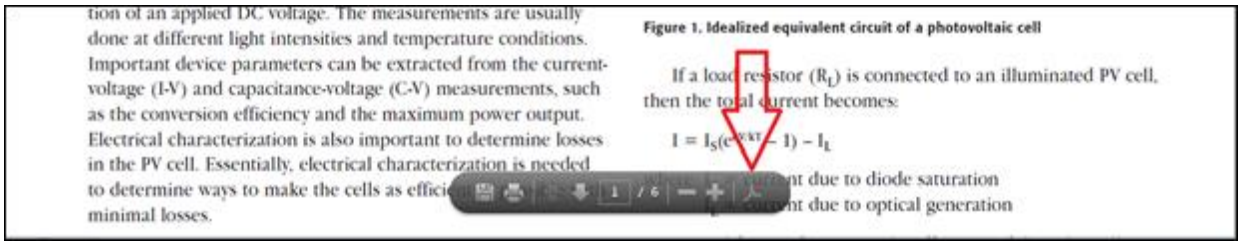
1. Create and use a project.
2. When work is complete, in My Projects, export the project. Refer to Export a project (on page 6-22).
3. Delete the project from My Projects. Refer to Delete a project (on page 6-25).
4. If you need to use the project again, import the project into the 4200A-SCS. Refer to Import a project (on page 6-22).

Enable scroll bars in Acrobat Reader

The viewing default for Adobe® Acrobat® Reader® may default to be full-screen touch mode, which means that no scroll bar or other tools are visible on the screen when you open PDFs. You can change this setting.

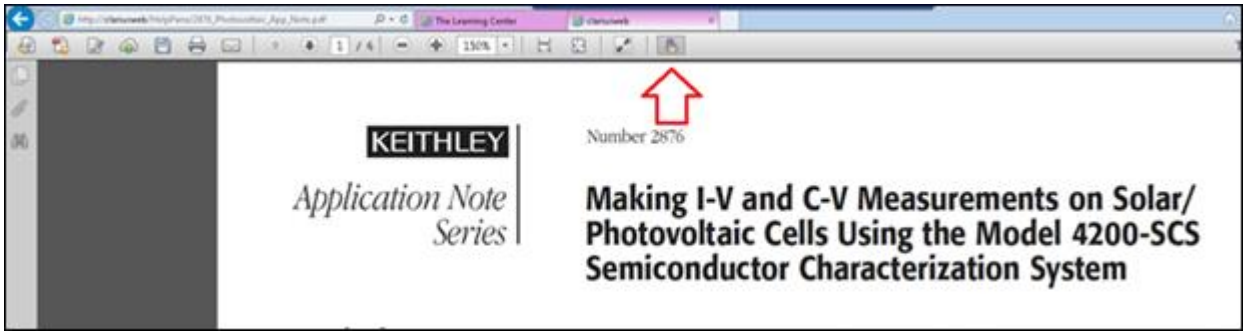
To display the scroll bars and other Reader controls:

1. Open a PDF.
2. Hover you mouse cursor or touch near the center-bottom of the viewer. A menu is displayed.
3. Select the Adobe Acrobat symbol. The tools and utilities of Reader are displayed.



4. Select **Toggle Touch Mode**, as shown in the following graphic. The scroll bar and other controls are now available.

Figure 582: Acrobat Toggle Touch Mode



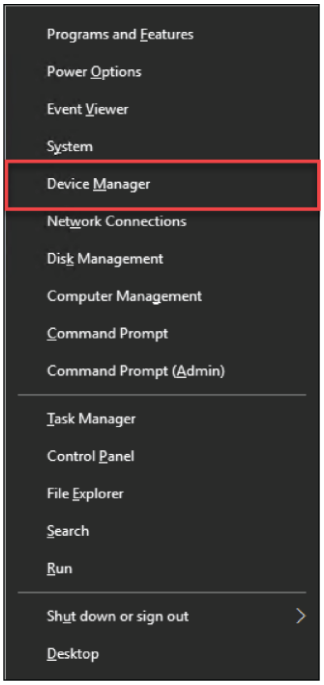
Disable the touchscreen

You can disable the touchscreen display of the 4200A-SCS.

To disable the touchscreen:

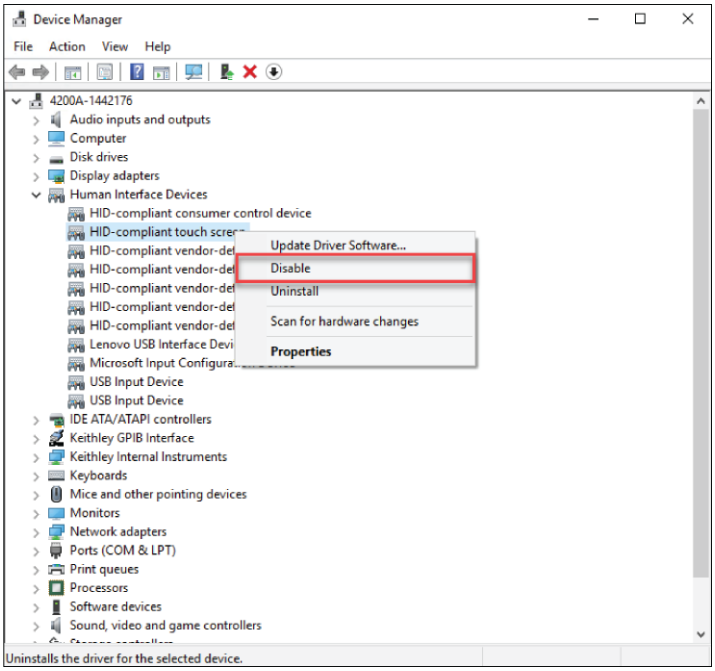
1. Select the <Windows> + X keys on your keyboard.
2. Select **Device Manager**.

Figure 583: Selecting Device Manager



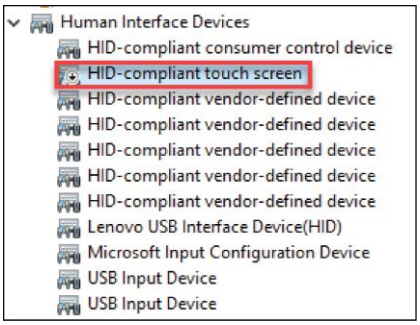
3. Select the arrow next to **Human Interface Devices** (HID) to display a list of available devices.
4. Right-click **HID-compliant touch screen**.
5. Select **Disable**.

Figure 584: Disabling the touchscreen



6. Select **Yes** to confirm. The HID-compliant touchscreen option displays an indicator that the device is disabled.

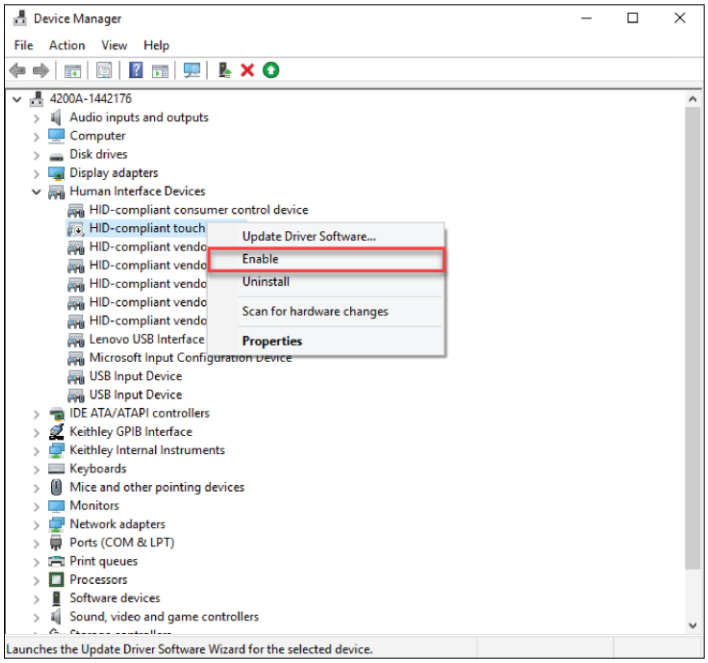
Figure 585: Touchscreen device disabled indicator



To enable the touchscreen:

1. To enable the touchscreen, right-click the HD compliant touchscreen option, then select **Enable**.

Figure 586: Enabling the touchscreen



System-level backup and restore software

Acronis True Image (OEM) is a software tool that allows 4200A-SCS users to create hard-disk images, including user data, environment settings, and operating system files. This software is preinstalled on every 4200A-SCS system by Keithley Instruments. Keithley recommends backup as the best way to preserve user application files and measurement data.

You can use the Acronis software tool to:

Restore the drive image to factory condition (on page 12-9)

Choose the files to be backed up (on page 12-10)

For additional information, refer to the *Acronis True Image OEM User's Guide*, available from The Learning Center.

Restore the drive image to factory condition

In addition to allowing you to create your own backups, the 4200A-SCS hard drive image is archived before shipment. This allows you to restore the contents of the 4200A-SCS hard drive to the condition it was in before shipment from Keithley Instruments.

The restore process takes about ten minutes.

CAUTION

Restoring the 4200A-SCS hard drive to factory condition deletes all files written to the hard drive after shipment. Ensure you back up all data and files before you restore the hard drive to factory condition.

To restore the 4200A-SCS hard drive to factory condition:

1. Reboot the 4200A-SCS. As the system boots up, there is a splash screen that contains the Keithley logo.
2. In the upper-left corner, either a prompt or menu with options is displayed depending on your 4200A-SCS version.
    a. If you see `Press F11 to run Acronis Startup Recovery Manager...`, press the **F11** key to begin the recovery process.
    b. If you see `Select an item by using the keyboard:` followed by multiple options, press the **1** key to begin the recovery process.

NOTE

You have approximately five seconds to press a key. Otherwise, Windows® will begin to boot.

3. Follow the instructions to restore the factory hard drive image.

For additional detail about the Acronis True Image HD software, please refer to the *Acronis True Image OEM User Guide*, available from The Learning Center.

Choose the files to be backed up

Keithley recommends backing up the following files and directories, where applicable:

- The default user directory
  When a 4200A-SCS is received from the factory, the `C:\s4200\kiuser` directory contains all installed test results and user application files. Clarius+ stores all user-created projects and tests in this location.

NOTE

Avoid selective backups of `C:\...\data` subfolders, such as `C:\s4200\kiuser\Projects\default\tests\data` . Such backups do not preserve the additional application files that are needed to restore the associated test setups and conditions.

- Special data files and directories
  Back up any additional, specialized data files or directories, such as the following:
  - A file into which a User Test Module (UTM) automatically places test results. For example, a UTM could be set to put results into `C:\TestData\test001.dat`.
  - A file that results from manual processing of data using Microsoft® Excel®.

NOTE

It is possible to back up the whole hard drive. However, the 4200A-SCS system files and application program files occupy a large portion of the storage and may not be properly recoverable unless backed up using special software. Such backups may be justifiable only if the user directory structure is too complex for selective backup and when storage space is plentiful.

You cannot back up files that are normally in use — for example, certain system files — without special backup software. Before considering use of such software, review the precautions in Protecting software integrity (on page 12-11).

Protect user files and system software

This section describes recommendations for software and data preservation. The recommendations apply to any 4200A-SCS that meets the following criteria:

- Was produced during or after 2016.
- Runs Windows 10 Enterprise.
- Runs Clarius+ applications version 1.0 or later.

Protect software integrity

The 4200A-SCS has been designed and tested for maximum system stability, reliability, and performance in the factory-standard configuration. To protect the system software, observe the following tips:

- Do not defragment the solid state drive (SSD) that is preinstalled on the 4200A-SCS. Defragmenting a SSD quickly reduces the life of the drive.
- Refer to the Embedded computer policy (on page 1-5) for information about upgrading Microsoft® Windows® software and installing third party software.
- Protect the system from viruses, which can reach the system through a facility network or a contaminated USB flash drive. The 4200A-SCS does not contain preinstalled protection against viruses, spyware, or malware. You are responsible for installing software protection packages.
- Do not attempt to format the system hard drive or reinstall the operating system. Attempting to do so will make the system inoperative and require factory repair.

CAUTION

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

Read and write permission access to USB ports

For enhanced system security and data integrity, you can use the Microsoft® Management Console (MMC) to enable, disable, or deny all access to removable storage devices used with the USB ports of the 4200A-SCS.

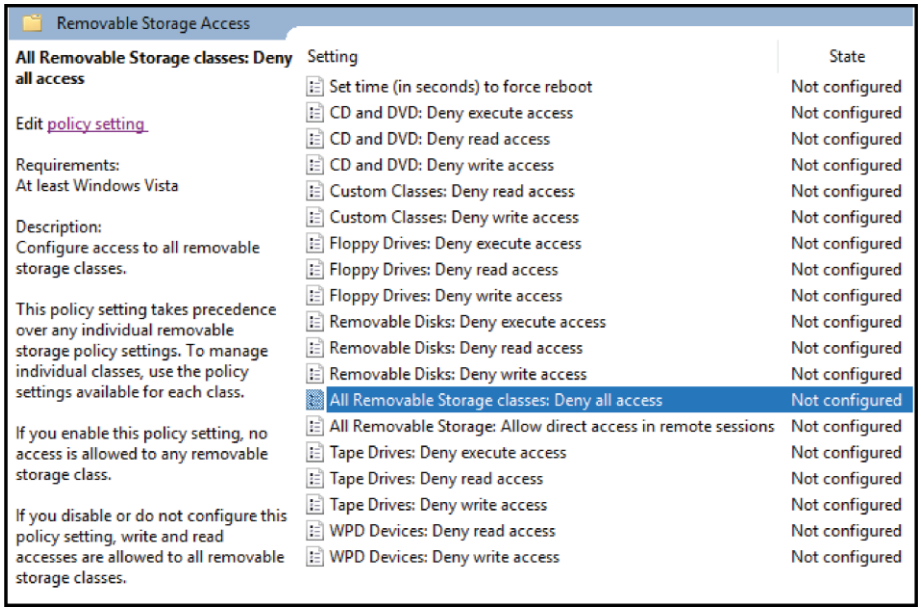
CAUTION

These steps are recommended only for advanced users.

To change access to USB ports:

1. Make sure that you are logged in to the Microsoft® Windows® administrative account. The default administrative account is `kiadmin`.
2. From the Windows start menu, type **gpedit.msc** in the search box. The results display the Windows Group Policy Editor.
3. Press <Enter> to open the application.
4. In the left pane, select **Computer Configuration > Administrative Templates > System > Removable Storage Access**.
5. In the middle pane of the MMC window, double-click a policy setting. In the next figure, the policy settings for all removable storage devices are displayed.
    a. Select either **Enabled** or **Disabled**.
    b. Select **Apply**, then **OK**.
    c. Repeat for any additional policy settings.

Figure 587: Policies for removable storage device access



6. Reboot the 4200A-SCS.

Add Clarius+ applications to the Windows startup menu

You can add Clarius+ applications to the Microsoft® Windows® startup menu to have them start automatically when you log into Windows.

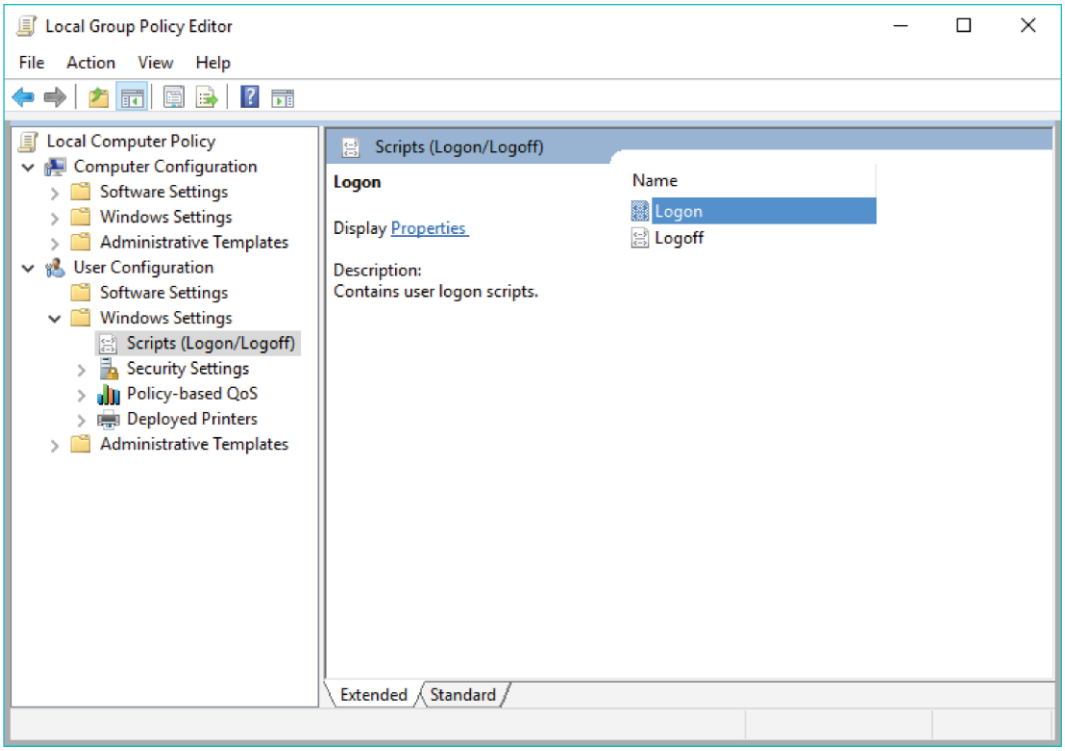
NOTE

Add either Clarius or KXCI to the startup menu.If you add both, it will cause conflicts when you start your 4200A-SCS.

To set up an application to start automatically:

1. Make sure that you are logged in to the Microsoft® Windows® administrative account. The default administrative account is `kiadmin`.
2. From the Windows start menu, type **gpedit.msc** in the search box. The results display the Windows Group Policy Editor.
3. Press <Enter> to open the application.
4. In the left pane, under **User Configuration > Windows Settings**, select **Scripts**.

Figure 588: Windows Group Policy Editor



5. Select **Logon** from the main window.
6. Select **Add**.
7. Enter one of the following application file paths in the **Script Name** field:
   - For Clarius+: `C:\S4200\sys\bin\Clarius.exe`
   - For KXCI: `C:\S4200\sys\bin\KXCI.exe`
8. Select **OK**.
9. Select **Apply**.
10. Close the Group Policy Editor window.

Section 13

Maintenance

In this section:

Introduction

The information in this section describes routine maintenance of the instrument that the operator can perform.

Line fuses

Rear-panel fuses protect the power line input of the unit.

⚠ WARNING

Turn off the power and disconnect the line cord before replacing the fuses. Failure to turn off the power and disconnect the line cord before replacing the fuses may result in personal injury or death due to electric shock.

To replace the line fuses:

1. Locate the fuses. The fuses are in two fuse holders above the AC receptacle.
2. Use a small slotted screwdriver to remove each fuse holder.
3. Push the fuses in and rotate them counterclockwise to remove the fuses from the fuse holders.
4. Replace the fuses with the correct type according to your line voltage:
   - **100 – 125 VAC**: 250 V, 15 A, 5 x 20 mm, slow-blow
   - **220 – 240 VAC**: 250 V, 8 A, 5 x 20 mm, slow-blow

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, correct the cause of the problem before replacing the fuses.

Front-panel display

Do not use sharp metal objects, such as tweezers or screwdrivers, or pointed objects, such as pens or pencils, to touch the touchscreen. It is strongly recommended that you use only fingers to operate the instrument. Use of clean-room gloves to operate the touchscreen is supported.

Cleaning the front-panel display

If you need to clean the front-panel LCD touchscreen display, use a soft dry cloth. If necessary, use a cloth dampened with an ammonia-free glass cleaner. Do not spray cleaning fluids onto the display.

Adjusting the display

You can use the Front Panel Control option to adjust the brightness of the screen.

The Front Panel Control option is in the Windows system tray at the bottom right of the screen.

Air intake ventilation screens

Your 4200A-SCS comes equipped with an internal axial fan for active cooling of the instrument cards. Your instrument must be properly cooled to maintain optimum operating specifications.

There are three air inlets on your instrument: One on each side and one on the bottom.

Every six months or when a screen appears to be clogged, use a vacuum cleaner with a soft brush attachment to carefully remove the accumulated particulates. The screens must remain unrestricted to provide sufficient cooling airflow.

CAUTION

Never use compressed air to blow debris from the screens. Doing so can force particulates through the screen and into the instrument.

Solid-state hard drive maintenance

Your 4200A-SCS is shipped with a solid-state hard drive (SSD). SSDs offer several advantages over traditional hard drives, such as:

- Superior data access time
- Increased reliability
- Lower power usage
- Less heat generation

The SSD installed in your instrument does not require conventional hard drive maintenance, such as defragmentation.

CAUTION

Do not perform or schedule defragment operations on your SSD. Defragmentation can cause premature wear on the SSD and reduce its usable capacity more quickly compared to normal use. Programs or operations that result in excessive write operations to the SSD will also reduce the capacity more quickly.

Repacking for shipment

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

- Call Keithley Instruments' repair department at 1-800-833-9200 for a Return Material Authorization (RMA) number.
- Let the repair department know the warranty status of the 4200A-SCS.
- Write ATTENTION REPAIR DEPARTMENT and the RMA number on the shipping label.

CAUTION

If you installed software that is not part of the standard application software for the 4200A-SCS, the non-standard software may be removed when the instrument is sent in for service. Back up the applications and any data related to them before sending the instrument in for service.

Firmware upgrade

When the system software is updated, you should upgrade firmware for each 4200A-SCS instrument. Before starting the firmware upgrade, make sure the 4200A-SCS is powered by an uninterruptable power source. Refer to the release notes for detailed instructions on the firmware upgrade of 4200A-SCS instruments, including the specific versions required for each instrument. See Accessing the release notes (on page 13-5) for more information.

CAUTION

Power the 4200A-SCS with an uninterruptable power supply during the firmware upgrade process. Interruption of the firmware upgrade process may damage an instrument card.

To upgrade the firmware:

1. From the Windows taskbar, select **Start**.
2. In the Keithley Instruments folder, select the **Firmware Upgrade** tool. If your instrument needs to be upgraded, the Upgrade button is active and the Status column shows "Upgrade Required."
3. Select an instrument that needs to be updated, and select **Upgrade**.

A progress bar is displayed during the upgrade. Note that each instrument card type is upgraded separately.

Accessing the release notes

Refer to the Learning Center for the release notes.

Reset the hardware

If you suspect a problem with any 4200A-SCS hardware, do not use Windows Task Manager to close any open applications. Both hardware and software issues may occur.

To reset the hardware:

1. Select **Start**.
2. Type **resethw**.
3. Select the instruments that need reset.
4. Click **Reset.**

Lithium battery

The 4200A-SCS contains a CR2032 cell ($LiMnO_2$) battery. Perchlorate material may require special handling. See Hazardous waste - perchlorate (dtsc.ca.gov/hazardouswaste/perchlorate).

This battery is not replaceable by the user.

Calibrate the system

To maintain SMU performance specifications, you must auto calibrate the 4200A-SCS every 24 hours or any time after the ambient temperature has changed more than ±1 °C.

The autocalibration routine recalibrates the current and voltage offsets for all source and measurement functions of all SMUs in the system.

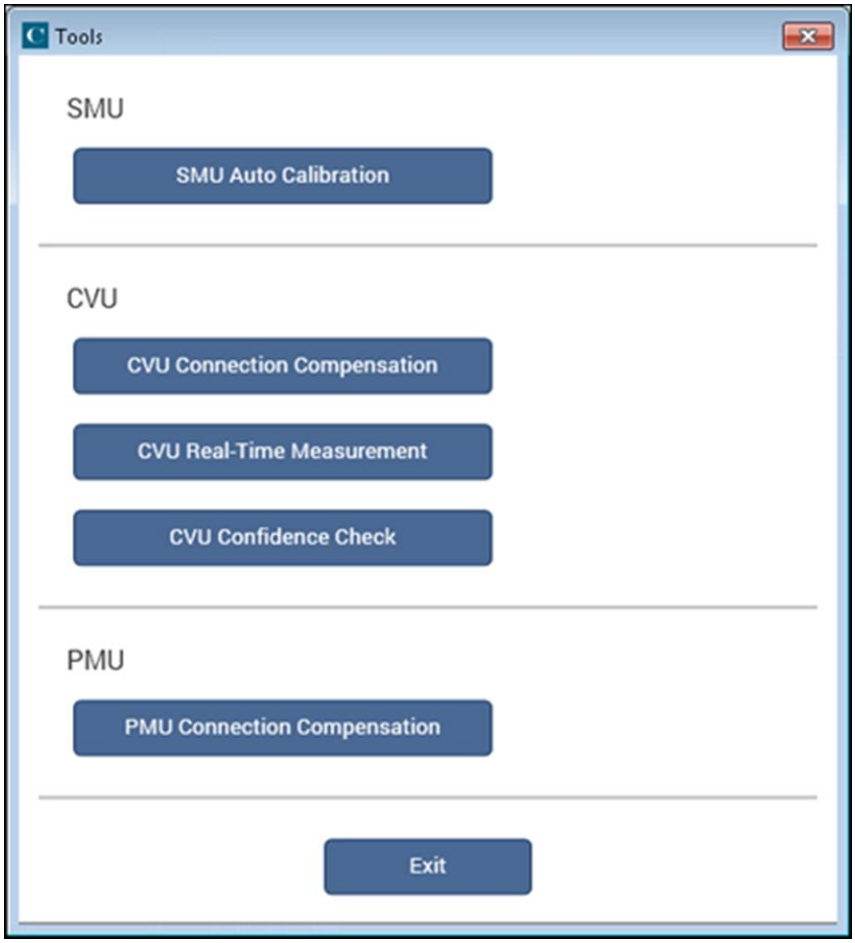
NOTE

Before initiating a calibration, allow the system to warm up for at least 30 minutes. Clarius will prevent autocalibration from occurring if the system is not sufficiently warmed up.

To auto-calibrate:

1. Allow the system to warm up for at least 30 minutes.
2. Remove connections to all SMUs in the 4200A-SCS.
3. Open Clarius.
4. Select **Tools**. The following dialog box is displayed.

Figure 589: Clarius Tools dialog box



5. Select **SMU Auto Calibration**. A warning dialog box is displayed.
6. Select **OK.** The SMU Auto Calibration dialog box opens, as shown in the following figure.

Figure 590: SMU Auto Calibration dialog box

7. Select **Start**. A progress bar is displayed.
   When autocalibration is complete, the message "Auto calibration successfully completed" is displayed.
8. Select **Close**. Autocalibration is complete.

Fan operation

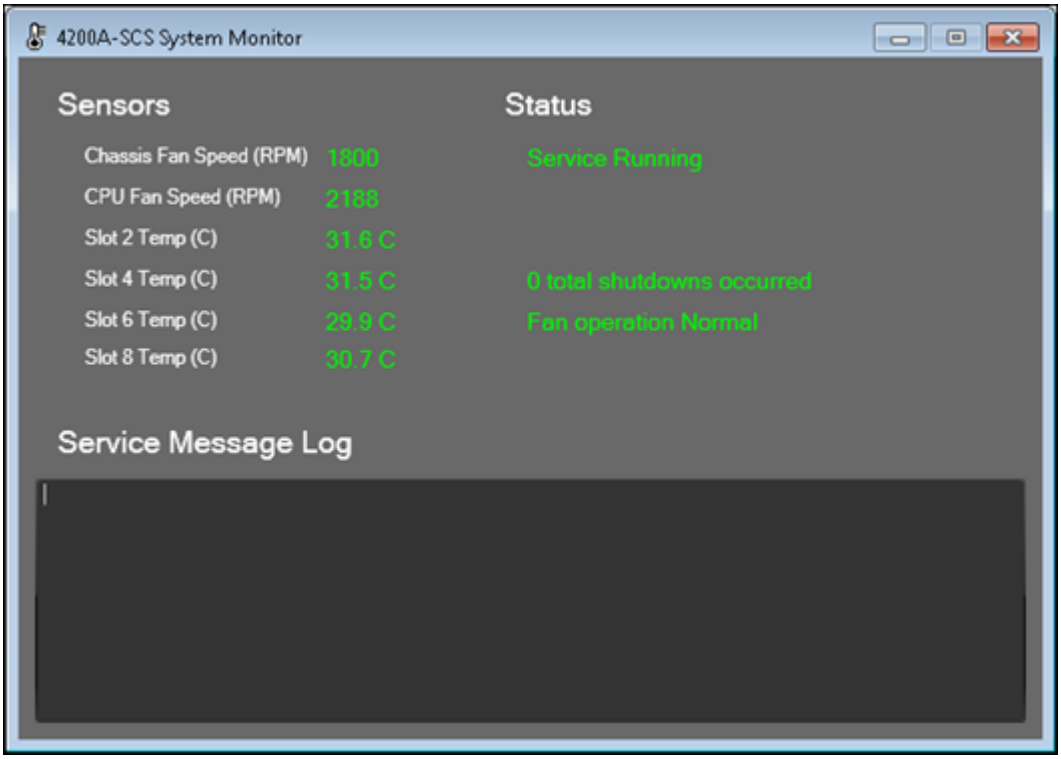
If the 4200A-SCS fan stops or is running too slowly, a warning message is displayed on the instrument. If the fan does not return to normal within five minutes of the initial warning, the 4200A-SCS is shut down to prevent system damage from overheating.

If the system was powered down because of a fan problem and is then powered up, another warning message is displayed to indicate the reason for the shutdown.

If you see this message repeatedly, contact Keithley Instruments. See Service tek.com/service for contact information.

For detail on fan operation, you can open the 4200A-SCS System Monitor. This is available in the Windows system tray.

Figure 591: 4200A-SCS System Monitor



Making stable measurements with SMUs

The following topics discuss various considerations when making stable measurements, including single-SMU stability, multiple-SMU stability, and avoiding oscillation.

Single SMU stability considerations

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

A SMU that is sourcing voltage is stable when driving capacitive loads up to 10 nF. However, at the lower current measurement ranges, large capacitive loads may increase settling time and may cause overshoot and ringing. To reduce this effect, you can add a small resistor in series with the capacitive load. Choose a resistor that provides an R_C time constant of 1 ms to 10 ms.

You can also increase the measurement delay factor for the test to reduce this effect:

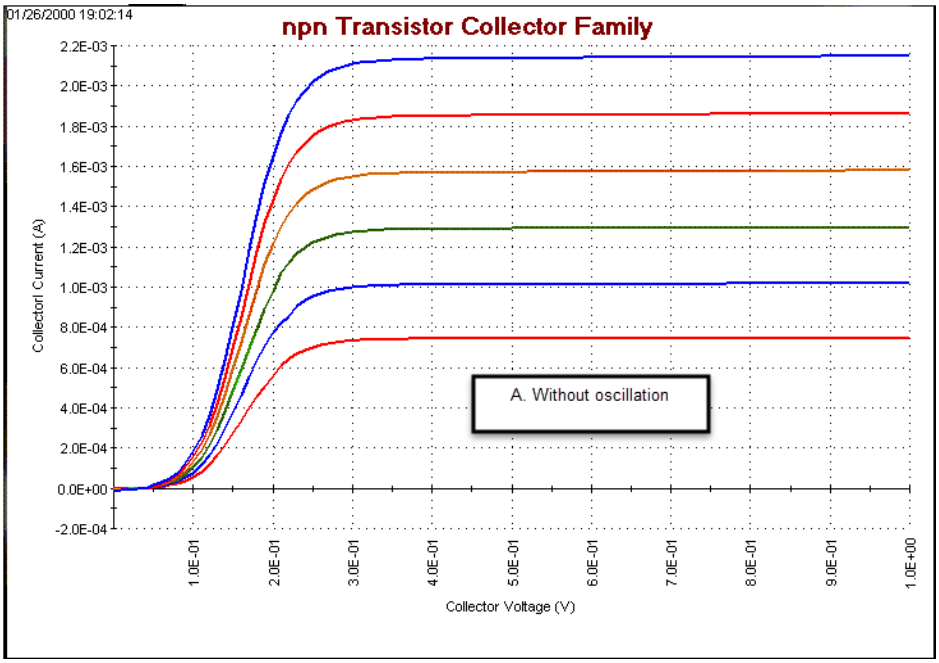
1. In Clarius, select the test.
2. Select **Configure**.
3. In the right pane, select **Test Settings**.
4. Select **Advanced**.
5. Set the Speed to **Custom**.
6. Increase the **Delay Factor**.

Refer to Delay Factor (on page 6-112) for additional information.

Multiple SMU stability considerations

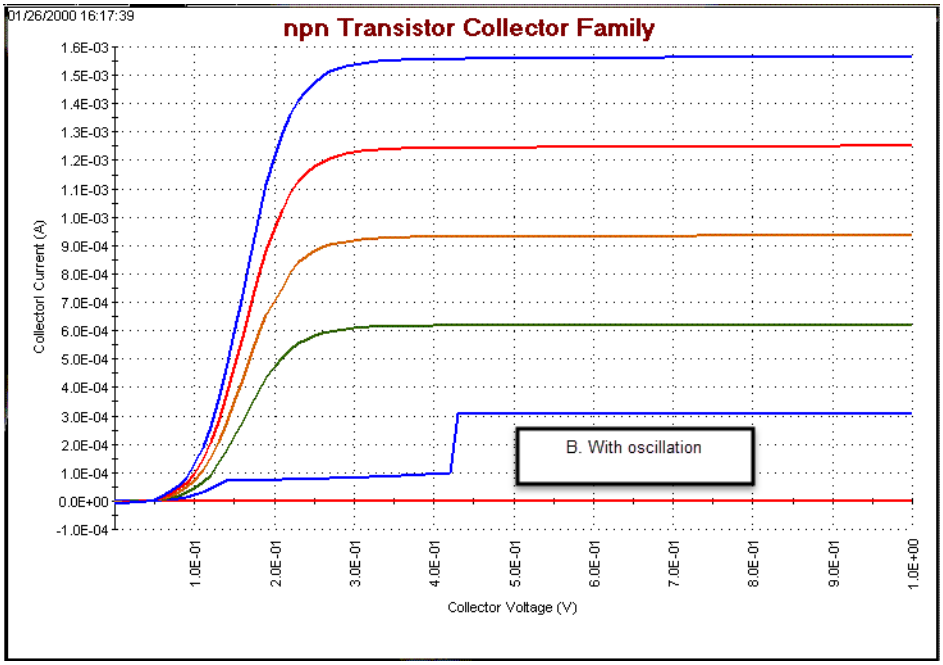
Using two or more SMUs to test an active device, such as a field-effect transistor (FET) or bipolar junction transistor (BJT), can aggravate system stability. The next figure shows an example of BJT characterization curves determined under stable conditions.

Figure 592: Effects of oscillation on test data: Without oscillation



The next figure shows an example of what can happen to a BJT characterization curve when the system oscillates.

Figure 593: Effects of oscillation on test data: With Oscillation



In general, oscillations can be classified in two categories: high-frequency oscillations (100 kHz through 200 MHz), and low frequency-oscillations (below 100 kHz). For solutions to both types of oscillation, refer to Eliminating oscillations (on page 13-11).

Eliminating oscillations

The measures needed to eliminate oscillations depend on whether the oscillations are high frequency or low frequency.

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 preamplifiers as close to the DUT as possible.
- Connect the COMMONs (outer shields) of all cables together at the DUT.
- Use lossy 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.

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: ZDrain SMU / ZSource SMU
- For a BJT (bipolar junction transistor): ZCollector SMU / ZEmitter SMU

A SMU measures current through 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 for a FET, try the following:

- Set (Drain-SMU current measure range) = (Source-SMU current measure range)
- If necessary, set both SMUs to autorange.
- For the source SMU, do not select the Common operation mode. The mode prevents you from configuring a current measurement range for the source SMU, which results in a lower impedance than at the drain SMU; a potentially high gain; and an increased likelihood of low-frequency oscillation. Instead, configure the source SMU for the Voltage Bias operation mode and set it to 0 V. This allows you to configure the current measurement range.

To avoid oscillations for a BJT, try the following:

1. Set (Collector-SMU current measure range) = (Emitter-SMU current measure range)
2. If necessary, set both SMUs to autorange.
3. For the emittter SMU, do not select the Common operation mode. This mode prevents prevents you from configuring a current measurement range for the emitter SMU, which results in a lower impedance than at the collector SMU; a potentially high gain; and an increased likelihood of low-frequency oscillation. Instead, configure the emitter SMU for the Voltage Bias operation mode 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 autorange.

NOTE

For instructions on configuring tests, refer to Configure a complex test (on page 6-187).

Low-current measurements

Low-current measurements made with a SMU or preamplifier are subject to error sources that can have a serious impact on measurement accuracy. The following topics discuss low-current measurement considerations, including leakage currents, generated currents, noise and source impedance, and voltage burden. Refer to the Keithley Instruments *Low Level Measurements Handbook* for more information.

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 are a common source of leakage. Typically, insulation resistance between conductors in the type of triaxial cables supplied with the SMUs and preamplifiers is approximately 1 PΩ (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 triaxial cables to the SMU or preamplifier automatically drives the inner cable shield at guard potential, minimizing the effects of cable leakage currents. See Guarding (on page 3-32) for details.

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 (on page 3-34) for details.

Generated currents

Any extraneous generated currents in the test system add to the expected current, which can cause errors. Currents can be internally generated, as in the case of preamplifier 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. The next table summarizes the typical ranges of a number of generated currents.

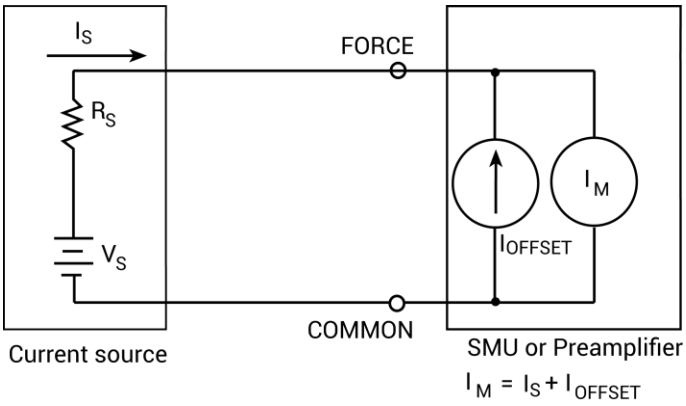
Typical generated currents

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

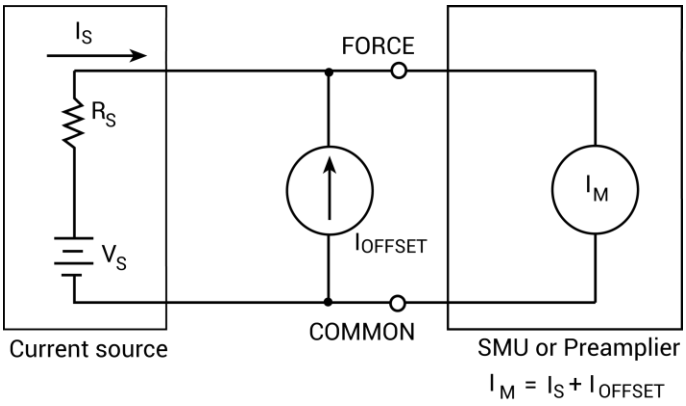
The preamplifier has a small current, known as the input offset current, that flows at all times. As shown in the figure below, the input offset current adds to the measured current so that the SMU measures the sum of the two currents. Note that input offset current can be brought to within specifications by calibrating the system. Refer to Calibrate the system (on page 13-5).

Figure 594: Input offset currents



Offset currents can also be generated externally from such sources as triboelectric (on page 13-16) and piezoelectric effects (on page 13-17). As shown in the next figure, 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 using the Formulator. For more information, refer to The Formulator (on page 6-290) and Analyze data using the Run tab.

Figure 595: External offset current



Triboelectric effects

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

The triaxial cables supplied with the SMU and preamplifier 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 triaxial cable creates some noise when subjected to vibration and expansion or contraction. Therefore, all connections should be kept short, away from temperature changes (which can 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 preamplifier as close as possible to the DUT.

NOTE

A temporary triboelectric current is generated when a triaxial 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.

CAUTION

Do not bend cables to make tight corners. Do not let long cables hang. Lay all cables on a flat surface.

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% relative humidity. Also be sure that all insulators are kept clean and free of contamination. If insulators become contaminated with, clean them thoroughly with a pure solvent such as methanol. To clean ionic contamination, use a de-ionized (DI) water wash.

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 for the current caused by dielectric absorption to dissipate.

Voltage burden

As shown in the next figure, the SMU or preamplifier 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 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:

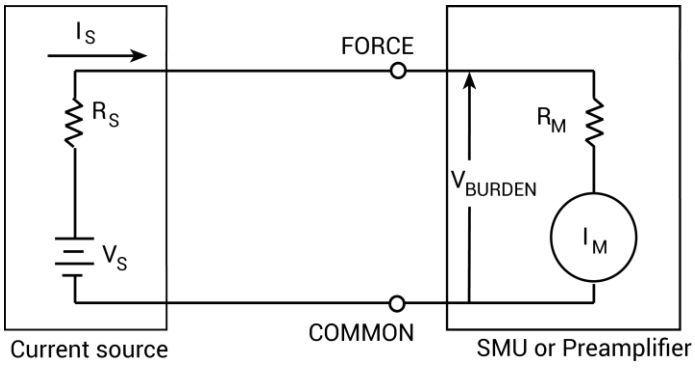
$$E = \frac{V_{BURDEN}}{R_S} \times 100$$

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.

Figure 596: 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 affects the noise performance of the SMU or preamplifier. 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. The next table summarizes minimum recommended source resistance values for various measurement ranges.

Minimum recommended source resistance values

Range	Minimum recommended source resistance
1 pA to 100 pA	1 GΩ to 100 GΩ
1 nA to 100 nA	1 MΩ to 100 MΩ
1 µA to 100 µA	1 kΩ to 100 kΩ
1 mA to 100 mA	1 Ω to 100 Ω

Source capacitance

DUT source capacitance also affects the noise performance of the preamplifier. 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 increases the voltage burden. For example, the source resistance values listed in the previous table results in a voltage burden between 1 mV and 1 V.

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 GΩ will result in an R_C time constant of approximately 100 ms. 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 (on page 3-32)).

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

Test system performance

When making a semiconductor I-V measurement, there is always a speed-noise trade-off. Even with given measurement settings, changes to the system configuration (such as cable length or adding a switch matrix) changes the measurement results. The 4200A-SCS has four settings to allow optimal I-V measurements. There are three fixed settings, fast, normal, and quiet, and a custom setting.

To achieve a low-noise measurement, the quiet setting is recommended. The trade-off is that measurement speed is slower 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 fixed settings are tuned to the 4200A-SCS for standard cable lengths connected to the DUT. In general, this should be sufficient to make good measurements. However, when extra long cables or a switch matrix are used in the system, these settings may not be adequate. A typical phenomenon is 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 takes 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 require a larger delay factor.

Interference

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 preamplifier 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.

To minimize electrostatic interference, you can use:

- **Shielding**: Possibilities include a shielded room, a shielded booth, shielding the sensitive circuit (test fixture), and 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 television or radio broadcast signals. It can also 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.

To minimize the effects of RFI:

- Keep DUT as far away from the RFI source as possible.
- Shield the test equipment, DUT, and test cables.
- In extreme cases, use a specially-constructed screen room to 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 the next figure shows a ground loop that is created by connecting both 4200A-SCS 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 the following figure. Note, however, that removing the COMMON-to-chassis link may result in oscillations (refer to Making stable measurements (on page 13-8)).

Figure 597: Ground loops

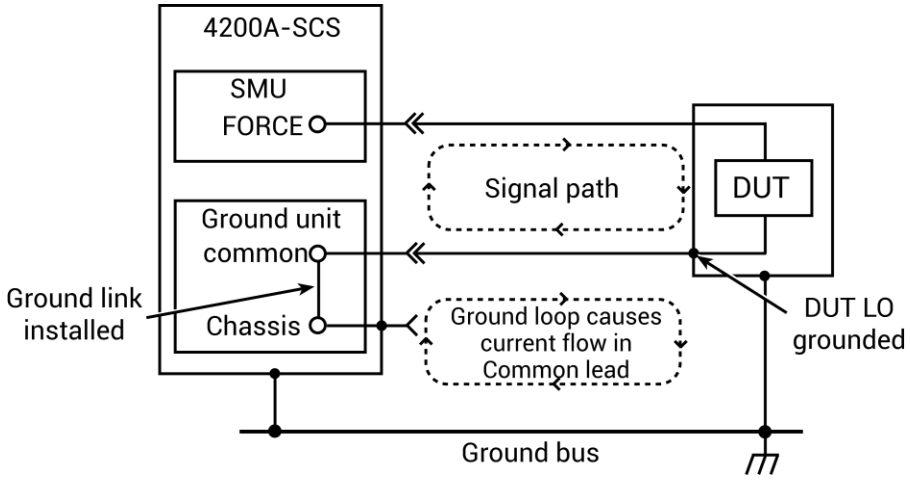
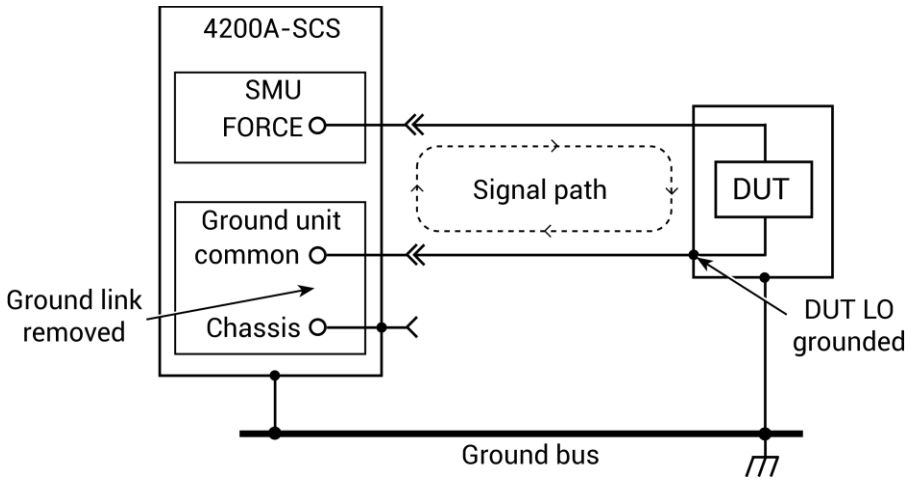


Figure 598: Eliminating ground loops



Section 14

LPT library function reference

In this section:

LPT library reference

The Keithley Instruments Linear Parametric Test Library (LPT library) is a high-speed data acquisition and instrument control software library. It is the programmer's lowest level of command interface to the system instrumentation. You can use the library commands to configure the relay matrix and instrumentation for parametric tests.

This section lists the commands included in the LPT library and describes how to use them. The descriptions include:

- A brief description of the command.
- Usage, which shows how the command should be organized and descriptions of each parameter. The parameters that you need to supply are shown in italics. For example, for the command `int delay(long n);`, replace *n* with the duration of the delay.
- Detailed information about the command.
- Examples that show a typical use of the command in a test sequence.

The following conventions are used when explaining the commands:

- All LPT library commands are case-sensitive and must be entered as lower case when writing program code.
- Period strings (`...`) indicate additional arguments or commands that can be added.
- Periods (`.`) indicate data not shown in an example because it is not necessary to help explain the specific command.

- A capital letter *X* in a command name indicates that you must select from a list of replacement suffixes. For example, in `forceX`, replace the *X* with either a `v` for voltage or `i` for current. The following is a table of possible suffixes, the parameter each represents, and the units used throughout the LPT library for that parameter.

Suffix	Parameter	Unit
`i`	Current	Amperes
`t`	Time	Seconds
`v`	Voltage	Volts
`f`	Frequency	Hertz

Lists of LPT library commands

These topics list the LPT library commands that are available in the 4200A-SCS. A brief description and links to full descriptions of each command are provided.

General operation commands

General operation commands include commands to control timing, execution, communications, and test status.

Command	Description
clrscn (on page 14-11)	Clears the measurement scan tables associated with a sweep.
clrtrg (on page 14-14)	Clears the user-selected voltage or current level that is used to set trigger points. This permits the use of the `trigXl` or `trigXg` command more than once with different levels in a single test sequence.
delay (on page 14-15)	Provides a user-programmable delay in a test sequence.
devint (on page 14-16)	Resets all active instruments in the system to their default states.
disable (on page 14-18)	Stops the timer and sets the time value to zero (0).
enable (on page 14-18)	Provides correlation of real time to measurements of voltage, current, conductance, and capacitance.
execut (on page 14-19)	Causes the system to wait for the preceding test sequence to be executed.
getinstattr (on page 14-19)	Returns configured instrument attributes.
getinstid (on page 14-21)	Returns the instrument identifier (ID) from the instrument name string.
getinstname (on page 14-21)	Returns the instrument name string from the instrument identifier (ID).
getlpterr (on page 14-21)	Returns the first LPT library error since the last `devint` command.
imeast (on page 14-22)	Forces a reading of the timer and returns the result.
inshld (on page 14-22)	Provided for compatibility with Model S400 LPT library.
kibcmd (on page 14-23)	Enables universal, addressed, and unaddressed GPIB bus commands to be sent through the GPIB interface.
kibdefclr (on page 14-24)	Defines the device-dependent command sent to an instrument connected to the GPIB interface.
kibdefdelete (on page 14-25)	Deletes all command definitions previously made with the `kibdefclr` (Keithley GPIB define device clear) and `kibdefint` (Keithley GPIB define device initialize) commands.
kibdefint (on page 14-26)	Defines a device-dependent command sent to an instrument connected to the GPIB interface.
kibrcv (on page 14-27)	Reads a device-dependent string from an instrument connected to the GPIB interface.
kibsnd (on page 14-28)	Sends a device-dependent command to an instrument connected to the GPIB interface.
kibspl (on page 14-29)	Serial polls an instrument connected to the GPIB interface.
kibsplw (on page 14-30)	Synchronously serial polls an instrument connected to the GPIB interface.
kspcfg (on page 14-31)	Configures and allocates a serial port for RS-232 communications.
kspdefclr (on page 14-32)	Defines a device-dependent character string sent to an instrument connected to a serial port.
kspdefdelete (on page 14-32)	Deletes all command definitions previously made with the `kspdefclr` (Keithley Serial Define Device Clear) and `kspdefint` (Keithley Serial Define Device Initialize) commands.
kspdefint (on page 14-33)	Defines a device-dependent character string sent to an instrument connected to a serial port.

Command	Description
ksprcv (on page 14-34)	Reads data from an instrument connected to a serial port.
kspsnd (on page 14-34)	Sends a device-dependent command to an instrument attached to a RS-232 serial port.
PostDataDouble (on page 14-35)	Posts double-precision floating point data from memory into the Clarius Analyze sheet.
PostDataInt (on page 14-37)	Posts an integer-type data point from memory to the Clarius Analyze sheet in the user test module and plots it on the graph.
PostDataString (on page 14-37)	Transfers a string from memory into the Clarius Analyze sheet in the user test module and plots it on the graph.
rdelay (on page 14-38)	Sets a user-programmable delay.
rtfary (on page 14-38)	Returns the force array determined by the instrument action.
savgX (on page 14-39)	Makes an averaging measurement for every point in a sweep.
scnmeas (on page 14-41)	Makes a single measurement on multiple instruments at the same time.
searchX (on page 14-42)	Used to determine the voltage or current required to get a current or voltage.
setmode (on page 14-45)	Sets instrument-specific operating mode parameters.
sintgX (on page 14-47)	Makes an integrated measurement for every point in a sweep.
smeasX (on page 14-49)	Allows a number of measurements to be made by a specified instrument during a `sweep`*X* command. The results of the measurements are stored in the defined array.
trigcomp (on page 14-50)	Causes a trigger when an instrument goes in or out of compliance.
trigXg, trigXl (on page 14-51)	Monitors for a predetermined level of voltage, current, or time.
tstdsl (on page 14-54)	Deselects a test station.
tstsel (on page 14-54)	Enables or disables a test station.

Math operation commands

Command	Description
kfpabs (on page 14-55)	Takes a user-specified positive or negative value and converts it into a positive value that is returned to a specified variable.
kfpadd (on page 14-55)	Adds two real numbers and stores the result in a specified variable.
kfpdiv (on page 14-56)	Divides two real numbers and stores the result in a specified variable.
kfpexp (on page 14-57)	Supplies the base of natural logarithms (e) raised to a specified power and stores the result as a variable.
kfplog (on page 14-57)	Returns the natural logarithm of a real number to the specified variable.
kfpmul (on page 14-58)	Multiplies two real numbers and stores the result as a specified variable.
kfpneg (on page 14-59)	Changes the sign of a value and stores the result as a specified variable.
kfppwr (on page 14-60)	Raises a real number to a specified power and assigns the result to a specified variable.
kfpsqrt (on page 14-61)	Performs a square root operation on a real number and returns the result to the specified variable.
kfpsub (on page 14-62)	Subtracts two real numbers and stores their difference in a specified variable.

SMU commands

Command	Description
adelay (on page 14-63)	Specifies an array of delay points to use with `asweepX` command calls.
asweepX (on page 14-63)	Generates a waveform based on a user-defined forcing array (logarithmic sweep or other custom forcing commands).
avgX (on page 14-66)	Makes a series of measurements and averages the results.
bmeasX (on page 14-67)	Makes a series of readings as quickly as possible. This measurement mode allows for waveform capture and analysis (within the resolution of the measurement instrument).
bsweepX (on page 14-69)	Supplies a series of ascending or descending voltages or currents and shuts down the source when a trigger condition is encountered.
devclr (on page 14-72)	Sets all sources to a zero state.
devint (on page 14-16)	Sets all sources to a zero state.
forceX (on page 14-74)	Programs a sourcing instrument to generate a voltage or current at a specific level.
getstatus (on page 14-76)	Returns the operating state of a specified instrument.
intgX (on page 14-78)	Performs voltage or current measurements averaged over a user-defined period (usually one AC line cycle).
limitX (on page 14-80)	Allows the programmer to specify a current or voltage limit other than the default limit of the instrument.
lorangeX (on page 14-81)	Defines the bottom autorange limit.
measX (on page 14-83)	Allows the measurement of voltage, current, or time.
mpulse (on page 14-85)	Uses a source-measure unit (SMU) to force a voltage pulse and measure both the voltage and current for exact device loading.
pulseX (on page 14-86)	Directs a SMU to force a voltage or current at a specific level for a predetermined length of time.
rangeX (on page 14-88)	Selects a range and prevents the selected instrument from autoranging.
rtfary (on page 14-38)	Returns the array of force values used during the subsequent voltage or frequency sweep.
segment_sweepX_list (on page 14-90)	Creates and returns up to a 4-segment linear sweep force table based on user-defined start, stop, and step values.
setauto (on page 14-91)	Re-enables autoranging and cancels any previous `rangeX` command for the specified instrument.
ssmeasX (on page 14-93)	Makes a series of readings until the change (delta) between readings is within a specified percentage.
sweepX (on page 14-95)	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.

PGU (pulse only) and PMU (pulse and measure) commands

In the LPT commands, the pulse-only module (4220-PGU) is referred to as VPU1, VPU2, and so on. The pulse-measure module (4225-PMU) is referred to as PMU1, PMU2, and so on. The 4210-CVU is referred to as CVU1, CVU2, and so on.

Note that the 4225-PMU and 4220-PGU support the PG2 commands.

Command	Description
dev_abort (on page 14-99)	PGU, PMU. Programmatically ends a test from within the user module (aborts a test) that was started with the `pulse_exec` command.
pmu_offset_current_comp (on page 14-103)	PMU. Collects offsets current constants from the 4225-PMU for offset compensation measurements.
PostDataDoubleBuffer (on page 14-101)	PMU. Posts PMU data retrieved from the buffer into the Clarius Analyze sheet (large data sets).
pulse_chan_status (on page 14-104)	PMU. Used to determine how many readings are stored in the data buffer.
pulse_conncomp (on page 14-105)	PMU. Enables or disables connection compensation.
pulse_exec (on page 14-107)	PGU, PMU. Used to validate the test configuration and start test execution.
pulse_exec_status (on page 14-109)	PGU, PMU. Used to determine if a test is running or completed.
pulse_fetch (on page 14-110)	PMU. Retrieves enabled test data and temporarily stores it in the data buffer.
pulse_float (on page 14-117)	PMU. Sets the state of the floating relay for the given pulse instrument
pulse_limits (on page 14-118)	PMU. Sets measured voltage and current thresholds at the DUT and sets the power threshold for each channel.
pulse_meas_sm (on page 14-119)	PMU. Configures spot mean measurements.
pulse_meas_timing (on page 14-121)	PMU. Sets the measurement windows.
pulse_meas_wfm (on page 14-123)	PMU. Configures waveform measurements.
pulse_measrt (on page 14-124)	PMU. Returns pulse source and measure data in pseudo real-time.
pulse_ranges (on page 14-125)	PGU, PMU. Sets the voltage pulse range and voltage/current measure ranges.
pulse_remove (on page 14-128)	PGU, PMU. Removes a pulse channel from the test.
pulse_sample_rate (on page 14-129)	PMU. Sets the measurement sample rate.
pulse_source_timing (on page 14-130)	PGU, PMU. Sets the pulse period, pulse width, rise time, fall time, and delay time.
pulse_step_linear (on page 14-132)	PGU, PMU. Configures the pulse stepping type.
pulse_sweep_linear (on page 14-135)	PGU, PMU. Configures the pulse sweeping type.
pulse_train (on page 14-138)	PGU, PMU. Configures the pulse card to output a pulse train using fixed voltage values.

Command	Description
rpm_config (on page 14-139)	PMU with 4225-RPM. Sends switching commands to the 4225-RPM.
seg_arb_sequence (on page 14-140)	PGU, PMU. Defines the parameters for a Segment Arb waveform pulse-measure sequence.
seg_arb_waveform (on page 14-144)	PGU, PMU. Creates a voltage segment waveform.
setmode (on page 14-145)	PMU. Sets the number of iterations for load-line effect compensation (LLEC) for the PMU. Also enables or disables offset current compensation.

Pulse source only (PG2) commands

In the LPT commands, the pulse-only module (4220-PGU) is referred to as VPU1, VPU2, and so on. The pulse-measure module (4225-PMU) is referred to as PMU1, PMU2, and so on. The 4210-CVU is referred to as CVU1, CVU2, and so on.

Note that the 4225-PMU and 4220-PGU support the PG2 commands.

Command	Description
arb_array (on page 14-148)	Used to define a full-arb waveform and name the file.
arb_file (on page 14-149)	Loads a waveform from an existing full-arb waveform file.
devclr (on page 14-72)	Sets all sources to a zero state.
devint (on page 14-16)	Resets all active instruments in the system to their default states.
getstatus (on page 14-76)	Returns the operating state of a specified instrument.
pg2_init (on page 14-155)	Resets the pulse card to the specified pulse mode (standard, full arb, or Segment Arb) and its default conditions.
pulse_burst_count (on page 14-156)	For the burst mode, this command sets the number of pulses to output during a burst sequence.
pulse_current_limit (on page 14-157)	Sets the current limit of the pulse card.
pulse_dc_output (on page 14-158)	Selects the DC output mode and sets the voltage level.
pulse_delay (on page 14-159)	Sets the delay time from the trigger to when the pulse output starts.
pulse_fall (on page 14-161)	Sets the fall transition time for the pulse output.
pulse_halt (on page 14-162)	Stops all pulse output from the pulse card.
pulse_init (on page 14-163)	Resets the pulse card to the default settings for the pulse mode that is presently selected.
pulse_load (on page 14-164)	Sets the output impedance for the load (DUT).
pulse_output (on page 14-165)	Sets the pulse output of a pulse card channel on or off.
pulse_output_mode (on page 14-166)	Sets the pulse output mode of a pulse card channel.
pulse_period (on page 14-167)	Sets the period for pulse output.
pulse_range (on page 14-168)	Sets a pulse card channel for low voltage (fast speed) or high voltage (slow speed).
pulse_rise (on page 14-169)	Sets the rise transition time for the pulse card pulse output.
pulse_ssrc (on page 14-171)	Controls the high-endurance output relay (HEOR) for each output channel of the PGU.
pulse_trig (on page 14-172)	Selects the trigger mode (continuous, burst, or trigger burst) and initiates the start of pulse output or arms the pulse card.
pulse_trig_output (on page 14-174)	Sets the output trigger on or off.

Command	Description
pulse_trig_polarity (on page 14-175)	Sets the polarity (positive or negative) of the pulse card output trigger.
pulse_trig_source (on page 14-176)	Sets the trigger source.
pulse_vhigh (on page 14-178)	Sets the pulse voltage high level.
pulse_vlow (on page 14-178, on page 14-180)	Sets the pulse voltage low value.
pulse_width (on page 14-182)	Sets the pulse width for pulse output.
seg_arb_define (on page 14-183)	Defines the parameters for a Segment Arb® waveform.
seg_arb_file (on page 14-186)	Used to load a waveform from an existing Segment Arb® waveform file.

CVU commands

Command	Description
adelay (on page 14-63)	Specifies an array of delay points to use with `asweep`*X* command calls.
asweepv (on page 14-188)	Does a DC voltage sweep using an array of voltage values.
bsweepX (on page 14-69)	Supplies a series of ascending or descending voltages or currents and shuts down the source when a trigger condition is encountered.
cvu_custom_cable_comp (on page 14-192)	Determines the delays needed to accommodate custom cable lengths.
devclr (on page 14-72)	Sets all sources to a zero state.
devint (on page 14-16)	Resets all active instruments in the system to their default states.
dsweepf (on page 14-195)	Performs a dual frequency sweep.
dsweepv (on page 14-197)	Performs a dual linear staircase voltage sweep.
forcev (on page 14-198)	Sets the DC bias voltage level.
getstatus (on page 14-199)	Returns various parameters pertaining to the state of the 4210-CVU.
measf (on page 14-200)	Returns the frequency sourced during a single measurement.
meast (on page 14-202)	Returns a timestamp referenced to a measurement or a system timer.
measv (on page 14-203)	Returns the DC bias voltage sourced during a single measurement.
measz (on page 14-204)	Makes an impedance measurement.
rangei (on page 14-205)	Selects an impedance measurement range.
rtfary (on page 14-89)	Returns the array of force values used during the subsequent voltage or frequency sweep.
segment_sweepX_list (on page 14-90)	Creates and returns up to a 4-segment linear sweep force table based on user-defined start, stop, and step values.
setauto (on page 14-207)	Selects the auto measure range.
setfreq (on page 14-208)	Sets the frequency for the AC drive.
setlevel (on page 14-209)	Sets the AC drive voltage level.
setmode (4210-CVU) (on page 14-209)	Sets operating modes specific to the 4210-CVU.
smeasf (on page 14-212)	Returns the frequencies used for a sweep.
smeasfRT (on page 14-213)	Returns the sourced frequencies (in real time) for a sweep.
smeast (on page 14-214)	Returns timestamps referenced to sweep measurements or a system timer.
smeastRT (on page 14-215)	Returns timestamps (in real time) referenced to sweep measurements or a system timer.
smeasv (on page 14-216)	Returns the DC bias voltages used for a sweep.
smeasvRT (on page 14-217)	Returns the sourced DC bias voltages (in real time) for a sweep.
smeasz (on page 14-217)	Performs impedance measurements for a sweep.
smeaszRT (on page 14-219)	Makes and returns impedance measurements for a voltage or frequency sweep in real time.
sweepf (on page 14-220)	Performs a frequency sweep.
sweepv (on page 14-222)	Performs a linear staircase DC voltage sweep.

Switch commands

Command	Description
addcon (on page 14-228)	Adds connections without clearing existing connections.
clrcon (on page 14-229)	Opens or de-energizes all device under test (DUT) pins and instrument matrix relays, disconnecting all crosspoint connections.
conpin (on page 14-230)	Connects pins and instruments.
conpth (on page 14-231)	Connects pins and instruments using a specific pathway.
cviv_config (on page 14-232)	Sends switching commands to the 4200A-CVIV Multi-Switch.
cviv_display_config (on page 14-233)	Configures the LCD display on the 4200A-CVIV Multi-Switch.
cviv_display_power (on page 14-234)	Sets the display state of the LCD display on the 4200A-CVIV.
devint (on page 14-16)	Resets all active instruments in the system to their default states.

LPT commands for general operations

General operation commands include commands to control timing, execution, communications, and test status.

clrscn

This command clears the measurement scan tables associated with a sweep.

Usage

```
int clrscn(void);
```

Details

When a single `sweepX` command is used in a test sequence, there is no need to program a `clrscn` command because the `execut` command clears the table.

The `clrscn` command is only required when multiple sweeps and multiple sweep measurements are used in a single test sequence.

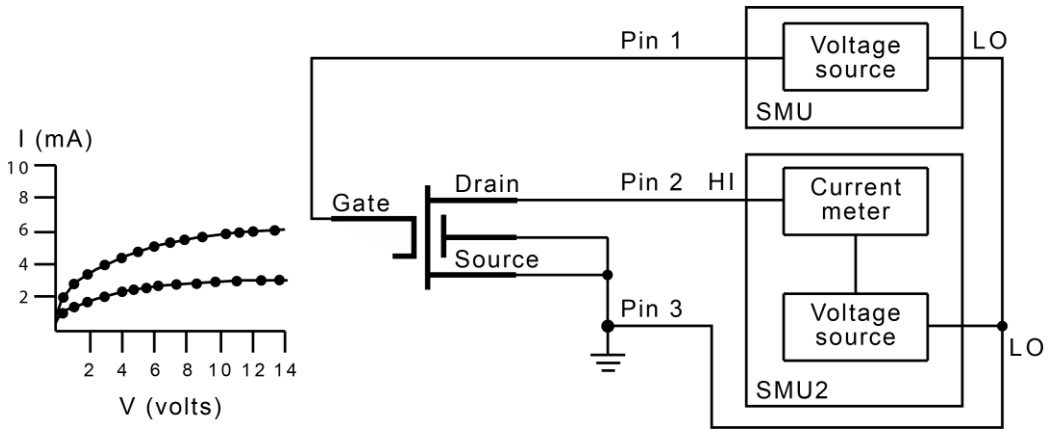
Example

```
double res1[14], res2[14];
.
conpin(SMU1, 1, 0);
conpin(SMU2, 2, 0);
conpin(GND, 3, 0);
forcev(SMU1, 4.0); /* Apply 4 V 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); /* Make */
 /* 14 measurements */
 /* over a range of 0 V to 14 V. */
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 V through 14 V. */
```

In this example, the `sweepX` command configures SMU2 to source a voltage that sweeps from 0 V through +14 V in 14 steps. The results of the first `sweepv` command are stored in an array called `res1`. Because of the `clrscn` command, the data and pointers associated with the first `sweepv` command are cleared. Then 5 V is forced to the gate, and the measurement process is repeated. Results from these second measurements are stored in an array called `res2`.

This example gets the measurement data needed to create a graph showing the gate voltage-to-drain current characteristics of a field-effect transistor (FET). 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 then with 5 V applied. The gate voltages are generated by SMU1.

Figure 599: Gate voltage-to-drain current characteristics



Also see

execut (on page 14-19)

sweepX (on page 14-95)

clrtrg

This command clears the user-selected voltage or current level that is used to set trigger points. This permits the use of the `trig`*`X`*`l` or `trig`*`X`*`g` command more than once with different levels in a single test sequence.

Usage

```
int clrtrg(void);
```

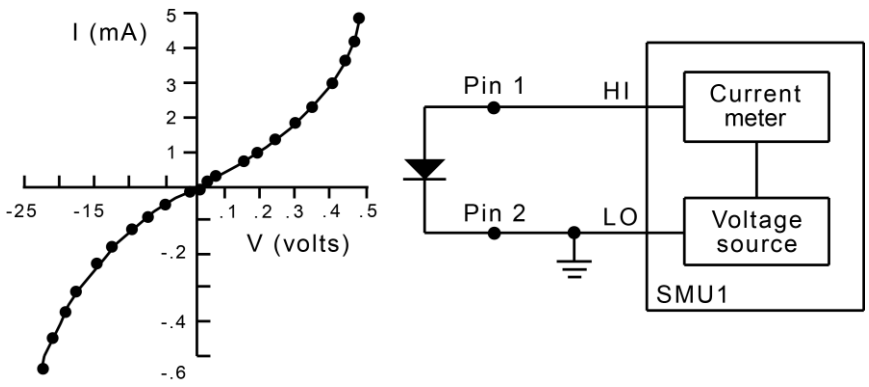
Details

The `search`*`X`*, `sweep`*`X`*, `asweep`*`X`*, or `bsweep`*`X`* command, each with different voltage or current levels, may be used repeatedly within a command if each is separated by a `clrtrg` command.

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, 5.0e-3); /* Output */
 /* 0 V to 0.5 V in 11 */
 /* 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, 5.00e-3); /* Output */
 /* 0 V to -30 V in 11 steps */
 /* each 5 ms in duration. */
```

This example collects data and creates a graph that shows the forward and reverse conduction characteristics of a diode. The `clrtrg` command allows multiple triggers to be programmed twice in the same test sequence. Each result is returned to a separate array.



Also see

asweepX (on page 14-63)
bsweepX (on page 14-69)
searchX (on page 14-42)
sweepX (on page 14-95)
trigXg, trigXl (on page 14-51)

delay

This command provides a user-programmable delay in a test sequence.

Usage

```
int delay(long n);
```

n	The duration of the delay in milliseconds

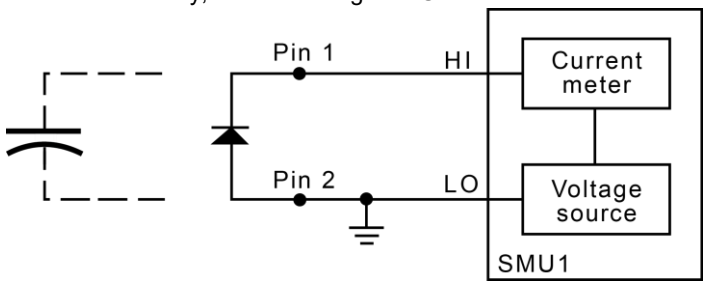
Details

The `delay` command can be called anywhere in the test sequence.

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. */
```

This example measures the leakage current of a variable-capacitance diode. 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. Because this type of diode has very high capacitance and low leakage current, a 20 ms delay is added. After the delay, current through SMU1 is measured and stored in the variable IR4.



Also see

rdelay (on page 14-38)

devint

This command resets all active instruments in the system to their default states.

Usage

```
int devint(void);
```

Details

Resets all active instruments in the system to their default states. It clears the system by opening all relays and disconnecting the pathways. Meters and sources are reset to their default states. Refer to the hardware manuals for the instruments in your system for listings of available ranges and the default conditions and ranges.

The `devint` command is implicitly called by the `execut` command.

To abort a running `pulse_exec` pulse test, see `dev_abort`.

`devint` does the following:

1. Clears all sources by calling `devclr`.
2. Clears the matrix crosspoints by calling `clrcon`.
3. Clears the trigger tables by calling `clrtrg`.
4. Clears the sweep tables by calling `clrscn`.
5. Resets GPIB instruments by sending the string defined with `kibdefint`.
6. Resets the active instrument cards.

Instrument cards are reset in the following order:

1. SMU instrument cards
2. CVU instrument cards
3. Pulse instrument cards (4225-PMU or 4220-PGU)

`devint` is implicitly called by `execut` and `tstdsl`. `devclr` is implicitly called by `clrcon`.

The SMUs return to the following states:

- 100 µA and 10 V ranges
- Autorange on
- Voltage source
- 0 V DCV bias

The 4210-CVU returns to the following states:

- 30 mV_{RMS} AC signal
- 0 V DCV bias
- 100 kHz frequency
- Autorange on
- Cable length compensation set to 0 m
- Open/Short/Load compensation disabled

The 4225-PMU or 4220-PGU returns to the following states:

- 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.
- 5 V and 10 mA ranges
- If in pulse mode:
  - Period of 1 µs
  - Transition Times (Rise and Fall) of 100 ns
  - Width of 500 ns
  - Voltage high and low of 0 V

  - Load of 50 Ω
- If in segmented arb mode, Start Voltage is 0 V
- If in arbitrary waveform mode, Table Length is 100

Also see

clrcon (on page 14-229)
clrscn (on page 14-11)
clrtrg (on page 14-14)
dev_abort (on page 14-99)
devclr (on page 14-72)
kibdefint (on page 14-26)

disable

This command stops the timer and sets the time value to zero (0).

Usage

```
int disable(int instr_id);
```

instr_id	The instrument identification code of the timer module (TIMER*n*)

Details

Timer reading is also stopped.

Sending `disable(TIMERn)` stops the timer and resets the time value to zero (0).

Also see

enable (on page 14-18)

enable

This command provides correlation of real time to measurements of voltage, current, conductance, and capacitance.

Usage

```
int enable(int instr_id);
```

instr_id	The instrument identification code of the timer module (TIMER*n*)

Details

Sending `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.

Also see

disable (on page 14-18)

execut

This command causes the system to wait for the preceding test sequence to be executed.

Usage

```
int execut(void);
```

Details

This command waits for all previous LPT library commands to complete and then sends the `devint` command.

Also see

devint (on page 14-16)

getinstattr

This command returns configured instrument attributes.

Usage

```
int getinstattr(int instr_id, char *attrstr, char *attrvalstr);
```

instr_id	The instrument identification code of the LPT library instrument
attrstr	The instrument attribute name string
attrvalstr	The value string of the requested attribute; see **Details**

Details

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.

If the attribute value string exists, the returned string will match one of the values shown in the Attribute value string column of the following table. If the requested attribute does not exist, the `attrvalstr` parameter is set to a null string.

Possible values for the `getinstattr` parameters are listed in the following table.

`getinstattr` **parameter values**

Instrument identification code	Attribute name string	Attribute value string
GPI*x*	GPIBADDR	1 to 30
	MODELNUM	GPI 2-terminal GPI 4-terminal
CMTR*x*	GPIBADDR	1 to 30
	MODELNUM	KI82 KI590 KI595 KI4284 KI4294
PGU*x*	GPIBADDR	1 to 30
	MODELNUM	KI3401 KI3402 HP8110 HP81110
SMU*x*	MODELNUM	KI4200 KI4210
MTRX1	MODELNUM	KI707 KI708
TF1	MODELNUM	KI8006 KI8007
	NUMOFPINS	12 72
PRBR1	NUMOFPINS	2 to 72
	MODELNUM	FAKE CC12K CM500 MANL MM40 PA200 MPI
CVU*x*	MODELNUM	KICVU4210
VPU*x* VPU*x*CH1 VPU*x*CH2	MODELNUM	KIVPU4220
PMU*x* PMU*x*CH1 PMU*x*CH2	MODELNUM	KIPMU4225
CVIV*x*	MODELNUM	KICVIV
GNDU	MODELNUM	GNDU

Also see

None

getinstid

This command returns the instrument identifier (ID) from the instrument name string.

Usage

```
int getinstid(char *instr_name, int *instr_id);
```

instr_name	The instrument name string
instr_id	The instrument identification code

Also see

None

getinstname

This command returns the instrument name string from the instrument identifier (ID).

Usage

```
int getinstname(int *instr_id, char *inst_name);
```

instr_id	The instrument identification code
inst_name	The returned instrument name string

Also see

None

getlpterr

This command returns the first LPT library error since the last `devint` command.

Usage

```
int getlpterr(void);
```

Details

This command returns the error code of the first error encountered since the last call to the `devint` command.

Also see

devint (on page 14-16)

imeast

This command forces a reading of the timer and returns the result.

Usage

```
int imeast(int instr_id, double *result);
```

instr_id	The instrument identification code of the device
result	The variable assigned to the measurement

Details

This command applies to all timers.Also see

None

inshld

Provided for compatibility with Model S400 LPT library.

Usage

```
int inshld(void);
```

Also see

None

kibcmd

This command enables universal, addressed, and unaddressed GPIB bus commands to be sent through the GPIB interface.

Usage

```
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 that contains the bytes to transfer over the GPIB interface

Details

These commands can consist of any command that is valid with the ATN line asserted, such as DCL, SDC, and GET. The following table lists these GPIB commands.

kibcmd does the following:

1. Asserts attention (ATN).
2. Sends byte string (command buffer).
3. De-asserts ATN.

GPIB command list

GPIB command	Data byte (Hex)	Comments
Universal		
LLO (local lockout) DCL (device clear) SPE (serial poll enable) SPD (serial poll disable)	11 14 18 19	Locks-out front panel controls. Returns instrument to default conditions. Enables serial polling. Disables serial polling.
Addressed		
SDC (selective device clear) GTL (go to local) GET (group execute trigger)	04 01 08	Returns instrument to default conditions. Sends go to local. Triggers instrument for reading.
Unaddressed		
UNL (unlisten) UNT (untalk) LAG (listen address group) TAG (talk address group) SCG (secondary command group)	3F 5F 20-3E 40-5E 60-7E	Removes all listeners from GPIB bus. Removes any talkers from GPIB bus. Place instrument at this primary address (0 through 30) in listen mode. Place instrument at this primary address (0 through 30) in talk mode. Place instrument at this secondary address (0 through 30) in listen mode.

Example

```
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 */
```

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.

Also see

None

kibdefclr

This command defines the device-dependent command sent to an instrument connected to the GPIB interface.

Usage

```
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 (0 to 30; the controller uses address 31)
sec_addr	The secondary address of the instrument (1 to 30; if the instrument device does not support secondary addressing, this parameter must be -1)
timeout	The GPIB timeout for the transfer 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 (the physical CLEAR string); a maximum of 1024 bytes is allowed

Details

This string is sent during any normal tester-based `devclr` command. It ensures that if the tester is calling the `devclr` command internally, any external GPIB device is cleared with the given string.

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 the `kibdefclr` or `kibdefint` command is the first string sent over the GPIB. The `devclr` (`kibdefclr`) strings are always sent before initialization.

The KIBLIB `devclr` strings are sent before the `devclr` and `devint` commands execute. This may be a problem when communicating with any Keithley-supported GPIB instruments. This may also have an effect on the `bsweep`*`X`* command, because the `bsweep`*`X`* command sends a call to the `devclr` command to clear active sources. It is not recommended to use GPIB instruments when performing tests with the `bsweep`*`X`* command.

Also see

bsweepX (on page 14-69)
devclr (on page 14-72)
devint (on page 14-16)
execut (on page 14-19)
kibdefint (on page 14-26)

kibdefdelete

This command deletes all command definitions previously made with the `kibdefclr` (Keithley GPIB define device clear) and `kibdefint` (Keithley GPIB define device initialize) commands.

Usage

```
int kibdefdelete(void);
```

Details

Once this command is issued, any previous definitions made using `kibdefclr` or `kibdefint` will no longer occur at `devint` or `devclr` time.

You can override this command by re-issuing the `kibdefint` and `kibdefclr` commands.

Also see

None

kibdefint

This command defines a device-dependent command sent to an instrument connected to the GPIB interface.

Usage

```
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 (0 to 30; the controller uses address 31)
sec_addr	The secondary address of the instrument (1 to 30; if the instrument device does not support secondary addressing, this parameter must be -1)
timeout	The GPIB timeout for the transfer 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 (the INITIALIZE string); a maximum of 1024 bytes is allowed

Details

This string is sent during any normal tester-based call to the `devint` command. It ensures that if the tester is calling the `devint` command internally, any external GPIB device is initialized with the rest of the known instruments.

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 the `kibdefclr` or `kibdefint` command is the first string sent over the GPIB. The `devclr` (`kibdefclr`) strings are always sent before initialization.

The KIBLIB `devclr` strings are sent before the `devclr` and `devint` commands execute. This may be a problem when communicating with any Keithley-supported GPIB instruments. This may also have an effect on the `bsweep`*X* command, because the `bsweep`*X* command sends a call to the `devclr` command to clear active sources. It is not recommended to use GPIB instruments when performing tests with the `bsweep`*X* command.

Also see

bsweepX (on page 14-69)
devclr (on page 14-72)
devint (on page 14-16)
execut (on page 14-19)
kibdefclr (on page 14-24)

kibrcv

This command reads a device-dependent string from an instrument connected to the GPIB interface.

Usage

```
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 (0 to 30; the controller uses address 31)
sec_addr	The secondary address of the instrument (1 to 30; 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 reading
timeout	The GPIB timeout for the transfer 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

Details

The `kibrcv` command receives a buffer from the GPIB interface by doing the following:

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 the device *rcv_buff* parameter 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.

The *rcv_size* parameter defines the maximum number of bytes physically allowed in the buffer. If the *rcv_size* parameter is greater than the byte string returned by the instrument, the device is short-cycled and only the maximum number of bytes is returned.

Also see

None

kibsnd

This command sends a device-dependent command to an instrument connected to the GPIB interface.

Usage

```
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 (0 to 30; the controller uses address 31)
sec_addr	The secondary address of the instrument (1 to 30; if the instrument device does not support secondary addressing, this parameter must be -1)
timeout	The GPIB timeout for the transfer 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

Details

The `kibsnd` command sends a buffer out through the GPIB interface by doing the following:

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 the *send_buff* parameter 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.

Also see

None

kibspl

This command serial polls an instrument connected to the GPIB interface.

Usage

```
int kibspl(int pri_addr, int sec_addr, unsigned int timeout,
   int *serial_poll_byte);
```

pri_addr	The primary address of the instrument (0 to 30; the controller uses address 31)
sec_addr	The secondary address of the instrument (1 to 30; if the instrument device does not support secondary addressing, this parameter must be -1)
timeout	The GPIB polling timeout in 100 ms units (for example, timeout = 40 = 4.0 s)
serial_poll_byte	The serial poll status byte returned by the device presently being polled

Details

The `kibspl` command does the following:

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 the *serial_poll_byte* parameter from the device (if data is available), else *serial_poll_byte* = 0 (indicating error; device not SRQing).
9. Assert ATN.
10. Send serial poll disable (SPD).
11. Send UNTalk (UNT).
12. Send UNListen (UNL).
13. De-assert ATN.

Also see

kibsplw (on page 14-30)

kibsplw

This command synchronously serial polls an instrument connected to the GPIB interface.

Usage

```
int kibsplw(int pri_addr, int sec_addr, unsigned int timeout, int *serial_poll_byte);
```

pri_addr	The primary address of the instrument (2 to 31)
sec_addr	The secondary address of the instrument (1 to 31; if the instrument device does not support secondary addressing, this parameter must be –1)
timeout	The GPIB polling timeout in 100 ms units (for example, a timeout of 40 = 4.0 s)
serial_poll_byte	The serial poll status byte variable name returned by the device presently being polled

Details

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.

The `kibsplw` command does the following:

1. Waits with timeout for general SRQ assertion on the GPIB.
2. Calls the `kibspl` command.

Also see

kibspl (on page 14-29)

kspcfg

This command configures and allocates a serial port for RS-232 communications.

Usage

```
int kspcfg(int port, int baud, int databits, int parity, int stopbits, int flowctl);
```

port	The RS-232 port to be used; only port 1 is supported
baud	The transmission rate to be used; valid rates are 2400, 4800, 9600, 14400, and 19200 baud
databits	The number of data bits to 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; 1 or 2
flowctl	Determines the type of flow control to be used: 0 (no flow control), 1 (XON/XOFF flow control), or 2 (hardware)

Details

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

```
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 */
```

This example uses `kspcfg` to set port 1 to 19200 baud, 8 data bits, odd parity, 1 stop bit, and XON/XOFF flow control.

Also see

None

kspdefclr

This command defines a device-dependent character string sent to an instrument connected to a serial port.

Usage

```
int kspdefclr(int port, double timeout, double delay, int buffsize, char *buffer);
```

port	The RS-232 port to be used; only port 1 is supported; this port must have been configured for communications with the `kspcfg` command
timeout	The serial communications timeout (0 to 600 s)
delay	The amount of time to delay after sending the string to the serial device (0 to 600 s)
buffsize	The length of the string to send to the serial device
buffer	A character string that contains the data to send to the serial device

Details

This string is sent during the normal tester `devclr` 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.

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 before normal (for example, a SMU) instrument `devclr` and `devint` execution.

Also see

kspcfg (on page 14-31)

kspdefdelete

This command deletes all command definitions previously made with the `kspdefclr` (Keithley Serial Define Device Clear) and `kspdefint` (Keithley Serial Define Device Initialize) commands.

Usage

```
int kspdefdelete( void );
```

Details

Once this command is issued, any previous definitions made using `kspdefclr` or `kspdefint` will no longer occur at `devint` or `devclr` time.

You can override this command by re-issuing the original `kspdefint` and `kspdefclr` commands.

Also see

None

kspdefint

This command defines a device-dependent character string sent to an instrument connected to a serial port.

Usage

```
int kspdefint(int port, double timeout, double delay, int buffsize, char *buffer);
```

port	The RS-232 port to be used; only port 1 is supported; this port must have been configured for communications with the `kspcfg` command
timeout	The serial communications timeout (0 to 600 s)
delay	The amount of time to delay after sending the string to the serial device (0 to 600 s)
buffsize	The length of the string to send to the serial device
buffer	A character string that contains the data to send to the serial device

Details

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.

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 before normal (for example, a SMU) instrument `devclr` and `devint` execution.

Also see

kspcfg (on page 14-31)

ksprcv

This command reads data from an instrument connected to a serial port.

Usage

```
int ksprcv(int port, char terminator, double timeout, int
   rcvsize, int *rcv_len, char *rcv_buffer);
```

port	The RS-232 port to be used; only port 1 is supported; this port must have been 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: 0 to 600 s
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

Also see

kspcfg (on page 14-31)

kspsnd

This command sends a device-dependent command to an instrument attached to a RS-232 serial port.

Usage

```
int kspsnd( int port, double timeout, int cmdlen, char *cmd);
```

port	The RS-232 port to be used; only port 1 is supported; this port must have been configured for communications with the `kspcfg` command
timeout	The serial communications timeout: 0 to 600 s
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

Also see

None

PostDataDouble

This command posts double-precision floating point data from memory into the Clarius Analyze sheet.

Usage

```
int PostDataDouble(char *ColName, double *array);
```

ColName	Column name for the data array in the Clarius Analyze sheet
array	An array of data values for the Clarius Analyze sheet

Pulsers

4225-PMU

Pulse mode

Standard and Segment Arb

Details

You can use the `PostDataDouble` command to post double-precision floating point data into the Clarius Analyze sheet. Up to 65,535 points (rows) of data can be posted into the Analyze sheet. These commands are used after one measurement point is finished and a data value is assigned to the corresponding output variable.

NOTE

If you do not need to analyze or manipulate the test data before posting it into the Analyze sheet, you can use `smeasxrt` for CVUs or `pulse_measrt` for PMUs, which retrieves all the test data in pseudo real-time and automatically posts it into the Analyze sheet.

Example

```
// 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]);
}
```

Posts spot mean measurement data into the Clarius Analyze sheet.
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 `pulse_meas_sm` to configure the spot mean test.
The code:

- 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 Clarius Analyze sheet. Low measurement data is not posted into the sheet.

Also see

Enabling real time plotting for UTMs (on page 8-17)
PostDataDoubleBuffer (on page 14-101)
pulse_fetch (on page 14-110)
pulse_meas_sm (on page 14-119)
pulse_measrt (on page 14-124)
smeasfRT (on page 14-213)
smeastRT (on page 14-215)
smeasvRT (on page 14-217)
smeaszRT (on page 14-219)

PostDataInt

This command posts an integer-type data point from memory to the Clarius Analyze sheet in the user test module and plots it on the graph.

Usage

```
PostDataInt(char *variableName, int *variableValue);
```

variableName	The variable name
variableValue	The value of the variable to be transferred

Details

The first parameter is the variable name, defined as `char *`. For example, if the output variable name is `DrainI`, then `DrainI` (with quotes) is first parameter.

The second parameter is the value of the variable to be transferred. For example, if `DrainI[10]` is transferred, then you call `PostDataInt("DrainI", DrainI[10])`.

Also see

None

PostDataString

This command transfers a string from memory into the Clarius Analyze sheet in the user test module and plots it on the graph.

Usage

```
PostDataString(char *variableName, int *variableValue);
```

variableName	The variable name
variableValue	The value of the variable to be transferred

Details

The first parameter is the variable name. For example, if the output variable name is `DrainI`, then `DrainI` (with quotes) is first parameter.

The second parameter is the value of the variable to be transferred. For example, if `DrainI[10]` is transferred, then you call `PostDataString("DrainI", DrainI[10])`.

Also see

None

rdelay

This command sets a user-programmable delay.

Usage

```
int rdelay(double n);
```

n	The delay duration in seconds

Example

```
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. */
```

This example measures the leakage current of a variable-capacitance diode. 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. Because of this, a 20 ms delay is added. After the delay, current through SMU1 is measured and stored in the variable `ir4`.

Also see

None

rtfary

This command returns the force array determined by the instrument action.

Usage

```
int rtfary(double *results);
```

results	The floating point array where the force values are stored

Details

This command eliminates the need to calculate the forced array in the application.

When used with the `bsweepX`, `sweepX`, or `searchX` commands, you can determine the exact forced value for each point in the sweep.

When the test sequence is executed, the sweep command initiates the first step of the voltage or current sweep. The sweep then logs the force point that the buffer specified by the `rtfary` command.

Locate the `rtfary` command before the sweep. The number of data points returned by the `rtfary` command is determined by the number of force points generated by the sweep.

Example

Refer to the examples for the `smeasX` and `sweepX` commands.

Also see

smeasX (on page 14-49)
sweepX (on page 14-95)

savgX

This command makes an averaging measurement for every point in a sweep.

Usage

```
int savgi(int instr_id, double *result, long count, double delay);
int savgv(int instr_id, double *result, long count, double delay);
```

instr_id	The instrument identification code of the measuring instrument
result	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

Details

This command creates an entry in the measurement scan table. During any of the sweeping commands, a measurement scan is done for every force point in the sweep. During each scan, a measurement is 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.

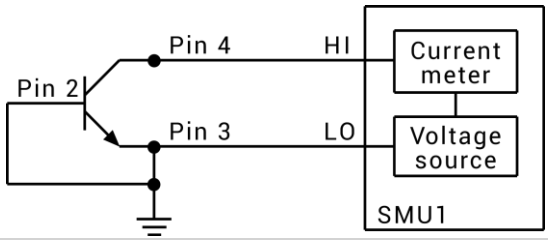
The `savgX` command sets up the new scan table entry to make an averaging measurement. The measurement results are stored in the array specified by the *result* parameter. Each time a measurement scan is made, a new measurement result is stored at the next location in the *result* array. If the scan table is not cleared, performing multiple sweeps will continue adding new measurement results to the end of the array. Care must be taken that the *result* array is large enough to hold all measurements made before the scan table is cleared. The scan table is cleared by an explicit call to the `clrscn` command or implicitly when the `devint` or `execut` command is called.

When making each averaged measurement, the number of actual measurements specified by the *count* parameter is made on the instrument at the interval specified by the *delay* parameter, and then the average is calculated. This average is the value that is stored in the results array.

Example

```
double res1[26];
.
.
conpin(GND, 3, 2, 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 V */
 /* to -50 V over 25 steps.*/
```

This example gets the measurement data that is needed to create a graph that shows 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`.



Also see

clrscn (on page 14-11)
devint (on page 14-16)

scnmeas

This command makes a single measurement on multiple instruments at the same time.

Usage

```
int scnmeas(void);
```

Details

This command behaves like a single point sweep. It makes a single measurement on multiple instruments at the same time. Any forcing or delaying must be done before calling `scnmeas`.

`smeas`*X*, `sintg`*X*, or `savg`*X* must be used to set up result arrays just as is done for a sweep call. Each call to `scanmeas` adds one element to the end of each array.

Calls to `scnmeas` may be mixed with calls to `sweep`*X*, and all results are appended to the result arrays in the same way multiple `sweep`*X* calls behave.

Also see

savgX (on page 14-39)
sintgX (on page 14-47)
smeasX (on page 14-49)

searchX

This command is used to determine the voltage or current required to get a current or voltage. It is useful in finding initial threshold points such as junction breakdown or transistor turn on.

Usage

```
int searchi(int instr_id, double min_val, double max_val, long iterate_no, double
    iterate_time, double *result);
int searchv(int instr_id, double min_val, double max_val, long iterate_no, double
    iterate_time, double *result);
```

instr_id	The instrument identification code of the sourcing instrument
min_val	The lower limit of the source range
max_val	The upper limit of the source range
iterate_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 the `searchv` command, or current, with the `searchi` command, applied during the last search operation

Details

The `trigXg` or `trigXl` command must be used with the `searchX` command. Triggers and the `searchX` command together initiate a search operation consisting of a series of steps referred to as iterations. During each iteration, the following events occur:

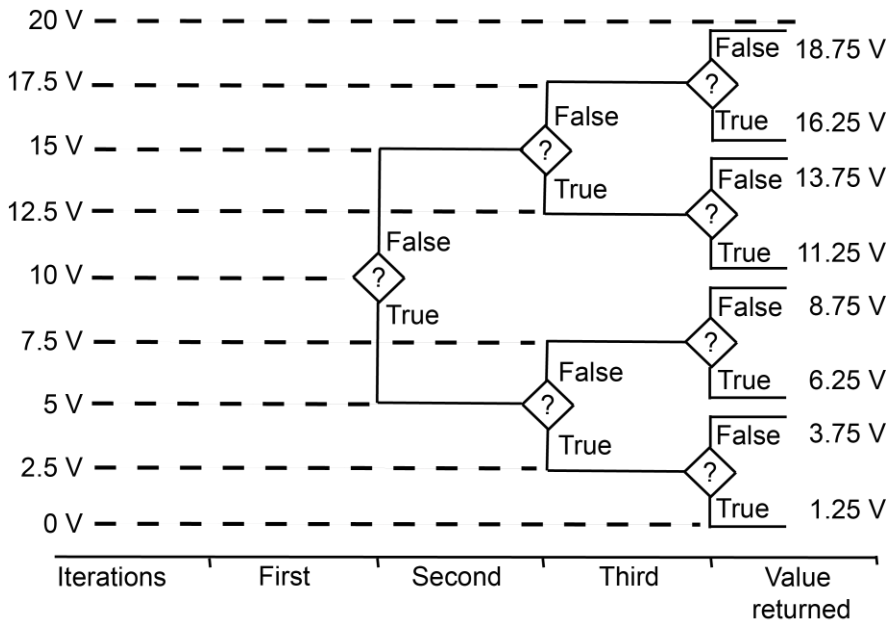
- A voltage or current is applied to a circuit node of the device under test (DUT).
- All triggers are evaluated.
- If the triggers evaluate true, the source value is moved toward the value specified in the *min_val* parameter. If the triggers do not evaluate true, the source value is moved toward the value specified in the *max_val* parameter. The source range is then divided in half for the next iteration.

A total of 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. 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 600: Minimum and maximum source values



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 command initiates a current or voltage from a source, its placement in a test sequence is critical. Therefore:

- Call the `limitX` and `rangeX` commands before the `searchX` command when all three refer to the same instrument.
- Call the `trigXg` or `trigXl` command before the `searchX` command.

The search operation determines the source voltage or current required at one circuit node to generate a 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 used by the instrument.

$$\frac{\text{voltage or current range}}{2^{(\text{iteration}+1)}}$$

For example, assume the minimum and maximum values of the source range are from 0 V to 20 V, and the number of iterations is 16. The 20 V level automatically initiates a source-measure unit (SMU) 20 V source range. As a result, the resolution of the final source voltage returned is:

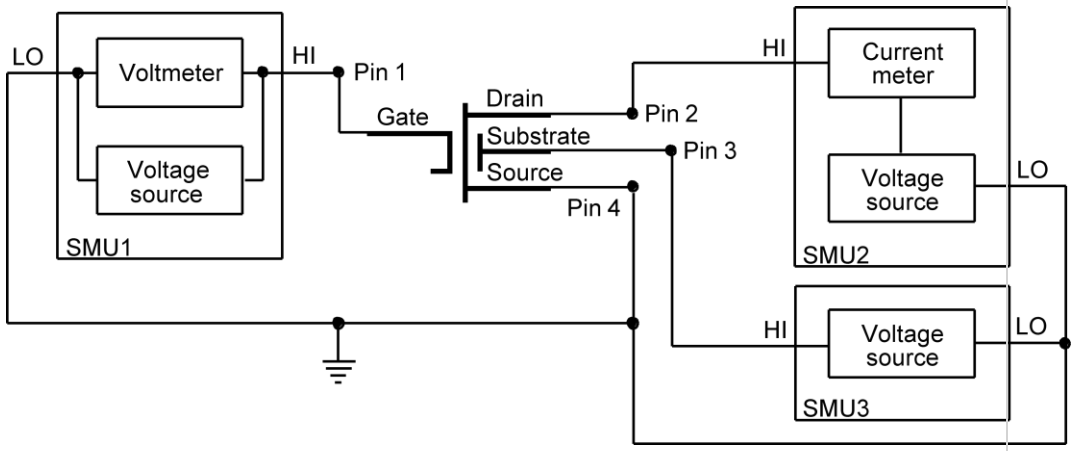
$$\frac{20}{2^{(16+1)}} = 1.2 \text{ mV}$$

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 rangeX (on page 14-88) for recommended command order.

Example

```
double ssbiasv, vgs1, vds1;
.
conpin(SMU1, 1, 0);
conpin(SMU2, 2, 0);
conpin(SMU3, 3, 0);
conpin(GND, 4, 0);
trigig(SMU2, +1.0E-6); /* Set trigger point for 1 uA. */
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, 1.0E-3, &vgs1); /* Set */
 /* for 8 steps from 0.6 to */
 /* 1.7 V at 1 ms.*/
 /* per iteration; return the */
 /* result to vgs1. */
```

This 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 the `searchv` command. 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 in the variable `vgs1`.



Also see

None

setmode

This command sets instrument-specific operating mode parameters.

Usage

```
int setmode(int instr_id, long modifier, double value);
```

instr_id	The instrument identification code of the instrument being operated on
modifier	The instrument-specific operating characteristic to change; see **Details**
value	The specified value of the operating parameter

Details

The `setmode` command allows you to control certain instrument-specific operating characteristics.

A special instrument ID named `KI_SYSTEM` is used to set operating characteristics of the system.

The following table describes `setmode` *`modifier`* parameters that are supported for `KI_SYSTEM`.

modifier	***value***	**Comment**

`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 make 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`.

The following `KI_SYSTEM` *`modifier`* parameters are accepted, but do no operations in the 4200A-SCS. They are included for compatibility so that existing S530 or S600 programs that use `setmode` can be ported to the 4200A-SCS without generating errors.

- `KI_MX_DEFMODE`
- `KI_HICURRENT`
- `KI_CC_AUTO`
- `KI_CC_SRC_DLY`
- `KI_CC_COMP_DLY`
- `KI_CC_MEAS_DLY`

The following `setmode` *`modifier`* parameters are supported for SMU instruments.

modifier	***value***	**Comment**
`KI_INTGPLC`	<*value*> (in units of line cycles)	Specifies the integration time the SMU will use for the `intg`*X* and `sintg`*X* commands. The default `devint` value is 1.0. The valid range is 0.01 to 10.0.

modifier	***value***	**Comment**
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.
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 is enforced by the SMU. 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.
KI_LIM_INDCTR	Any	Controls the measure value that is returned if the SMU is at its programmed limit. The devint default is SOURCE_LIMIT (7.0e22). NOTE: The SMU always returns INST_OVERRANGE (1.0e22) 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 the SMU returns 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.
KI_OUTP_RELAY_STATE	KI_OUTP_HIZ KI_OUTP_NORM	Only available if there are no preamplifiers. KI_OUTP_HIZ sets the state to high impedance (open). KI_OUTP_NORM sets the state to normal (closed, force V 0).

The following SMU *modifier* parameters are accepted, but do no operations in the 4200A-SCS. They are included for compatibility so that existing S530 or S600 programs that use setmode can be ported to the 4200A-SCS without generating errors.

- KI_IMTR
- KI_VMTR

Also see

None

sintgX

This command makes an integrated measurement for every point in a sweep.

Usage

```
int sintgi(int instr_id, double *result);
int sintgv(int instr_id, double *result);
```

instr_id	The instrument identification code of the measuring instrument
result	The floating point array where the results are stored

Details

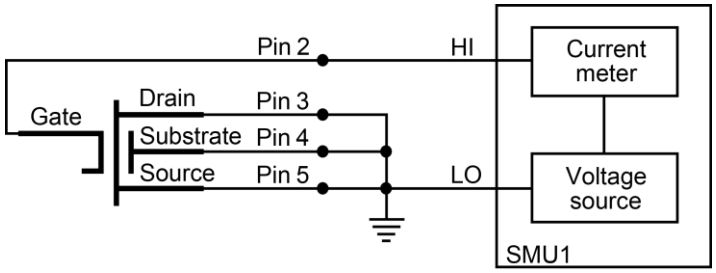
Use this command to create an entry in the measurement scan table. During any of the sweeping commands, a measurement scan is performed for every force point in the sweep. During each scan, a measurement is made for every entry in the scan table. The measurements are made in the same order in which the entries were made in the scan table.

The `sintgX` command sets up the new scan table entry to make an integrated measurement. The measurement results are stored in the array, specified by the *result* parameter. Each time a measurement scan is made, a new measurement result is stored at the next location in the results array. If the scan table is not cleared, making multiple sweeps will continue to add new measurement results to the end of the array. Care must be taken that the results array is large enough to hold all measurements that are made before the scan table is cleared. The scan table is cleared by an explicit call to the `clrscn` command or implicitly when the `devint` or `execut` command is called.

Example

```
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 20 nA. */
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 1 ms in */
. /* duration. */
```

This example collects information on the low-level gate leakage current of a metal-oxide field-effect transistor (MOSFET). Sixteen integrated measurements are made as the voltage is increased from 0 V to 25 V.



Also see

clrscn (on page 14-11)
devint (on page 14-16)
execut (on page 14-19)
sweepX (on page 14-95)

smeasX

This command allows a number of measurements to be made by a specified instrument during a `sweepX` command. The results of the measurements are stored in the defined array.

Usage

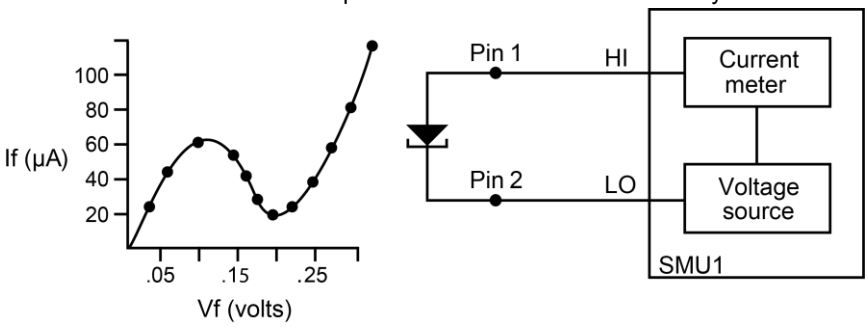
```
int smeasi(int instr_id, double *result);
int smeast(int instr_id, double *result);
int smeasv(int instr_id, double *result);
```

instr_id	The instrument identification code of the measuring instrument
result	The floating point array that stores the results

Details

This command is used to create an entry in the measurement scan table. During any of the sweep functions, a measurement scan is done for every force point in the sweep. During each scan, a measurement is 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.

The `smeasX` command sets up the new scan table entry to make an ordinary measurement. The measurement results are stored in the array specified by the *result* parameter. Each time a measurement scan is made, a new measurement result is stored at the next location in the *result* array. If the scan table is not cleared, doing multiple sweeps continues adding new measurement results to the end of the array. Care must be taken that the results array is large enough to hold all measurements that are made before the scan table is cleared. The scan table is cleared by an explicit call to the `clrscn` command or implicitly when the `devint` or `execut` command is called.This example determines the measurement data needed to create a graph showing the negative resistance characteristics of a tunnel diode. SMU1 generates a voltage ramp ranging from 0 to 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 named `resi`.



Also see

clrscn (on page 14-11)
devint (on page 14-16)
execut (on page 14-19)
sweepX (on page 14-95)

trigcomp

This command causes a trigger when an instrument goes in or out of compliance.

Usage

```
int trigcomp(int instr_id, int mode);
```

instr_id	The instrument identification code the trigger is set to
mode	Specifies whether to trigger when an instrument is in or out of compliance: ■ 1: Trigger when in compliance ■ 0: Trigger when out of compliance

Details

This command monitors 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.

Also see

None

trigXg, trigXl

This command monitors for a predetermined level of voltage, current, or time.

Usage

```
int trigig(int instr_id, double value);
int trigil(int instr_id, double value);
int trigtg(int instr_id, double value);
int trigtl(int instr_id, double value);
int trigvg(int instr_id, double value);
int trigvl(int instr_id, double value);
```

instr_id	The instrument identification code of the monitoring instrument
value	The voltage, current, or time specified as the trigger point; this trigger point value is reached when either of the following occurs: ■ The measured value is equal to or greater than the value argument of the `trigXg` command ■ The measured value is less than the value argument of the `trigXl` command

Details

The `trigXl` and `trigXg` commands are used with the `searchX` command or with one of the sweep measurement commands: `smeasX`, `sintgX`, or `savgX`.

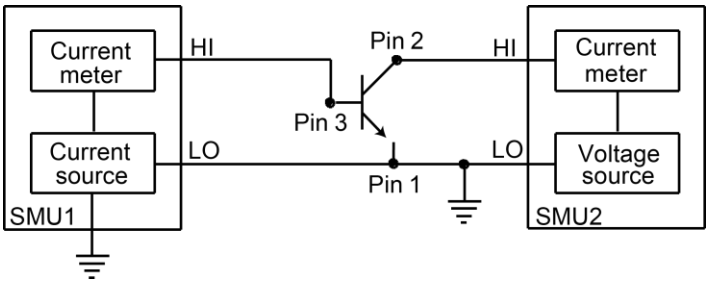
- The `trigXg` or `trigXl` command provides the `sweepX` command the digital feedback to allow for the increase or decrease in sourcing values.
- The `trigXl` and `trigXg` commands must be located before any associated `searchX` commands.
- Triggers are not automatically reset by the `searchX` or `sweepX` command. A single call to the `trigXl` or `trigXg` command can be followed by two or more calls to the *searchX* or `sweepX` commands.

The specified trigger point is automatically cleared when a `clrtrg`, `execut`, or `devint` command is executed.

Example 1

```
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 */
 /* current of 5 mA. */
searchi(SMU1, 5.0E-5, 2.0E-4, 15, 1.0E-3, &res22); /* Generate */
    /* a current ranging */
 /* from 50 uA to 200 uA in */
 /* 15 iterations. Return the */
 /* current resulting from the */
 /* last iteration as res22. */
```

This example uses the `trigig` and `searchi` commands 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-emitter voltage supplied by SMU2 is defined by the variable `vcc8`. The `searchi` command generates the base current from SMU1. This current ranges between 50 mA and 200 mA in 15 iterations. The `trigig` command continuously monitors the current through SMU1. The base current supplied by SMU1 is stored as the result `res22`.

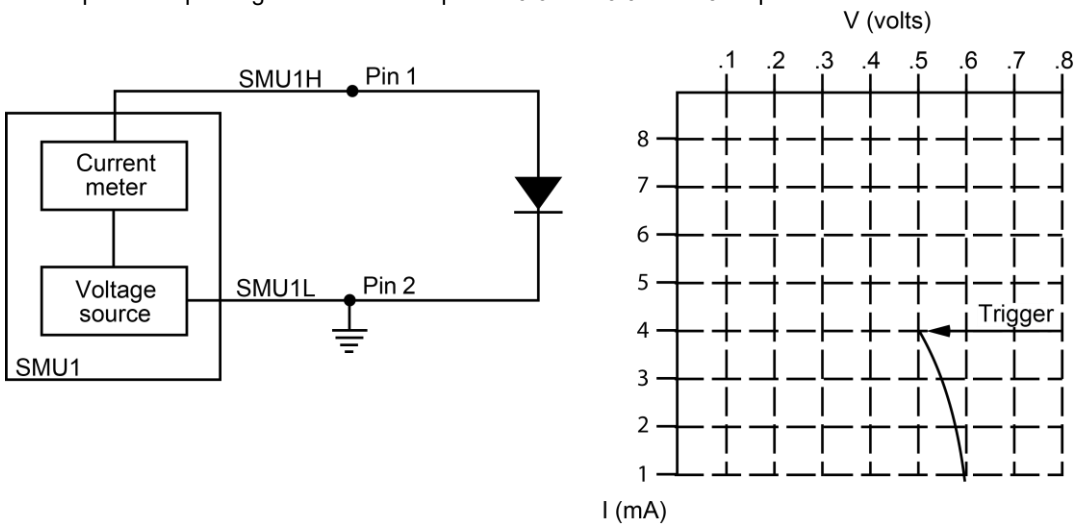


Example 2

```
double res1[20];
.
.
conpin(SMU1, 1, 0);
conpin(GND,2, 0);
trigil(SMU1, +4.0E-3); /* If less than +4 mA, */
 /* stop ramping. */

smeasi(SMU1, res1); /* Measure current at each of */
 /* the 19 levels; return */
 /* results to the res1 array. */
sweepv(SMU1, 0.0, 0.6, 18, 1.00E-3); /* Generate */
    /* 0.0 V to 0.6 V */
 /* in 18 steps. */
```

This example sets up and generates a sweep from 0.6 V to 0.0 V in 19 steps.



Also see

savgX (on page 14-39)
searchX (on page 14-42)
sintgX (on page 14-47)
smeasX (on page 14-49)
sweepX (on page 14-95)

tstdsl

This command deselects a test station.

Usage

```
tstdsl(void);
```

Details

To relinquish control of an individual test station, a new test station must be selected using `tstsel` before any subsequent test control commands are run.

The `tstdsl` command has the same effect as the `tstsel(0)` command.

NOTE

`tstdsl` is not required for use in a user test module (UTM).

Example

```
tstdsl( );    /* Disables test station.*/
```

Also see

tstsel (on page 14-54)

tstsel

This command enables or disables a test station.

Usage

```
tstsel(int x);
```

x	The test station number: `0` or `1`

Details

`tstsel` is normally called at the beginning of a test program.

`tstsel(1)` selects the first test station and loads the instrumentation configuration.

NOTE

The `tstsel` command is not required for use in a user test module (UTM).

Also see

tstdsl (on page 14-54)

LPT commands for math operations

The following commands provide math operations.

kfpabs

This command takes a user-specified positive or negative value and converts it into a positive value that is returned to a specified variable.

Usage

```
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 is stored

Example

```
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*/
```

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`.

Also see

None

kfpadd

This command adds two real numbers and stores the result in a specified variable.

Usage

```
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* is stored

Details

The values referenced by *x* and *y* are summed and the result is stored in the location pointed to by *z*. If an overflow occurs, the result is `±Inf`. If an underflow occurs, the result is zero (0).

Example

```
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. */
.
.
```

This example adds the data in `res1` to the data in `res2`. The result is stored in the `resia` variable.

Also see

None

kfpdiv

This command divides two real numbers and stores the result in a specified variable.

Usage

`int kfpdiv(double *x, double *y, double *z);`

x	The dividend
y	The divisor
z	A variable where the result of *x*/*y* is stored

Details

The value referenced by *x* is divided by the value referenced by *y*. The result is stored in the location pointed to by *z*. If an overflow occurs, the result is `±Inf`. If an underflow occurs, the result is zero (0).

Example

```
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. */
.
.
```

This example divides the data in `res1` by the data in `res2`. The result is stored in the `resia` variable.

Also see

None

kfpexp

This command supplies the base of natural logarithms (e) raised to a specified power and stores the result as a variable.

Usage

```
int kfpexp(double *x, double *z);
```

x	The exponent
z	The variable where the result of e^x is stored

Details

e raised to the power of the value referenced by *x* is stored in the location pointed to by *z*. If an overflow occurs, the result is ±Inf. If an underflow occurs, the result is zero (0).

Example

```
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. */
.
.
```

In this example, kfpexp raises the base of natural logarithms to the power specified by the exponent res4. The result is stored in res4e.

Also see

None

kfplog

This command returns the natural logarithm of a real number to the specified variable.

Usage

```
int kfplog(double *x, double *z);
```

x	A variable containing a floating point number
z	A variable where the result of ln (*x*) is stored

Details

This command returns a natural logarithm, not a common logarithm. The natural logarithm of the value referenced by *x* is 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 is generated and the result is NaN (not a number).

Example

```
double res1, logres;
.
.
measv(SMU1, &res1);/* Measure SMU1; store in res1. */
kfplog(&res1, &logres);/* Convert res1 to a natural */
/* LOG and store in logres. */
.
```

This example calculates the natural logarithm of a real number (`res1`). The result is stored in `logres`.

Also see

None

kfpmul

This command multiplies two real numbers and stores the result as a specified variable.

Usage

```
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* is stored

Details

The value referenced by *x* is multiplied by the value referenced by *y*, and the result is stored in the location pointed to by *z*. If an overflow occurs, the result is ±Inf. If an underflow occurs, the result is zero (0).

Example

```
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. */
.
.
```

This example multiplies variables `res1` and `res2`. The result is stored in the variable `pwr2`.

Also see

None

kfpneg

This command changes the sign of a value and stores the result as a specified variable.

Usage

```
int kfpneg(double *x, double *z);
```

x	A variable containing the number to be converted
z	A variable where the result of -*x* is stored

Details

If the value is positive, it is converted to a negative; if the value is negative, it is converted to a positive.

Example

```
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. */
.
```

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.

Also see

None

kfppwr

This command raises a real number to a specified power and assigns the result to a specified variable.

Usage

```
int kfppwr(double *x, double *y, double *z);
```

x	A variable that contains a floating point number
y	A variable that contains the exponent
z	A variable where the result of x^y is stored

Details

The value referenced by *x* is raised to the power of the value referenced by *y*, and the result is stored in the location pointed to by *z*. If an overflow occurs, the result is ±Inf. If an underflow occurs, the result is zero (0).

If *x* points to a negative number, a power of a negative number error is generated, and the result returned is -Inf.

If *x* points to a value of zero (0) and y points to a negative number, a divide by zero (0) error is generated, and the result returned is +Inf.

If *x* points to a value of 1.0, the result is 1.0, regardless of the exponent.

Example

```
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. */
.
```

Raises the variable res2 by the power of three. The result is stored in pwres2.

Also see

None

kfpsqrt

This command performs a square root operation on a real number and returns the result to the specified variable.

Usage

```
int kfpsqrt(double *x, double *z);
```

x	A variable that contains a floating point number
z	A variable where the result, the square root of x, is stored

Details

The square root of the value referenced by *x* is stored in the location pointed to by *z*.

If *x* points to a negative number, a square root of negative number error is generated, and the result is NaN (not a number).

Example

```
double res1, sqres2;
.
.
measv(SMU1, &res1);/* Measure SMU1; store result */
./* in res1. */
kfpsqrt(&res1, &sqres2);/* Find square root of res1; */
/* return result to sqres2. */
.
```

This example converts a real number (res1) into its square root. The result is stored in sqres2.

Also see

None

kfpsub

This command subtracts two real numbers and stores their difference in a specified variable.

Usage

```
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* is stored

Details

The value referenced by *y* is subtracted from the value referenced by *x*. The result is stored in the location pointed to by *z*. If an overflow occurs, the result is ±Inf. If an underflow occurs, the result is zero (0).

Example

```
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. */
```

This example subtracts res2 from res1. The result is returned to diff2.

Also see

None

LPT commands for SMUs

The following information explains the commands in the LPT library for the SMUs.

adelay

This command specifies an array of delay points to use with `asweepX` command calls.

Usage

```
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

Details

The delay is specified in units of seconds, with a resolution of 1 ms. The minimum delay is 0 s.

Each delay in the array is added to the delay specified in the `asweepX` command. For example, if the array contains four delays (0.04 s, 0.05 s, 0.06 s, and 0.07 s) and the delay specified in the `asweepX` command is 0.1 s, then the resulting delays are 0.14 s, 0.15 s, 0.16 s, and 0.17 s.

Also see

asweepX (on page 14-63)

asweepX

This command generates a waveform based on a user-defined forcing array (logarithmic sweep or other custom forcing commands).

Usage

```
int asweepi(int instr_id, long num_points, double delay_time, double *force_array);
int asweepv(int instr_id, long num_points, double delay_time, double *force_array);
```

instr_id	The instrument identification code of the sourcing instrument
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

Details

The `asweepX` command is used with the `smeasX`, `sintgX`, or `savgX` commands.

The `trigXl` or `trigXg` command can also be used with the `asweepX` command. 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 `asweepX` command. 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 because the source is held at a constant output.

The `asweepv` and `asweepi` commands are sourcing-type commands. When called, an automatic limit is imposed on the sourcing device. Refer to the limitX (on page 14-80) command for additional information.

The maximum number of times data is measured (using the `smeas`*`X`*, `sintg`*`X`*, or `savg`*`X`* command) is determined by the `num_points` argument in the `asweep`*`X`* command. A one-dimensional result array with the same number of data elements as the selected value of the *`num_points`* parameter must be defined in the test program.

When multiple calls to the `asweep`*`X`* command are executed in the same test sequence, the `smeas`*`X`*, `sintg`*`X`*, or `savg`*`X`* arrays are loaded sequentially. This appends the measurements from the second `asweep`*`X`* command to the previous results. If the arrays are not dimensioned correctly, access violations occur. The measurement table remains intact until the `devint` or `clrscn` command is executed.

Defining new test sequences using the `smeas`*`X`*, `sintg`*`X`*, or `savg`*`X`* command appends the command to the active measure list. Previous measures are still defined and will be used. The `clrscn` command is used to eliminate previous buffers for the second sweep. Using the `smeas`*`X`*, `sintg`*`X`*, and `savg`*`X`* commands after calling the `clrscn` or `execut` command causes the appropriate new measures to be defined and used.

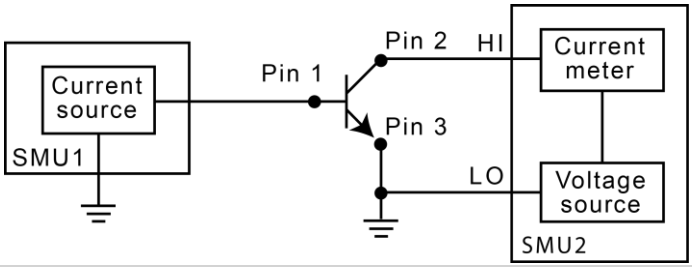
Note that changing the source mode of the SMU can modify the measure range. If the sourcing mode is changed from voltage to current sourcing (or from current to voltage sourcing), the measure range may be changed to minimize variations in the SMU output level. See rangeX (on page 14-88) for recommended command order.

If `adelay` is called before `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

```
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. */
```

This 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 the `asweepi` command. The collector current applied by SMU2 is measured 10 times by the `smeasi` command. The data gathered is then stored in the *`icmeas`* array.



Also see

limitX (on page 14-80)
savgX (on page 14-39)
sintgX (on page 14-47)
smeasX (on page 14-49)
trigXg, trigXI (on page 14-51)

avgX

This command makes a series of measurements and averages the results.

Usage

```
int avgi(int instr_id, double *result, long stepno, double steptime);
int avgv(int instr_id, double *result, long stepno, double steptime);
```

instr_id	The instrument identification code of the measuring instrument
result	The variable assigned to the result of the measurement
stepno	The number of steps averaged in the measurement (1 to 32,767)
steptime	The interval in seconds between each measurement; the minimum practical time is approximately 2.5 ms

Details

The `avgX` command is used primarily to get measurements when:

- The device under test (DUT) being tested acts in an unstable manner.
- Electrical interference is higher than can be tolerated if the `measX` command is used.

The programmer specifies the number of samples and the duration between each sample.

After this command 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 commands that return multiple results are more efficient than performing multiple measurement commands.

The `rangeX` command directly affects the operation of the `avgX` command. The use of the `rangeX` command prevents the addressed instrument from automatically changing ranges. This can result in an overrange condition similar to what 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 the `rangeX` command is not 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` command is used to select the range. For autoranging, the `avgi` or `avgv` commands triggers a needed range change before the measurement is made. See Compliance limits (on page 3-4) for details.

Example

```
double leakage;

.
.
limiti(SMU1, 1.0e-06); /* Limit the maximum current */
 /* to 1 µA */
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 */
```

This example illustrates how to use the `avgX` command to make five current readings and return the average of the measurements to the variable *leakage*.

Also see

measX (on page 14-83)
rangeX (on page 14-88)

bmeasX

This command makes a series of readings as quickly as possible. This measurement mode allows for waveform capture and analysis (within the resolution of the measurement instrument).

Usage

```
int bmeasi(int instr_id, double *result, long numrdg,

   double delay, int timerid, double *timerdata);
int bmeasv(int instr_id, double *result, long numrdg,
   double delay, int timerid, double *timerdata);
```

instr_id	The instrument identification code of the measuring instrument
result	The result name of the array to receive readings; the array must be large enough to hold the 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)
timerdata	The array used to receive the time points at which the readings were made; if *timerID* = `0`, the timer is not read and this array is not updated; if used, the array must be large enough to hold the readings

Details

This command collects data using the presently selected range. The measurement range is typically the same as the force range. If you need a different range, you must change the measurement range before calling the `bmeasX` command.

When used with the time module, the measurements and the times for each measurement are stored. The specific timer is defined in the command, and the time array is returned with the *timerdata* array.

Example 1

```
double irange, volts, rdng[5], timer[5];
:
.
.
enable(TIMER1); /* Enable the timer module. */
.
.
conpin(GND, 11, 0); /* Make connections. */
conpin(SMU3, 14, 0);
.
.
forcev(SMU3, volts); /* Perform the test. */
measi(SMU3, &irange); /* Set the I range of the SMU based */
rangei(SMU3, irange); /* on the initial measurement. */
.
forcev(SMU3, volts);
bmeasi(SMU3, rdng, 5, 0.0001, TIMER1, timer); /* gather a block of    measurements
   */
 /* I measurement of 5 */
 /* readings using SMU3 with */
 /* 100 us delay between */
 /* readings, using TIMER1 with */
 /* time data labeled timer. */
```

This example shows how the `bmeas`*X* command is used with a timer. Each measurement is associated with a timestamp. This timestamp marks the interval when each reading is made. This information is useful when determining how much time was required to obtain a specific reading.

Example 2

```
double volts, rdng[5];
:
.
conpin(GND, 11, 0); /* Make connections. */
conpin(SMU3, 14, 0);
.
forcev(SMU3, volts); /* Perform the test. */
.
bmeasi(SMU3, rdng, 5, 0, 0, 0); /* Block current measurement */
 /* of 5 readings using SMU3. */
```

This example shows how the `bmeas`*X* command is used without a timer. When used without a timer, the returned measurement is not associated with a timestamp.

Also see

None

bsweepX

This command supplies a series of ascending or descending voltages or currents and shuts down the source when a trigger condition is encountered.

Usage

```
int bsweepi(int instr_id, double startval, double endval, long num_points, double
    delay_time, double *result);
int bsweepv(int instr_id, double startval, double endval, long num_points, double
    delay_time, double *result);
```

instr_id	The instrument identification code of the sourcing instrument
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 the *startval* and *endval* parameters (1 to 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

Details

`bsweepi` is only available for SMUs.

The `bsweepX` command is used with the `trigXg`, `trigXl`, or `trigcomp` command. These trigger commands 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, the `bsweepX` command terminates the sweep and clears all sources by executing a `devclr` command internally. The standard `sweepX` command continues to force the last value. This is useful for device characterization curves but can cause problems when used in device breakdown conditions.

The `bsweepX` command can also be used with the `smeasX`, `sintgX`, `savgX`, or `rtfary` command. Measurements are stored in a one-dimensional array in the order in which they were made.

The system maintains a measurement scan table consisting of devices to test. This table is maintained using calls to the `smeasX`, `sintgX`, `savgX`, or `clrscn` command. As multiple calls to `sweepX` commands are made, these commands are appended to the measurement scan table. Measurements are made after the time programmed by the *`delay_time`* parameter has elapsed at the beginning of each `bsweepX` command step.

When multiple calls to the `bsweepX` command are executed in the same test sequence, the arrays defined by calls to the `smeasX`, `sintgX`, or `savgX` command are all loaded sequentially. The results from the second call to the `bsweepX` command are appended to the results of the previous `bsweepX` command call. 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`, `execut`, or `clrscn` command completes.

Defining new test sequences using the `smeasX`, `sintgX`, or `savgX` command adds the command to the active measure list. The previous measurements are still defined and used; however, previous measurements for the second sweep can be eliminated by calling the `clrscn` command. New measurements are defined and used by calling the `smeasX`, `sintgX`, or `savgX` command after a `clrscn` command.

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 rangeX (on page 14-88) for recommended command order.

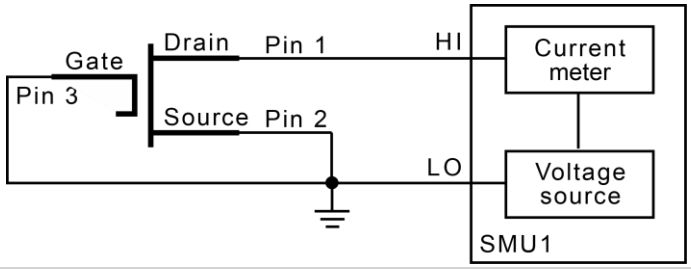
NOTE

It is recommended that you do not use GPIB instruments when doing sweeps with the `bsweepX` command. Refer to kibdefint (on page 14-26) for additional information.

Example

```
double bvdss;
.
.
conpin(SMU1, 1, 0);
conpin(GND, 2, 3, 0);
limiti(SMU1, 100e-6); /* Define the I limit for the device. */
rangei(SMU1, 100e-6); /* Select a fixed range */
 /* measurement. */
trigil(SMU1, -10e-6); /* Set the trigger point to -10 uA. */
bsweepv(SMU1, 10.0, 50.0, 40, 10.0e-3, &bvdss); /* Sweep */
 /* from 10 V to 50 V in 40 */
 /* steps with 10 ms settling */
 /* time per step. */
```

This example measures the drain to source breakdown voltage of a field-effect transistor (FET). A linear voltage sweep is generated from 10.0 V to 50.0 V by SMU1 using the `bsweepv` command. The breakdown current is set to 10 mA by using the `trigil` command. The voltage at which this current is exceeded is stored in the variable `bvdss` and is returned to the application processor by the `execut` command.



Also see

clrscn (on page 14-11)
devclr (on page 14-72)
execut (on page 14-19)
rtfary (on page 14-38)
savgX (on page 14-39)
sintgX (on page 14-47)
smeasX (on page 14-49)
sweepX (on page 14-95)
trigXg, trigXl (on page 14-51)
trigcomp (on page 14-50)

devclr

This command sets all sources to a zero state.

Usage

```
int devclr(void);
```

Details

This command clears all sources sequentially in the reverse order from which they were originally forced. Before clearing all Keithley supported instruments, GPIB-based instruments are cleared by sending all strings defined with the `kibdefclr` command.`devclr` is implicitly called by `clrcon`, `devint`, `execut`, and `tstdsl`.

For C-V testing, this command turns off the DC bias voltage.

Also see

clrcon (on page 14-229)
devint (on page 14-16)
execut (on page 14-19)
kibdefclr (on page 14-24)
tstdsl (on page 14-54)

devint

This command resets all active instruments in the system to their default states.

Usage

```
int devint(void);
```

Details

Resets all active instruments in the system to their default states. It clears the system by opening all relays and disconnecting the pathways. Meters and sources are reset to their default states. Refer to the hardware manuals for the instruments in your system for listings of available ranges and the default conditions and ranges.

The `devint` command is implicitly called by the `execut` command.

To abort a running `pulse_exec` pulse test, see `dev_abort`.

`devint` does the following:

1. Clears all sources by calling `devclr`.
2. Clears the matrix crosspoints by calling `clrcon`.
3. Clears the trigger tables by calling `clrtrg`.
4. Clears the sweep tables by calling `clrscn`.
5. Resets GPIB instruments by sending the string defined with `kibdefint`.
6. Resets the active instrument cards.

Instrument cards are reset in the following order:

1. SMU instrument cards
2. CVU instrument cards
3. Pulse instrument cards (4225-PMU or 4220-PGU)

`devint` is implicitly called by `execut` and `tstdsl`. `devclr` is implicitly called by `clrcon`.

The SMUs return to the following states:

- 100 µA and 10 V ranges
- Autorange on
- Voltage source
- 0 V DCV bias

The 4210-CVU returns to the following states:

- 30 mV_{RMS} AC signal
- 0 V DCV bias
- 100 kHz frequency
- Autorange on
- Cable length compensation set to 0 m
- Open/Short/Load compensation disabled

The 4225-PMU or 4220-PGU returns to the following states:

- 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.
- 5 V and 10 mA ranges
- If in pulse mode:
  - Period of 1 µs
  - Transition Times (Rise and Fall) of 100 ns
  - Width of 500 ns
  - Voltage high and low of 0 V

  - Load of 50 Ω
- If in segmented arb mode, Start Voltage is 0 V
- If in arbitrary waveform mode, Table Length is 100

Also see

clrcon (on page 14-229)
clrscn (on page 14-11)
clrtrg (on page 14-14)
dev_abort (on page 14-99)
devclr (on page 14-72)
kibdefint (on page 14-26)

forceX

This command programs a sourcing instrument to generate a voltage or current at a specific level.

Usage

```
int forcei(int instr_id, double value);
int forcev(int instr_id, double value);
```

instr_id	The instrument identification code
value	The level of the bipolar voltage or current forced in volts or amperes

Details

The `forcev` and `forcei` commands generate either a positive or negative voltage, as directed by the sign of the value argument. With both `forcev` and `forcei` commands:

- 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.

The `forcev` command accepts both `CMTR1H` and `CMTR1L` for the *instr_id* parameter to support differential CVU biasing. By forcing one polarity on CMTR1H and an opposite polarity on CMTR1L, total bias can be up to 60 V, centered in relationship to ground. Note that it is not possible to exceed ± 30 V in relationship to ground.

When using the `limitX`, `rangeX`, and `forceX` commands on the same source at the same time in a test sequence, call the `limitX` and `rangeX` commands before the `forceX` command. See Compliance limits (on page 3-4) for details.

The ranges of currents and voltages available from a voltage or current source vary with the instrument type. For more detailed information, refer to the 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 4200A-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:

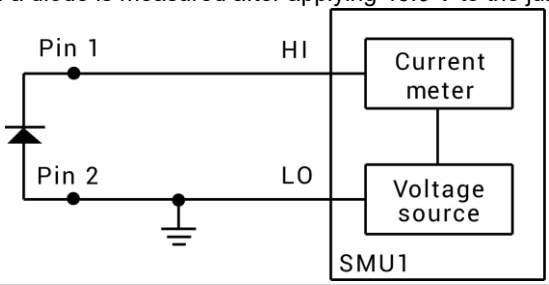
- A `measi` call, which causes the 4200A-SCS to autorange to a higher current limit.
- A `rangei` call to an appropriate fixed current, which results in a fixed current limit.

Note that changing the source mode of the source-measure unit (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 rangeX (on page 14-88) for recommended command order.

Example

```
double ir12;
.
.
conpin(2, GND, 0);
conpin(SMU1, 1, 0);
limiti(SMU1, 2.0E-4); /* Limit 1 mA to 200 uA. */
forcev(SMU1, 40.0); /* Apply 40.0 V. */
measi(SMU1, &ir12); /* Measure leakage; */
 /* return results to ir12. */
```

The reverse bias leakage of a diode is measured after applying 40.0 V to the junction.



Also see

None

getstatus

This command returns the operating state of a specified instrument.

Usage

```
int getstatus(int instr_id, long parameter, double *result);
```

instr_id	The instrument identification code
parameter	The parameter of query; see **Details**
result	The data returned from the instrument; the `getstatus` command returns one item

Details

If you see the `UT_INVLDPRM` invalid parameter error returned from the `getstatus` command, it indicates that the status item parameter is illegal for this device. The requested status code is invalid for the selected device.

A list of supported `getstatus` command values for *parameter* for a source-measure unit (SMU) and a pulse card (VPU) are provided in the following tables.

No status values are provided for measurement-specific conditions.

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	Compliance status of last reading	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	Range compliance status of last reading	Returns 1 if in range compliance

Supported pulse 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
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

Also see

getinstid (on page 14-21)

intgX

This command performs voltage or current measurements averaged over a user-defined period (usually one AC line cycle).

Usage

```
int intgi(int instr_id, double *result);
int intgv(int instr_id, double *result);
```

instr_id	The instrument identification code of the measuring instrument, such as SMU1
result	The variable assigned to the result of the measurement

Details

The 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.

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` command, the default integration time is set to 1 PLC. For 60 Hz 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.

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 by using the `setmode` command.

`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 in the test sequence before the associated `intgX` command.

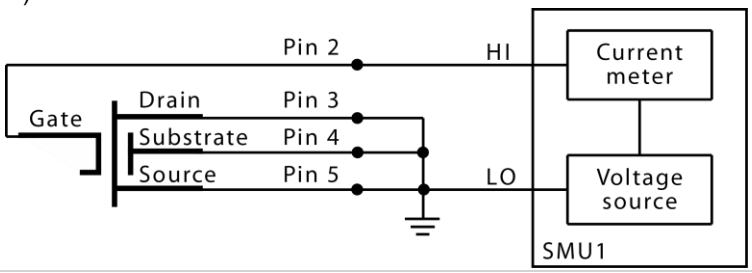
In general, measurement commands that return multiple results are more efficient than sending multiple measurement commands.

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` command is used to select the range. For autoranging, `intgi` or `intgv` triggers a needed range change before the measurement is made. See Compliance limits (on page 3-4) for details.

Example

```
double idss;
.
.
conpin(GND, 5, 4, 3, 0);
conpin(SMU1, 2, 0);
limiti(SMU1, 2.0E-8); /* Limits to 20.0 nA. */
rangei(SMU1, 2.0E-8); /* Select range for 20.0 nA */
forcev(SMU1, 25.0); /* Apply 25 V to the gate. */
intgi(SMU1, &idss); /* Measure gate leakage; */
 /* return results to idss. */
```

This example measures the relatively low leakage current of a metal-oxide semiconductor field-effect transistor (MOSFET).



Also see

devint (on page 14-16)
measX (on page 14-83)
rangeX (on page 14-88)
setmode (on page 14-45)

limitX

This command allows the programmer to specify a current or voltage limit other than the default limit of the instrument.

Usage

```
int limiti(int instr_id, double limit_val);
int limitv(int instr_id, double limit_val);
```

instr_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; see **Details**

Details

The parameter *limit_val* is bidirectional. For example, the command `limitv(SMU1, 10.0)` limits the voltage of the current source SMU1 to ±10.0 V. The command `limiti(SMU1, 1.5E-3)` limits the current of the voltage source SMU1 to ±1.5 mA.

Use the `limiti` command to limit the current of a voltage source. Use the `limitv` command 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 command must be called in the test sequence before the associated `force`*X*, `pulse`*X*, `bsweep`*X*, `sweep`*X*, or `search`*X* command is used to generate the voltage or current. The `limit`*X* command also sets the top measurement range of an autoranged measurement.

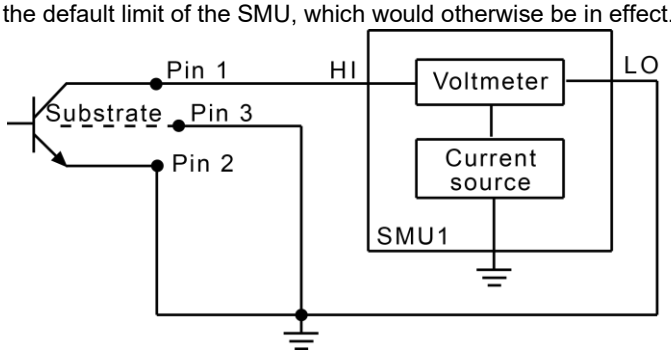
The limits set within a particular test sequence are cleared when the `devint` or `execut` command is called.

If you need a voltage limit greater than 20 V at a source-measure unit (SMU) that has been set to force zero current, call the `measv` command to set the SMU to autorange to a higher range, or use the `rangev` command to set a higher voltage range. Similarly, if you need a current limit of greater than 10 mA at a SMU that has been set to force zero volts, call the `measi` command to set the SMU to autorange to a higher range or use the `rangev` command to set a higher current range.

Example

```
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 the DUT. */
measv(SMU1, &vbceo); /* Measure breakdown voltage; */
. /* return results to vbceo. */
.
```

This example measures the breakdown voltage of a device. The limit is set at 150 V. This limit is necessary to override the default limit of the SMU, which would otherwise be in effect.



Also see

bsweepX (on page 14-69)
pulseX (on page 14-86)
devint (on page 14-16)
execut (on page 14-19)
forceX (on page 14-74)
measX (on page 14-83)
rangeX (on page 14-88)
searchX (on page 14-42)
sweepX (on page 14-95)

lorangeX

This command defines the bottom autorange limit.

Usage

```
int lorangei(int instr_id, double range);
int lorangev(int instr_id, double range);
```

instr_id	The instrument identification code
range	The value of the instrument range, in volts or amperes

Details

The `lorangeX` command is used with autoranging to limit the number of range changes, which saves test time.

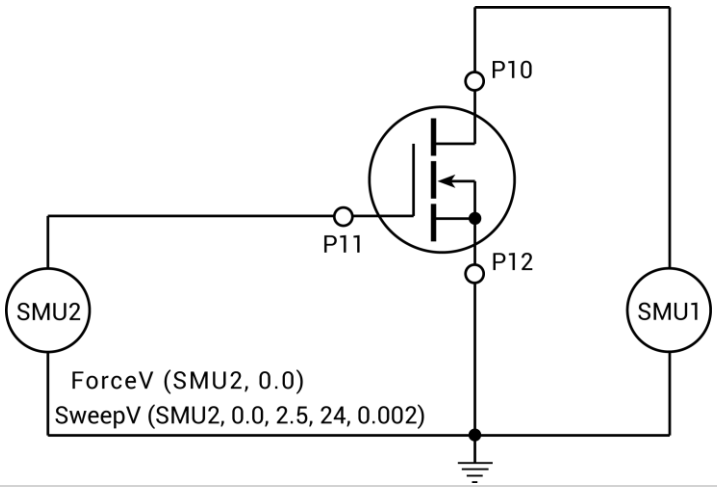
If the instrument is on a range lower than the one specified by the `lorangeX` command, the range is changed. The 4200A-SCS automatically provides any settling delay for the range change that may be necessary due to this potential range change.Once defined, the `lorangeX` command is in effect until a `devclr`, `devint`, `execut`, or another `lorangeX` command executes.

Example

```
double idatrg[25];

.
.
conpin(SMU1, 10, 0);
conpin(SMU2, 11, 0):
conpin(12, GND, 0);
lorangei(SMU1, 2.0E-6); /* Select 2 uA as minimum */
 /* range during autoranging. */
smeasi(SMU1, idatvg); /* Set up sweep measurement */
 /* of IDS. */
sweepv(SMU2, 0.0, 2.5, 24, 0.002); /* Sweep */
 /* gate from 0 V to 2.5 V. */
```

This example illustrates how you would select the bottom autorange limit.



Also see

devclr (on page 14-72)
devint (on page 14-16)
execut (on page 14-19)

measX

This command allows the measurement of voltage, current, or time.

Usage

```
int meast(int instr_id, double *result);
int measi(int instr_id, double *result);
int measv(int instr_id, double *result);
```

instr_id	The instrument identification code
result	The variable assigned to the result of the measurement

Details

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` command, 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 command 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.

The `rangeX` command directly affects the operation of the `measX` command. The use of the `rangeX` command prevents the instrument addressed from automatically changing ranges when the `measX` command is called. This can result in an overrange condition such that would occur when measuring 10 V on a 4.0 V range. An overrange condition returns the value 1.0E+22 as the result of the measurement.

If used, the `rangeX` command must be in the test sequence before the associated `measX` command.

All measurements except the `meast` command invoke a timer snapshot measurement to be made by all enabled timers. This timer snapshot can then be read with the `meast` command.

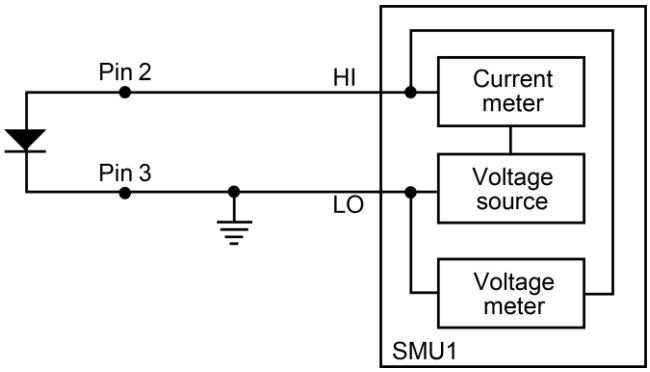
In general, measurement commands that return multiple results are more efficient than performing multiple measurement commands.

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` command is used to select the range. For autoranging, the `measi` or `measv` command will trigger a needed range change before the measurement is performed. See Compliance limits (on page 3-4) for details.

Example

```
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. */
```

In this example, the forward bias voltage of the diode is obtained from a single source-measure unit (SMU).



Also see

intgX (on page 14-78)
rangeX (on page 14-88)

mpulse

This command uses a source-measure unit (SMU) to force a voltage pulse and measure both the voltage and current for exact device loading.

Usage

```
int mpulse(long instr_id, double pulse_amplitude, double pulse_duration, double
    *v_meas, double *i_meas);
```

instr_id	The instrument identification code of the instrument under control
pulse_amplitude	The pulse height in volts
pulse_duration	The pulse width in seconds; the measurements are made at the end of the pulse before the `mpulse` command is shut down
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 made simultaneously with the voltage, so the combined values are an exact representation of the device load at pulse termination

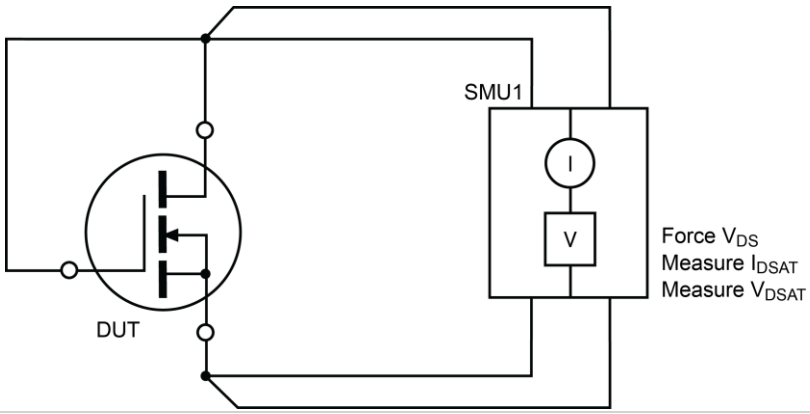
Details

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

```
double vdsat, idsat, vds;
.
.
mpulse(SMU1, vds, 1.0E-3, &vdsat, &idsat);
 /* Pulse output of SMU1. */
```

This example measures the drain current of a metal-oxide semiconductor field-effect transistor (MOSFET) when drain-source voltage (V_{DS}) equals gate-source voltage (V_{GS}). A voltage pulse, V_{DS}, is applied to the drain. The pulse duration is 1 ms. Voltage across the MOS transistor, V_{DSAT}, and drain current, I_{DSAT}, are measured.



Also see

None

pulseX

This command directs a SMU to force a voltage or current at a specific level for a predetermined length of time.

Usage

```
int pulsei(int instr_id, double forceval, double time);
int pulsev(int instr_id, double forceval, double time);
```

instr_id	The instrument identification code
forceval	The level of voltage in volts or current in amperes to force; see **Details**
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 a source-measure unit (SMU) source is dependent on the voltage or current level being sourced and the impedance of the device under test (DUT)

Details

The *forceval* parameter can be positive or negative. For example, sending `pulsev(SMU1, 10.0, 10e-3)` generates +10 V for 10 ms, and sending `pulsei(SMU1, -1.5e-3, 10e-3)` generates -1.5 mA for 10 ms.

The ranges of current and voltage available vary with the instrument type. For more detailed information, refer to the hardware manuals of the instruments in your system.

After `pulseX` is executed, the output is turned off. In order to make 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 made using the `pulseX` and `measX` commands are made 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 `limitX` (on page 14-80) 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 rangeX (on page 14-88) for recommended command order.

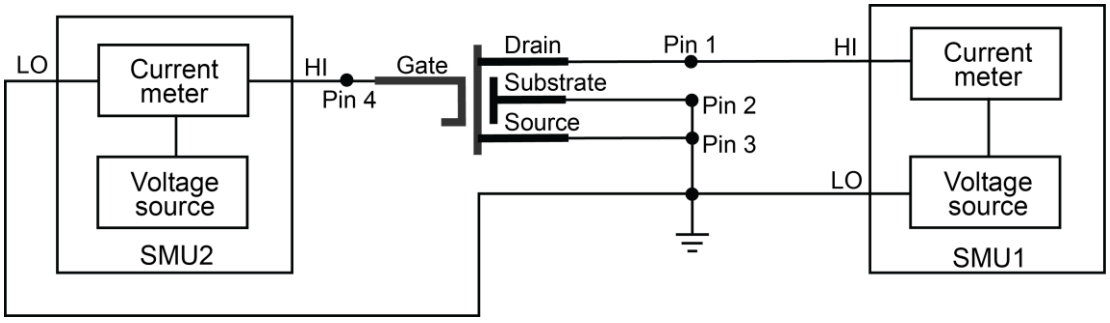
Example

```
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 */
 /* 10 mA. */
searchv(SMU2, 0.0, 3.0, 7, 2.0E-5, &res1); /* Increase */
 /* voltage until */
 /* trigger point occurs. */
 /* Return results to res1. */
pulsev(SMU2, 20.0, 5.E-1); /* Apply a 20 V pulse to the */
    /* gate for 500 ms. */
searchv(SMU2, 0.0, 3.0, 7, 2.0E-5, &res2); /* Increase */
 /* voltage until */
 /* trigger point occurs. */
 /* Return results to res2. */
```

This example measures the threshold voltage shift of an FET by calling two `searchv` commands:

1. The `searchv` command measures the gate voltage required to initiate a drain current of 10 µA.
2. The `searchv` command 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` command was called without reprogramming the `trigil` command. This is possible because the clear trigger command (`clrtrg`) was not used.



Also see

None

rangeX

This command selects a range and prevents the selected instrument from autoranging.

Usage

```
int rangei(int instr_id, double range);
int rangev(int instr_id, double range);
```

instr_id	The instrument identification code
range	The value of the highest measurement to be made (the most appropriate range for this measurement is selected); if *range* is set to 0, the instrument autoranges

Details

Use the `rangeX` command to eliminate the time required by automatic range selection on a measuring instrument. Because the `rangeX` command prevents autoranging, an overrange condition can occur (for example, when measuring 10 V on a 2 V range). The value 1.0e+22 is returned when this occurs.

The `rangeX` command can also reference a source, because a source-measure unit (SMU) can be either of the following:

- Simultaneously a voltage source, voltmeter, and ammeter.
- Simultaneously a current source, ammeter, and voltmeter.

The range of a SMU is the same for the source and the measure commands.

Compliance limits – When selecting a range below the limit value, whether it is explicitly programmed or the default value, an instrument temporarily uses the full-scale value of the range as the limit. This does not change the programmed limit value, and if the instrument range is restored to a value higher than the programmed limit value, the instrument again uses the programmed limit value. See Compliance limits (on page 3-4) for more information.

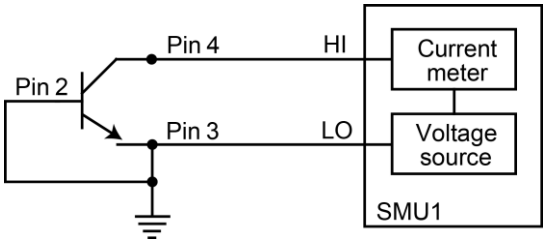
When changing the instrument range, be careful not to overrange the instrument. For example, a test initially performed on the 10 mA range with a 5 mA limit is changed to test in the 1 mA range with a 1 mA limit. Notice that the limit is lowered from 5 mA to 1 mA to avoid overranging the 1 mA setting.

When source mode of the SMU changes, 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 correct 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

```
double icer2;
.
.
conpin(GND, 3, 2, 0);
conpin(SMU1, 4, 0);
limiti(SMU1, 1.0E-3); /* Limit current to 1.0 mA. */
rangei(SMU1, 2.0E-3); /* Select range for 2 mA. */
forcev(SMU1, 35.0); /* Force 35 V. */
measi(SMU1, &icer2); /* Measure leakage; return */
 /* results to icer2. */
```

This example specifies connections, sets a 1 mA limit on the 2 mA range and forces 35 V, then measures current leakage and returns the results to the variable `icer2`.



Also see

asweepX (on page 14-63)
bsweepX (on page 14-69)
forceX (on page 14-74)
pulseX (on page 14-86)
searchX (on page 14-42)
sweepX (on page 14-95)

rtfary

This command returns the array of force values used during the subsequent voltage or frequency sweep.

Usage

```
int rtfary(double *forceArray);
```

forceArray	Array of force values for voltage or frequency

Details

This command is used to return an array of voltage or frequency force values for a sweep. Send this command before calling any sweep command.

NOTE

To prevent a memory exception error, make sure that the array that will receive the sourced values is large enough.

The following examples show the proper command sequence for using `rtfary`:

Example 1: Valid command sequence for voltage sweep	**Example 2: Valid command sequence for frequency sweep**
`smeasz`	`smeasz`
`smeast`	`rtfary`
`rtfary`	`dsweepf`
`dsweepv`	

Example

Programming example #2 (on page 14-224) returns the array of force values for the voltage sweep.

Also see

None

segment_sweepX_list

This command creates and returns up to a 4-segment linear sweep force table based on user-defined start, stop, and step values.

Usage

```
int segment_sweepv_list (double startVal, double *stopArray, double *stepArray, int
    numSegments, double *forceArray, int forceArraysize, int *numListpts);
int segment_sweepi_list (double startVal, double *stopArray, double *stepArray, int
    numSegments, double *forceArray, int forceArraysize, int *numListpts);
```

startVal	Starting voltage value
stopArray	A single dimension array containing stop values
stepArray	A single dimension array containing step values
numSegments	Number of segments
forceArray	A single dimension array returned with force values
forceArraysize	Size allocated for `forceArray`
numListpts	Number of total points in returned `forceArray`

Details

The `segment_sweepX` command is used with the `asweepX` command.

A forcing table is created with the `segment_sweepX_list` command and the force array table is sent using the `asweepX` command.

Example

```
startVoltage = 0.0V
stopArray[] = {5.0, -5.0, 0}
stepArray[] = {0.1, -0.5, 0.25}
segmentpts = 3
arraysize = 1000

    segment_sweepv_list(startVoltage, stopArray, stepArray, segmentpts,
   forceArray, arraysize, numListpts);
    forcev(SMU1, 0.0);
    rtfary(Programmed_V);
    smeasi(SMU1, Measured_I);
    asweepv(SMU1, *numListpts, delayValue, &forceArray[0]);
```

Also see

asweepX (on page 14-63)
forceX (on page 14-74)
rtfary (on page 14-89)
smeasX (on page 14-49)

setauto

This command re-enables autoranging and cancels any previous `rangeX` command for the specified instrument.

Usage

`int setauto(int instr_id);`

instr_id	The instrument identification code

Details

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.

```
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 2 uA 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 V to 2.5 V. */
```

Also see

None

ssmeasX

This command makes a series of readings until the change (delta) between readings is within a specified percentage.

Usage

```
int ssmeasi(int instr_id, double *result, double delta, unsigned int max_read, double
    delay);
int ssmeasv(int instr_id, double *result, double delta, unsigned int max_read, double
    delay);
```

instr_id	The instrument identification code of the measuring instrument
result	The floating point variable assigned to the result of the measurement
delta	The termination definition; this is the percentage of the first reading that defines the steady-state condition
max_read	The maximum number of readings made to determine whether or not the reading is steady
delay	The delay between readings to wait in seconds

Details

This command is used when device stability is uncertain. It continually reads the instrument until the resulting measurement is stable and provides the fastest measurement possible.

If the reading never stabilizes because of factors such as oscillations or charge and discharge, this reading count expires and a reading of `MEAS_NOT_PERFORMED (1.00E23)` is returned.

Any instrument that uses the `meas`*X* command can use the `ssmeas`*X* command. This command calls the `meas`*X* command for each reading. Any `range`*X* command rule applies to this command.

The `ssmeas`*X* command is used when making single-point readings. It is not used for any of the combination measurements, such as the *X*`sweep`*Y* and `trig`*XY* commands.

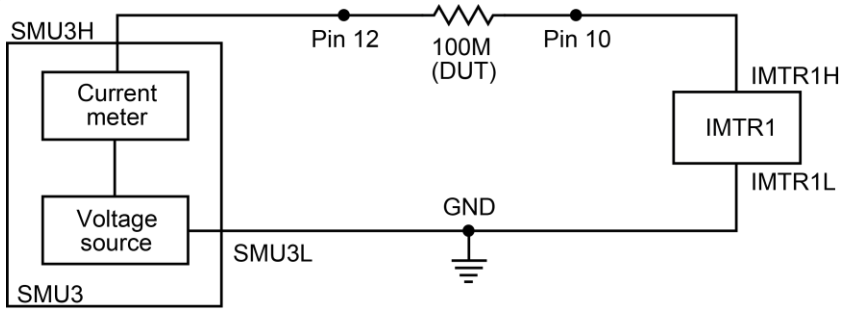
Under certain test conditions, the `ssmeas`*X* command is not ideal. For example, an oscillation where two contiguous measurements are within the given percentage will return a stable reading, even though the device cannot be measured.

Example

```
double meascur;
.
.
conpin(SMU3, 12, 0);       /* Make connections. */
conpin(SMU2, 10, 0);
.
.
forcev(SMU3, 0.1);  /* Perform the test. */
ssmeasi(SMU2, &meascur, 0.1, 300, 0.015);      /* Steady */
    /* state measurement /*
 /* with delta of 0.1%, with */
 /* maximum of 300 readings */
. /* before error, wait 15 ms */
. /* between readings. */
```

This example makes a series of measurements and tests to verify if the present measurement and the previous measurement are within 0.1%. If the measurements are within 0.1%, the result of the last measurement is stored and the program continues. If the measurements are not within 0.1%, the program waits 15 ms before making another measurement. It then compares this measurement with previous measurements. If the measurements are within 0.1%, the result of the last measurement is stored and the program continues. If the measurements are not within 0.1% it repeats the comparison until the change is within 0.1%. If, after 300 attempts, the change is not within the specified limit, the following error is returned:

`"MEAS_NOT_PERFORMED (1.000E23)."`



Also see

measX (on page 14-83)
rangeX (on page 14-88)
smeasX (on page 14-49)

sweepX

This command 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.

Usage

```
int sweepi(int instr_id, double startval, double endval, long stepno, double
    step_delay);
int sweepv(int instr_id, double startval, double endval, long stepno, double
    step_delay);
```

instr_id	The instrument identification code of the sourcing instrument
startval	The initial voltage or current level output from the sourcing instrument, which is 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

Details

The `sweepX` command is always used with the `smeasX`, `sintgX`, `savgX`, or `rtfary` command.

The `sweepX` command 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 done at each step.

NOTE

The actual number of forced data points is one more than the number of steps specified. This means that the number of measurements made is the number of steps specified 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 made.

The `trigXg`, `trigXl`, and `trigcomp` commands can be used with the `sweepX` command, even though they are also used with the `smeasX`, `sintgX`, or `savgX` commands. 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 the `smeasX`, `sintgX`, `savgX`, or `clrscn` command. As multiple calls to these commands are made, the commands are appended to this table.

When multiple calls to the `sweep`*X* command are executed in the same test sequence, the `smeas`*X*, `sintg`*X*, or `savg`*X* arrays are loaded sequentially. This appends the measurements from the second `sweep`*X* call to the previous results. If the arrays are not dimensioned correctly, access violations occur. The measurement table remains intact until the `clrscn`, `execut`, or `devint` command is executed.

Defining new test sequences using the `smeas`*X*, `sintg`*X*, or `savg`*X* commands adds commands to the active measure list. The previous measures are still defined and used. The `clrscn` command is used to eliminate the previous measures for the second sweep. Using the `smeas`*X*, `sintg`*X*, or `savg`*X* command after a `clrscn` command causes the appropriate new measures to be defined and used.

When the first sweep point is nonzero, it may be necessary to precharge the circuit so that the `sweep`*X* command will return a stable value for the first measured point without penalizing remaining points in the sweep. For example:

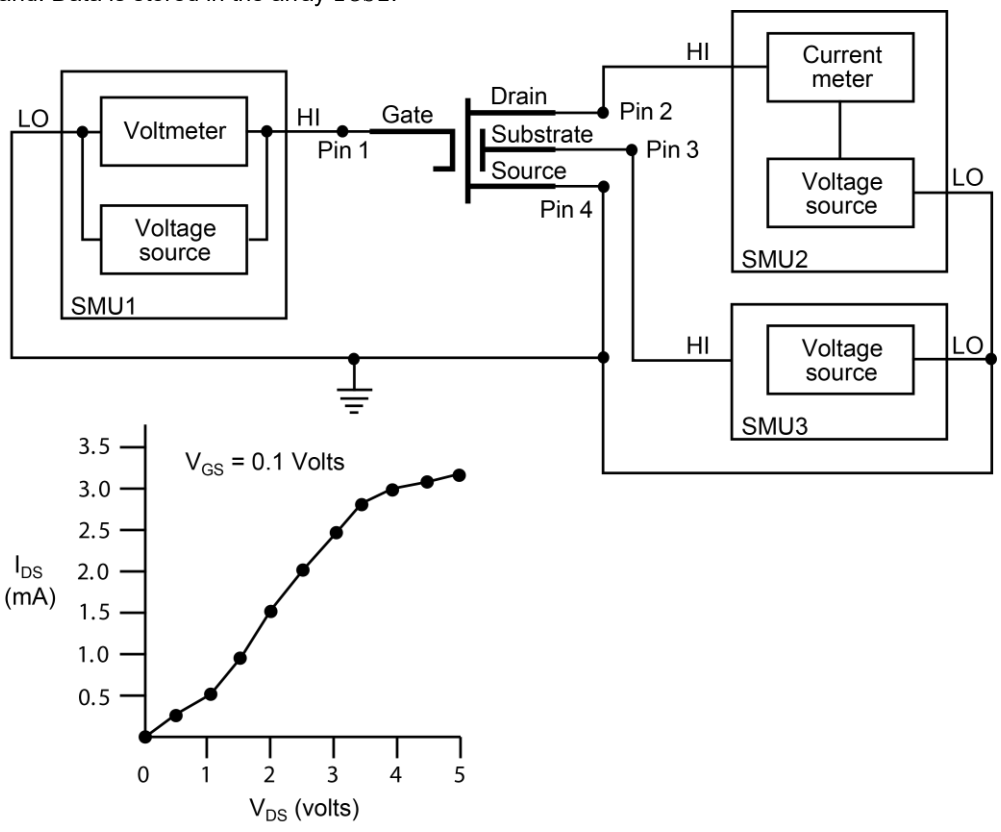
```
double ires[6];
conpin(SMU1, 10, 0);
conpin(2, GND 0);
forcev(SMU1, 5.0); /* Force 5 V to charge. */
delay(10); /* Wait for precharge. */
smeasi(SMU1, ires); /* Set up measurement. */
sweepv(SMU1, 5.0, 10.0, 5, 2.5E-3); /* Make the real measurement. */
```

Note that changing the source mode of the source-measure unit (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 rangeX (on page 14-88) for recommended command order.

Example

```
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 */
 /* voltage SSBIASV. */
forcev(SMU1, -0.1); /* 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, 2.5E-3); /* Generate */
 /* 11 steps and 11 */
 /* points each 2.5 ms duration, */
 /* ranging from 0 to 5 V. */
```

This example gathers data to create a graph showing the common drain-source characteristics of a field-effect transistor (FET). A fixed gate-to-source voltage is generated by SMU1. A voltage ramp from 0 V to 5 V is generated by SMU2. Drain current applied by SMU2 is measured 11 times by the `smeasi` command. Data is stored in the array `resi`.



Also see

rtfary (on page 14-38)
savgX (on page 14-39)
sintgX (on page 14-47)
smeasX (on page 14-49)

LPT commands for PGUs and PMUs

NOTE

The 4200A-SCS has built-in project tests that use the PGU and PMU LPT commands. Refer to the `pmu-dut-examples` project for a simple example of coding a PMU user test module (UTM).

The following information explains the commands in the LPT library for the 4220-PGU and 4225-PMU. The model names are abbreviated as PGU (pulse-generator unit) and PMU (pulse-measure unit). 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) group (on page 14-6) for a summary of the functions for the PGU and PMU.

The pulse commands for the 4220-PGU and 4225-PMU require `pulse_exec` to execute. In addition, they support external triggering, but do not support trigger input from external input signals or instruments.

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 (on page 14-140) and seg_arb_waveform (on page 14-144) 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 (on page 14-148) and arb_file (on page 14-149)).

Use the following instrument ID (identification) for LPT commands for the PGU and PMU:

- 4220-PGU: The instrument ID is VPU (VPU1, VPU2, and so on)
- 4225-PMU: The instrument ID is PMU (PMU1, PMU2, and so on)

NOTE

The 4220-PGU and 4225-PMU support the pulsing and external triggering commands of the obsolete 4205-PG2.

dev_abort

This command programmatically ends a test from within the user module (aborts a test) that was started with the `pulse_exec` command.

Usage

```
int dev_abort(NULL);
```

Pulsers

4220-PGU
4225-PMU

Pulse mode

Standard and Segment Arb

Details

Use this command to abort a pulse test from in 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 to save test time.

Example

```
// 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);
 }
```

This example illustrates 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.

Also see

None

PostDataDoubleBuffer

This buffer posts PMU data retrieved from the buffer into the Clarius Analyze sheet (large data sets).

Usage

```
int PostDataDoubleBuffer(char *ColName, double *array, int length);
```

ColName	Column name for the data array in the Clarius Analyze sheet
array	An array of data values for the Clarius Analyze sheet
length	Number of data points (up to 65,535) to post into the Clarius Analyze sheet

Pulsers

4225-PMU

Pulse mode

Standard and Segment Arb

Details

You can use the `PostDataDouble` and `PostDataDoubleBuffer` commands to post double-precision floating point data into the Clarius Analyze sheet. Up to 65,535 points (rows) of data can be posted into the Analyze sheet. These commands are used after one measurement point is finished and a data value is assigned to the corresponding output variable.

You can use either of these commands to post data into the sheet. However, you should use the `PostDataDoubleBuffer` command to post the large data sets that are typically generated by PMU waveform measurements.

The following sequence summarizes the process to post data into the Analyze sheet:

- Run a test.
- Use `pulse_fetch` to retrieve the data from the buffer. You can analyze or manipulate the retrieved data.
- Use `PostDataDouble` or `PostDataDoubleBuffer` to post data into the Analyze sheet.

When you use `pulse_fetch`, you can either 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`" in the pulse_fetch (on page 14-110) command Details section.

NOTE

If you do not need to analyze or manipulate the test data before posting it into the Analyze sheet in Clarius, you can use `pulse_measrt`, which retrieves all the test data in pseudo real-time and automatically posts it into the Clarius Analyze sheet.

NOTE

`PostDataDoubleBuffer` is not compatible with using KXCI to call user libraries remotely (see Calling KULT user libraries remotely (on page 10-103)). Use `PostDataDouble` for user routines (UTMs) that will be called using KXCI.

Example

```
// 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);
```

Posts waveform measurement data into the Analyze sheet. This example assumes that a PMU waveform test is configured to perform 20,000 (or more) voltage and current measurements. Use `pulse_meas_wfm` to configure the waveform test.
The code:

- 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 Clarius Analyze sheet.

Also see

PostDataDouble (on page 14-35)
pulse_fetch (on page 14-110)
pulse_meas_wfm (on page 14-123)
pulse_measrt (on page 14-124)

pmu_offset_current_comp

This command is used to collect offset current constants from the 4225-PMU. The offset (open) correction readings are stored in a local file.

Usage

```
int pmu_offset_current_comp(int instr_id);
```

instr_id	The instrument identification code of the pulse generator: PMU1, PMU2, and so on.

Pulsers

4225-PMU

Pulse mode

Segment Arb

Details

Use this command to collect current constants for offset current compensation. The correction readings are stored in a file. You can then use the `setmode` command to configure the state of the offset current compensation.

Example

```
int pmu_offset_current_comp(PMU1)
```

This example collects offset current constants from a 4225-PMU assigned PMU1.

Also see

setmode (4225-PMU) (on page 14-145)

pulse_chan_status

This command is used to determine how many readings are stored in the data buffer.

Usage

```
int pulse_chan_status(int instr_id, int chan, int *buffersize);
```

instr_id	The instrument identification code: PMU1, PMU2, and so on
chan	Channel number of the pulse card: 1 or 2
buffersize	User-defined name for the returned size value

Pulsers

4225-PMU

Pulse mode

Standard and Segment Arb

Details

Use this command to return the number of readings presently stored in the data buffer. This command 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 command is typically called after the sweep is complete to determine the total number of readings stored in the buffer. For a long test, you can use this command to track the progress of the test. A long test is typically Segment Arb with test time in minutes, hours, or days.

Example

```
pulse_chan_status(PMU1, 1, buffersize);
```

This command returns the number of readings stored in the buffer for channel 1.

Also see

None

pulse_conncomp

This command enables or disables connection compensation.

Usage

```
int pulse_conncomp(int instr_id, int chan, int type, int index);
```

instr_id	The instrument identification code: PMU1, PMU2, and so on
chan	Channel number of the PMU: 1 or 2
type	Type of compensation to enable: ■ 1: Short ■ 2: Delay
index	Connection compensation values: ■ 0: Disable 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 (see **Details**)

Pulsers

4225-PMU

Pulse mode

Standard and Segment Arb

Details

Errors caused by the connections and cable length between the 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 command to control connection compensation. You can select short or delay. Short compensation corrects for the measured resistance of the cabling and connections. Delay compensation measures and corrects for cable delay (the time it takes a signal to transit the cable).

You can 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.

Custom connection compensation is a two-part process:

1. Perform connection compensation from the Clarius interface (see Performing connection compensation (on page 5-41)). Connection compensation data is generated for short and delay conditions. The compensation values are stored in tables.
2. Use this command (`pulse_conncomp`) to enable the custom connection compensation values.

When a test is run, each measurement factors in the enabled compensation values. If connection compensation is disabled, the compensation values are not used by the test.

Example

```
pulse_conncomp(PMU1, 1, 1, 3);
```

This example assumes connection compensation was done using the Clarius interface. This command enables short connection compensation using the custom compensation values.

Also see

None

pulse_exec

This command is used to validate the test configuration and start test execution.

Usage

```
int pulse_exec(long mode);
```

mode	The mode of execution: ■ 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

Pulsers

4220-PGU
4225-PMU

Pulse mode

Standard and Segment Arb

Details

Use this command 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` command is nonblocking, which means that if this command is called to execute the test, the program continues and does 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 user test module (UTM).

There are two commands that affect a pulse test while it is running:

- The `pulse_remove` command removes a PMU channel from the test.
- The `dev_abort` command aborts the test.

NOTE

The Internal Trigger Bus trigger source (see `pulse_trig_source` command) is used only by the 4220-PGU and 4225-PMU for triggering. The `pulse_exec` command automatically uses the internal trigger bus. A trigger input to start a `pulse_exec` test is not available.

CAUTION

Do not exit the user module while the test is still running. Incorrect readings or device damage may result.

Example

```
// 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
```

This example uses `pulse_exec` to set the execution type to simple two-level pulse operation (no analysis), and executes the test. The code pauses the program to monitor the status of the test. It uses 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.

Also see

dev_abort (on page 14-99)
pulse_remove (on page 14-128)
pulse_trig_source (on page 14-176)

pulse_exec_status

This command is used to determine if a test is running or completed.

Usage

```
int pulse_exec_status(double *elapsedt);
```

elapsedt	Name of the user-defined pointer for elapsed time

Pulsers

4220-PGU
4225-PMU

Pulse mode

Standard and Segment Arb

Details

This command 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.

The elapsed time is the Clarius 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 command is typically used in a `while` loop to allow the test to finish before retrieving the data using the `pulse_fetch` command.

It is the responsibility of the user test module (UTM) programmer to ensure that the pulse test is complete before exiting the UTM. If the UTM program ends before the test is complete, Clarius responds with two messages. These messages are displayed in the Clarius messages area:

1. 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)" is displayed.
2. Clarius continues to wait for the UTM to finish, interrupting further test execution.
3. 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."
4. After this five minute wait, Clarius releases control to the user interface or to the next test in the project (if using repeat executing or looping).

Example

```
// 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
```

This example uses `pulse_exec` to set the execution type to simple two-level pulse operation (no analysis), and executes the test. The code pauses the program to monitor the status of the test. It uses 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.

Also see

pulse_exec (on page 14-107)
pulse_fetch (on page 14-110)

pulse_fetch

This command retrieves enabled test data and temporarily stores it in the data buffer.

Usage

```
int pulse_fetch(int instr_id, int chan, int StartIndex, int StopIndex, double *Vmeas,
   double *Imeas, double *Timestamp, unsigned long *Status);
```

instr_id	The instrument identification code: PMU1, PMU2, and so on
chan	Channel number of the pulse card: `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

Pulsers

4225-PMU

Pulse mode

Standard and Segment Arb

Details

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 command to pause for a specified period of time, or you can use the `pulse_exec_status` command in a `while` loop to wait until the test is completed.

Use this command 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 Clarius Analyze sheet.

Typically, this command is used with the `pulse_exec_status` command to allow the test to finish before retrieving the 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 valid parameters for these fields.

NOTE

The return of all readings must be enabled by the `pulse_meas_sm` command. If disabled, the arrays will not be retrieved.

For spot mean measurements, data can be mixed; the 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 the `pulse_meas_sm` command 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, and so on. 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 the following table.

NOTE

If you do not need to analyze or manipulate the test data before posting it to the Clarius Analyze sheet, you can use the `pulse_measrt` command. The `pulse_measrt` command retrieves all the test data in pseudo real-time and automatically posts it into the Clarius Analyze sheet.

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

Data retrieval options for `pulse_fetch`

There are two options to retrieve data:

- Wait until the test is completed
- Retrieve blocks of data while the test is running

Because `pulse_exec` is a non-blocking command, 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 Clarius before the test is complete. 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 complete 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 Clarius Analyze sheet. Use the `PostDataDouble` or `PostDataDoubleBuffer` command to post the data.

Retrieve blocks of data while the test is running

An advantage of the `pulse_exec` command 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` command as shown in the following program fragment.

Program fragment 2

```
// Code to initialize the data arrays
for (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 command 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` command to post the data to the Clarius Analyze sheet.

Example 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
```

This example uses `pulse_exec` to set the execution type to simple two-level pulse operation (no analysis), and executes the test. The code pauses the program to monitor the status of the test. It uses 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.

Example 2

```
pulse_fetch(PMU1, 1, 0, 49, Drain_Vmeas, Drain_Imeas, T_Stamp, NULL);
```

This command retrieves 50 points of data from the buffer, 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)

Also see

PostDataDouble (on page 14-35)
PostDataDoubleBuffer (on page 14-101)
pulse_meas_sm (on page 14-119)
pulse_measrt (on page 14-124)

pulse_float

This command sets the state of the floating relay for the given pulse instrument.

Usage

```
int pulse_float(int instr_id, int state);
instr_id The instrument identification code: PMU1, PMU2
state OFF (default), ON (float)
Pulsers
4220-PGU, 4225-PMU
Pulse Mode
Standard
```

Details

This command is used to float the PGU/PMU card. Use this in conjunction with the 4205-PCU to allow higher voltage pulse output. The 4205-PCU combines two pulser cards to provide a single output signal.This command is to be used with the 4205-PCU to combine four channels from two 4220-PGU or 4225-PMU instruments.

Example

```
pulse_float(PMU1, OFF);
```

This turns off the floating relay on PMU1 instrument.

Also see

None

pulse_limits

This command sets measured voltage and current thresholds at the DUT and sets the power threshold for each channel.

Usage

```
int pulse_limits(int instr_id, int chan, double V_Limit, double I_Limit, double
    Power_Limit);
```

instr_id	The instrument identification code: PMU1, PMU2, and so on
chan	Channel number of the pulse card: 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 = V_{meas} x I_{meas})

Pulsers

4225-PMU

Pulse mode

Standard

Details

This feature differs from a SMU compliance setting in that threshold checking is done 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 does not prevent the set thresholds from being reached or exceeded. After detecting a threshold breach, it 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

```
pulse_limits(PMU1, 1, 42, 1, 10);
```

This example sets thresholds for channel 1 of the PMU, where:

- *instr_id* = PMU1
- *chan* = Channel 1
- *V_Limit* = 42 V
- *I_Limit* = 1 A
- *Power_Limit* = 10 W

Also see

None

pulse_meas_sm

This command configures spot mean measurements.

Usage

```
int pulse_meas_sm(int instr_id, int chan, Int AcquireType, int AcquireMeasVAmpl, int
    AcquireMeasVBase, int AcquireMeasIAmpl, int AquireMeasIBase, int AquireTimeStamp,
    int LLEComp);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: `1` or `2`
AcquireType	Acquisition type: ■ Discrete: `0` ■ Average: `1`
AcquireMeasVAmpl	Return amplitude voltage measurements: ■ Disable: `0` ■ Enable: `1`
AcquireMeasVBase	Return base level voltage measurements: ■ Disable: `0` ■ Enable: `1`
AcquireMeasIAmpl	Return amplitude current measurements: ■ Disable: `0` ■ Enable: `1`
AcquireMeasIBase	Return base current level measurements: ■ Disable: `0` ■ Enable: `1`
AcquireTimeStamp	Return time stamp readings: ■ Disable: `0` ■ Enable: `1`
LLEComp	Load-line effect compensation (LLEC): ■ All LLEC disabled: `0` ■ Voltage LLEC on for pulse amplitude only: `1`

Pulsers

4225-PMU

Pulse modes

Standard

Details

Use the `pulse_meas_sm` command to configure spot mean measurements; select the data acquisition type, set the readings to be returned, enable or disable time stamp, and set load-line effect compensation (LLEC).

LLEC is only performed for standard pulse IV testing using PMU measure ranges. It is not performed when using 4225-RPM measure ranges. The active RPM circuitry provides its own analog LLEC (assuming a short cable from the RPM to the DUT).

Also see

Load-line effect compensation (on page 5-44)
Measurement types (on page 5-91)
Spot mean measurements (on page 5-91)

pulse_meas_timing

This command sets the measurement windows.

Usage

```
int pulse_meas_timing(int instr_id, int chan, double, StartPercent, double StopPercent,
    int NumPulses);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: `1` or `2`
StartPercent	Start location for measurements: ■ Spot mean measurements: Start location, specified as a percentage of the widths for the amplitude and base level (see **Details**) ■ Waveform: Pre-data for the amplitude, specified as a percentage of the amplitude pulse duration (see **Details**)
StopPercent	Stop location for measurements: ■ Spot mean measurements: Stop location, specified as a percentage of the widths for the amplitude and base level (see **Details**) ■ Waveform: Post-data for the amplitude, specified as a percentage of the amplitude pulse duration (see **Details**)
NumPulses	Number of pulses to output and measure (1 to 10,000)

Pulsers

4225-PMU

Pulse modes

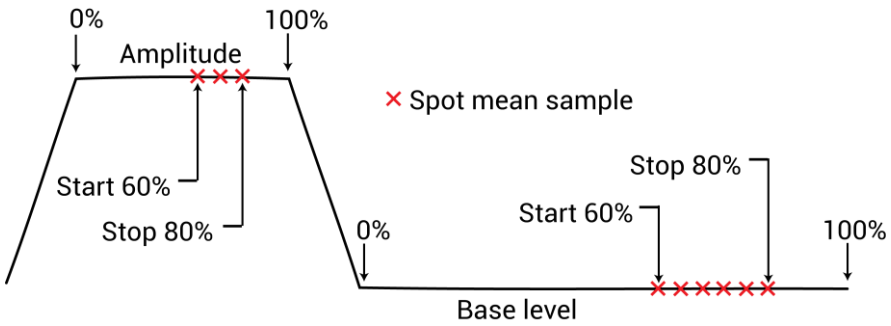
Standard

Details

Use the `pulse_meas_timing` command 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 (on page 5-96) for details on pulse measurement timing.

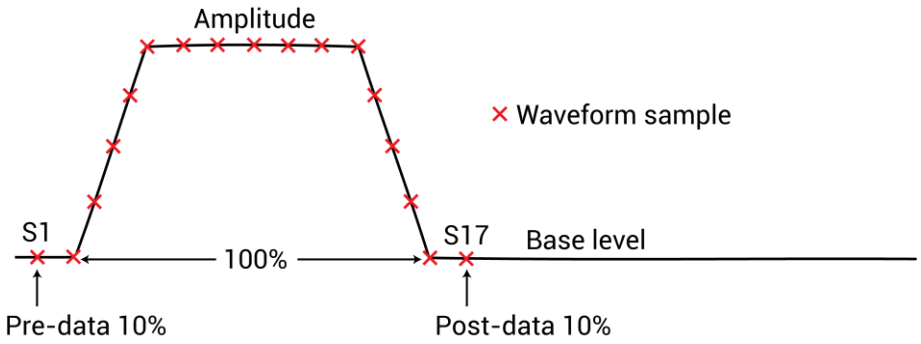
The figure below shows example start and stop locations spot mean measurements are made. Three measured samples are taken on the amplitude and six samples are taken on the base level. The start and stop percentage values indicate the portions of the pulse that are sampled.

Figure 601: Spot mean measurements example



The figure below shows example a waveform measurement with 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 602: Waveform measurements with pre-data and post-data



NOTE

Use the `pulse_sample_rate` command to set the sampling rate for pulse measurements.

NOTE

Before calling the `pulse_meas_timing` command, use the `pulse_meas_sm` or `pulse_meas_wfm` command to configure the measurement type.

Also see

Measurement timing (on page 5-96)
Measurement types (on page 5-91)
pulse_meas_sm (on page 14-119)
pulse_sample_rate (on page 14-129)
pulse_meas_wfm (on page 14-123)

pulse_meas_wfm

This command configures waveform measurements.

Usage

```
int pulse_meas_wfm(int instr_id, int chan, int AcquireType, int AcquireMeasV, int
    AcquireMeasI, int AquireTimeStamp, int LLEComp);
```

instr_id	The instrument identification code of the PMU, such as PMU1 or PMU2
chan	Channel number of the pulse card: 1 or 2
AcquireType	Acquisition type: ■ Discrete: 0 ■ Average: 1
AcquireMeasV	Return voltage measurements: ■ Disable: 0 ■ Enable: 1
AcquireMeasI	Return current measurements: ■ Disable: 0 ■ Enable: 1
AcquireTimeStamp	Return time stamp readings (must be enabled to measure waveforms): ■ Disable: 0 ■ Enable: 1
LLEComp	Load line effect compensation (LLEC): ■ LLEC disabled: 0 ■ LLEC enabled: 1

Pulsers

4225-PMU

Pulse modes

Standard

Details

Use the pulse_meas_wfm command to configure waveform measurements; select the data acquisition type, set the readings to be returned, enable or disable time stamp, and set load-line effect compensation (LLEC). See Waveform measurements (on page 5-94) for details.

LLEC is only performed for standard pulse IV testing using PMU measure ranges. It is not performed when using 4225-RPM measure ranges. The active RPM circuitry provides its own analog LLEC (assuming a short cable from the RPM to the DUT).

Also see

Load-line effect compensation (on page 5-44)
Measurement types (on page 5-91)

pulse_measrt

This command returns pulse source and measure data in pseudo real-time.

Usage

```
int pulse_measrt(int instr_id, int chan, char *VMeasColName, char *IMeasColName, char
    *TimeStampColName, char *StatusColName);
```

instr_id	The instrument identification code: PMU1, PMU2, and so on
chan	Channel number of the pulse card: `1` or `2`
VMeasColName	Column name for V-measure data in the Clarius Analyze sheet
IMeasColName	Column name for I-measure data in the Clarius Analyze sheet
TimeStampColName	Column name for the time stamp data in the Clarius Analyze sheet
StatusColName	Column name for the status data in the Clarius Analyze sheet

Pulsers

4225-PMU

Pulse mode

Standard and Segment Arb

Details

Use this command to return and display test data in pseudo real-time. As measurements are performed, the data is automatically placed in the Clarius Analyze sheet. This command is also used to name the columns in the Clarius Analyze sheet.

This command must be called before calling `pulse_exec` to start the test.

NOTE

The `pulse_measrt` command is not compatible with using KXCI to call user libraries remotely (see Calling KULT user libraries remotely (on page 10-103)). Use `PostDataDouble` for user test modules (UTMs) that will be called using KXCI.

Example

```
pulse_measrt(PMU1, 1, "V-Measure", "I-Measure", "Timestamp", "Status");
```

This example configures channel 1 of PMU1 to return data in pseudo real-time.

Also see

pulse_exec (on page 14-107)
pulse_fetch (on page 14-110)

pulse_ranges

This command sets the voltage pulse range and voltage/current measure ranges.

Usage

```
int pulse_ranges(int instr_id, int chan, double VSrcRange, int Vrange_type, double
    Vrange, int Irange_type, double IRange);
```

instr_id	The instrument identification code, such as VPU1, VPU2, PMU1, or PMU2
chan	Channel number of the pulse card: `1` or `2`
VSrcRange	Voltage source range: ■ `5` or `20` (into 50 Ω) ■ `10` or `40` (into ≥1 MΩ)
Vrange_type	Voltage measure range type (PMU): ■ Auto: `0` ■ Limited auto: `1` ■ Fixed: `2`
Vrange	Voltage measure range (PMU) in volts: `10` or `40`; ignored if autorange is selected
Irange_type	Current measure range type (PMU): ■ Auto: `0` ■ Limited auto: `1` ■ Fixed: `2`
IRange	Current measure range in amps; see **Details**; ignored if autorange is selected

Pulsers

4220-PGU
4225-PMU
4225-RPM

Pulse modes

Standard, Full Arb, Segment Arb

Details

The *Vrange_type*, *Vrange*, *Irange_type*, and *Irange* parameters are ignored by the PGU.

Autorange (0) and limited autorange (1) are not valid for the Segment Arb pulse mode.

You can set the source range independently for each PGU 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 (on page 6-136) can be as short as 20 ns and Pulse Width (on page 6-136) can be set as short as 10 ns. This setting takes effect when the next pulse trigger is initiated.

For the PGU, use this command to set the voltage source range for pulse output.

For the PMU, use this command to:

- Set the voltage source range for pulse output.
- Set the voltage and current measure range types.
- Set the actual voltage and current measure ranges.

The measure range types for the PMU are:

- 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 when autorange is enabled, but must be a valid range.

The current ranges available depend on the source range and whether the system includes a 4225-RPM, as shown in the next table.

Current measure range (A)	PMU source range (V)	RPM source range (V)
0.8	n/a	40
0.2	10	n/a
0.01	10	10 or 40
0.001	n/a	10
0.0001	n/a	10 or 40
0.00001	n/a	10
0.000001	n/a	10
0.0000001	n/a	10

Auto or limited autoranging is available only when using the advanced mode in the `pulse_exec` command. Ranging is controlled per channel and may be combined with load-line effect compensation (LLEC) and thresholds. See `pulse_limits` command for thresholds.

The Segment Arb pulse mode does not allow range changes (no autorange) in a Segment Arb® waveform definition. Only fixed ranging is available for the Segment Arb pulse mode.

Example

```
pulse_ranges(PMU1, 1, 10, 0, 10, 0, 0.2);
```

This example sets the source-measure ranges for channel 1 of PMU1, 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)

Also see

PMU and RPM measure ranges are not source ranges (on page 5-57)
pulse_exec (on page 14-107)
pulse_limits (on page 14-118)
rpm_config (on page 14-139)

pulse_remove

This command removes a pulse channel from the test.

Usage

```
int pulse_remove(int instr_id, int chan, double voltage, unsigned long state);
```

instr_id	The instrument identification code: VPU1, VPU2, PMU1, PMU2, and so on
chan	Channel number of the pulse card: `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)

Pulsers

4220-PGU
4225-PMU

Pulse mode

Standard and Segment Arb

Details

Use this command 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. You can use it to remove a channel from a long-term reliability test while allowing other channels to continue running.

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 the 4225-PMU block diagram in Models 4220-PGU and 4225-PMU (on page 5-1).

When you remove a channel from a test that is not running, the `voltage` and `state` parameters are ignored.

Example

```
pulse_remove(PMU2, 1, 0, 0);
```

This example removes channel 1 for PMU2, sets the voltage to 0 V, and opens the output relay.

Also see

None

pulse_sample_rate

This command sets the measurement sample rate.

Usage

```
int pulse_sample_rate(INSTR_ID instr_id, double Sample_rate);
```

instr_id	The instrument identification code: PMU1, PMU2, and so on
Sample_rate	Sample rate: 200E6, 100E6, 50E6, 40E6, 33E6, 29E6, ... 1E3

Pulsers

4225-PMU

Pulse mode

Standard and Segment Arb

Details

Use this card-based command 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 is used. For example, if 90E6 samples per second is sent, the sampling rate is set to 100E6 samples per second (200E6/2).

Example

```
pulse_sample_rate(PMU1, 100E6);
```

This example command sets the sampling rate of the PMU to 100E6 samples per second.

Also see

None

pulse_source_timing

This command sets the pulse period, pulse width, rise time, fall time, and delay time.

Usage

```
int pulse_source_timing(int instr_id, int chan, double period, double delay, double
    width, double rise, double fall);
```

instr_id	The instrument identification code: VPU1, VPU2, PMU1, PMU2, and so on
chan	Channel number of the pulse card: `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

Pulsers

4220-PGU
4225-PMU

Pulse mode

Standard

Details

Use this command 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 (on page 5-51) for more information on these pulse parameters.

This command 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 command. 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` command, the period values for the sweep override the period setting for this command.

Example

```
pulse_source_timing(PMU1, 1, 0.02, 0.005, 0.01, 0.001, 0.001);
```

This example the following pulse source timing settings for the PMU, 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)

Also see

pulse_step_linear (on page 14-132)
pulse_sweep_linear (on page 14-132)

pulse_step_linear

This command configures the pulse stepping type.

Usage

```
int pulse_step_linear(int instr_id, int chan, int StepType, double start, double stop,
    double step);
```

instr_id	The instrument identification code: VPU1, VPU2, PMU1, PMU2, and so on
chan	Channel number of the pulse generator: `1` or `2`
StepType	Step type: ■ `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
stop	Final value for stepping
step	Step size for stepping

Pulsers

4220-PGU
4225-PMU

Pulse mode

Standard

Details

The relationship between a step function and a sweep function for SMUs is illustrated in Operation mode timing diagrams (on page 3-47). The step/sweep relationship for pulsing is similar (see Operation Mode (PMU) (on page 6-86)). 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 6-96)).

Use the `start`, `stop`, and `step` parameters to configure stepping. 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 card can step or sweep amplitude (with base level fixed) or step or 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 card can step or 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 Pulse period (on page 6-136)). SweepType example:

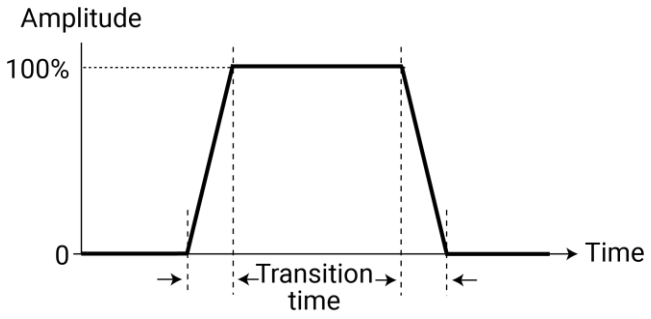
`PULSE_PERIOD_SP` (stepping or sweeping): Start = 0.01 s, stop = 0.05 s, step = 0.01 s

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. The transition time is the interval between corresponding 0% and 100% amplitude points on the rising and falling edge of the pulse, as shown in the following figure.

Figure 603: Transition time



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 Pulse width (on page 6-136). 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 Dual Sweep Option (on page 6-96) for a diagram of this example.

Also see

pulse_sweep_linear (on page 14-135)
pulse_vhigh (on page 14-178)
pulse_vlow (on page 14-180)

pulse_sweep_linear

This command configures the pulse sweeping type.

Usage

```
int pulse_sweep_linear(int instr_id, int chan, int SweepType, double start, double stop,
    double step);
```

instr_id	The instrument identification code: VPU1, VPU2, PMU1, PMU2, and so on
chan	Channel number of the pulse generator: 1 or 2
SweepType	Sweep type: ■ 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 sweeping
stop	Final value for sweeping
step	Step size for sweeping

Pulsers

4220-PGU
4225-PMU

Pulse mode

Standard

Details

The relationship between a step function and a sweep function for SMUs is illustrated in Operation mode timing diagrams (on page 3-47). The step/sweep relationship for pulsing is similar (see Operation Mode (PMU) (on page 6-86)). 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 6-96)).

Use the start, stop, and step parameters to configure stepping. 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 card can step or sweep amplitude (with base level fixed) or step or 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 card can step or 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 Pulse period (on page 6-136)). SweepType example:

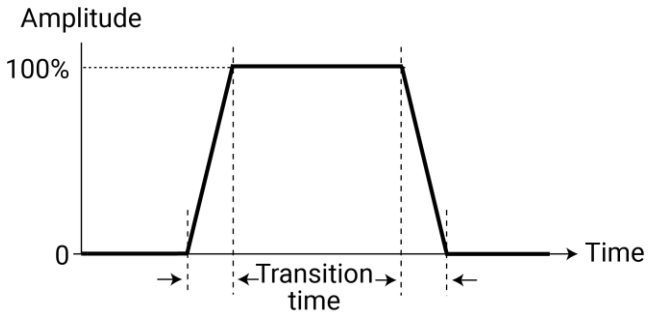
`PULSE_PERIOD_SP` (stepping or sweeping): Start = 0.01 s, stop = 0.05 s, step = 0.01 s

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. The transition time is the interval between corresponding 0% and 100% amplitude points on the rising and falling edge of the pulse, as shown in the following figure.

Figure 604: Transition time



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 Pulse width (on page 6-136). 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 Dual Sweep Option (on page 6-96) for a diagram of this example.

Example

```
pulse_sweep_linear(PMU1, 1, PULSE_AMPLITUDE_SP, 1, 5, 1);
```

This example configures channel 1 of the PMU to perform an amplitude sweep from 1 V to 5 V in 1 V steps.

Also see

pulse_step_linear (on page 14-132)
pulse_vhigh (on page 14-178)
pulse_vlow (on page 14-180)

pulse_train

This command configures the pulse card to output a pulse train using fixed voltage values.

Usage

```
int pulse_train(int instr_id, int chan, double Vbase, double Vamplitude);
```

instr_id	The instrument identification code: VPU1, VPU2, PMU1, PMU2, and so on
chan	Channel number of the pulse card: `1` or `2`
Vbase	Voltage level for pulse base level
Vamplitude	Voltage level for pulse amplitude

Pulsers

4220-PGU
4225-PMU

Pulse mode

Standard

Details

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` command cannot be used by itself in a test. When using a PMU, at least one PMU channel in a test must be a valid `pulse_sweep_linear` function call.

Example

```
pulse_train(PMU1, 1, 0, 5);
```

This example configures channel 1 of the PMU to output a 0 to 5 V pulse train.

Also see

pulse_step_linear (on page 14-132)
pulse_sweep_linear (on page 14-132)

rpm_config

This command sends switching commands to the 4225-RPM.

Usage

```
int rpm_config(int instr_id, long chan, long modifier, long value);
```

instr_id	The instrument identification code: PMU1, PMU2, and so on
chan	Channel number of the pulse generator: `1` or `2`
modifier	Parameter to modify: `KI_RPM_PATHWAY`
value	Value to set modifier: ■ `KI_RPM_PULSE` or `0`: Selects pulsing (4225-RPM) ■ `KI_RPM_CV_2W` or `1`: Selects 2-wire CVU (4210-CVU) ■ `KI_RPM_CV_4W` or `2`: Selects 4-wire CVU (4210-CVU) ■ `KI_RPM_SMU` or `3`: Selects SMU (4200-SMU)

Pulsers

4225-PMU with the 4225-RPM

Pulse mode

Standard (two-level pulsing), Segment Arb, and full arb

Details

The 4225-RPM includes input connections for the 4210-CVU and 4200-SMU. Use this command 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

```
rpm_config(PMU1, 1, KI_RPM_PATHWAY, KI_RPM_PULSE);
```

This example sets channel 1 of the RPM for pulsing.

Also see

clrcon (on page 14-229)

seg_arb_sequence

This command defines the parameters for a Segment Arb waveform pulse-measure sequence.

Usage

```
int seg_arb_sequence(int instr_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	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: 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 (for trigger output only): ■ Trigger low: 0 ■ Trigger high: 1
SSR	An array of values to control the high endurance output relay: ■ Open: 0 ■ Closed: 1
MeasType	PGU: 0 PMU: An array of measure types: ■ No measurements for this segment: 0 ■ Spot mean discrete (see Spot mean measurements (on page 5-91)): 1 ■ Waveform discrete (see Waveform measurements (on page 5-94)): 2 ■ Spot mean average (see Spot mean average readings (on page 5-93)): 3 ■ Waveform average (see Waveform average readings (on page 5-95)): 4
MeasStart	PGU: 0 PMU: 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	PGU: 0 PMU: An array of stop measurement times (in seconds, with 10 ns resolution); this is the elapsed time, within the segment, when the measurement stops

Pulsers

4220-PGU
4225-PMU

Pulse mode

Segment Arb

Details

Use this command to configure each channel to output a unique Segment Arb® waveform. For the PMU, this also configures each channel to make measurements.

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), and output relay state (open or closed). For PMUs, each can have a unique pulse measurement type, measurement start time, and measurement stop time.

A defined sequence is uniquely identified by its specified channel number and sequence ID number. This command defines the sequences, or building blocks, that are typically used for a BTI (bias temperature instability semiconductor reliability) test.

A sequence is defined as three or more segments with seamless voltage transitions. Seamless means that there are no voltage differences — 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 combined into a Segment Arb waveform using the `seg_arb_waveform` command. All sequence transitions must also be seamless. The example below shows an example of a waveform that consists of three sequences.

The 4220-PGU does not have pulse-measure capability. When this command for the PGU is called, the parameter values for *`MeasType`*, *`MeasStart`*, and *`MeasStop`* are ignored.

NOTE

See Segment Arb waveforms (on page 11-7) for details on using KPulse to output Segment Arb waveforms.

Example

This command defines the Segment Arb sequence shown in the following figure.

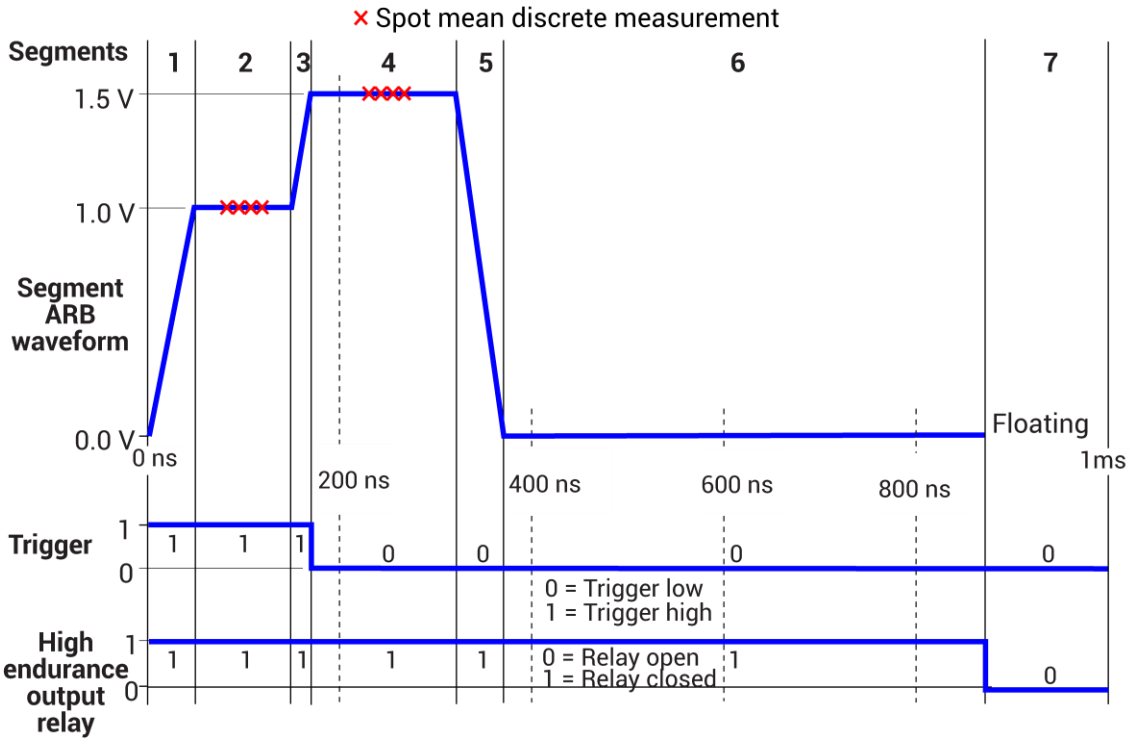
```
seg_arb_sequence(PMU1, 1, 1, 7, Start_Volt, Stop_Volt, Time_Interval, Trig_Level,
    Output_Relay, Meas_Type, Meas_Start, Meas_Stop);
```

The arrays for the `seg_arb_function` are:

```
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};
```

This figure shows an example of a Segment Arb sequence defined by the `seg_arb_sequence` command. Spot mean discrete measurements are performed on segments two and four.

Figure 605: Segment Arb sequence example



This table lists the `seg_arb_sequence` parameter arrays for the Segment Arb sequence shown in the example.

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

Also see

Segment Arb waveform (on page 5-72)
seg_arb_waveform (on page 14-144)

seg_arb_waveform

This command creates a voltage segment waveform.

Usage

```
int seg_arb_waveform(int instr_id, long chan, long NumSeq, long *Seq, double
   *SeqLoopCount);
```

instr_id	The instrument identification code, such as VPU1 or VPU2
chan	Channel number of the pulse card: 1 or 2
NumSeq	Total number of sequences in waveform definition (512 maximum)
Seq	An array of sequences using the sequence number ID
SeqLoopCount	An array of loop values (number of times to output a sequence); loop value range is 1 to 1E12

Pulsers

4220-PGU
4225-PMU

Pulse modes

Segment Arb

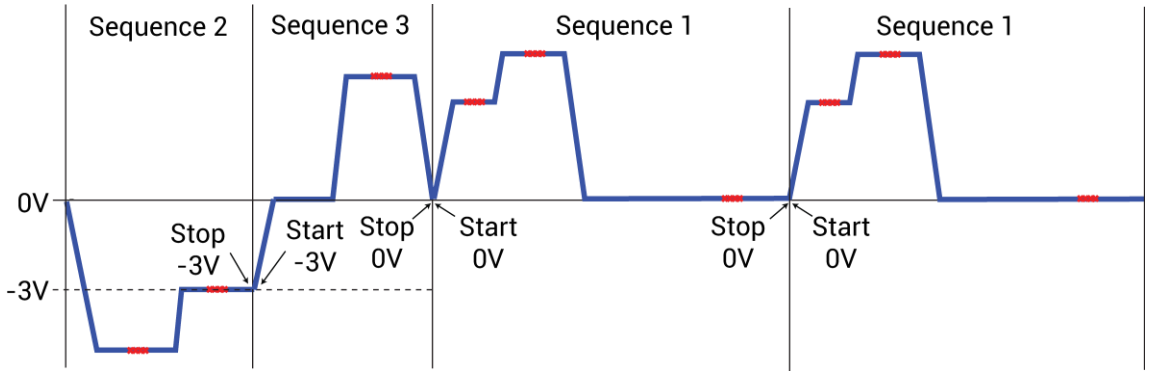
Details

Use this command to create a voltage segment waveform from the sequences defined by the seg_arb_sequence command. 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 command.

You can use this command 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. The figure below 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 606: Three-sequence waveform (with looping)



Example

```
seg_arb_waveform(PMU1, 1, 3, Seq_Num, Seq_Loop_Count);
```

This function configures channel 1 of the PMU for a single three-sequence Segment Arb® waveform (as shown in the figure in the **Details**). This example assumes that the three sequences shown in the figure have already been defined by the `seg_arb_sequence` command.

The arrays for the waveform are:

```
int Seq_Num[3] = {2, 3, 1};
double Seq_Loop_Count[3] = {1, 1, 2};
```

Also see

seg_arb_sequence (on page 14-140)

setmode (4225-PMU)

This command sets operating modes specific to the PMU.

Usage

```
int setmode(int instr_id, long modifier, double value);
```

instr_id	The instrument identification code of the pulse generator: PMU1, PMU2, and so on
modifier	Specific operating characteristic to change; see table in **Details**
value	Parameter value for the *modifier*; see table in **Details**

Pulsers

4225-PMU

Pulse mode

Standard

Details

The `setmode` command allows control over the following 4225-PMU operating characteristics:

- 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 (DUT) resistance and reach the programmed output value at the DUT. This algorithm is not guaranteed to reach the programmed target value. Therefore, there are controls to fine-tune the LLEC performance.

  When enabled, the LLEC algorithm performs a number of iterations to determine the appropriate output voltage. The `pulse_meas_sm` and `pulse_meas_wfm` commands enable or disable LLEC. See Load-line effect compensation (LLEC) (on page 5-44) 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 continues until either the output voltage to the DUT falls within the acceptance window or until the maximum number of iterations are performed.

  The LLEC tolerance window:

  LLEC window = LLC_TOLERANCE * Preferred Voltage + LLC_OFFSET

  LLEC is satisfied when:

  Measured voltage < Preferred voltage ± LLEC Window

  For example, assume the programmed pulse output is 1 V and the acceptance window is set to 0.1 (10%) and offset to 10 mV. LLEC performs iterations until the output voltage falls within the 0.9 V to 1.1 V window. Note that setting a smaller tolerance results in voltage steps that are much closer to the preferred voltage steps sizes, but at the expense of longer test times.

- Offset current compensation. This compensation method is configured by collecting offset current constants from the 4225-PMU and then enabling the constants. Use the `pmu_offset_current_comp` command to collect constants and then enable the constants with the `KI_PMU_Chx_OFFSET_CURR_COMP` command.

Parameters		
modifier	***value***	**Comment**
KI_PXU_LLC_MAX_ITERATIONS	1 to 1000; 20 to 30 typical	Set the maximum number of LLEC iterations
KI_PXU_CHx_LLC_TOLERANCE	0.0001 to 1 (0.01% to 100%); typical range is 0.001 to 0.01 (0.1% to 1%). The typical value is 0.003 (0.3%)	Set the gain of the channel 1 or channel 2 LLEC tolerance window as a percentage of the desired signal level.
KI_PXU_CHx_LLC_OFFSET	0 to 1.0	Sets the channel 1 or channel 2 LLEC DC bias offset.
KI_PMU_CHx_OFFSET_CURR_COMP	0 = OFF 1 = ON	Enable or disable constants for channel 1 or channel 2 offset current compensation.

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

```
setmode(PMU1, KI_PXU_CH1_LLC_TOLERANCE, 0.01);
```

This command sets the LLEC for channel 1 of the PMU for a 1% acceptance window.

Also see

pmu_offset_current_comp (on page 14-103)
pulse_meas_sm (on page 14-119)
pulse_meas_wfm (on page 14-123)
setmode (on page 14-145) (SMU)
setmode (on page 14-209) (4210-CVU)

LPT commands for pulse source only (PG2)

Use the following instrument ID (identification) for LPT commands for the PGU and PMU:

- 4220-PGU: The instrument ID is VPU (VPU1, VPU2, and so on)
- 4225-PMU: The instrument ID is PMU (PMU1, PMU2, and so on)

NOTE

The 4220-PGU and 4225-PMU support the pulsing and external triggering commands of the obsolete 4205-PG2.

arb_array

This command is used to define a full-arb waveform and name the file.

Usage

```
int arb_array(int instr_id, long ch, double TimePerPt, long length, double *levelArr,
    char *fname);
```

instr_id	The instrument identification code, such as VPU1 or VPU2
ch	The pulse card channel: `1` or `2`
TimePerPt	Sets the time interval between waveform points: 20 ns to 1 s
length	The number of waveform points (values): 262,144 maximum
levelArr	An array of voltage values for each point in the waveform (see **Details**)
fname	A name for the full-arb waveform

Pulse modes

Full Arb

Details

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.

This command 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 for each 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.

For additional information on this pulse mode and an example of a Full Arb waveform, refer to Full arb waveform (on page 5-78).

.kaf waveform file for KPulse: You can copy the arbitrary waveform data defined by the `arb_file` command into a `.kaf` file. Use a text editor to format the file. You can then import the `.kaf` file into KPulse. By default, `.kaf` waveform files for KPulse are saved in the ArbFiles folder at the command path location `C:\s4200\kiuser\KPulse\ArbFiles`. Refer to KPulse (on page 11-1) for details on using KPulse.

Also see

arb_file (on page 14-149)
pulse_output (on page 14-165)
pulse_trig (on page 14-172)
seg_arb_define (on page 14-183)

arb_file

This command loads a waveform from an existing full-arb waveform file.

Usage

```
int arb_file(int instr_id, long chan, char *fname);
```

instr_id	The instrument identification code of the pulse card: VPU1, VPU2, and so on
chan	Channel number of the pulse card: `1` or `2`
fname	The name of the waveform file; the name must be in quotes

Details

Use this command to load a waveform from an existing full-arb `.kaf` waveform file into the pulse card. You can load a full-arb waveform for each channel of the pulse 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
```

You can create a full-arb waveform using KPulse, and then save it as a `.kaf` waveform file (refer to KPulse (on page 11-1) for details).

You can modify a waveform in an existing `.kaf` file using a text editor or KPulse.

Example

```
arb_file(VPU1, 1, "C:\\s4200\\kiuser\\KPulse\\ArbFiles\\SINE.kaf")
```

This example loads a full-arb file named `SINE.kaf` (saved in the `ArbFiles` folder) into the pulse card for channel 1.

Also see

arb_array (on page 14-148)
pulse_output (on page 14-165)
pulse_trig (on page 14-172)
seg_arb_file (on page 14-186)
seg_arb_define (on page 14-183)

devclr

This command sets all sources to a zero state.

Usage

```
int devclr(void);
```

Details

This command clears all sources sequentially in the reverse order from which they were originally forced. Before clearing all Keithley supported instruments, GPIB-based instruments are cleared by sending all strings defined with the `kibdefclr` command.`devclr` is implicitly called by `clrcon`, `devint`, `execut`, and `tstdsl`.

For C-V testing, this command turns off the DC bias voltage.

Also see

clrcon (on page 14-229)
devint (on page 14-16)
execut (on page 14-19)
kibdefclr (on page 14-24)
tstdsl (on page 14-54)

devint

This command resets all active instruments in the system to their default states.

Usage

```
int devint(void);
```

Details

Resets all active instruments in the system to their default states. It clears the system by opening all relays and disconnecting the pathways. Meters and sources are reset to their default states. Refer to the hardware manuals for the instruments in your system for listings of available ranges and the default conditions and ranges.

The `devint` command is implicitly called by the `execut` command.

To abort a running `pulse_exec` pulse test, see `dev_abort`.

`devint` does the following:

1. Clears all sources by calling `devclr`.
2. Clears the matrix crosspoints by calling `clrcon`.
3. Clears the trigger tables by calling `clrtrg`.
4. Clears the sweep tables by calling `clrscn`.
5. Resets GPIB instruments by sending the string defined with `kibdefint`.
6. Resets the active instrument cards.

Instrument cards are reset in the following order:

1. SMU instrument cards
2. CVU instrument cards
3. Pulse instrument cards (4225-PMU or 4220-PGU)

`devint` is implicitly called by `execut` and `tstdsl`. `devclr` is implicitly called by `clrcon`.

The SMUs return to the following states:

- 100 µA and 10 V ranges
- Autorange on
- Voltage source
- 0 V DCV bias

The 4210-CVU returns to the following states:

- 30 mV_{RMS} AC signal
- 0 V DCV bias
- 100 kHz frequency
- Autorange on
- Cable length compensation set to 0 m
- Open/Short/Load compensation disabled

The 4225-PMU or 4220-PGU returns to the following states:

- 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.
- 5 V and 10 mA ranges
- If in pulse mode:
  - Period of 1 µs
  - Transition Times (Rise and Fall) of 100 ns
  - Width of 500 ns
  - Voltage high and low of 0 V

  - Load of 50 Ω
- If in segmented arb mode, Start Voltage is 0 V
- If in arbitrary waveform mode, Table Length is 100

Also see

clrcon (on page 14-229)
clrscn (on page 14-11)
clrtrg (on page 14-14)
dev_abort (on page 14-99)
devclr (on page 14-72)
kibdefint (on page 14-26)

getstatus

This command returns the operating state of a specified instrument.

Usage

```
int getstatus(int instr_id, long parameter, double *result);
```

instr_id	The instrument identification code
parameter	The parameter of query; see **Details**
result	The data returned from the instrument; the `getstatus` command returns one item

Details

If you see the `UT_INVLDPRM` invalid parameter error returned from the `getstatus` command, it indicates that the status item parameter is illegal for this device. The requested status code is invalid for the selected device.

A list of supported `getstatus` command values for *parameter* for a source-measure unit (SMU) and a pulse card (VPU) are provided in the following tables.

No status values are provided for measurement-specific conditions.

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	Compliance status of last reading	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	Range compliance status of last reading	Returns 1 if in range compliance

Supported pulse 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
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

Also see

getinstid (on page 14-21)

pg2_init

This command resets the pulse card to the specified pulse mode (standard, full arb, or Segment Arb) and its default conditions.

Usage

```
int pg2_init(int instr_id, long mode);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
mode	The pulse mode: ■ Standard pulse: `0` ■ Segment Arb: `1` ■ Full Arb: `2`

Pulse modes

Standard, Full Arb, Segment Arb

Details

This command resets both channels of the pulse card to the default settings of the specified pulse mode. The default settings for each parameter are listed in the table below.

If you want to reset the pulse card for the presently selected pulse mode, use the `pulse_init` command.

Standard pulse defaults	Full Arb and Segment Arb pulse defaults
Pulse high and pulse low = 0 V Source range = 5 V fast speed Pulse period = 1 µs Pulse width = 500 ns Pulse count = 1 Rise and fall time = 100 ns Pulse delay = 0 s Pulse load = 50 Ω Pulse trigger source = Software Pulse trigger mode = Continuous Pulse trigger output = On* Trigger polarity = Positive Complement mode = Normal pulse Current limit = 105 mA Pulse output = Off	Source range = 5 V fast speed Pulse count = 1 Pulse delay = 0 s Pulse load = 50 Ω Pulse trigger source = Software Pulse trigger mode = Continuous Pulse trigger output = Off* Trigger polarity = Positive Current limit = 105 mA Pulse output = Off
* Turns on when a pulse is initiated with `pulse_trig`	

Example

```
pg2_init(VPU1, 1)
```

Resets the pulse card to the Segment Arb pulse mode and its default settings.

Also see

pulse_init (on page 14-163)

pulse_burst_count

For the burst mode, this command sets the number of pulses to output during a burst sequence.

Usage

```
int pulse_burst_count(int instr_id, long chan, unsigned long count);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: `1` or `2`
count	Number of pulses to output: 1 to (2^{32}-1); default 1

Pulse modes

Standard, Full Arb, Segment Arb

Details

Each channel of the pulse card can have a unique burst count. When a burst sequence is triggered, the card outputs the specified number of pulses and then stops. The `pulse_trig` command is used to start (or arm) the burst sequence (Burst or Trig Burst).

You can set burst count independently for each pulse card channel.

This command can also be used with `pulse_exec`.

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

```
pulse_burst_count(VPU1, 1, 10)
```

Sets the burst count for the pulse card channel 1 to a count of 10.

Also see

pulse_exec (on page 14-107)
pulse_trig (on page 14-172)
pulse_trig_source (on page 14-176)

pulse_current_limit

This command sets the current limit of the pulse card.

Usage

```
int pulse_current_limit(int instr_id, long chan, double ilimit);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: `1` or `2`
ilimit	Current limit value (in amps, range and load dependent): ■ 5 V range, 50 Ω load : –0.2 to +0.2 ■ 20 V range, 50 Ω load: –0.4 to +0.4 ■ 20 V range: –0.8 to +0.8 Default is 5 V range, 0.105

Pulse modes

Standard, Full Arb, Segment Arb

Details

You can set the current limit independently for each pulse card channel.

Current limit protects the DUT by using the specified DUT load to calculate the voltage required to reach the current limit. A pulse card channel will not exceed the voltage required to reach the set current limit value at the specified DUT load.

For information on the effect of loading on the limits, refer to DUT resistance determines pulse voltage across DUT (on page 5-80). For an example and values for load-line effect, refer to Example 5: Maximum voltage and current, high voltage range (on page 5-84).

Example

```
pulse_current_limit(VPU1, 1, 1e-3)
```

Sets the current limit of pulse card channel 1 to 1 mA.

Also see

pulse_load (on page 14-164)

pulse_dc_output

This command selects the DC output mode and sets the voltage level.

Usage

```
int pulse_dc_output(int instr_id, long chan, double dcvalue);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: `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 (50 Ω load) ■ Default: N/A

Pulse modes

Standard

Details

You can set each pulse card channel to output a fixed DC voltage level instead of pulses.

The maximum and minimum output voltage is range dependent. See `pulse_vhigh` and `pulse_vlow` for details.

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 change the voltage level immediately, before the pulsing is started with a `pulse_trig` command.

Example

```
pulse_dc_output(VPU1, 1, 10)
```

Selects channel 1 DCV output and sets voltage to +10 V.

Also see

pulse_load (on page 14-164)
pulse_vhigh (on page 14-178)
pulse_vlow (on page 14-180)

pulse_delay

This command sets the delay time from the trigger to when the pulse output starts.

Usage

```
int pulse_delay(int instr_id, long chan, double delay);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: `1` or `2`
delay	Time delay in seconds: ■ Fast speed: `0` to (Period – 10e-9) ■ Slow speed: `0` to (Period – 10e-9) ■ Default: `0`

Pulse modes

Standard, Full Arb, Segment Arb

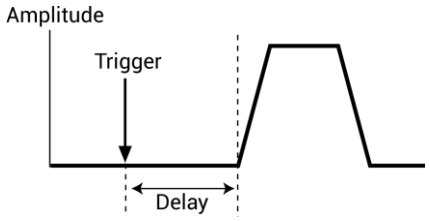
Details

NOTE

Use the `pulse_source_timing` command to set the pulse delay time for the 4220-PGU and 4225-PMU.

Pulse delay can be set independently for each pulse card channel. For both speeds, pulse delay can be set from 0 ns to (Period – 10 ns). The pulse delay is set in 10 ns increments. The `pulse_range` command 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.



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).

Example

```
pulse_delay(VPU1, 1, 300e-9)
```

Sets the pulse delay for channel 1 to 300 ns.

Also see

pulse_period (on page 14-167)
pulse_range (on page 14-168)
pulse_source_timing (on page 14-130)
pulse_trig (on page 14-172)

pulse_fall

This command sets the fall transition time for the pulse output.

Usage

```
int pulse_fall(int instr_id, long chan, double fallt);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: `1` or `2`
fallt	Pulse fall time in seconds (floating point number): ■ Fast speed: 10e-9 to 33e-3 ■ Slow speed: 4220-PGU and 4225-PMU: 50e-9 to 33e-3 ■ Default: 100e-9

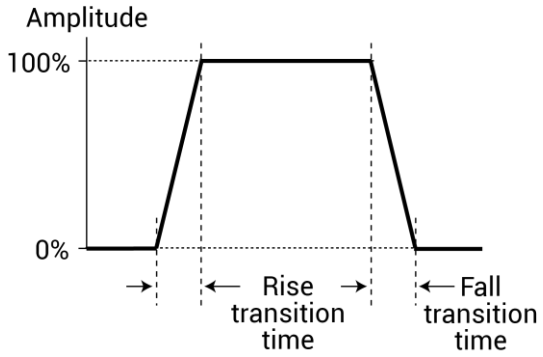
Pulse modes

Standard

Details

Rise and fall transition time can be set independently for each pulse card 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` command 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.



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` command is used to trigger continuous or burst output.

For slow speed, note that the minimum transition time for pulse source only (no measurement) on the 40 V range is 50 ns for the 4225-PMU and 4220-PGU.

NOTE

Use the `pulse_source_timing` command to set the pulse fall time for the 4220-PGU and 4225-PMU.

Example

```
pulse_fall(VPU1, 1, 50e-9)
```

For fast speed, the sets the pulse fall time for channel 1 of the pulse card to 50 ns.

Also see

pulse_range (on page 14-168)
pulse_rise (on page 14-169)
pulse_source_timing (on page 14-130)
pulse_trig (on page 14-172)

pulse_halt

This command stops all pulse output from the pulse card.

Usage

```
int pulse_halt(int instr_id);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2

Pulse modes

Standard, Full Arb, Segment Arb

Details

This command stops all pulse output from the pulse card and turns the pulse card channels off. Pulse output can be restarted by turning the outputs on with `pulse_output` and then using the `pulse_trig` command to restart the test.

Example

```
pulse_halt(VPU1)
```

Stops pulse output.

Also see

pulse_output (on page 14-165)
pulse_trig (on page 14-172)

pulse_init

This command resets the pulse card to the default settings for the pulse mode that is presently selected.

Usage

```
int pulse_init(int instr_id);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2

Pulse modes

Standard, Full Arb, Segment Arb

Details

This command resets both channels of the pulse card to the default settings. The default settings are listed in the following table.

If you want to specify the pulse mode to reset, use the `pg2_init` command.

Standard pulse defaults	Full Arb and Segment Arb pulse defaults
Pulse high and pulse low = 0 V Source range = 5 V fast speed Pulse period = 1 µs Pulse width = 500 ns Pulse count = 1 Rise and fall time = 100 ns Pulse delay = 0 s Pulse load = 50 Ω Pulse trigger source = Software Pulse trigger mode = Continuous Pulse trigger output = On* Trigger polarity = Positive Complement mode = Normal pulse Current limit = 105 mA Pulse output = Off	Source range = 5 V fast speed Pulse count = 1 Pulse delay = 0 s Pulse load = 50 Ω Pulse trigger source = Software Pulse trigger mode = Continuous Pulse trigger output = Off* Trigger polarity = Positive Current limit = 105 mA Pulse output = Off
* Turns on when a pulse is initiated with `pulse_trig`	

Example

```
pulse_init(VPU1)
```

Resets the pulse card to the default settings for the presently selected pulse mode.

Also see

pg2_init (on page 14-155)

pulse_load

This command sets the output impedance for the load (DUT).

Usage

```
int pulse_load(int instr_id, long chan, double load);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: 1 or 2
load	Output impedance (in ohms): 1 to 10e6 (default 50)

Pulse modes

Standard, Full Arb, Segment Arb

Details

DUT impedance can be independently set for each pulse card channel. The DUT impedance can be set from 1 Ω to 10 MΩ, depending on the programmed pulse high and low values.

Maximum power transfer is achieved when the DUT impedance matches the output impedance of the pulse card. 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).

The purpose of setting the DUT load to a value other than 50 Ω is to simplify the calculation of the output levels. For example, if the DUT load is set to 50 Ω, but the actual DUT load has a high impedance of 1 MΩ, setting a voltage level of 2 V will result in a 4 V pulse at the DUT. Setting the DUT load to 1 MΩ 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.

Example

```
pulse_load(VPU1, 1, 100)
```

Sets the output impedance of pulse card channel 1 to 100 Ω.

Also see

None

pulse_output

This command sets the pulse output of a pulse card channel on or off.

Usage

```
int pulse_output(int instr_id, long chan, long out_state);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: `1` or `2`
out_state	Pulse output state: ■ Off: `0` (default) ■ On: `1`

Pulse modes

Standard, Full Arb, Segment Arb

Details

This command configures the channel to output and close the output relay.

If no 4225-RPM is used, this command connects the source to the device under test (DUT). `devclr` resets the pulse card source and disconnects the source from the DUT. `pulse_output(PMU`*x*`, chan, 0)` clears the physical connection to the DUT and resets the PMU source.

If a 4225-RPM is used with the PMU, this command prepares the pulse source when using a PMU with RPMs, but it does not close the output relay. The `rpm_config` command establishes the physical connection to the DUT. The `clrcon` command clears the physical connection to the DUT.

You can control each channel of the pulse card individually (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 starts when a pulse trigger is initiated. Note that if a pulse delay has been set, pulse output starts 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.

The `pulse_ssrc` command controls the high-endurance output relays (HEORs), and the `seg_arb_define` command defines a Segment Arb® waveform, which includes HEOR control.

Example

```
pulse_output(VPU1, 1, 0)
```

Turns off the output for pulse card channel 1.

Also see

clrcon (on page 14-229)
devclr (on page 14-72)
pulse_delay (on page 14-159)
pulse_ssrc (on page 14-171)
pulse_trig (on page 14-172)
pulse_current_limit (on page 14-157)
rpm_config (on page 14-139)
seg_arb_define (on page 14-183)

pulse_output_mode

This command sets the pulse output mode of a pulse card channel.

Usage

```
int pulse_output_mode(int instr_id, long chan, long mode);
```

instr_id	The instrument identification code of the pulse card, such as `VPU1` or `VPU2`
chan	Channel number of the pulse card: `1` or `2`
mode	Pulse output state: ■ `NORMAL` or `0` (default) ■ `COMPLEMENT` or `1`

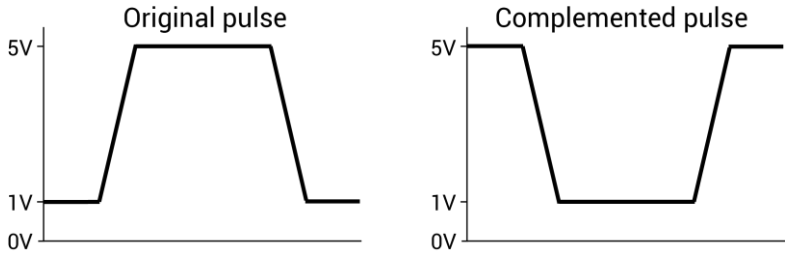
Pulse modes

Standard

Details

When a pulse card channel is set to `COMPLEMENT`, the V Low and V High voltage settings are swapped.

As shown in the following figure, when pulse is complemented, low pulse goes to the high level, and high pulse goes to the low level.



Example

```
pulse_output_mode(VPU1, 1, COMPLEMENT)
```

Sets the output mode for pulse card channel 1 to `COMPLEMENT`.

Also see

None

pulse_period

This command sets the period for pulse output.

Usage

```
int pulse_period(int instr_id, double period);
```

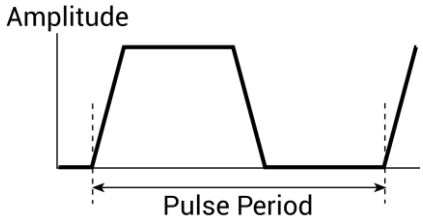
instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
period	Pulse period (in seconds): ■ 5 V range: 20e-9 to 1 ■ 20 V range: 500e-9 to 1 ■ Default: 1e-6

Pulse modes

Standard

Details

This command sets the pulse period for both channels of the pulse card. As shown below, the pulse period is measured at the median point (50 percent between the high and low pulse values) from the rising transition of the pulse to the rising transition of the next pulse.



The pulse period setting takes effect immediately during continuous pulse output. Otherwise, the period setting takes effect when the next trigger is initiated. The `pulse_trig` command is used to trigger continuous or burst output.

Example

```
pulse_period(VPU1, 200e-9)
```

Sets the pulse period of the pulse card to 200 ns.

Also see

pulse_trig (on page 14-172)

pulse_range

This command sets a pulse card channel for low voltage (fast speed) or high voltage (slow speed).

Usage

```
int pulse_range(int instr_id, long chan, double range);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: `1` or `2`
range	Pulse range (in volts): `5` or `20` (default 5 V)

Details

Setting the pulse range of a pulse card 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 and fall times can be set to 10 ns.

Setting the pulse range of a pulse card 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 and 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

Use `pulse_range` before setting the voltage levels. When you use the `pulse_range` command, if you change the source range after setting the voltage levels in any pulse mode, it may result in voltage levels that are invalid for the new range setting.

NOTE

This command can also be used to set the voltage source range of the 4220-PGU and 4225-PMU. Use the `pulse_ranges` command to set the source and measure ranges of the 4225-PMU.

Example

```
pulse_range(VPU1, 1, 20)
```

Selects the high-voltage (slow speed) range for pulse card channel 1.

Also see

pulse_fall (on page 14-161)
pulse_vhigh (on page 14-178)
pulse_vlow (on page 14-180)
pulse_period (on page 14-167)
pulse_ranges (on page 14-125)
pulse_rise (on page 14-169)
pulse_width (on page 14-182)

pulse_rise

This command sets the rise transition time for the pulse card pulse output.

Usage

```
int pulse_fall(int instr_id, long chan, double riset);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: `1` or `2`
riset	Pulse rise time in seconds (floating point number): ■ Fast speed: 10e-9 to 33e-3 ■ Slow speed, 4220-PGU and 4225-PMU: 50e-9 to 33e-3 ■ Default: 100e-9

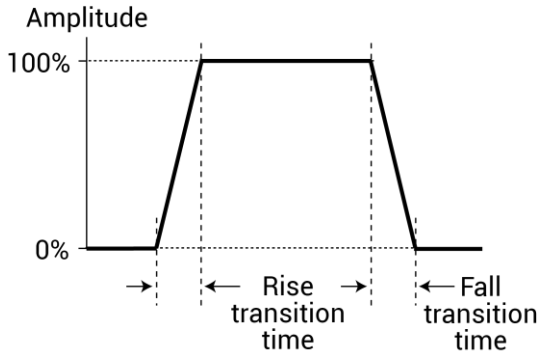
Pulse modes

Standard

Details

Rise and fall transition time can be set independently for each pulse card 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` command is used to set pulse speed.

As shown below, the pulse rise time occurs between 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 pulse values.



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 `pulse_trig` command is used to trigger continuous or burst output.

For slow speed, note that the minimum transition time for pulse source only (no measurement) on the 40 V range is 50 ns for the 4225-PMU and 4220-PGU.

NOTE

Use the `pulse_source_timing` command to set the pulse fall time for the 4220-PGU and 4225-PMU.

Example

```
pulse_rise(VPU1, 1, 50e-9)
```

For fast speed, the sets the pulse rise time for channel 1 of the pulse card to 50 ns.

Also see

pulse_fall (on page 14-161)
pulse_range (on page 14-168)
pulse_source_timing (on page 14-130)
pulse_trig (on page 14-172)

pulse_ssrc

This command controls the high-endurance output relay (HEOR) for each output channel of the PGU.

Usage

```
int pulse_ssrc(int instr_id, long chan, long state, long ctrl);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: 1 or 2
state	Open: 0 Close: 1 (default)
ctrl	How the HEOR will be controlled: ■ Auto (the Segment Arb pulse mode controls the HEOR): 0 (default) ■ Manual (*state* parameter opens or closes relay): 1 ■ Trigger out driven (relay state follows the trigger output): 2

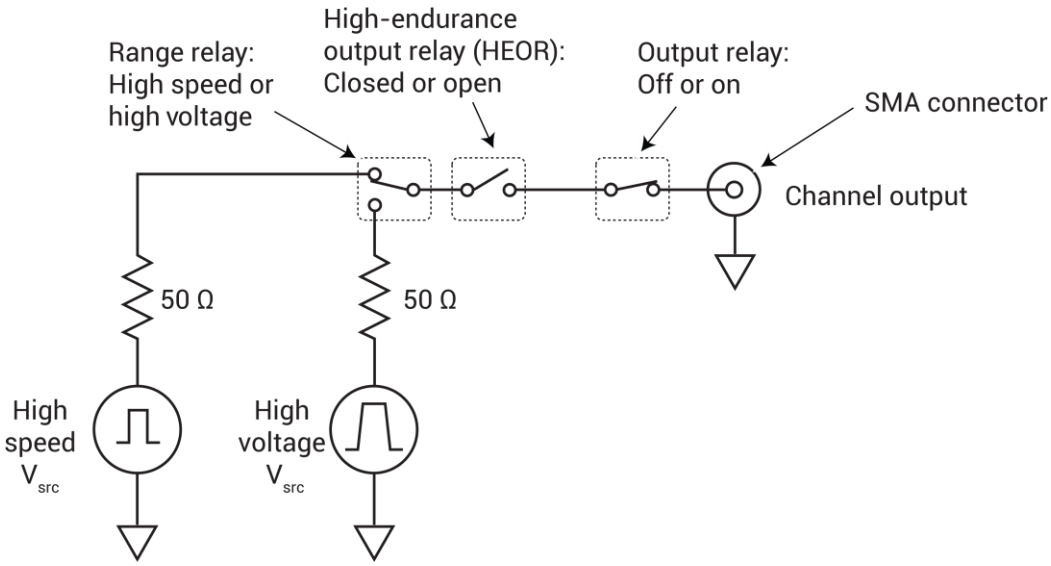
Pulse modes

Standard, Full Arb, Segment Arb

Details

The high-endurance output relay (HEOR) is a solid-state relay (SSR) on each channel of the pulse card. Note that this setting is independent of the output relay (see pulse_output). A simplified schematic showing the relays is shown here.

Figure 607: Simplified schematic of a 4220-PGU channel



Example

```
pulse_ssrc(VPU1, 1, 0, 1)
```

Selects manual control and opens the relay.

Also see

pulse_output (on page 14-165)
seg_arb_define (on page 14-183)
seg_arb_file (on page 14-186)

pulse_trig

This command selects the trigger mode (continuous, burst, or trigger burst) and initiates the start of pulse output or arms the pulse card.

Usage

```
int pulse_trig(int instr_id, long mode);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
mode	Trigger mode: ■ Burst: 0 ■ Continuous: 1 ■ Trigger burst: 2

Pulse modes

Standard, Full Arb, Segment Arb

Details

With the software trigger source selected, this command sets the trigger mode (continuous, burst, or trig burst) for both pulse card channels, and initiates the start of pulse output.

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. 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` command is used to set the burst count.

NOTE

See Triggering (on page 5-86) 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 more than 0, pulse output will start after the delay period expires

This setting affects both output channels.

Example

```
pulse_trig(VPU1, 0)
```

Initiates (triggers) burst pulse output.

Also see

pulse_burst_count (on page 14-156)
pulse_delay (on page 14-159)
pulse_halt (on page 14-162)
pulse_output (on page 14-165)
pulse_trig_source (on page 14-176)

pulse_trig_output

This command sets the output trigger on or off.

Usage

```
int pulse_trig_output(int instr_id, long state);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
state	Output trigger state: ■ Off: 0 (default for Segment Arb and full arb) ■ On: 1 (default for standard pulse)

Pulse modes

Standard, Full Arb, Segment Arb

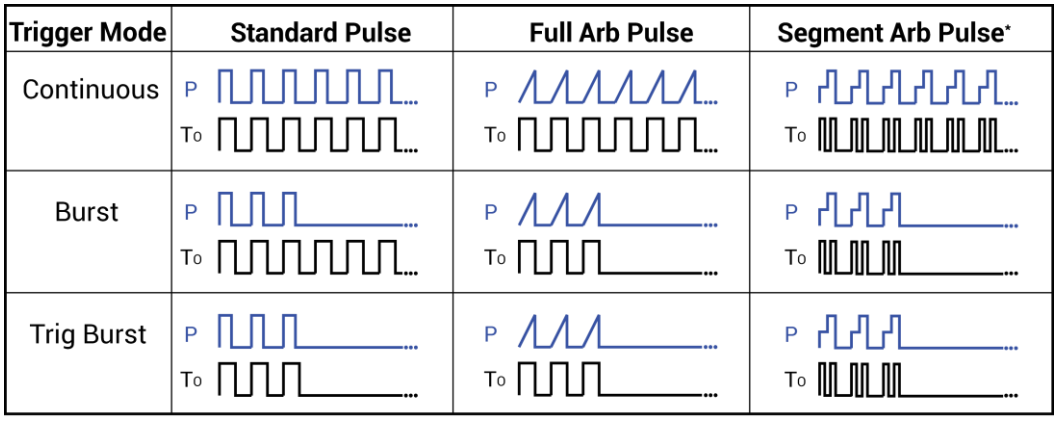
Details

This command turns the TTL-level trigger output pulse on or off. The pulse is used to synchronize pulse output with the operations of an external instrument. When connected to a scope, each output pulse of triggers a scope waveform measurement.

When output trigger is 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 figure below 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 trigger burst mode, output triggers stop when the pulse output stops.

Figure 608: Pulse generator card output trigger



P = Pulse output
T_0 = Trigger output

*Segment Arb has user defined trigger output (0 or 1) for each segment.

Example

```
pulse_trig_output(VPU1, 1)
```

Sets the pulse card trigger output on.

Also see

Pulse generator card output trigger (on page 5-88)
pulse_trig_polarity (on page 14-175)

pulse_trig_polarity

This command sets the polarity (positive or negative) of the pulse card output trigger.

Usage

```
int pulse_trig_polarity(int instr_id, long polarity);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
polarity	Output trigger polarity: ■ Negative, falling edge: `0` ■ Positive, rising edge: `1` ■ Default: `1`

Pulse modes

Standard, Full Arb, Segment Arb

Details

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. It is used to synchronize pulse outputs with the operations of an external instrument.

The external instrument that is connected to the pulse card external trigger may require a positive-going (rising-edge) pulse or a negative-going (falling-edge) pulse for triggering.

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

4220-PGU and 4225-PMU: Do not use the two external falling trigger sources (`pulse_trig_source` function) with the positive trigger output polarity (`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 used to supply the trigger pulses to the 4220-PGU and 4225-PMU. This applies to all three pulse modes (standard pulse, Segment Arb, and full arb).

Example

```
pulse_trig_polarity(VPU1, 0)
```

Sets the pulse card trigger output for negative polarity.

Also see

pulse_trig_output (on page 14-174)
pulse_trig_source (on page 14-176)
Triggering (on page 5-86)

pulse_trig_source

This command sets the trigger source.

Usage

```
int pulse_trig_source(int instr_id, long source);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
source	Trigger source: ■ Software: 0 (default) ■ External – initial trigger only – rising: 1 ■ External – initial trigger only – falling: 2 ■ External – trigger per pulse – rising: 3 ■ External – trigger per pulse – falling: 4 ■ Internal trigger bus: 5

Pulse modes

Standard, Full Arb, Segment Arb

Details

This command sets the trigger source that is used to trigger the pulse card to start its output.

If the software trigger source selected, the `pulse_trig` command will select the trigger mode (continuous, burst, or trig burst), and initiate the start of pulse output.

If an external trigger source selected, the `pulse_trig` command 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 of the pulse card. 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.

NOTE

4220-PGU and 4225-PMU: Do not use the two external falling trigger sources (`pulse_trig_source` function) with the positive trigger output polarity (`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 used to supply the trigger pulses to the 4220-PGU and 4225-PMU. This applies to all three pulse modes (standard pulse, Segment Arb, and full arb).

NOTE

Because 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.

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.

There are four Trigger In 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 commands (`pulse_vhigh`, `pulse_vlow`, `pulse_width`, and so on). This trigger source is used only by the 4220-PGU and 4225-PMU.

The internal bus trigger source is used for synchronizing multiple PMU/PGU cards for standard pulse using the legacy pulse commands (`pulse_vhigh`, `pulse_vlow`, `pulse_width`, and so on). This trigger source is used only by the 4220-PGU and 4225-PMU.

Example

```
pulse_trig_source(VPU1, 1)
```

Sets the trigger source to external – initial trigger only – rising.

Also see

pulse_trig (on page 14-172)
pulse_trig_polarity (on page 14-175)

pulse_vhigh

This command sets the pulse voltage high level.

Usage

```
int pulse_vhigh(INSTR_ID instr_id, long chan, double vhigh);
```

instr_id	The instrument identification code, such as VPU1 or VPU2
chan	Channel number of the pulse card: 1 or 2
vhigh	Pulse voltage high value in volts (floating point number): ■ Fast speed: –5 to +5 ■ Slow speed: –20 to +20 ■ Default: 0

Pulse modes

Standard

Details

Pulse voltage high can be set independently for each pulse card channel.

For a 50 Ω load:

- 5 V range (lower voltages and higher transitions): Pulse high and pulse low can be set from –5 V to +5 V.
- 20 V range (higher voltages and lower transitions): Pulse high and pulse low can be set from –20 V to +20 V.

For a 1 MΩ 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.

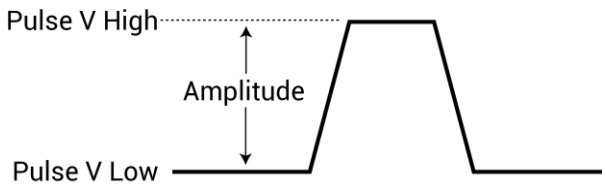
The `pulse_range` command sets the pulse voltage range.

NOTE

Set the `pulse_range` command before setting the voltage levels. When using the `pulse_range` command, changing the source range after setting voltage levels (in any pulse mode) will result in voltage levels that are invalid for the new range setting.

As shown below, the pulse voltage high is typically set as the greater pulse voltage value. However, voltage high can be any valid voltage value. That means pulse voltage high can be less than voltage low. When started, the pulse transitions from voltage low to voltage high and then back to voltage low. The voltage remains at voltage low for the remainder of the pulse period.

Figure 609: Pulse V Low and Pulse V High



The pulse voltage high setting takes effect immediately during continuous pulse output. Otherwise, the voltage high setting takes effect when the next trigger is initiated. The `pulse_trig` command is used to trigger continuous or burst output.

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 change the voltage level immediately, before the pulsing is started with a `pulse_trig` command.

Example

```
pulse_vhigh(VPU1, 1, 2.5)
```

Sets the pulse voltage high value for channel 1 of the pulse card to 2.5 V.

Also see

pulse_dc_output (on page 14-158)
pulse_output (on page 14-165)
pulse_range (on page 14-168)
pulse_trig (on page 14-172)
pulse_vlow (on page 14-180)

pulse_vlow

This command sets the pulse voltage low value.

Pulse modes

Standard

Usage

```
int pulse_lhigh(int instr_id, long chan, double vlow);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: `1` or `2`
vlow	Pulse voltage low value in volts (floating point number): ■ Fast speed: –5 to +5 ■ Slow speed: –20 to +20 ■ Default: 0

Details

Pulse voltage low can be set independently for each pulse card channel.

For a 50 Ω load:

- 5 V range (lower voltages and higher transitions): Pulse high and pulse low can be set from –5 V to +5 V.
- 20 V range (higher voltages and lower transitions): Pulse high and pulse low can be set from –20 V to +20 V.

For a 1 MΩ 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.

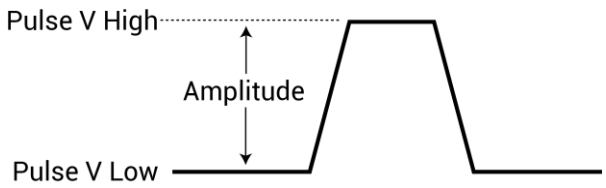
The `pulse_range` command determines the pulse voltage range.

NOTE

Set the `pulse_range` command before setting the voltage levels. When using the `pulse_range` command, changing the source range after setting voltage levels (in any pulse mode) will result in voltage levels that are invalid for the new range setting.

As shown below, the pulse voltage low is typically set as the lower pulse voltage value. However, voltage low can be any valid voltage value. That means pulse voltage low can be less than voltage high. When started, the pulse transitions from voltage low to voltage high and then back to voltage low. The voltage remains at voltage low for the remainder of the pulse period.

Figure 610: Pulse V Low and Pulse V High



The pulse voltage low setting takes effect immediately during continuous pulse output. Otherwise, the voltage low setting takes effect when the next trigger is initiated. The `pulse_trig` command is used to trigger continuous or burst output.

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 change the voltage level immediately, before the pulsing is started with a `pulse_trig` command.

Example

```
pulse_vlow(VPU1, 1, 0.5)
```

Sets the pulse voltage low value for channel 1 of the pulse card to 0.5 V.

Also see

pulse_dc_output (on page 14-158)
pulse_output (on page 14-165)
pulse_range (on page 14-168)
pulse_trig (on page 14-172)
pulse_vhigh (on page 14-178)

pulse_width

This command sets the pulse width for pulse output.

Usage

```
int pulse_width(int instr_id, long chan, double width);
```

instr_id	The instrument identification code of the pulse card, such as `VPU1` or `VPU2`
chan	Channel number of the pulse card: `1` or `2`
width	Pulse width in seconds: ■ Fast speed (5 V): 10e-9 to (Period – 10e-9) ■ Slow speed (20 V): 250e-9 to (Period – 10e-9) ■ Default: 500e-9

Pulse modes

Standard

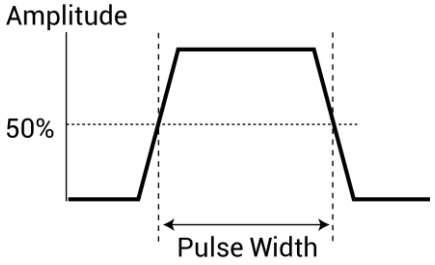
Details

NOTE

Use the `pulse_source_timing` command to set the pulse width for the 4220-PGU and 4225-PMU.

You can set the pulse width independently for each pulse card channel. The `pulse_range` command is used to set pulse speed.

Pulse card pulse width is based on the full width at half-maximum 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.



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 fast speed 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` command is used to trigger continuous or burst output.

Example

```
pulse_width(VPU1, 1, 250e-9)
```

Sets the pulse width for channel 1 to 250 ns.

Also see

pulse_period (on page 14-167)
pulse_range (on page 14-168)
pulse_source_timing (on page 14-130)
pulse_trig (on page 14-172)

seg_arb_define

This command defines the parameters for a Segment Arb® waveform.

Usage

```
int seg_arb_define(int instr_id, long chan, long nsegments, double *startvals, double
   *stopvals, double *timevals, long *triggervals, long *outputRelayVals);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: `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: ■ Trigger low: `0` ■ Trigger high: `1`
outputRelayVals	An array of values to control the high endurance output relay: ■ Open: `0` ■ Closed: `1`

Pulsers

4220-PGU
4225-PMU

Pulse modes

Source, Segment Arb

Details

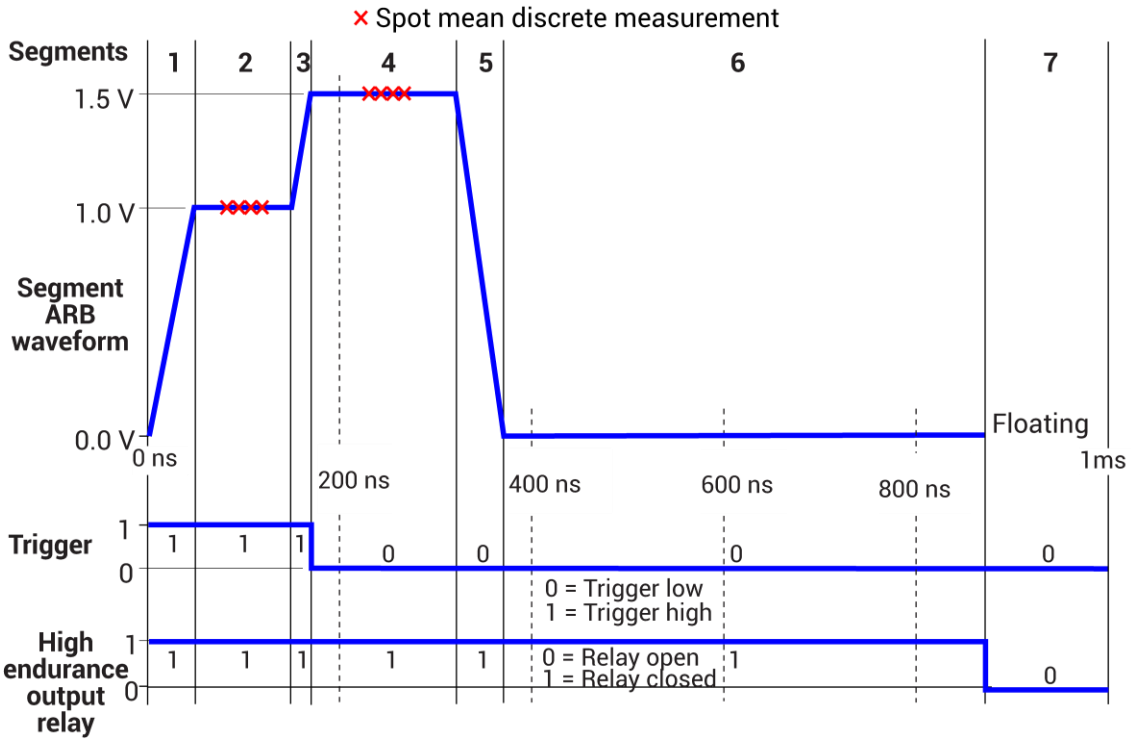
You can configure each channel to output its own unique Segment Arb waveform. A Segment Arb waveform is made up of user-defined segments. Each segment can have a unique time interval, start value, stop value, output trigger level (TTL high or low), and output relay state (open or closed).

See Segment Arb (on page 5-72) for details on this pulse mode. For triggering, refer to Pulse-measure synchronization (on page 5-87).

To configure each channel to output a unique Segment Arb® waveform, refer to seg_arb_sequence (on page 14-140).

The following arrays are required for the example Segment Arb waveform shown here.

Figure 611: Segment Arb sequence example



Start	Stop	Time	Trigger	Output relay
startvals[0] = 0.0	stopvals[0] = 1.0	timevals[0] = 50e-9	triggervals[0] = 1	outputRelayVals[0] = 0
startvals[1] = 1.0	stopvals[1] = 1.0	timevals[1] = 100e-9	triggervals[1] = 1	outputRelayVals[1] = 0
startvals[2] = 1.0	stopvals[2] = 1.5	timevals[2] = 20e-9	triggervals[2] = 1	outputRelayVals[2] = 0
startvals[3] = 1.5	stopvals[3] = 1.5	timevals[3] = 150e-9	triggervals[3] = 0	outputRelayVals[3] = 0
startvals[4] = 1.5	stopvals[4] = 0.0	timevals[4] = 50e-9	triggervals[4] = 0	outputRelayVals[4] = 0
startvals[5] = 0.0	stopvals[5] = 0.0	timevals[5] = 500e-9	triggervals[5] = 0	outputRelayVals[5] = 0
startvals[6] = 0.0	stopvals[6] = 0.0	timevals[6] = 130e-9	triggervals[6] = 0	outputRelayVals[6] = 1

Also see

arb_file (on page 14-149)
arb_array (on page 14-148)
seg_arb_file (on page 14-186)

seg_arb_file

This command is used to load a waveform from an existing Segment Arb® waveform file.

Usage

```
int seg_arb_file(INSTR_ID instr_id, long chan, char *fname);
```

instr_id	The instrument identification code of the pulse card, such as VPU1 or VPU2
chan	Channel number of the pulse card: `1` or `2`
fname	The name of the waveform file; name must be in quotes

Pulse modes

Source only, Segment Arb

Details

This command loads a waveform from an existing Segment Arb `.ksf` waveform file into the pulse card. A Segment Arb waveform can be loaded for each channel of the pulse card. Once loaded, use `pulse_output` to turn on the appropriate channel. Use `pulse_trig` to select the trigger mode and start (or arm) pulse output.

When specifying the file name, 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 saved as a `.ksf` waveform file (refer to KPulse (for Keithley Pulse Generator Cards) (on page 11-1) for details).

You can modify a waveform in an existing `.ksf` file using a text editor.

Example

```
seg_arb_file(VPU1, 1, "C:\\s4200\\kiuser\\KPulse\\SarbFiles\\sarb3.ksf")
```

Loads a Segment Arb file named `sarb3.ksf` (saved in the `SarbFiles` folder) into the pulse card for channel 1.

Also see

arb_array (on page 14-148)
arb_file (on page 14-149)
pulse_output (on page 14-165)
pulse_trig (on page 14-172)
seg_arb_define (on page 14-183)

LPT commands for the CVUs

The LPT commands for the 4210-CVU are listed in CVU commands (on page 14-10). LPT command details are presented here in alphabetic order.

adelay

This command specifies an array of delay points to use with `asweepX` command calls.

Usage

```
int adelay(long numberOfPoints, double *delayArray);
```

numberOfPoints	Total number of sweep points
delayArray	An array of delay values (in seconds)

Details

NOTE

This command can be used with any of the `asweepX` commands. The following information pertains specifically to the 4210-CVU.

This command 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. For example: Assume `asweepv` is configured to sweep four points, 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` command would be configured as follows:

```
delayArray(0) = 0.5
delayArray(1) = 0.25
delayArray(2) = 0.5
delayArray(3) = 0.25
```

Example

See Programming example #5 (on page 14-227), which shows how to set up an array of delay times for a voltage array sweep.

Also see

asweepX (on page 14-63)

asweepv

This command does a DC voltage sweep using an array of voltage values.

Usage

```
int asweepv(int instr_id, long numberOfPoints, double delayTime, double *forceArray);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
numberOfPoints	Total number of sweep points (1 to 4096)
delayTime	Delay time before each measurement (0 to 999 s)
forceArray	Array of DC voltage values

Details

NOTE

The following supplemental information on the voltage array sweep pertains specifically to the 4210-CVU. See `asweepX` in LPT commands for SMUs and general operations (on page 14-62) for additional information.

This command performs a DC voltage sweep using an array of voltage values. The number of voltage values in the array must match the *numberOfPoints* parameter value.

The *delayTime* parameter sets the user-programmed delay before each measurement. Note that there is an additional inherent system delay 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` commands to set the AC drive frequency and voltage for the sweep.

Example

Refer to Programming example #4 (on page 14-226) for an example of a voltage array sweep.

Also see

adelay (on page 14-187)
asweepX (on page 14-63)
dsweepf (on page 14-195)
dsweepv (on page 14-197)
sweepf (on page 14-220)
setfreq (on page 14-208)
setlevel (on page 14-209)
sweepv (on page 14-222)

bsweepX

This command supplies a series of ascending or descending voltages or currents and shuts down the source when a trigger condition is encountered.

Usage

```
int bsweepi(int instr_id, double startval, double endval, long num_points, double
    delay_time, double *result);
int bsweepv(int instr_id, double startval, double endval, long num_points, double
    delay_time, double *result);
```

instr_id	The instrument identification code of the sourcing instrument
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 the *startval* and *endval* parameters (1 to 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

Details

`bsweepi` is only available for SMUs.

The `bsweepX` command is used with the `trigXg`, `trigXl`, or `trigcomp` command. These trigger commands 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, the `bsweepX` command terminates the sweep and clears all sources by executing a `devclr` command internally. The standard `sweepX` command continues to force the last value. This is useful for device characterization curves but can cause problems when used in device breakdown conditions.

The `bsweepX` command can also be used with the `smeasX`, `sintgX`, `savgX`, or `rtfary` command. Measurements are stored in a one-dimensional array in the order in which they were made.

The system maintains a measurement scan table consisting of devices to test. This table is maintained using calls to the `smeasX`, `sintgX`, `savgX`, or `clrscn` command. As multiple calls to `sweepX` commands are made, these commands are appended to the measurement scan table. Measurements are made after the time programmed by the *`delay_time`* parameter has elapsed at the beginning of each `bsweepX` command step.

When multiple calls to the `bsweepX` command are executed in the same test sequence, the arrays defined by calls to the `smeasX`, `sintgX`, or `savgX` command are all loaded sequentially. The results from the second call to the `bsweepX` command are appended to the results of the previous `bsweepX` command call. 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`, `execut`, or `clrscn` command completes.

Defining new test sequences using the `smeasX`, `sintgX`, or `savgX` command adds the command to the active measure list. The previous measurements are still defined and used; however, previous measurements for the second sweep can be eliminated by calling the `clrscn` command. New measurements are defined and used by calling the `smeasX`, `sintgX`, or `savgX` command after a `clrscn` command.

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 rangeX (on page 14-88) for recommended command order.

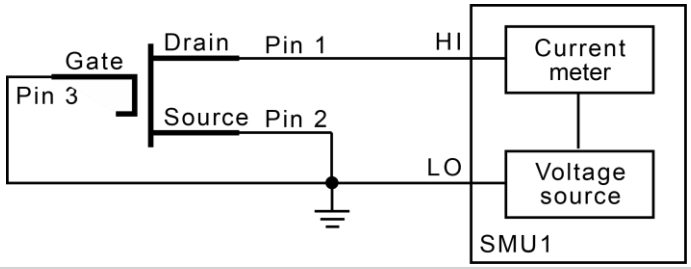
NOTE

It is recommended that you do not use GPIB instruments when doing sweeps with the `bsweepX` command. Refer to kibdefint (on page 14-26) for additional information.

Example

```
double bvdss;
.
.
conpin(SMU1, 1, 0);
conpin(GND, 2, 3, 0);
limiti(SMU1, 100e-6); /* Define the I limit for the device. */
rangei(SMU1, 100e-6); /* Select a fixed range */
 /* measurement. */
trigil(SMU1, -10e-6); /* Set the trigger point to -10 uA. */
bsweepv(SMU1, 10.0, 50.0, 40, 10.0e-3, &bvdss); /* Sweep */
 /* from 10 V to 50 V in 40 */
 /* steps with 10 ms settling */
 /* time per step. */
```

This example measures the drain to source breakdown voltage of a field-effect transistor (FET). A linear voltage sweep is generated from 10.0 V to 50.0 V by SMU1 using the `bsweepv` command. The breakdown current is set to 10 mA by using the `trigil` command. The voltage at which this current is exceeded is stored in the variable `bvdss` and is returned to the application processor by the `execut` command.



Also see

clrscn (on page 14-11)
devclr (on page 14-72)
execut (on page 14-19)
rtfary (on page 14-38)
savgX (on page 14-39)
sintgX (on page 14-47)
smeasX (on page 14-49)
sweepX (on page 14-95)
trigXg, trigXl (on page 14-51)
trigcomp (on page 14-50)

cvu_custom_cable_comp

This command determines the delays needed to accommodate custom cable lengths.

Usage

```
cvu_custom_cable_comp(int instr_id);
```

instr_id	The instrument identification code of the CVU (`CVU1`)

Details

The custom cable length measure gathers a specific set of timing coefficients to be applied during the C-V testing for a custom length cable. They are used to compensate the calibrated measurements made from the CVU.

Custom cable lengths are any lengths that are not 0 m, 1.5 m, or 3 m.

Once this command is run, these values are applied if you select a cable length of Custom in Clarius Tools > CVU Connection Compensation.

Possible return values are:

- `0`: OK
- `-907`: LPOT/LCUR fail
- `-908`: HPOT/HCUR fail

Also see

Connection compensation (on page 4-14)

devclr

This command sets all sources to a zero state.

Usage

```
int devclr(void);
```

Details

This command clears all sources sequentially in the reverse order from which they were originally forced. Before clearing all Keithley supported instruments, GPIB-based instruments are cleared by sending all strings defined with the `kibdefclr` command.`devclr` is implicitly called by `clrcon`, `devint`, `execut`, and `tstdsl`.

For C-V testing, this command turns off the DC bias voltage.

Also see

clrcon (on page 14-229)
devint (on page 14-16)
execut (on page 14-19)
kibdefclr (on page 14-24)
tstdsl (on page 14-54)

devint

This command resets all active instruments in the system to their default states.

Usage

```
int devint(void);
```

Details

Resets all active instruments in the system to their default states. It clears the system by opening all relays and disconnecting the pathways. Meters and sources are reset to their default states. Refer to the hardware manuals for the instruments in your system for listings of available ranges and the default conditions and ranges.

The `devint` command is implicitly called by the `execut` command.

To abort a running `pulse_exec` pulse test, see `dev_abort`.

`devint` does the following:

1. Clears all sources by calling `devclr`.
2. Clears the matrix crosspoints by calling `clrcon`.
3. Clears the trigger tables by calling `clrtrg`.
4. Clears the sweep tables by calling `clrscn`.
5. Resets GPIB instruments by sending the string defined with `kibdefint`.
6. Resets the active instrument cards.

Instrument cards are reset in the following order:

1. SMU instrument cards
2. CVU instrument cards
3. Pulse instrument cards (4225-PMU or 4220-PGU)

`devint` is implicitly called by `execut` and `tstdsl`. `devclr` is implicitly called by `clrcon`.

The SMUs return to the following states:

- 100 µA and 10 V ranges
- Autorange on
- Voltage source
- 0 V DCV bias

The 4210-CVU returns to the following states:

- 30 mV_{RMS} AC signal
- 0 V DCV bias
- 100 kHz frequency
- Autorange on
- Cable length compensation set to 0 m
- Open/Short/Load compensation disabled

The 4225-PMU or 4220-PGU returns to the following states:

- 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.
- 5 V and 10 mA ranges
- If in pulse mode:
  - Period of 1 µs
  - Transition Times (Rise and Fall) of 100 ns
  - Width of 500 ns
  - Voltage high and low of 0 V

  - Load of 50 Ω
- If in segmented arb mode, Start Voltage is 0 V
- If in arbitrary waveform mode, Table Length is 100

Also see

clrcon (on page 14-229)
clrscn (on page 14-11)
clrtrg (on page 14-14)
dev_abort (on page 14-99)
devclr (on page 14-72)
kibdefint (on page 14-26)

dsweepf

This command performs a dual frequency sweep.

Usage

```
int dsweepf(int instr_id, double startf, double stopf, long *NumPts, double delaytime);
```

instr_id	The instrument identification code of the 4210-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)

Details

NOTE

Use the `sweepf` command to perform a single frequency sweep.

This command is used to perform a dual frequency sweep (see **Example**). The 4210-CVU provides test frequencies from 1 kHz to 10 MHz in the following steps:

- 1 kHz through 10 kHz in 1 kHz 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 is selected (for example, 15 kHz input selects 20 kHz). If a specified frequency is equidistant from two adjacent frequencies, it is rounded 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

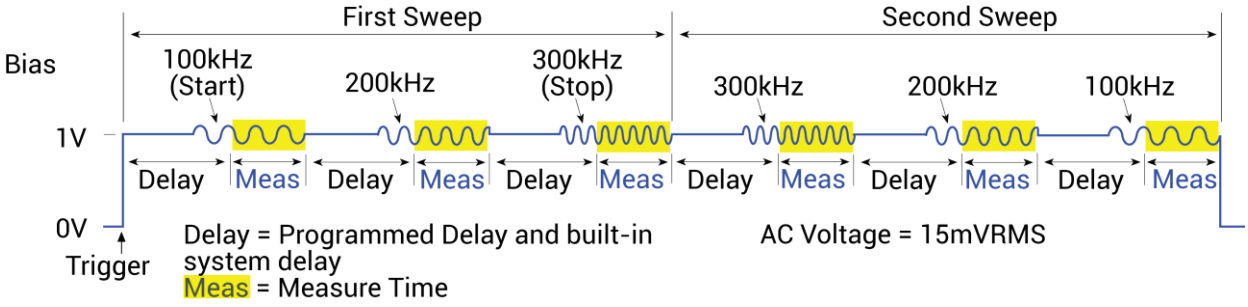
The total number of sweep points is returned in the *NumPts* parameter. For the above example, *NumPts* is assigned a value of 10.

The *delayTime* parameter sets the delay that occurs before each measurement. Note that there is an inherent system overhead delay on each frequency step of the sweep.

Use the `forcev` command to set the DC bias level and `setlevel` command to set the AC drive voltage.

Example

Figure 612: Dual frequency sweep example



Also see

forcev (on page 14-198)
setlevel (on page 14-209)
sweepf (on page 14-220)

dsweepv

This command performs a dual linear staircase voltage sweep.

Usage

```
int dsweepv(int instr_id, double startv, double stopv, long numSteps, double delaytime);
```

instr_id	The instrument identification code of the 4210-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); see **Details**
delaytime	Delay before each measurement (0 to 999 s)

Details

This command is used to perform a dual staircase sweep (see the figure below). The linear step size to sweep is set using the *startv*, *stopv*, and *NumSteps* parameters. The linear step size for the sweep is then calculated as follows:

StepSize (in volts) = (*stopv* – *startv*) / (*numSteps*)

numSteps 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, set *numSteps* to 10. The result is a 20-point sweep (10 up and 10 down).

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.

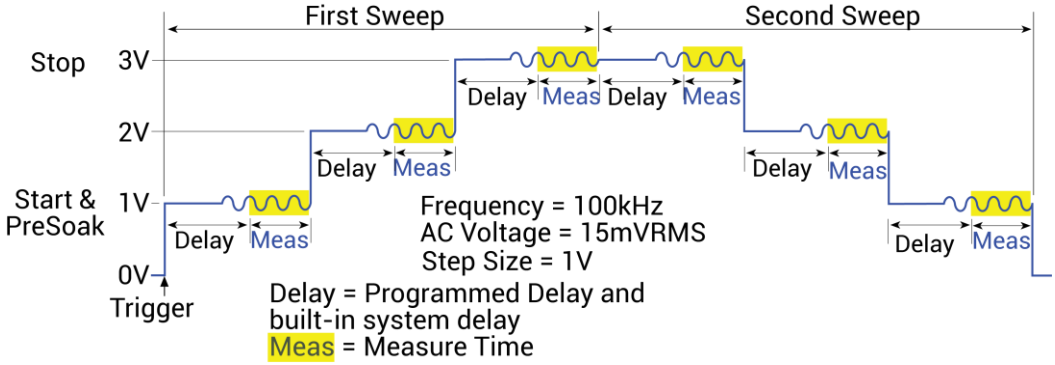
The *delayTime* parameter sets the delay that occurs before each measurement. Note that there is an inherent system overhead delay on each step of the sweep.

Use the `setfreq` and `setlevel` commands to set the AC drive frequency and voltage for the sweep.

NOTE

Use the `sweepv` command to perform a single linear staircase voltage sweep.

Example



Also see

asweepv (on page 14-188)
dsweepf (on page 14-195)
setfreq (on page 14-208)
setlevel (on page 14-209)
sweepf (on page 14-220)
sweepv (on page 14-222)

forcev

This command sets the DC bias voltage level.

Usage

```
int forcev(int instr_id, double voltage);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
voltage	DC bias voltage level (–30 V to 30 V)

Details

This command sets a DC bias level for a single impedance measurement and a frequency sweep. Use the `setfreq` and `setlevel` commands 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; the AC frequency and source value are only used during `measz` operations.

Example

Programming example #1 (on page 14-223) makes a single impedance measurement. Note that the `rdelay` command provides a settling time before the measurement.

Also see

measz (on page 14-204)
setfreq (on page 14-208)
setlevel (on page 14-209)

getstatus

This command returns various parameters pertaining to the state of the 4210-CVU.

Usage

```
int getstatus(int instr_id, long parameter, double *value);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
parameter	Parameter to be queried; the macros for the `KI_CVU` parameters are defined in `lptparam.h`; see **Details**
value	Returned value for the queried parameter

Details

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 1.5 µA, 50 µA or 1.5 mA for fixed range)
`KI_CVU_ACI_PRESENT_RANGE`	AC current range (1.5 µA, 50 µA or 1.5 mA); returns range used for last measurement, even if on auto range
`KI_CVU_ACV_LEVEL`	AC voltage level (10 mV to 100 mV_{RMS})
`KI_CVU_APERTURE`	A/D aperture time (0.006 to 10.002 PLCs)
`KI_INTGPLC`	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 707)
`KI_CVU_FREQUENCY`	Drive frequency (1 kHz to 10 MHz)
`KI_CVU_MEASURE_MODEL`	Impedance measure model (0 through 5; see "Measurement model parameter values" table below)
`KI_CVU_MEASURE_SPEED`	Measurement speed (fast, normal, quiet or custom)
`KI_CVU_MEASURE_STATUS`	Measurement status (for last reading); the measurement status codes are listed and explained in Status codes (on page 6-244)
`KI_CVU_MEASURE_TSTAMP`	Measurement timestamp (for last reading)

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`
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

Also see

None

measf

This command returns the frequency sourced during a single measurement.

Usage

```
int measf(int instr_id, double *freq);
```

instr_id	The instrument identification code of the 4210-CVU: CVU1
freq	Returned frequency

Details

This command returns the present test frequency being used for a single impedance measurement. Use the measz command to make a single measurement.

NOTE

Use the smeasf or smeasfRT command to return the frequencies used for a sweep.

Also see

measz (on page 14-204)
smeasf (on page 14-212)
smeasfRT (on page 14-213)

meass

This command returns the status referenced to a single measurement.

Usage

```
int meass(INSTR_ID instr_id, double* result);
```

instr_id	The instrument identification code for the 421x-CVU: `CVU1`
result	Returned 32 bit measurement status

Details

This command returns the measurement status for a single measurement.

Key Below:

Measurement Status Result Key

Bit	Description
31	Measurement Timeout
30:28 - 27	CVH ABB Lock Fault
26 - 25:24	CVH Overflow Indicator (V & I)
23:20 - 19	CVL ABB Lock Fault
18 - 17:16	CVL Overflow Indicator (V & I)
15:2 - 1:0	IAC Measure Range Index (0: 1uA; 1:30uA; 2: 1mA)

NOTE

Use the `measz` command to make a single measurement.

Use the `smeass` command to return the measurement status values used for a sweep.

Also see

measz (on page 14-204)
smeass (on page 14-213)

meast

This command returns a timestamp referenced to a measurement or a system timer.

Usage

```
int meast(long timerID, double *timestamp);
```

timerID	The instrument identification code: CVU1, TIMER1, TIMER2, and so on
timestamp	Returned timestamp

Details

This command 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 command after the call to perform a measurement (measz command) will return the timestamp for that measurement.

When the timerID parameter is set for a timer, the meast command can be called at any time and will return a timestamp that is referenced to a system timer. The enable command is used to start the timer (starts at zero when called).

NOTE

Use the smeast or smeastRT command to acquire timestamps for a sweep.

Examples

Programming example #1 (on page 14-223) acquires a timestamp for the measurement.

Programming example #2 (on page 14-224) measures the execution time of the code.

Also see

smeast (on page 14-214)
smeastRT (on page 14-215)

measv

This command returns the DC bias voltage sourced during a single measurement.

Usage

```
int measf(int instr_id, double *biasV);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
biasV	Returned DC bias voltage

Details

This command returns the DC bias voltage presently being used for a single measurement.

Use the `measz` command to make a single measurement.

NOTE

Use the `smeasv` or `smeasvRT` command to return the DC bias voltages used for a sweep.

Also see

measz (on page 14-204)
smeasv (on page 14-216)
smeasvRT (on page 14-217)

measz

This command makes an impedance measurement.

Usage

```
int measz(int instr_id, int model, int speed, double *result1 double *result2);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
model	Measurement model; see table in **Details**
speed	Measure speed: ■ `KI_CVU_SPEED_FAST`: Fast measurements (higher noise) ■ `KI_CVU_SPEED_NORMAL`: Selects a balance between speed and low noise ■ `KI_CVU_SPEED_QUIET`: Low-noise measurements ■ `KI_CVU_SPEED_CUSTOM`: Selects custom settings; the delay factor, filter factor, and aperture are set using the `setmode` command
result1	First result of the selected measure *model*
result2	Second result of the selected measure *model*

Details

This command makes a single impedance measurement.

Before calling `measz`, use the `forcev` command to set the DC bias level, the `setfreq` command to set the AC drive frequency, and the `setlevel` command to set the AC drive voltage.

The parameter values for the measurement *model* are listed in the following table.

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`
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`

NOTE

Use the `smeasz` or `smeaszRT` command to measure and return the impedance readings for a sweep.

Also see

forcev (on page 14-198)
setfreq (on page 14-208)
setlevel (on page 14-209)
setmode (on page 14-209)
smeasz (on page 14-217)
smeaszRT (on page 14-219)

rangei

This command selects an impedance measurement range.

Usage

```
int rangei(int instr_id, double range);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
range	Impedance measure range (0, 1 µA, 30 µA, or 1 mA)

Details

Use this command to set the 4210-CVU to a current measure range for impedance measurements. Setting *range* to 0 selects auto range. The CVU automatically goes to the most sensitive (optimum) range to make the measurement. This is the same as calling the `setauto` command.

The other `range` parameter values select a fixed measure range. The CVU remains on the fixed range until auto range is enabled or the CVU is reset (`devint` called).

Example

Programming example #1 (on page 14-223) uses the 1 mA measure range for the impedance measurement.

Also see

devint (on page 14-16)
setauto (on page 14-207)

rtfary

This command returns the array of force values used during the subsequent voltage or frequency sweep.

Usage

```
int rtfary(double *forceArray);
```

forceArray	Array of force values for voltage or frequency

Details

This command is used to return an array of voltage or frequency force values for a sweep. Send this command before calling any sweep command.

NOTE

To prevent a memory exception error, make sure that the array that will receive the sourced values is large enough.

The following examples show the proper command sequence for using `rtfary`:

Example 1: Valid command sequence for voltage sweep

```
smeasz
smeast
rtfary
dsweepv
```

Example 2: Valid command sequence for frequency sweep

```
smeasz
rtfary
dsweepf
```

Example

Programming example #2 (on page 14-224) returns the array of force values for the voltage sweep.

Also see

None

setauto

This command selects the auto measure range.

Usage

```
int setauto(int instr_id);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`

Details

This command sets the 4210-CVU for auto range measurements. When `setauto` is called, the CVU goes to the most sensitive range to make the measurement. Calling `devint` also selects autorange.

You can also use the `rangei` command to enable autorange, or select a fixed measurement range. Autorange remains enabled until a fixed range is selected.

Example

Programming examples (on page 14-223) 2 through 5 use auto range for impedance measurements.

Also see

devint (on page 14-16)
rangei (on page 14-205)

setfreq

This command sets the frequency for the AC drive.

Usage

```
int setfreq(int instr_id, double frequency);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
frequency	Frequency of the AC drive

Details

This command sets the frequency of the AC drive. The 4210-CVU provides test frequencies from 1 kHz to 10 MHz in the following steps:

- 1 kHz through 10 kHz in 1 kHz 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 is selected (for example, 15 kHz input selects 20 kHz). You can use the `getstatus` command to retrieve the selected frequency value.

AC drive (AC voltage level and frequency) does not turn on until a measurement is made. The AC drive turns off after the measurement is completed. Note that the DC voltage source stays on for the whole test.

Example

Programming examples (on page 14-223) 1, 2, 4 and 5 use the `setfreq` command to set the AC drive frequency.

Also see

getstatus (on page 14-199)
setlevel (on page 14-209)

setlevel

This command sets the AC drive voltage level.

Usage

```
int setlevel(int instr_id, double signalLevel);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
signalLevel	Voltage level of the AC drive (10 mV to 100 mV_{RMS})

Details

This command is used to set the voltage of the AC drive.

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.

Example

All the Programming examples (on page 14-223) use the `setlevel` command to set the AC drive voltage.

Also see

setfreq (on page 14-208)

setmode (4210-CVU)

This command sets operating modes specific to the 4210-CVU.

Usage

```
int setmode(int instr_id, long modifier, double value);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
modifier	Specific operating characteristic to change; see table in **Details**
value	Parameter value for the *modifier*

Details

> **NOTE**
>
> The following information is specific to the CVU. For details on using `setmode` for other instruments, see the setmode (on page 14-45) command.

The `setmode` command allows control over the following 4210-CVU operating characteristics:

- Connection compensation control for open, short and load. When disabled, saved compensation constants are not applied to the measurements. Whenever the connection setup has changed, connection compensation needs to be performed to acquire and save new compensation constants. Connection compensation is performed from Clarius. For details, see Connection compensation (on page 4-14).
- Setting for cable length compensation (0 m, 1.5 m, or 3 m). This setting is made from the window that is used to enable compensation. For details, see Enabling compensation (on page 4-22).
- Settings (delay factor, filter factor and aperture) for `KI_CUSTOM` measurement speed, which is set by `measz`, `smeasz`, or `smeaszRT`.

Parameters		Comment
modifier	***value***	**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: No cable compensation ■ 1.5: 1.5 m CVU cable ■ 3.0: 3.0 m CVU cable ■ 5.0: 1.5 m CVIV cable; 2-wire mode ■ 6.0: 1.5 m CVU to CVIV cable with 0.75 m CVIV to DUT cable; 4-wire mode ■ 7.0: 1.5 m CVU to CVIV cable with 0.61 m CVIV to DUT cable; 4-wire mode	Cable length setting.
KI_CVU_SET_CONSTANT_FILE	0 to 1000	The number in the file name that contains the open, short, and load compensation values for the CVU.
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 707	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.

Also see

measz (on page 14-204)
setmode (on page 14-45)
smeasz (on page 14-217)
smeaszRT (on page 14-219)

smeasf

This command returns the frequencies used for a sweep.

Usage

```
int smeasf(int instr_id, double *freq_arr);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
freq_arr	Returned array of test frequencies

Details

This command returns the present test frequencies used for a sweep. The frequency values are returned in an array. The frequency values are posted to Clarius in Analyze after the test has finished.

NOTE

You can use the `smeasfRT` command to return sourced sweep frequency values in an array. It posts the frequency values to Clarius in real time.

NOTE

Use the `measf` command to return the frequency used for a single measurement.

Also see

measf (on page 14-200)
smeasfRT (on page 14-213)

smeasfRT

This command returns the sourced frequencies (in real time) for a sweep.

Usage

```
int smeasfRT(int instr_id, double *freq_arr, char *colname);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
freq_arr	Returned array of test frequencies
colname	Column name (character string) to pass into Clarius for the data sheet column

Details

Like the `smeasf` command, the test frequencies for a sweep are returned in an array. However, the frequency values are posted to the Clarius Analyze sheet and graph in real time (after each step of the sweep is executed).

Note that the values are only available in real-time if Clarius is running. Otherwise, they are stored in an array in the usual fashion.

Example

```
smeasfRT(CVU1, freq_arr, "freq_arr");
```

This command posts the frequency values into the Clarius Analyze sheet under a column named `freq_arr`.

Also see

smeasf (on page 14-212)

smeass

This command returns the measurement status values for every point in a sweep.

Usage

```
int meass(INSTR_ID instr_id, double* result);
```

instr_id	The instrument identification code for the 421x-CVU: `CVU1`
result	Returned array of 32 bit measurement status values

Details

This command returns the measurement status values for every point in a sweep. The values are returned in an array.

See `meass` for the Measurement Status Result Key.

NOTE

Use the `meass` command to return the measurement status for a single measurement.

Also see

smeass (on page 14-213)

smeast

This command returns timestamps referenced to sweep measurements or a system timer.

Usage

```
int meast(long timerID, double *tarray);
```

timerID	The ID of the 4210-CVU or timer: CVU1, TIMER1, TIMER2, and so on
tarray	Returned array of timestamps

Details

This command acquires the timestamp for each measurement step of a sweep. The timestamps are returned in an array. The timestamps are posted to Clarius Analyze after the test has finished.

NOTE

You can also use the smeastRT command to return timestamps in an array. It posts the frequency values to Clarius in real time.

The timestamp can be referenced to the 4210-CVU (*timerID* = CVU1) or to a system timer (for example, *timerID* = TIMER1). This command is similar to the meast command, but is synchronized with a sweep to return a timestamp referenced to each measurement. If you need a timestamp for a single measurement, use the meast command.

NOTE

LPT maintains a list of measurements to be done at each sweep point after the forcing instrument has stepped its source (V, I, or F). The *smeasX* and smeas*X*RT commands register the measurement with a master list. If the time measurement precedes the Z measurement, then the wrong timestamp is returned (the one from the previous measurement).

Example

Programming example #2 (on page 14-224) acquires a timestamp for each measurement in the sweep.

Also see

meast (on page 14-202)
smeastRT (on page 14-215)

smeastRT

This command returns timestamps (in real time) referenced to sweep measurements or a system timer.

Usage

```
int measRT(long timerID, double *tarray, char *colname);
```

timerID	The ID of the 4210-CVU or timer: CVU1, TIMER1, TIMER2, and so on
tarray	Returned array of timestamps
colname	Column name to pass into Clarius (case-sensitive character string)

Details

Returns the timestamps are returned in an array and posts the timestamps to the Clarius Analyze sheet and graph in real time. Each timestamp appears in the sheet and graph after each measurement is made.

Note that the values are only available in real time if Clarius is running. Otherwise, they are stored in an array.

The *colname* parameter specifies the name for the data sheet column in Clarius.

NOTE

LPT maintains a list of measurements to be done at each sweep point after the forcing instrument has stepped its source (V, I, or F). The *smeasX* and smeas*X*RT commands register the measurement with a master list. If the time measurement precedes the Z measurement, then the wrong timestamp is returned (the one from the previous measurement).

Example

```
smeastRT(CVU1, time_arr, "time_arr");
```

This command posts the timestamp values into the Clarius Analyze sheet in the column named time_arr.

Also see

smeast (on page 14-214)

smeasv

This command returns the DC bias voltages used for a sweep.

Usage

```
int smeasv(int instr_id, double *varray);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
varray	Returned array of DC bias voltages

Details

This command returns the DC bias voltages used in a sweep. The values are returned in an array. The voltage values are posted to the Clarius Analyze sheet and graph after the test has finished.

NOTE

You can also use the `smeasvRT` command to return sourced sweep DC bias voltage values in an array. It posts the voltage values to Clarius in real time.

NOTE

Use the `measv` command to return the DC bias voltage used for a single measurement.

Also see

measv (on page 14-203)
smeasvRT (on page 14-217)

smeasvRT

This command returns the sourced DC bias voltages (in real time) for a sweep.

Usage

```
int smeasvRT(int instr_id, double *varray, char *colname);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
varray	Returned array of DC bias voltages
colname	Column name to pass into Clarius (character string)

Details

This command is similar to `smeasv` command. It returns the sourced DC bias voltages for a sweep in an array. However, the voltage values are posted to the Clarius Analyze sheet and graph in real time. Each voltage value appears in the sheet and graph after each step of the sweep is executed.

Note that the values are only available in real time if Clarius is running. Otherwise, they are stored in an array.

The *colname* parameter specifies a name for the data sheet column in Clarius.

Example

```
smeasvRT(CVU1, volt_arr, "volt_arr");
```

This command posts the voltage values into the Clarius data sheet under a column named `volt_arr`.

Also see

smeasv (on page 14-216)

smeasz

This command performs impedance measurements for a sweep.

Usage

```
int smeasz(int instr_id, long model, long speed, double *result1, double *result2);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
model	Measure model; refer to "Measurement model parameter values" table in the **Details**
speed	Speed settings: ■ `KI_CVU_SPEED_FAST`: Fast measurements (higher noise) ■ `KI_CVU_SPEED_NORMAL`: Selects a balance between speed and low noise ■ `KI_CVU_SPEED_QUIET`: Low-noise measurements ■ `KI_CVU_SPEED_CUSTOM`: Selects custom settings; the delay factor, filter factor, and aperture are set using the `setmode` command
result1	Array of the first result of the selected measure model
result2	Array of the second result of the selected measure model

Details

This command makes 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 the Clarius Analyze sheet and graph after the test has finished.

Before calling `smeasz`, use the `forcev` command to set the DC bias level, the `setfreq` command to set the AC drive frequency and the `setlevel` command to set the AC drive voltage.

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`
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`

NOTE

You can also use the `smeaszRT` command to measure and return sweep impedance measurements. It posts the measured readings to Clarius in real time.

NOTE

To return a single impedance measurement, use `measz`.

Example

Programming examples (on page 14-223) 2 through 5 use the `smeas` command for impedance measurements.

Also see

forcev (on page 14-198)
measz (on page 14-204)
setfreq (on page 14-208)
setlevel (on page 14-209)
setmode (on page 14-209)
smeaszRT (on page 14-219)

smeaszRT

This command makes and returns impedance measurements for a voltage or frequency sweep in real time.

Usage

```
int smeaszRT(int instr_id, long model, long speed, double *result1, char *colname1,
    double *result2, char *colname2);
```

instr_id	The instrument identification code of the 4210-CVU: `CVU1`
model	Measure model (see "Measurement model parameter values" table in **Details**)
speed	Speed settings: ■ `KI_CVU_SPEED_FAST`: Fast measurements (higher noise) ■ `KI_CVU_SPEED_NORMAL`: Selects a balance between speed and low noise ■ `KI_CVU_SPEED_QUIET`: Low-noise measurements ■ `KI_CVU_SPEED_CUSTOM`: Selects custom settings; the delay factor, filter factor, and aperture are set using the `setmode` command
result1	Array of the first result of the selected measure model
colname1	Column name to pass into Clarius for *result1* array (character string)
result2	Array of the second result of the selected measure model
colname2	Column name to pass into Clarius for *result2* array (character string)

Details

This command is similar to the `smeasz` command; both commands return the measured impedance readings for a sweep returned in arrays. However, the readings from `smeaszRT` are posted to the Clarius Analyze sheet and graph in real time. Two measurement results appear in the sheet and graph after each step of the sweep is executed.

Note that the values are only available in real-time if Clarius is running. Otherwise, they are stored in an array.

The *colname1* and *colname2* parameters specify names for data sheet columns in Clarius.

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`
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`

Example

```
smeaszRT(CVU1, 2, KI_NORMAL, result1, "result1", result2, "result2");
```

This command posts the results into the Clarius data sheet under columns named *result1_arr* and *result2_arr*.

Also see

smeasz (on page 14-217)

sweepf

This command performs a frequency sweep.

Usage

```
int sweepf(int instr_id, double startf, double stopf, long *NumPts, double delaytime);
```

instr_id	The instrument identification code of the 4210-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)

Details

This command is used to perform a frequency sweep. The 4210-CVU provides test frequencies from 1 kHz to 10 MHz in the following steps:

- 1 kHz through 10 kHz in 1 kHz 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, 15 kHz 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 4210-CVU steps 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 steps through the following frequency points: 800 kHz, 900 kHz, 1 MHz, 2 MHz, 3 MHz.

The *`NumPts`* query returns the total number of sweep points. For the above example, *`NumPts`* returns `5`.

The *`delayTime`* parameter sets the delay that occurs before each measurement. Note that there is an inherent system overhead delay on each frequency step of the sweep.

Use the `forcev` command to set the DC bias level and `setlevel` command to set the AC drive voltage.

NOTE

Use the `dsweepf` command to perform a dual frequency sweep.

Example

Programming example #3 (on page 14-225) performs a frequency sweep.

Also see

asweepv (on page 14-188)
dsweepf (on page 14-195)
dsweepv (on page 14-197)
forcev (on page 14-198)
setlevel (on page 14-209)
sweepv (on page 14-222)

sweepv

This command performs a linear staircase DC voltage sweep.

Usage

```
int sweepv(int instr_id, double startv, double stopv, long NumSteps, double delaytime);
```

instr_id	The instrument identification code of the 4210-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)

Details

This command is used to perform a staircase sweep. 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 on each step of the sweep.

Use the `setfreq` and `setlevel` commands to set the AC drive frequency and voltage for the sweep.

NOTE

Use the `dsweepv` command to do a dual linear staircase voltage sweep.

Example

Refer to Programming example #2 (on page 14-224) for an example of a single staircase voltage sweep.

Also see

asweepv (on page 14-188)
dsweepf (on page 14-195)
dsweepv (on page 14-197)
setfreq (on page 14-208)
setlevel (on page 14-209)
sweepf (on page 14-220)

Programming examples

These programming examples provide examples of how to use LPT commands to:

- Make a single CsRs impedance measurement: Programming example #1 (on page 14-223)
- Do a single staircase sweep and measure CpGp for each step: Programming example #2 (on page 14-224)
- Do a single frequency sweep and measure CpGp for each step: Programming example #3 (on page 14-225)
- Do a voltage array sweep: Programming example #4 (on page 14-226)
- Do a voltage array sweeps with an array of delay values used for the sweep: Programming example #5 (on page 14-227)

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

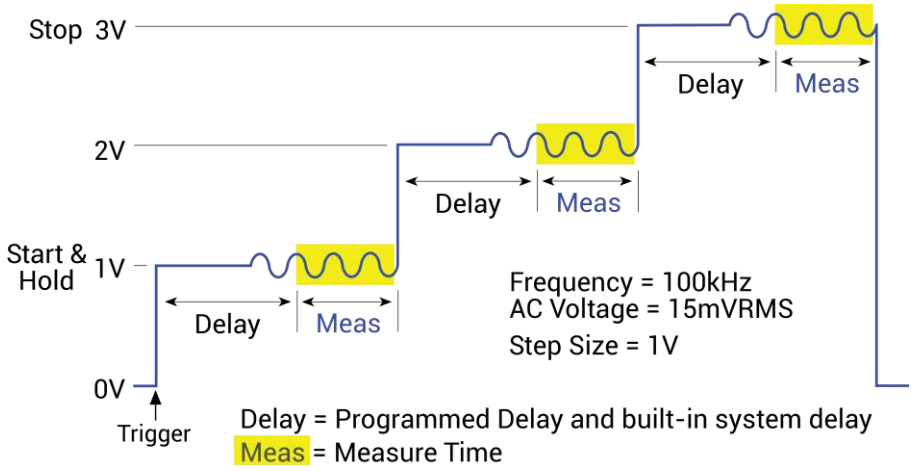
This example also acquires a timestamp for the measurement.

```
double result1, result2, timeStamp;
forcev(CVU1, 1);          /*  Set DC bias to 1 V. */
setfreq(CVU1, 100e3);     /*  Set AC drive frequency to 100 kHz. */
setlevel(CVU1, 15e-3);    /*  Set AC drive voltage to 15 mV RMS. */
rdelay(0.1);              /*  Set settling time to 100 ms. */
rangei(CVU1, 1.0e-3);     /* Select 1 mA measure range. */
measz(CVU1, KI_CVU_TYPE_CSRS, KI_CVU_SPEED_NORMAL, &result1, &result2);
                                  /*  Measure CsRs. */
meast(CVU1, &timeStamp);  /*  Return timestamp for measurement. */
devint();                 /*  Reset CVU. */
```

Programming example #2

Performs a single staircase voltage sweep, as shown in the figure below.

Figure 613: Voltage sweep example



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

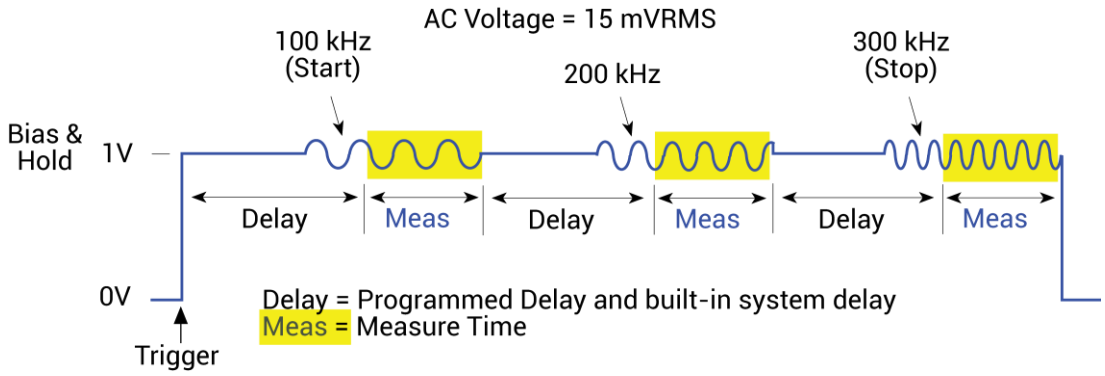
This example also returns a timestamp for each measurement and measures the execution time of the code.

```
double result1[4], result2[4], timeStamp1[4], timeStamp2;
enable(TIMER1);                 /*  Start timer at 0 seconds. */
setfreq(CVU1, 100e3);           /*  Set AC drive frequency to 100 kHz. */
setlevel(CVU1, 15e-3);          /*  Set AC drive voltage to 15 mV RMS. */
setauto(CVU1);                  /* Select auto measure range. */
smeasz(CVU1, KI_CVU_TYPE_CPGP, KI_CVU_SPEED_NORMAL, result1, result2);
                                        /*  Configure CpGp measurements. */
smeast(CVU1, timeStamp1);       /* Return timestamps for all measurements. */
rtfary(forceArray);             /*  Return array of force voltages. */
sweepv(CVU1, 1, 3, 3, 0.05);    /*  Configure and perform sweep. */
meast(TIMER1, &timeStamp2);     /*  Return execution time for above. */
                                        /*  block of code. */
devint();                               /*  Reset CVU. */
```

Programming example #3

Performs a single frequency sweep shown in the following figure.

Figure 614: Frequency sweep example



CpGp is measured on each frequency step of the sweep.

Test parameters:

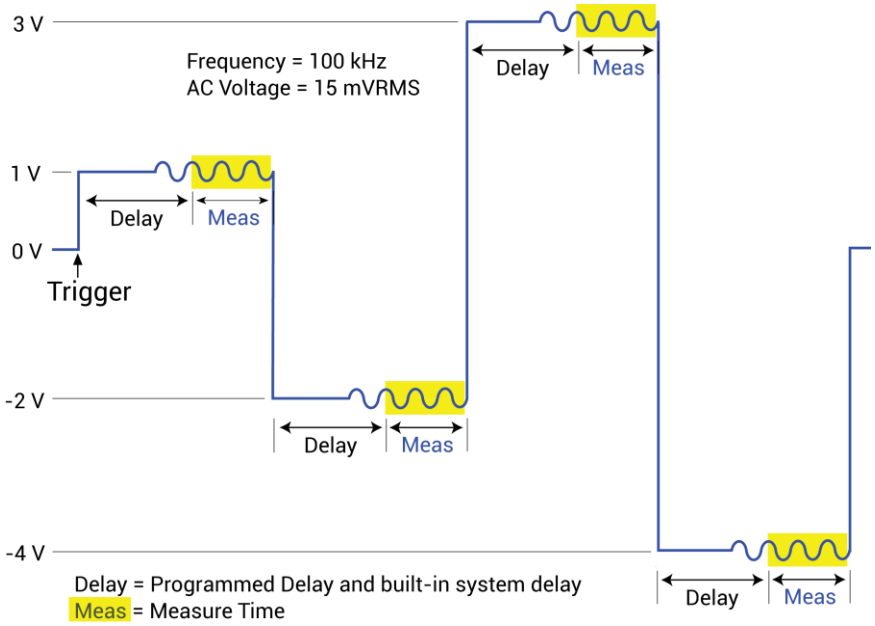
- AC drive voltage = 15 mV_{RMS}
- Measure mode = CpGp
- Measure range = Auto
- Start frequency = 100 kHz
- Stop frequency = 300 kHz
- Number of frequency steps = 3 (this value is returned from the command to the *NumPts* variable, and not passed by the user)
- Delay = 50 ms

```
double result1[6], result2[6], forceArray[6];
long NumPts;
setlevel(CVU1, 15e-3);       /*  Set AC drive voltage to 15 mV RMS. */
setauto(CVU1);               /* Select auto measure range. */
smeasz(CVU1, KI_CVU_TYPE_CPGP, KI_CVU_SPEED_NORMAL, result1, result2);
                                    /*  Configure CpGp measurements. */
rtfary(forceArray);          /*  Return array of force frequencies. */
dsweepf(CVU1, 100e3, 300e3, &NumPts, 0.05);  /*  Configure and perform sweep. */
devint();                    /*  Reset CVU. */
```

Programming example #4

This example performs a voltage array sweep as shown in the figure below.

Figure 615: Voltage array sweep example



The above figure shows an example of a voltage array sweep using the following voltage values: 1 V, -2 V, 3 V, -4 V. The voltage array is configured as follows:

```
forceArray[0] = 1
forceArray[1] = -2
forceArray[2] = 3
forceArray[3] = -4
```

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];
setfreq(CVU1, 100e3);          /*  Set AC drive frequency to 100 kHz. */
setlevel(CVU1, 15e-3);         /*  Set AC drive voltage to 15 mV RMS. */
setauto(CVU1);                 /* Select auto measure range. */
smeasz(CVU1, KI_CVU_TYPE_CPGP, KI_CVU_SPEED_NORMAL, result1, result2);
                                        /*  Configure CpGp measurements. */
asweepv(CVU1, 4, 0.05, forceArray);  /*  Configure and perform sweep. */
devint();                      /*  Reset CVU. */
```

Programming example #5

Similar to Programming example #4 (on page 14-226), 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];
setfreq(CVU1, 100e3);         /*  Set AC drive frequency to 100 kHz. */
setlevel(CVU1, 15e-3);        /* Set AC drive voltage to 15 mV RMS. */
setauto(CVU1);                /* Select auto measure range. */
smeasz(CVU1, KI_CVU_TYPE_CPGP, KI_CVU_SPEED_NORMAL, result1, result2);
                                      /*  Configure CpGp measurements. */
adelay(4, delayArray);        /* Call a delay array for asweepv. */
asweepv(CVU1, 4, 0, forceArray);  /* Configure and perform sweep. */
devint();                     /* Reset CVU. */
```

LPT commands for switching

These LPT commands are used with the Series 700 Switching System, the 4200A-CVIV Multi-Switch, and the 4225-RPM.

addcon

This command adds connections without clearing existing connections.

Usage

```
int addcon(int exist_connect, int connect1, [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 the `addcon`, `conpin`, or `conpth` command
connect1	A pin number or an instrument terminal ID
connectn	A pin number or an instrument terminal ID

Details

`addcon` can be used to make additional connections on a matrix. `addcon` will 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.

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.

Before making the new connections, the `addcon` command clears all active sources by calling the `devclr` command.Also see

clrcon (on page 14-229)
conpin (on page 14-230)
conpth (on page 14-231)
delcon (on page 14-234)

clrcon

This command opens or de-energizes all device under test (DUT) pins and instrument matrix relays, disconnecting all crosspoint connections.

Usage

```
int clrcon(void);
```

Details

The `clrcon` command is called automatically by the `devint`, `pulse_output` (only for RPMs), and `execut` commands. The first in a series of one or more connection-type commands automatically calls a `clrcon` command. Because this command is automatically called, it is not normally used by a programmer.

If any sources are actively generating current or voltage, the `devclr` command is automatically called before the relay matrix is de-energized.

Also see

devclr (on page 14-72)
devint (on page 14-16)
execut (on page 14-19)
pulse_output (on page 14-165)

conpin

This command connects pins and instruments.

Usage

```
int conpin(int InstrTermID, int connect1, [connectn, [...]] 0);
```

InstrTermID	The instrument terminal ID, such as SMU1, GNDU, or PMU1CH1
connect1	A pin number or an instrument terminal ID
connectn	A pin number or an instrument terminal ID

Details

`conpin` connects every item in the argument list together. If no connection rules are violated, the pin or terminal is connected to the additional items, along with everything to which it is already connected.

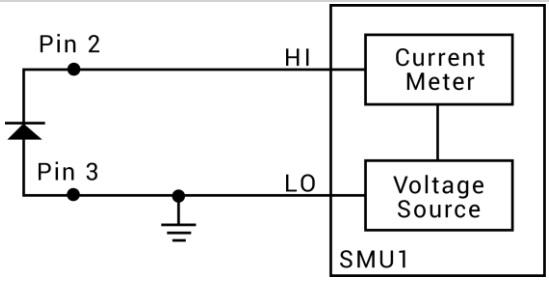
The first `conpin` or `conpth` after any other LPT library call clears all sources by calling `devclr` and then clears all matrix connections by calling `clrcon` before making the new connections.

The value `-1` is 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.

Example

```
conpin(3, GND, 0); /* Connect pin 3 to SMU1 */
 /* and ground. */
conpin(2, SMU1, 0); /* Connect pin 2 to SMU1. */
.
.
```



Also see

addcon (on page 14-228)
clrcon (on page 14-229)
conpth (on page 14-231)
devclr (on page 14-72)

conpth

This command connects pins and instruments using a specific pathway.

Usage

```
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

Details

You can force the system 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` command after any other LPT library call clears all sources by calling the `devclr` command and then clears all matrix connections by calling the `clrcon` command before making the new connections.

The value `-1` for any item in the connection list is ignored by `conpth` and is considered a valid entry in the connection list.

The `conpth` command 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.

Also see

addcon (on page 14-228)
clrcon (on page 14-229)
conpin (on page 14-230)
devclr (on page 14-72)

cviv_config

This command sends switching commands to the 4200A-CVIV Multi-Switch.

Usage

```
int cviv_config(int instr_id, int channel, int mode);
```

instr_id	The instrument identification code of the 4200A-CVIV: `CVIV1`
channel	4200A-SCS channel: `1` to `4` 4200A-SCS all channels: `5`
mode	For channels 1 to 4, the switch settings for the selected channel: ■ Open connection to output terminal: `KI_CVIV_OPEN` or `0` ■ Connect channel to SMU (4200-SMU or 4210-SMU): `KI_CVIV_SMU` or `1` ■ Connect channel to CVU HI (4210-CVU): `KI_CVIV_CVH` or `2` ■ Connect channel to CVU LO (4210-CVU): `KI_CVIV_CVL` or `3` ■ Connect CV guard to the output connector shell with AC ground to center: `KI_CVIV_CV_GRD` or `4` ■ Connect channel to ground unit: `KI_CVIV_GNDU` or `5` ■ Connect channel to AC-coupled AC ground: `KI_CVIV_AC_COUPLED_AC_GND` or `6` ■ Connect channel to bias tee SMU CV HI: `KI_CVIV_BT_CVH` or `7` ■ Connect channel to bias tee SMU CV LO: `KI_CVIV_BT_CVL` or `8` ■ Connect channel to bias tee low current SMU CV HI: `KI_CVIV_BT_LOI_CVH` or `9` ■ Connect channel to bias tee low current SMU CV LO: `KI_CVIV_BT_LOI_CVL` or `10` ■ Connect channel to bias tee AC ground: `KI_CVIV_BT_AC_GND` or `11` If *channel* is set to `5` (all channels), the switch settings for the 4200A-SCS instrument are: ■ All CV channels to C-V 2-wire: `KI_CVIV_CVU_2WIRE` or `1` ■ All CV channels to C-V 4-wire: `KI_CVIV_CVU_4WIRE` or `0`

Details

The 4200A-SCS includes input connections for four 4200-SMU or 4210-SMU cards and one 4210-CVU. Use this command to control switching inside the 4200A-SCS to connect the SMU and CVU instruments to the output terminals.

The 4200A-SCS connections are cleared by the `clrcon` command.

Example

```
cviv_config(CVIV1, 1, KI_CVIV_SMU);
```

This command connects channel 1 of the CVIV to a SMU.

Also see

clrcon (on page 14-229)
cviv_display_config (on page 14-233)
cviv_display_power (on page 14-234)

cviv_display_config

This command configures the LCD display on the 4200A-CVIV Multi-Switch.

Usage

```
int cviv_display_config(int instr_id, int channel, int identifier, char *value);
```

instr_id	The instrument identification code of the 4200A-SCS: `CVIV1`
channel	4200A-SCS channel (use to set a terminal name): `1` to `4` 4200A-SCS virtual channel (use to set the test name): `5` See **Details**
identifier	Display the name of the terminal: `KI_CVIV_TERMINAL_NAME` or `1` Display the name of the test: `KI_CVIV_TEST_NAME` or `0` See **Details**
value	A string that defines the name (up to 16 characters for a test name or 6 characters for a terminal name)

Details

Sets the name for the channel terminal or test that is displayed on the 4200A-SCS for the selected channel.

The *channel* and *identifier* settings must be set for either terminal or test name. For example, if *channel* is set to `2`, *identifier* must be set to `KI_CVIV_TERMINAL_NAME`.

If the `clrcon` command is sent, the 4200A-CVIV display is updated to show the change in connections. If the 4200A-CVIV display is turned off, it remains off after a `clrcon`.

Example

```
cviv_display_config(CVIV1, 2, KI_CVIV_TERMINAL_NAME, "Source");
```

This command sets the name of the channel 2 terminal on the 4200A-SCS display.

Also see

clrcon (on page 14-229)
cviv_config (on page 14-232)
cviv_display_power (on page 14-234)

cviv_display_power

This command sets the display state of the LCD display on the 4200A-CVIV.

Usage

```
int cviv_display_power(int instr_id, int state);
```

instr_id	The instrument identification code of the 4200A-SCS: `CVIV1`
state	Display on: `KI_CVIV_DISPLAY_ON` or `1` Display off: `KI_CVIV_DISPLAY_OFF` or `0`

Details

This command turns the display of the 4200A-SCS on or off.

When the display is turned off, the 4200A-SCS clears the displays. A small green circle is displayed to indicate that the 4200A-SCS instrument is powered.

When the display is turned on, the latest configuration is displayed.

If the `clrcon` command is sent, the 4200A-CVIV display is updated to show the change in connections. If the 4200A-CVIV display is turned off, it remains off after a `clrcon`.

Example

```
cviv_display_power(CVIV1, KI_CVIV_DISPLAY_OFF);
```

Turns off the 4200A-SCS display.

Also see

cviv_config (on page 14-232)
cviv_display_config (on page 14-233)

delcon

This command removes specific matrix connections.

Usage

```
int delcon(int InstrTermID, int exist_connect, [int exist_connectn, [...]] 0);
```

InstTermID	The instrument terminal ID, such as SMU1, GNDU, or PMU1CH1
exist_connect	A pin number or an instrument terminal ID
exist_connectn	A pin number or an instrument terminal ID

Details

This command disconnects all connections to each terminal or pin listed. Before disconnecting the pins or terminals, the `delcon` command clears all active sources by calling the `devclr` command.

If a SMU remains connected, GND must be reconnected using `addcon` or an error is generated when the first LPT library command after the connection sequence executes.

A programmer can run a series of tests in a single test sequence using the `addcon` and `delcon` commands together without breaking existing connections. Only the required terminal and pin changes are made before the next sourcing and measuring operations.

Example

```
double i1, i2;
conpin(3, GND, 0);
conpin(1, SMU1, 0);
conpin(2, SMU2, 0);
forcev(SMU1, 1.0);
forcei(SMU2, 0.001);
measi(SMU1, &i1);
delcon(SMU2, 0); /* Remove SMU2 from the circuit */
forcev(SMU1, 1.0); /* because delcon cleared sources. */
measi(SMU1, &i2);
```

Also see

addcon (on page 14-228)
clrcon (on page 14-229)
conpin (on page 14-230)
conpth (on page 14-231)
devclr (on page 14-72)

devint

This command resets all active instruments in the system to their default states.

Usage

```
int devint(void);
```

Details

Resets all active instruments in the system to their default states. It clears the system by opening all relays and disconnecting the pathways. Meters and sources are reset to their default states. Refer to the hardware manuals for the instruments in your system for listings of available ranges and the default conditions and ranges.

The `devint` command is implicitly called by the `execut` command.

To abort a running `pulse_exec` pulse test, see `dev_abort`.

`devint` does the following:

1. Clears all sources by calling `devclr`.
2. Clears the matrix crosspoints by calling `clrcon`.
3. Clears the trigger tables by calling `clrtrg`.
4. Clears the sweep tables by calling `clrscn`.
5. Resets GPIB instruments by sending the string defined with `kibdefint`.
6. Resets the active instrument cards.

Instrument cards are reset in the following order:

1. SMU instrument cards
2. CVU instrument cards
3. Pulse instrument cards (4225-PMU or 4220-PGU)

`devint` is implicitly called by `execut` and `tstdsl`. `devclr` is implicitly called by `clrcon`.

The SMUs return to the following states:

- 100 µA and 10 V ranges
- Autorange on
- Voltage source
- 0 V DCV bias

The 4210-CVU returns to the following states:

- 30 mV_{RMS} AC signal
- 0 V DCV bias
- 100 kHz frequency
- Autorange on
- Cable length compensation set to 0 m
- Open/Short/Load compensation disabled

The 4225-PMU or 4220-PGU returns to the following states:

- 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.
- 5 V and 10 mA ranges
- If in pulse mode:
  - Period of 1 µs
  - Transition Times (Rise and Fall) of 100 ns
  - Width of 500 ns
  - Voltage high and low of 0 V

    - Load of 50 Ω
- If in segmented arb mode, Start Voltage is 0 V
- If in arbitrary waveform mode, Table Length is 100

Also see

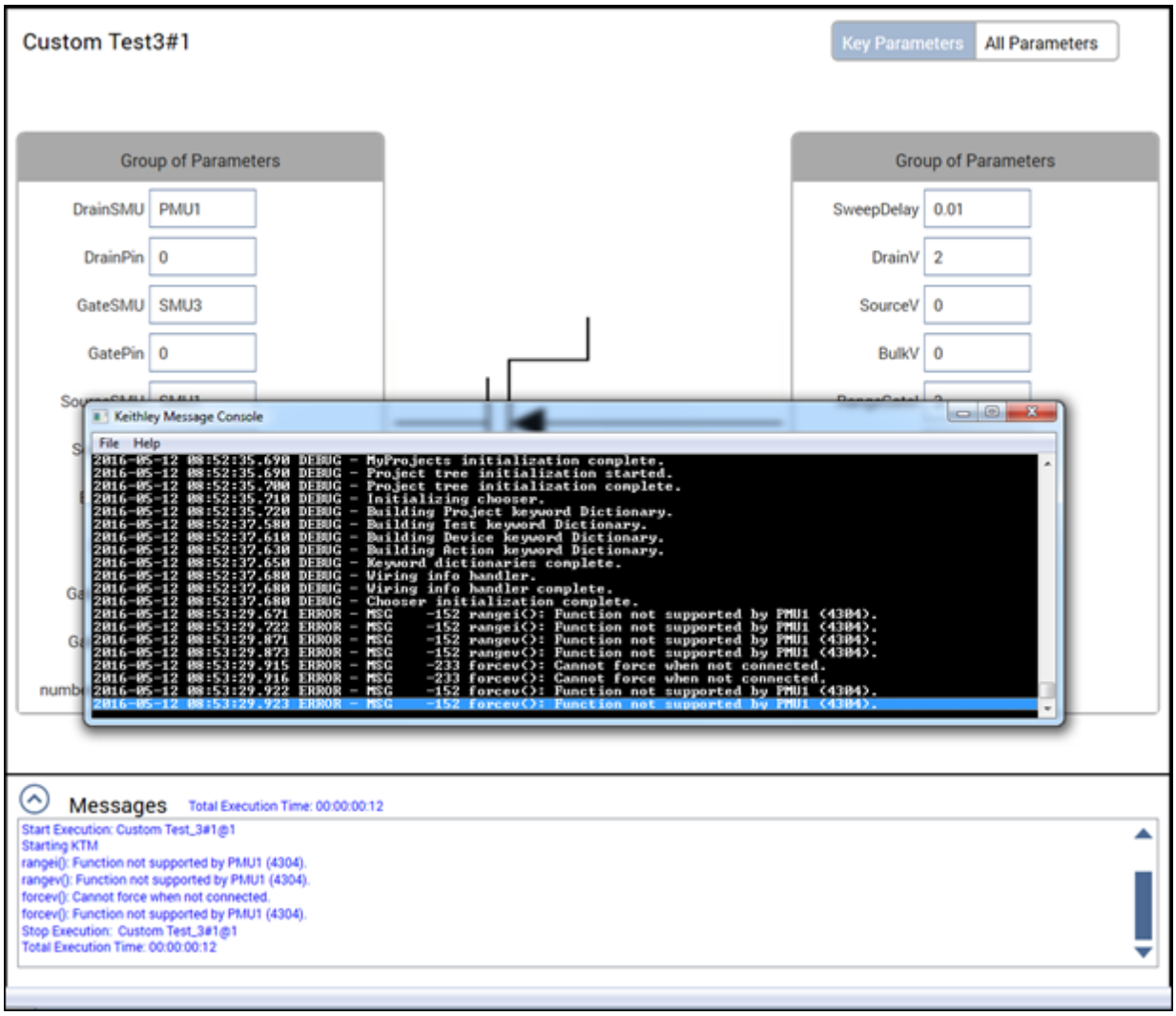
clrcon (on page 14-229)
clrscn (on page 14-11)
clrtrg (on page 14-14)
dev_abort (on page 14-99)
devclr (on page 14-72)
kibdefint (on page 14-26)

LPT Library Status and Error codes

Error codes are displayed whenever an invalid parameter or configuration occurs. The messages associated with the error codes describe the error condition to help the user module programmer or user determine how to address the error. Once an error occurs, the response of the user module to the error depends on how the user module is programmed. If a user module does not have any error handling, an initial error could cause additional errors on following LPT commands.

Library status and error codes are reported in Clarius in the message area.

Figure 616: LPT error codes in the Clarius message areas



Codes with positive values are statuses or updates. Codes with negative values are errors and warnings.

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 Customized error texts (on page 14-239) 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 reports a condition. This is usually a large number with an exponent of 10^{22} or 10^{23}. See Large number reported readings and explanations (on page 14-246) for the conditions associated with these large numbers.

Customized error texts

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

Code status or error titles

Code	Status or error titles
2802 to 2807	RPM: Invalid Configuration Requested
2801	RPM: Returned ID Error Response
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
1902	PMU: Transmission to analog from digital error
1901	PMU: Handshake from analog to digital error
1900	PMU: DA Communication Timeout
400 to 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.

Code	Status or error titles
-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.
-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.
-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.

Code	Status or error titles
-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.
-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.

Code	Status or error titles
-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.
-301	PCI ID read back on send error, slot.
-455	Protocol version mismatch.
-510	No command byte available (read) or SRQ not asserted.

Code	Status or error titles
-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.

Code	Status or error titles
-802	VPU in slot %d has reached thermal limit.
-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.

Code	Status or error titles
-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 segments 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 µs while relay settles.
-842	This test 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.
-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 available.
-10000 to -20000	User Module (UTM) error codes. Refer to user module description (help) for details.

Large number reported readings and explanations

Measurement Value	Condition
1.0000E+22 or 10.0000E+21	SMU is in range compliance (see Compliance limits (on page 3-4)), where the reading is at the maximum of a fixed range.
5.0000E+22	SMU is in range compliance, while auto ranging, where the reading is not at the maximum voltage or current range.
7.0000E+22	SMU in real compliance (see Compliance limits (on page 3-4)).
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.

LPT library and Clarius interaction when using UTMs

ITMs and UTMs are typically independent. However, an ITM and a UTM are not independent if the UTM occurs before an ITM in the project and the UTM configures a switch matrix. Under these conditions, the following occur:

- Clarius assumes that the ITM depends on the UTM-created switch configuration.
- Clarius maintains the UTM-created switch configuration during execution of the ITM.

Clarius actions affected by ITM and UTM sequence

Test sequence in the project	Clarius action
A UTM precedes an ITM	Before the ITM executes, the `devint` command 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 LPT library calls occur.
An ITM precedes a UTM	Before the UTM executes, the `devint` command initializes all devices, including the switch matrix.

Appendix A

Using switch matrices

In this appendix:

Typical test systems using a switch matrix

A switch matrix enhances the connectivity of the 4200A-SCS by allowing any SMU or preamplifier 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 preamplifiers.

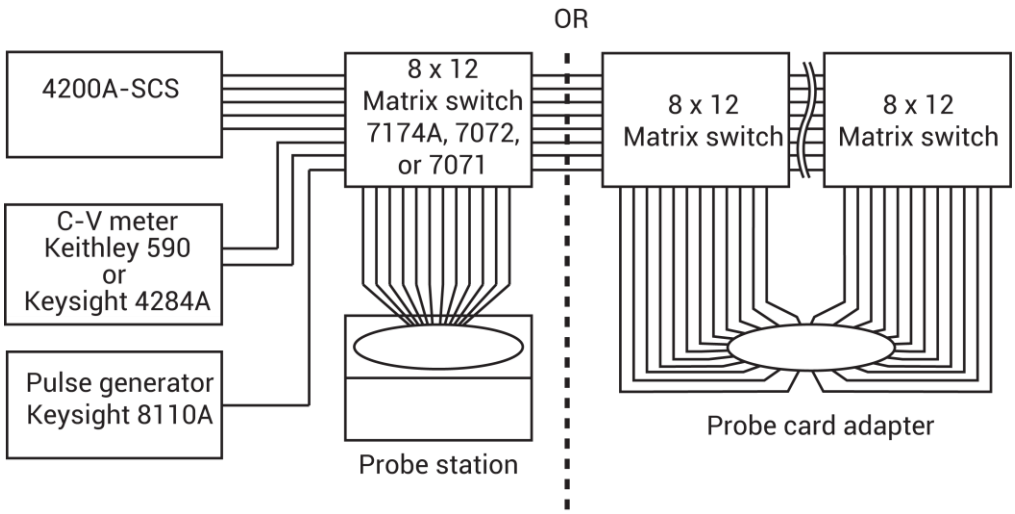
A switch matrix provides automatic switching for test instrumentation and devices under test (DUTs). Typical switch matrix systems are shown in the following figure.

The 4200A-SCS supports the Keithley Instruments Series 700 Switching System as external instruments. This series includes the 707, 707A, and 707B, which have six slots for matrix cards. This provides up to 72 pins of switching. This series also includes the 708, 708A, and 708B, which support a single matrix card for 12 pins of matrix switching.

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 user modules in the `Matrixulib` user library.

The following figure 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 617: Typical systems using a switch matrix



Matrix card types

The recommended Keithley Instruments matrix cards are:

- Model 7071 8 x 12 General Purpose Matrix Card, <100 pA offset current
- Model 7072 8 x 12 Semiconductor Matrix Card, <1 pA offset current
- Model 7136 (not available to purchase)
- Model 7174A 8 x 12 Low Current Matrix Card, <100 fA offset current
- Model 9174 8 x 12 Semiconductor Matrix Card , <100 fA offset current (not available to purchase)

Note that a key characteristic of these cards is low offset current to minimize the negative effects of offset currents on low-current measurements.

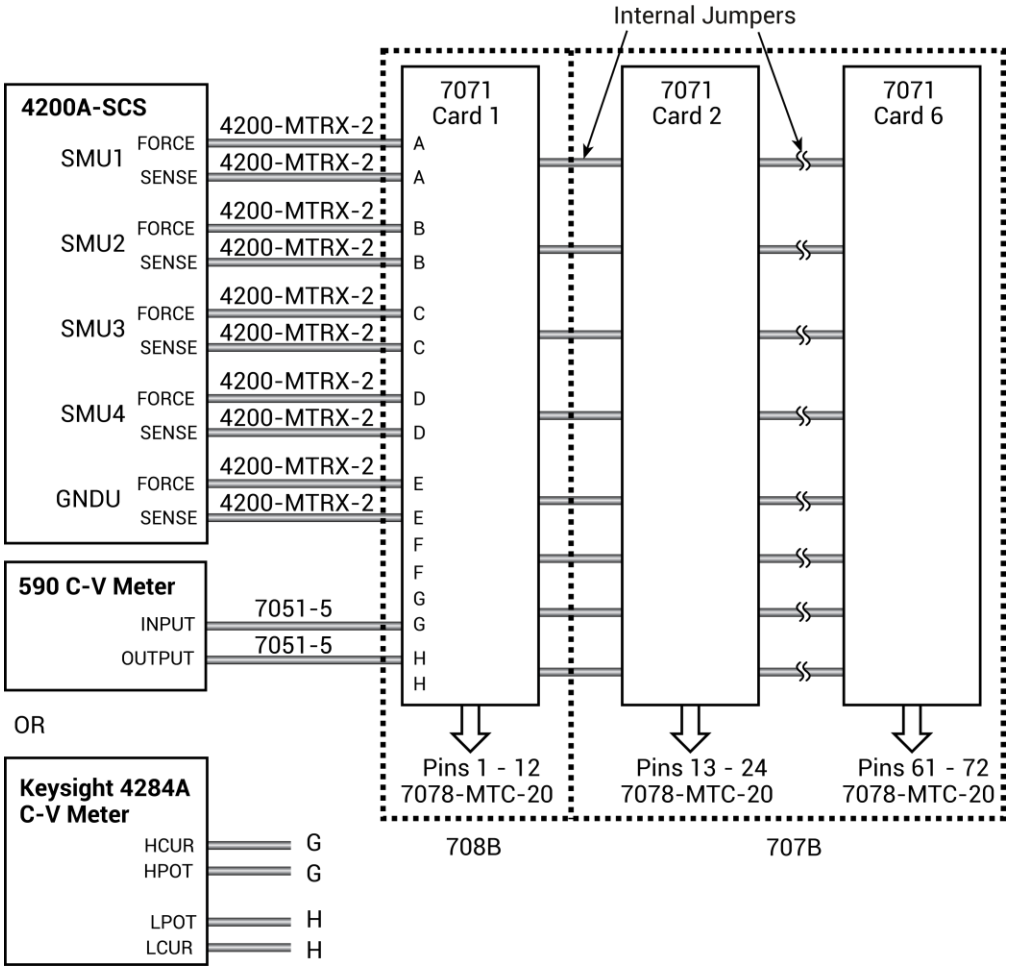
7071 General Purpose Matrix Card

The 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.

The next figure shows a test system using 7071 matrix cards. The triaxial and BNC cables are unterminated on one end to allow direct hard-wire connections to the screw terminals of the matrix card. It shows how signals are routed through the 7071 matrix switches to a DUT.

The test system in the next figure uses remote sensing. Notice that each FORCE and SENSE terminal-pair of the 4200A-SCS shares the same path (row).

Figure 618: Test system using 7071 matrix cards



7072 Semiconductor Matrix Card

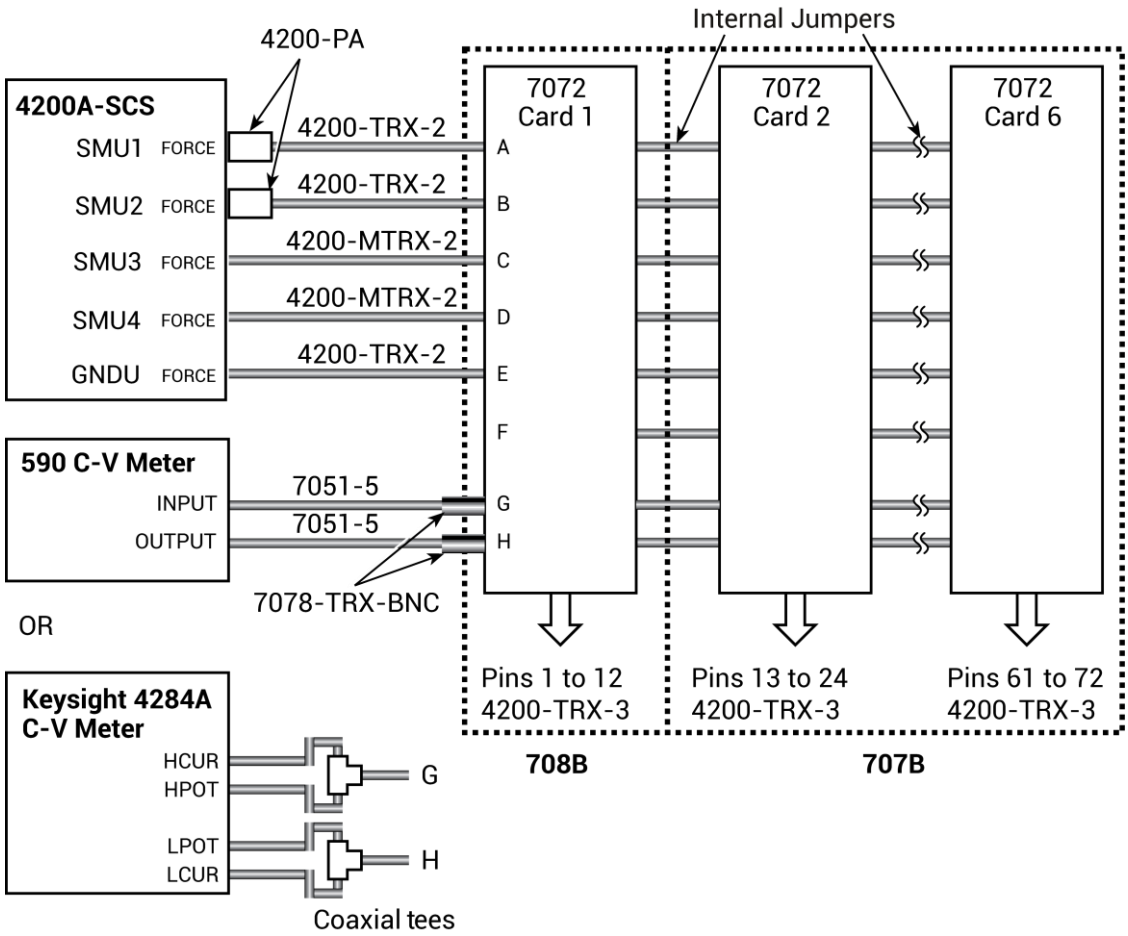
The 7072 provides two two-pole low current paths that have <1 pA offset current (rows A and B), two 1-pole CV paths for characterization from DC to 1 MHz (rows G and H), and four two-pole paths for general purpose switching (rows C, D, E, and F). The card is equipped with 3-lug triaxial connectors for signal connections. The maximum signal level is 200 V, 1 A. The maximum leakage is 0.01 pA/V and the 3 dB bandwidth is 5 MHz (CV channals).

The next figure shows a test system using 7072 matrix cards. The connection requirements for this card are the same as the connection requirements for the 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 preamplifiers with the 4200A-SCS, they should be connected to the first two rows of the 7072 matrix card.

The next figure and the C-V Analyzer signal paths (on page A-27) figures show how signals are routed through 7072 matrix switches to a DUT.

Figure 619: Test system using 7072 matrix cards



7174A Low Current Matrix Card

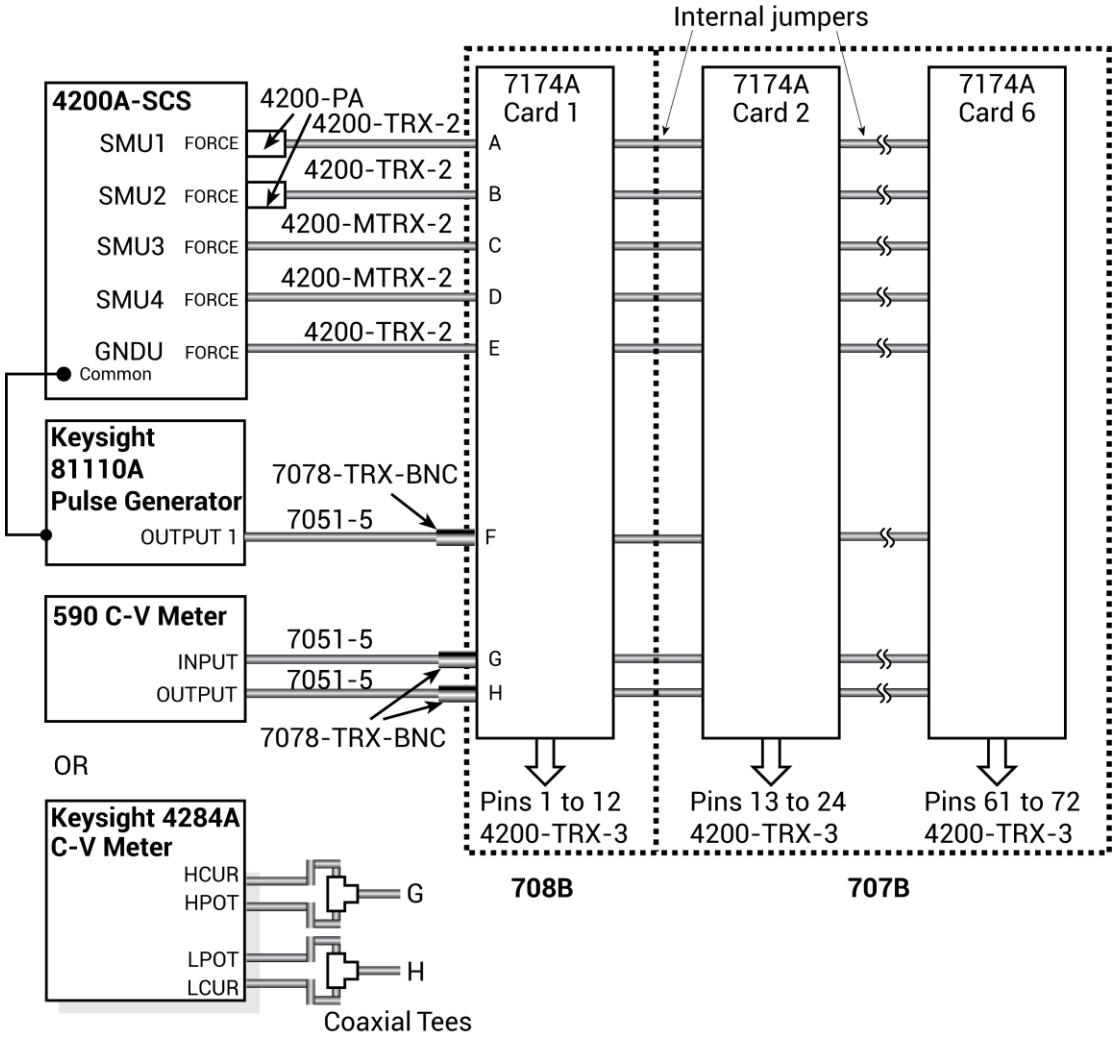
The 7174A provides high quality, high performance switching of I-V and C-V signals. This matrix card uses 3-pole switching (HI, LO, Guard) with 10 fA typical offset current. The card is equipped with 3-lug triax connectors for signal connections.

The following figures show test systems using 7174A matrix cards. The supplied triaxial cables connect the 4200A-SCS directly to matrix rows. The other instruments in the system are fitted with BNC connectors that require the use of BNC-to-triaxial adapters.

7174A connections for local sensing

The following figure shows a system that uses local sensing. Note that coaxial tees are used to adapt the Keysight 4284A C-V meter for two-terminal operation.

Figure 620: Test system using 7174A matrix cards



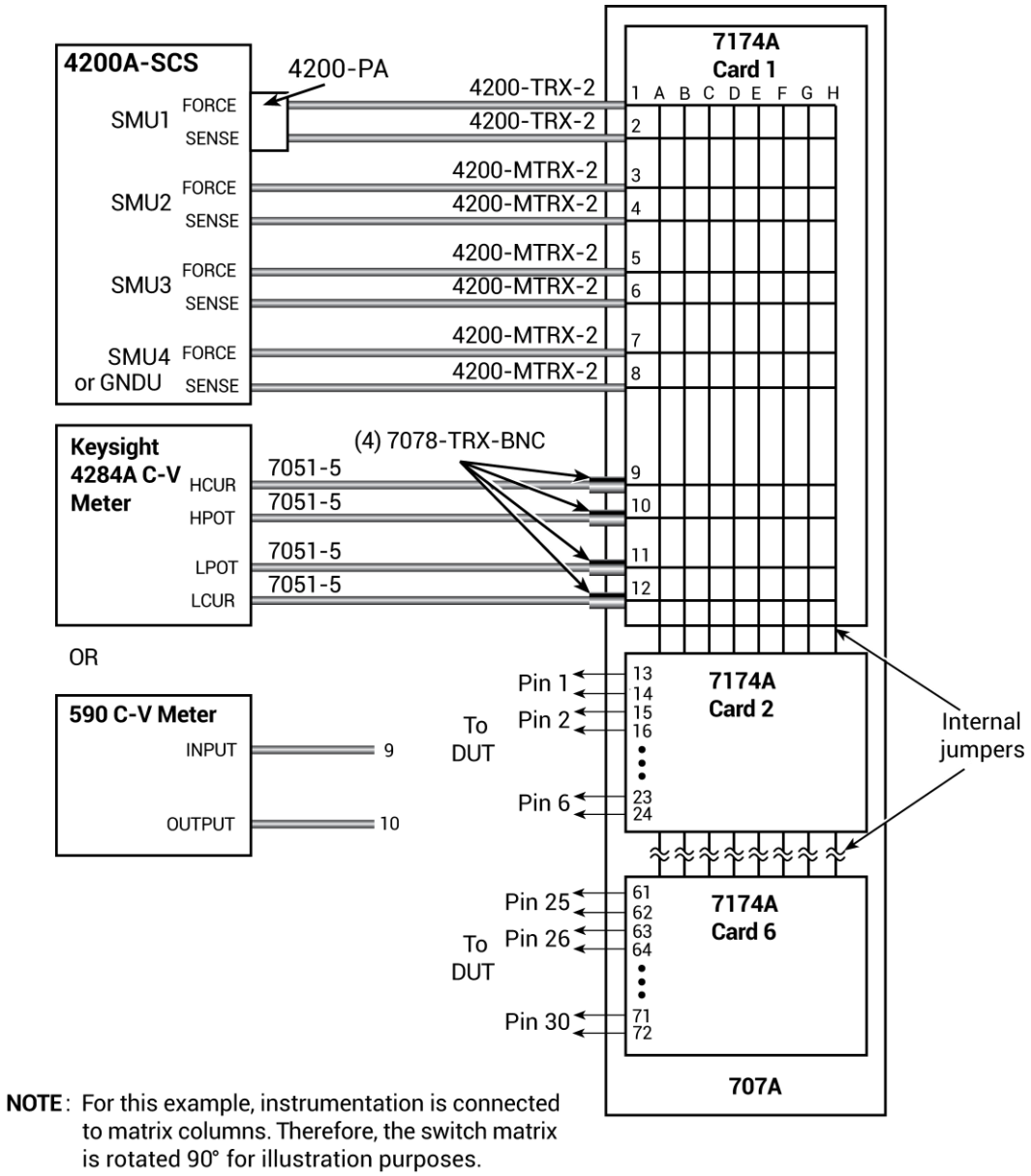
7174A connections for remote sensing

The next figure 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 device under test (DUT). With five DUT matrix cards installed in a Series 700 Switching System mainframe, up to 30 DUT pin-pairs can be used.

NOTE

In the next figure, the C-V Analyzer signal paths (on page A-27) for the Keysight Model 4980A and Keysight Model 8110A pulse generator signal path (on page A-30) show how signals are routed through 7174A matrix switches to a DUT.

Figure 621: Remote sense test system using 7174A matrix cards



NOTE: For this example, instrumentation is connected to matrix columns. Therefore, the switch matrix is rotated 90° for illustration purposes.

Switch matrix mainframes

The 4200A-SCS provides a user library containing preconfigured data acquisition and control user modules for the Series 700 Switch System.

You can use the 4200A-SCS with switch matrices from other vendors. However, you will need to develop software to control these matrices from Clarius+. See Keithley User Library Tool (KULT) (on page 8-1) for information about developing user modules and libraries.

Card installation

Refer to the instructions for your matrix card for card installation instructions.

GPIB connections

The 4200A-SCS controls the switch matrix using the GPIB interface. Connect the GPIB port of the switch matrix to the 4200A-SCS using a 7007-1 or 7007-2 GPIB cable.

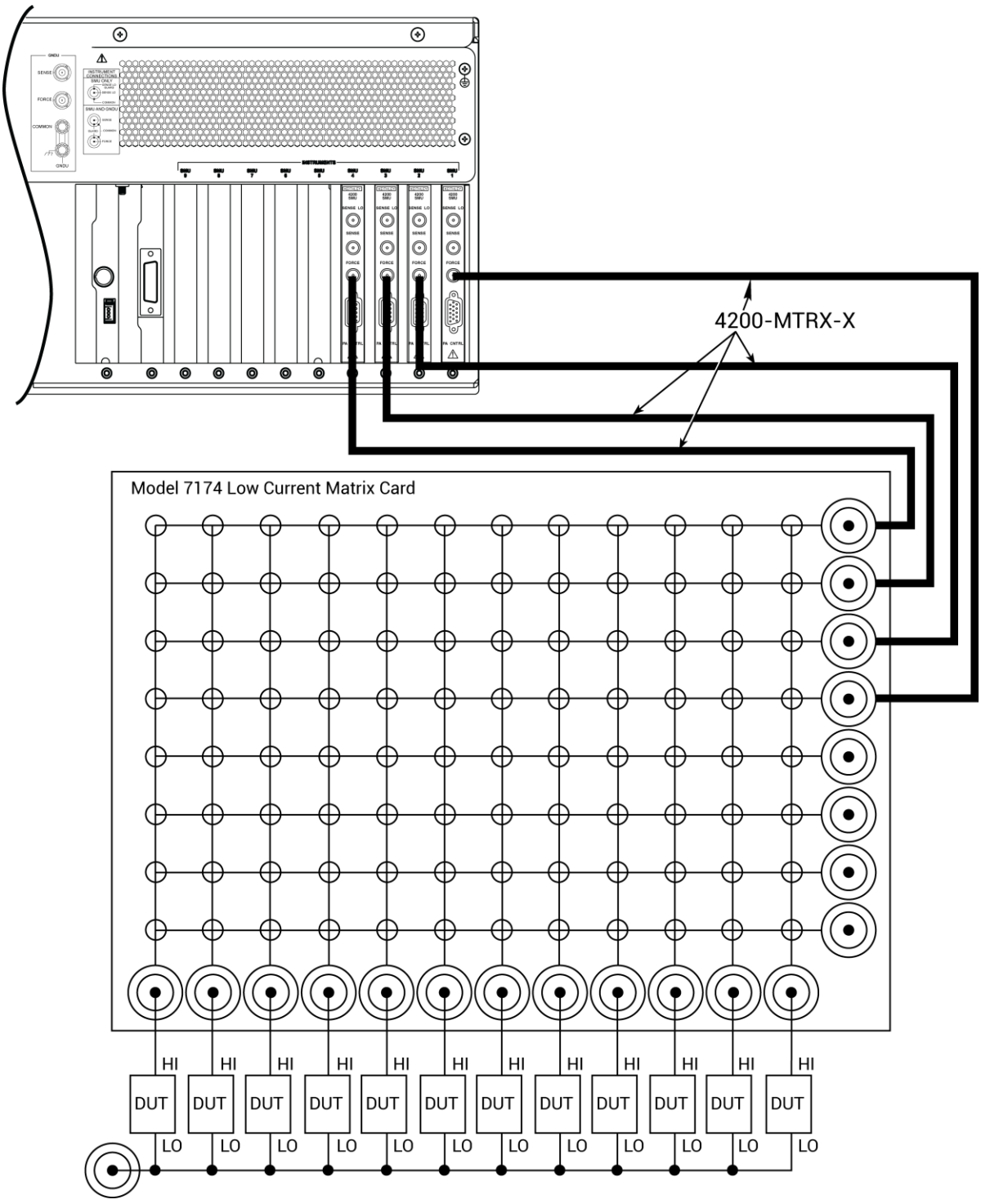
Switch matrix connections

A switch matrix enhances the connectivity of the 4200A-SCS by allowing any SMU or preamplifier signal to be connected to any DUT pin. Typically, devices are connected to columns and instruments are connected to rows. The following topics summarize recommended switching mainframes and matrix cards. They also show typical connection schemes with SMUs and preamplifiers.

Typical SMU matrix card connections

The figure below 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. Any SMU FORCE terminal can be connected to any DUT HI terminal simply by closing the appropriate matrix crosspoint.

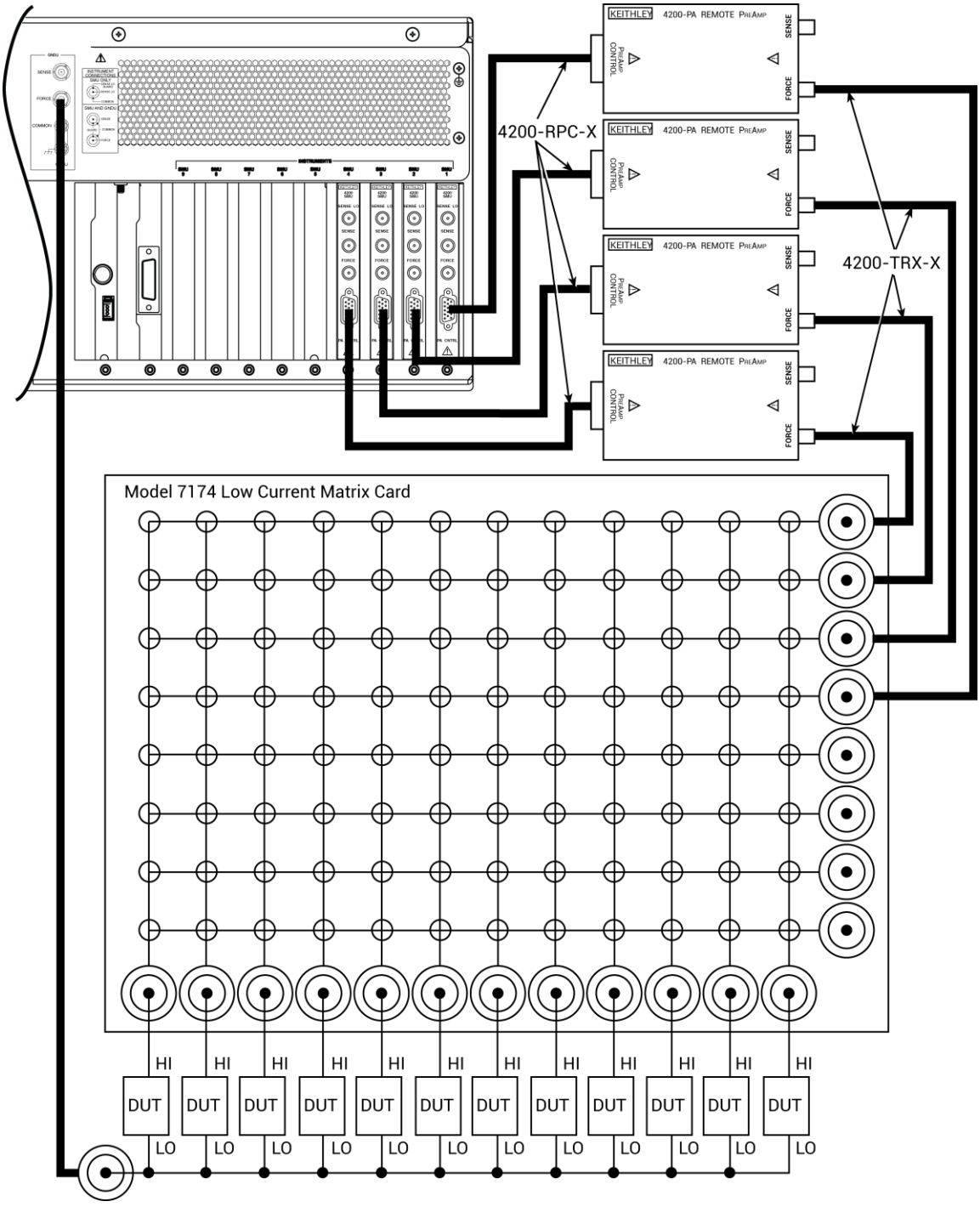
Figure 622: Typical SMU matrix card connections



Typical preamplifier matrix card connections

The following figure shows typical preamplifier matrix card connections using local sensing. This configuration is similar to the SMU configuration shown in the previous figure, except that preamplifiers are added for low-current source-measure capabilities. The preamplifier 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. Any preamplifier FORCE terminal can be connected to any DUT HI terminal by closing the appropriate matrix crosspoint.

Figure 623: Preamplifier matrix card connections



Typical CVU matrix card connections

In your project, you can automate the use of a CVU and other instrumentation using a switching matrix and actions to control the switching. When the project is run, the switching matrix automatically makes the required instrument connections for each test in the project.

The next figures show typical connections for a switch system using a Series 700 Switching System with the 7174A Matrix Card installed.

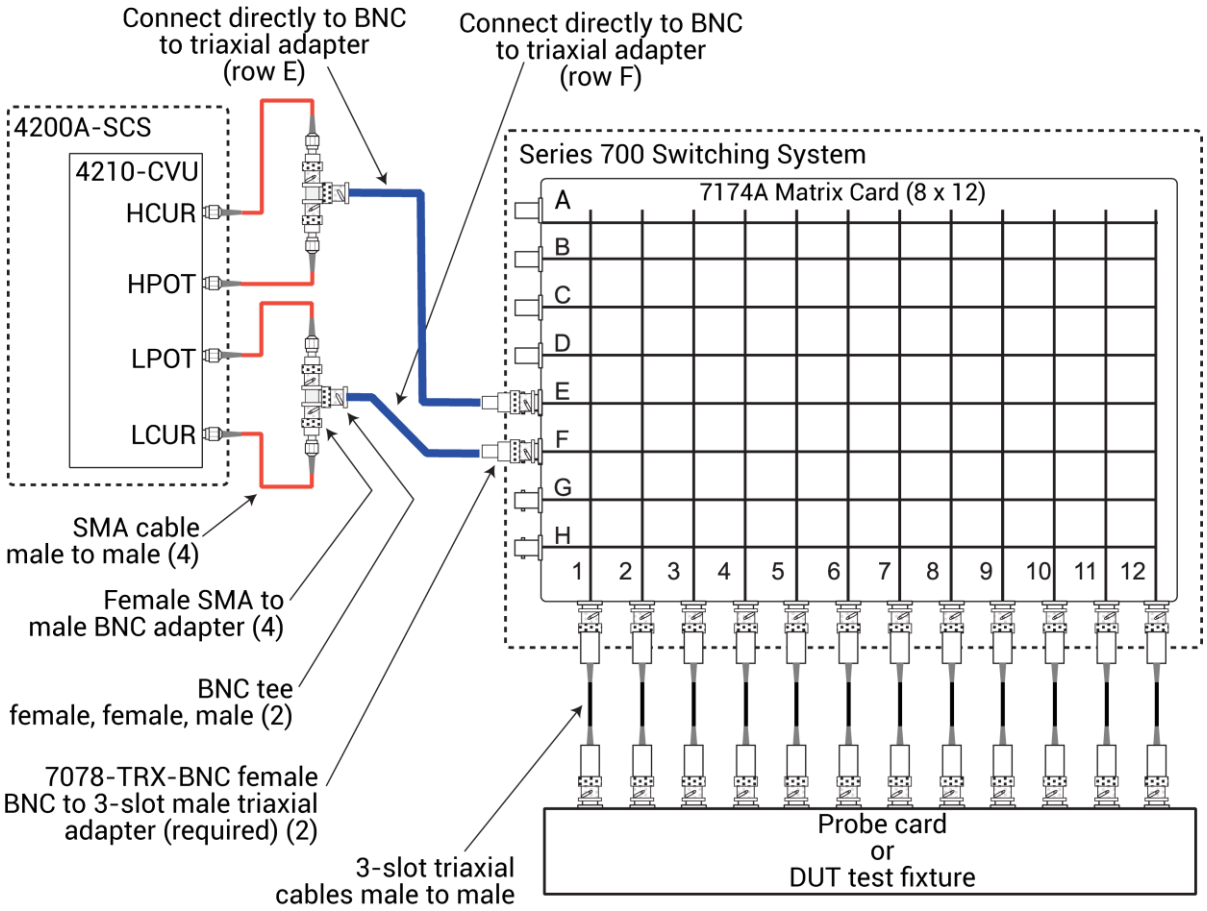
NOTE

You can also use the 7072 Matrix Card for C-V testing. However, you must use rows G and H and local (2-wire) sensing.

The SMA cables and adapters shown in the following drawings are supplied with the CVU or the 4210-CVU-Prober-Kit. The triaxial and BNC cables are not supplied. The prober kit includes two types of BNC-to-triaxial adapters that connect directly to the rows of the matrix. The 7078-TRX-BNC has the guard connected to the inner shield of the adapter. The 7078-TRX-GND has the guard disconnected.

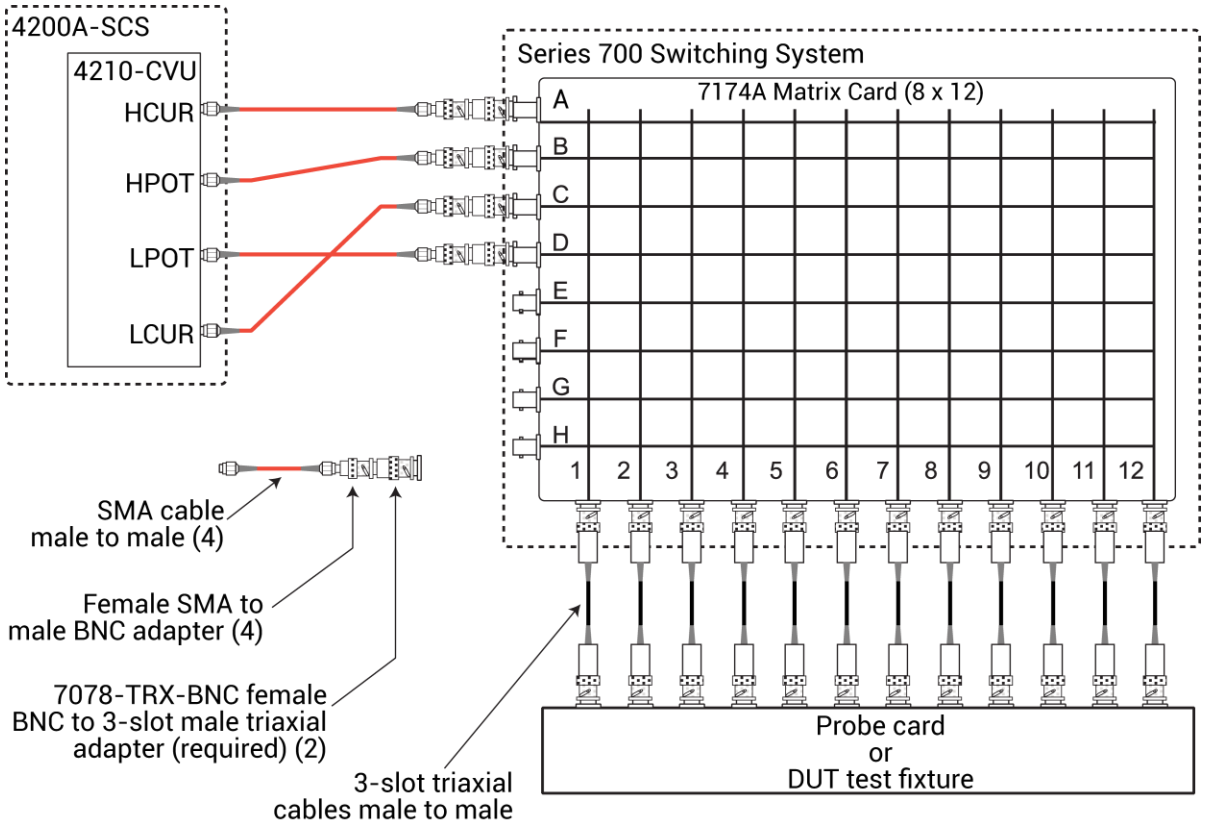
This figure shows connections for local (2-wire) sensing. It shows the 4210-CVU connected to rows E and F of the matrix. This is the connection scheme for the `cap-iv-cv-matrix` project. For details, see cap-iv-cv-matrix (on page 4-49).

Figure 624: Test connections for a switch matrix - local (2-wire) sensing



The following figure shows connections for remote (4-wire) sensing.

Figure 625: Test connections for a switch matrix - remote (4-wire) sensing



NOTE

The 7078-TRX-BNC adapters must be used in order to extend SMA shielding through the matrix card.

NOTE

The shields of the SMA cables must be connected together and extended as far as possible to the DUT, as shown in Typical 4210-CVU test connections to a DUT (on page 4-8).

Connection scheme settings

The following connection scheme settings are set from the Keithley Configuration Utility (KCon) when the switch matrix is added to the system configuration. See Using KCon to add a switch matrix to the system (on page A-31).

Row-column or instrument card settings

You select the scheme for interconnections between the instruments, the switch-matrix rows and columns, and the test system (prober or test fixture). You can select:

- **Row-Column:** Connect instruments to rows and prober or test fixture to columns.
- **Instrument Card:** Both instruments and prober or test fixture are connected to columns. Matrix rows are not used.

The row-column setting is the simplest connection scheme. In this scheme, instruments are connected to the switch-matrix rows. The prober/test fixture pins or the device under test (DUT) are connected to the switch-matrix columns (see Switch matrix control (on page A-21) and 4200A-SCS signal paths (on page A-22)).

When you set up a matrix, you also select the sense. You can select:

- **Local sense:** 2-wire conections. Connections are only to instrument FORCE terminals.
- **Remote sense:** 4-wire connections. Connections are to both instrument FORCE and SENSE terminals.

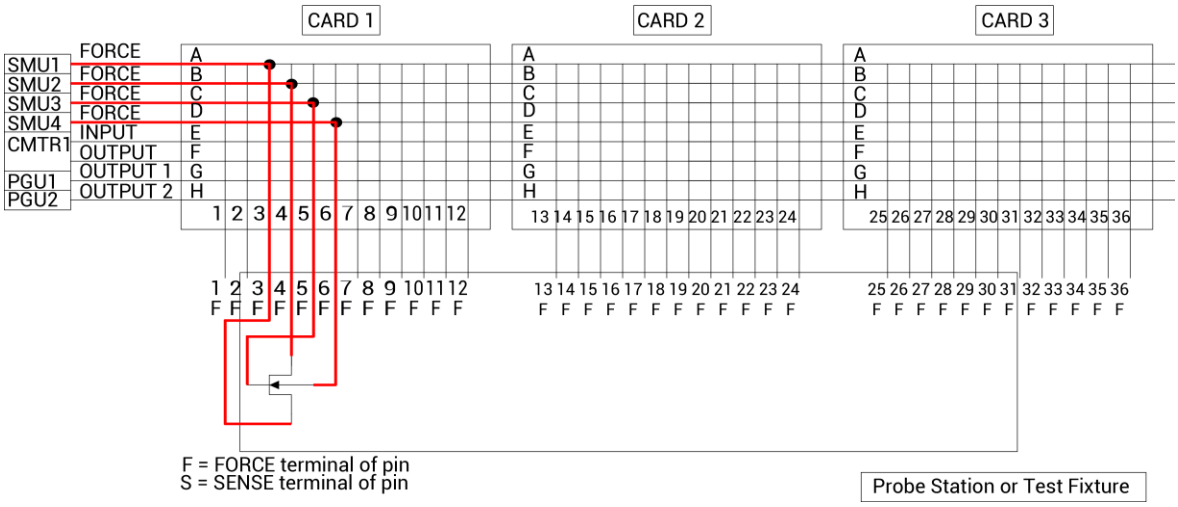
For more information regarding local and remote sense, refer to Remote sensing (on page 3-39).

Row-column scheme

The row-column setting is the simplest connection scheme. In this scheme, instruments are connected to the switch-matrix rows. The prober/test fixture pins or the device under test (DUT) are connected to the switch-matrix columns (see Switch matrix control (on page A-21) and 4200A-SCS signal paths (on page A-22)).

Instrument signals can route to prober/test-fixture pins through only one matrix card, as shown in the following figure. However, the Row-Column scheme limits the number of external instruments. If the instrumentation requirements exceed eight paths (rows), you must use the instrument card configuration.

Figure 626: Row-Column, Local Sense Connection Scheme example



Instrument card scheme for 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

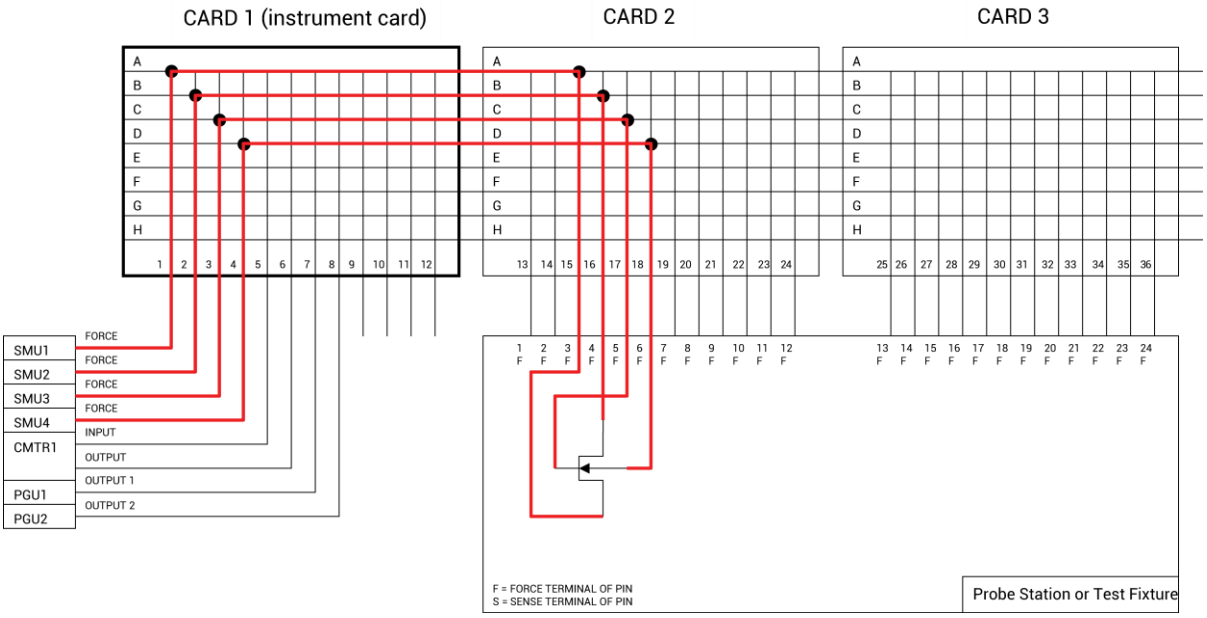
Current flowing through the measurement pathway creates a voltage drop (an error voltage) that is directly proportional to the pathway resistance. This error voltage is present in all local sense voltage measurements.

When local sense is selected, only the connection paths specified by the connected action are completed. For example, in the figure in Switch matrix control (on page A-21), the specified connection paths would be:

- SMU2, 6 (connect SMU2 to Pin 6)
- GNDU, 3 (connect GNDU to Pin 3)

For the instrument card scheme, both the instrumentation and the prober/text-fixture pins or DUT are connected to switch-matrix columns. No external connections are made to matrix rows. In this configuration, two switch relays are closed to complete a path from an instrument to a DUT. Instrument signals route to the prober/test-fixture pins through two or more matrix cards, as shown in the following figure. This connection scheme can support large systems with numerous instruments by removing the eight-row instrument connection limitation.

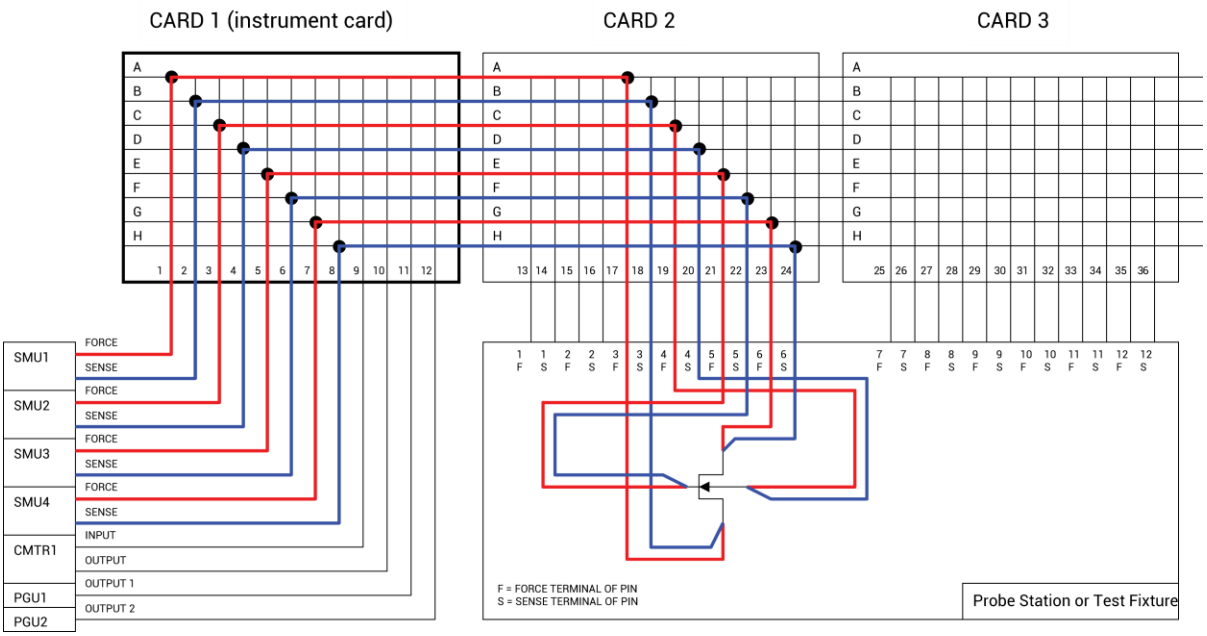
Figure 627: Instrument Card, Local Sense Connection Scheme example



Instrument card scheme for remote sense

Use remote sense to eliminate the effects of measurement pathway resistance. The following figure 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 628: Instrument Card, Remote Sense Connection Scheme example



When remote sense is 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` action, the paired connection path is also completed. For example, in the figure in 4200A-SCS signal paths (on page A-22), the specified connection paths would be:

- SMU1, 4 (connect SMU1 to Pin 4)
- GNDU, 3 (connect GNDU to Pin 3)

Switch matrix control

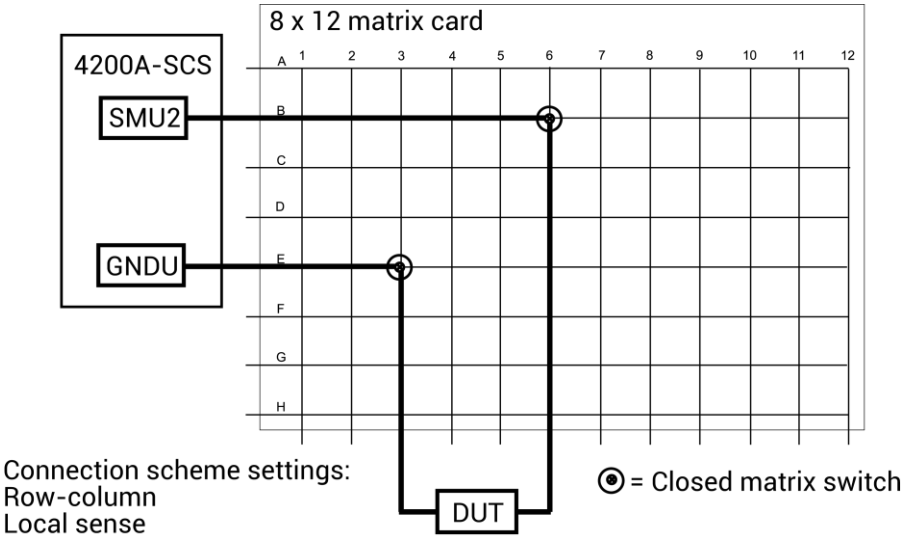
You can use the `connect` action in the `ivcvswitch` project to control switching. You can also use the `ConnectPins` user module in the `Matrixulib` user library.

The `connect` action uses the `ConnectPins` user module to control a switch matrix. You specify the instrument terminal and pin pairs. For example, for the row-column connection scheme shown in the figure below, you set the parameters:

- `TermIDStr2` to `SMU2` and `Pin2` to `6`, which connects SMU2 to pin 6.
- `TermIDStr8` to `GNDU` and `Pin8` to `3`, which connects GNDU (ground unit) to pin 3.

A matrix control example using the `ConnectPins` user module is provided in Switch matrix control example (on page A-39). Detailed information for `ConnectPins` is provided in Matrixulib user library (on page 6-390).

Figure 629: Row-column connection scheme



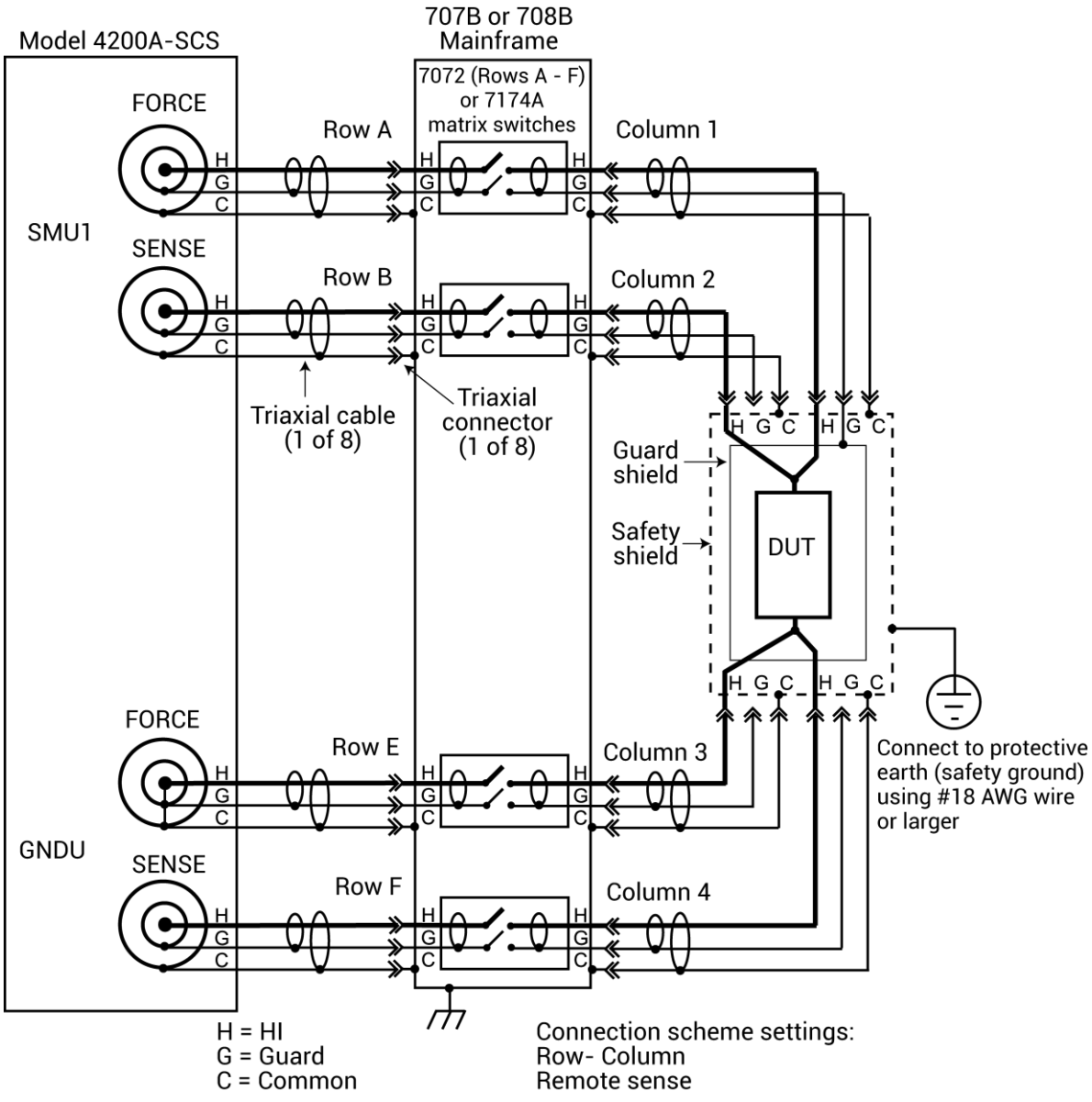
Signal paths to a DUT

The following figures show signal path examples from the various test instruments through the matrix switches to a DUT.

4200A-SCS signal paths

The following figure shows remote sensing (4-wire) signal paths through a matrix card using two-pole switching. Two-pole switching is provided by the 7174A and 7072 (rows A through F).

Figure 630: 4200A-SCS signal paths through a two-pole matrix card using remote sensing



Sense setting

To make the connections shown in 4200A-SCS signal paths (on page A-22), you must select remote sensing.

When remote sensing is selected, the 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 are also closed.

For local sensing (2-wire), the connections from the SENSE terminals of the 4200A-SCS are not used.

NOTE

See Connection scheme settings (on page A-15) for details on local and remote sensing.

Connection setting

The row-column setting must be used when connecting instrumentation to matrix rows, as shown in 4200A-SCS signal paths (on page A-22).

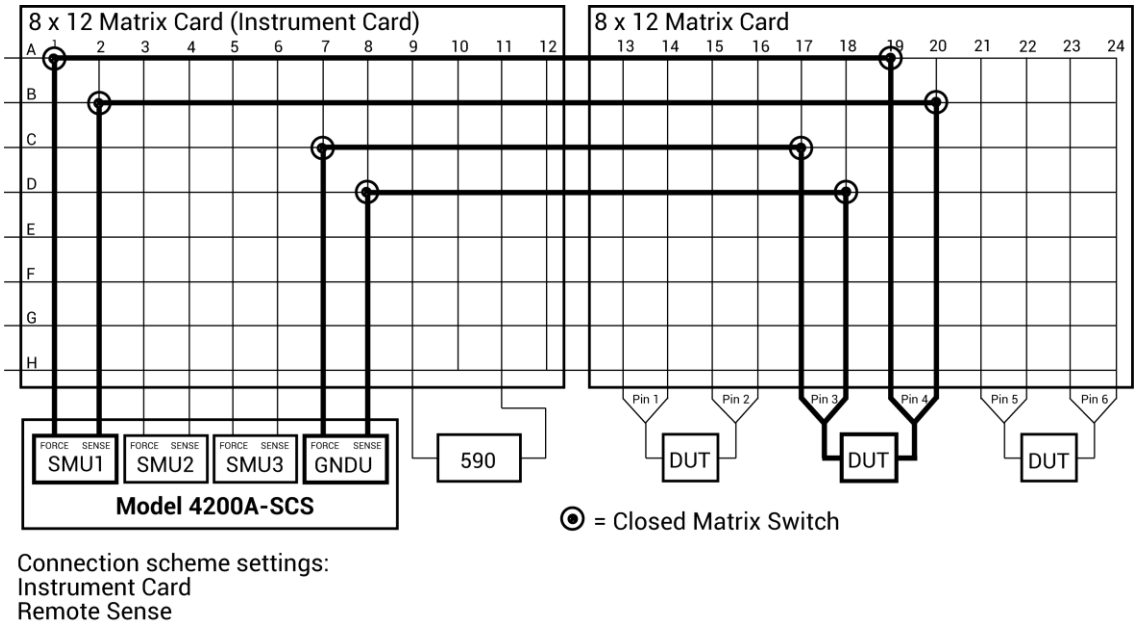
The maximum number of rows available to the test system is eight. If instrumentation needs more than eight pathways, they must be connected to matrix columns, and the instrument card setting must be used.

The figure below shows a test system with both the instruments and the DUT connected to matrix columns.

NOTE

See Connection scheme settings (on page A-15) for details on the row-column and instrument card settings.

Figure 631: Instrument card connection scheme



NOTE

The 4200A-SCS automatically selects the first available rows to make connections to the DUT. In this example, rows A through D are the first available rows.

The following shows 4200A-SCS signal paths through a 3-pole 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 4200A-SCS signal paths (on page A-22), 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 4200A-SCS.

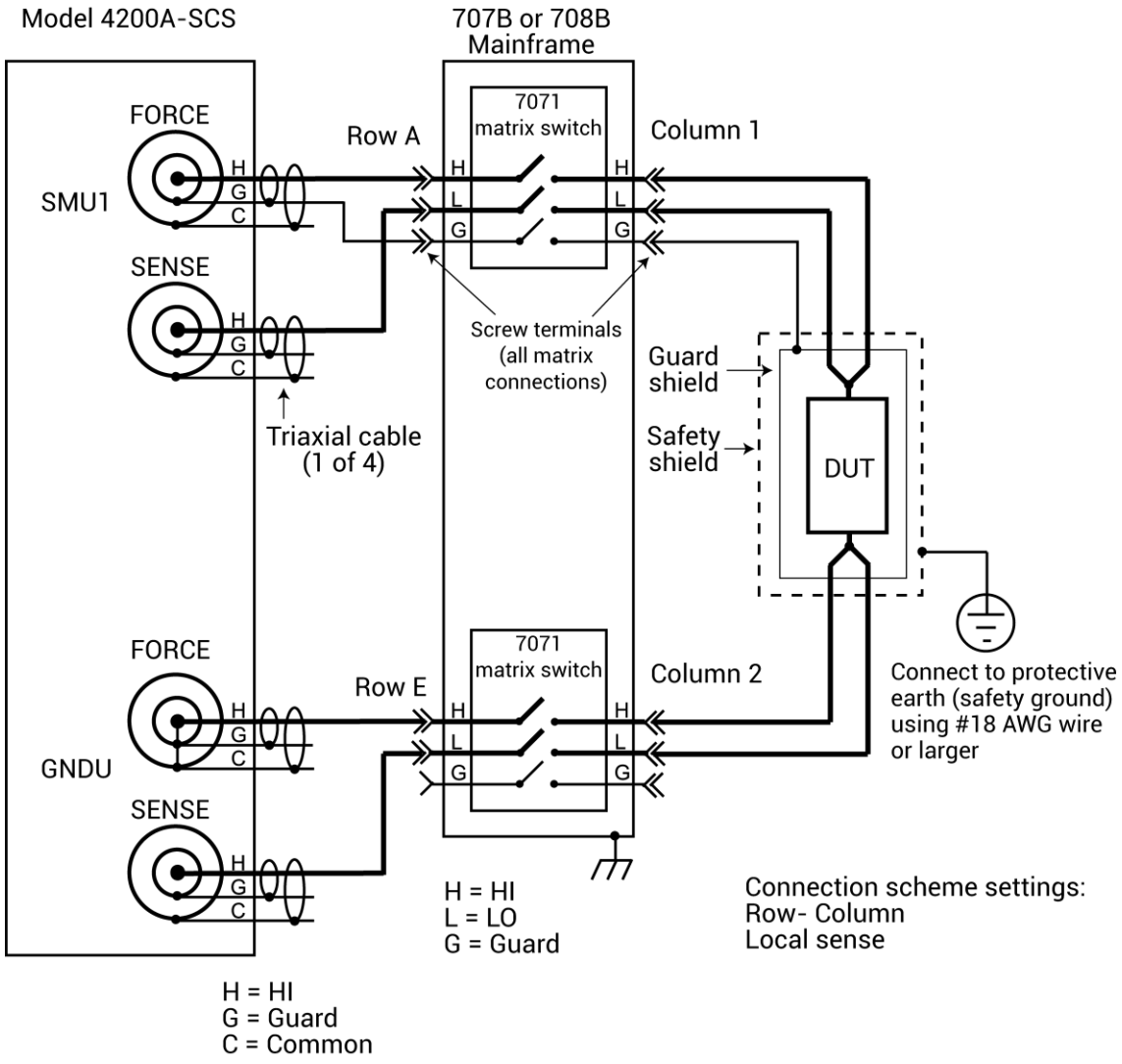
WARNING

To avoid high voltage exposure that could result in personal injury or death, whenever the interlock of the 4200A-SCS is asserted, the FORCE and GUARD terminals of the SMUs and preamplifier should be considered to be at high voltage, even if they are programmed to a non-hazardous voltage current.

NOTE

See Connection scheme settings (on page A-15) for details on sense settings.

Figure 632: 4200A-SCS signal paths through a 3-pole matrix card using remote sensing



C-V Analyzer signal paths

The next two figures show local sense C-V Analyzer signal paths through rows B and H of a 7072 matrix card. A C-V analyzer can be used with any of the three matrix card types; however, rows G and H of the 7072 are optimized for C-V measurements.

Figure 633: 590 signal paths through 7072 matrix card using local sensing

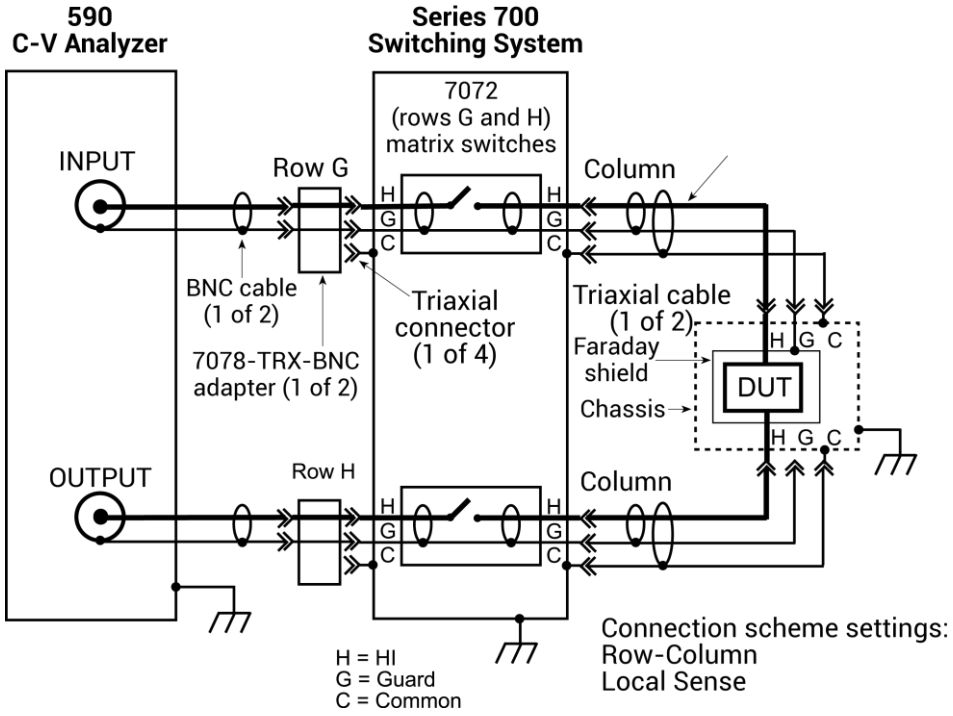
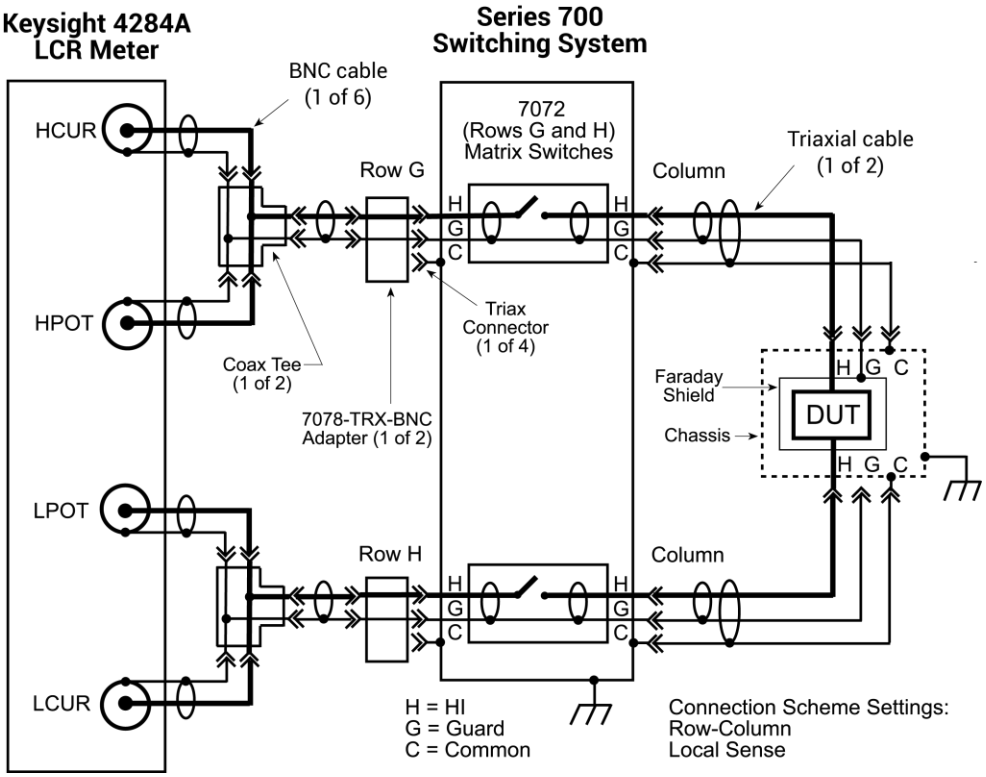
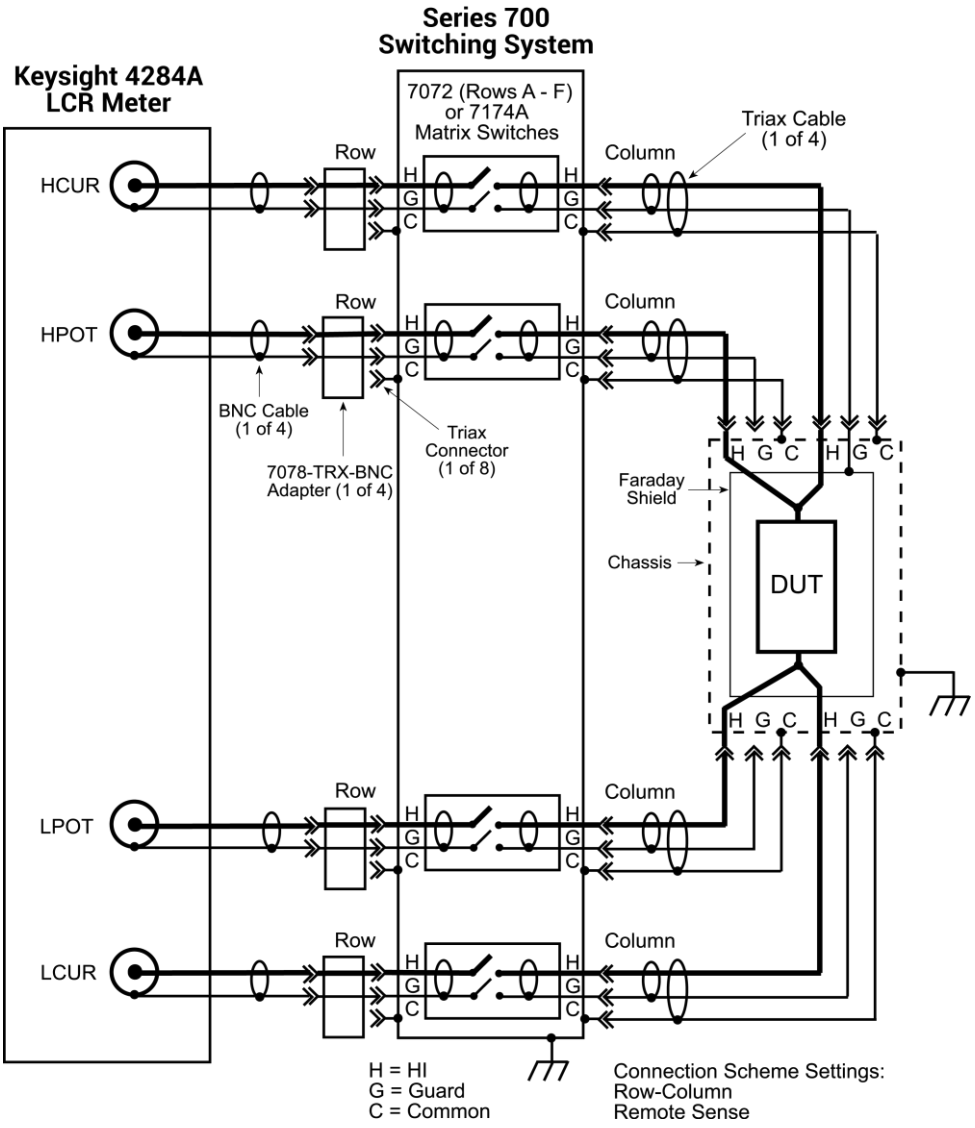


Figure 634: Keysight Model 4980A signal paths through 7072 matrix card using local sensing



The next figure shows the remote sense signal paths for the Keysight Model 4980A LCR meter through a 2-pole matrix card. Since row pairing is required, the remote sense setting must be used.

Figure 635: Keysight Model 4980A signal paths through a two-pole matrix card using remote sensing

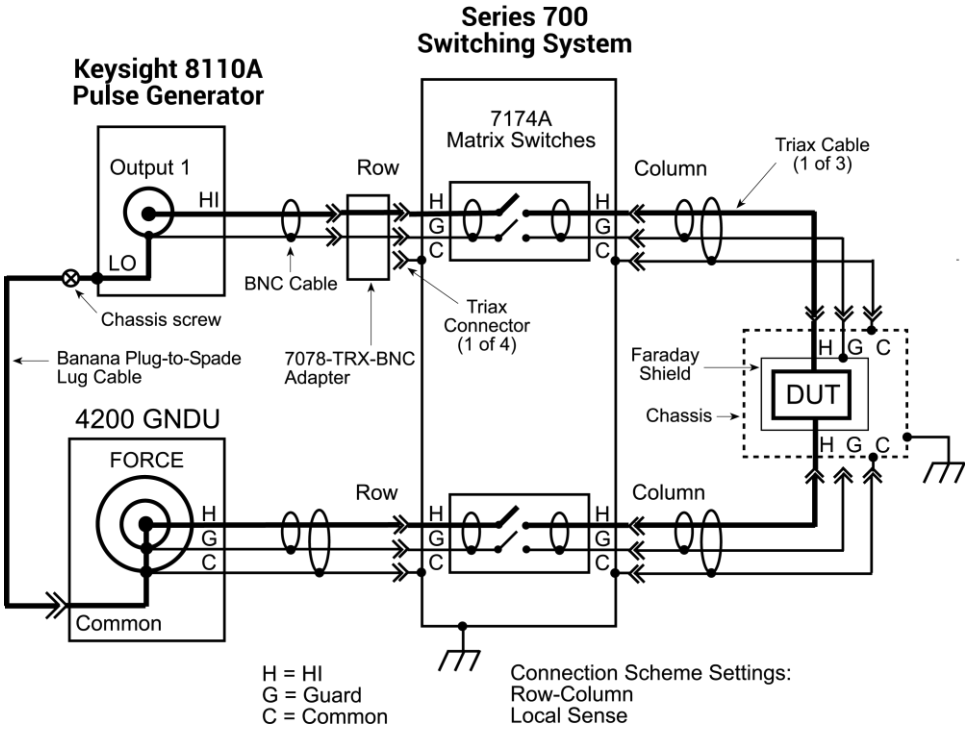


Keysight Model 8110A pulse generator signal path

The next figure shows the HI signal path through the 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 the next figure, 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 636: Keysight Model 8110A signal path through a 7174A matrix card



Use KCon to add a switch matrix to the system

You use Keithley Configuration Utility (KCon) to manage the configuration of all instrumentation controlled by the 4200A-SCS software. To use the 4200A-SCS to control a switch matrix, you must add the switch matrix to the system configuration using KCon.

If you are testing discrete device under test (DUTs), you must use the switch matrix with a test fixture. If you are testing a wafer, you must use the switch matrix with a probe station. The test fixture or probe station is also added to the system configuration using KCon.

You specify physical instrument-to-card and card-to-prober or fixture connections in KCon.

These and other KCon switch matrix settings result in simplified matrix connections. Initially, you need to:

- Add the test fixture or probe station.
- Configure the Instrument Connection Scheme and Switch Cards areas.
- Specify the physical instrument-to-card and card-to-prober/fixture connections.
- Physically make the specified instrument-to-card and card-to-prober/fixture connections.

After the initial setup, you can specify instrument-to-prober/fixture connections by specifying the corresponding terminal and prober/fixture pins in a Clarius user test module (UTM). You do not need to specify matrix cross points. The 4200A-SCS automatically routes the signals through the matrix.

For additional detail on KCon, refer to Keithley Configuration Utility (KCon) (on page 7-1).

NOTE

The Series 700 Switching System must be set to DDC compatibility mode to be used with the 4200A-SCS. Refer to the Series 700 documentation for information on how to make this setting.

Step 1. Exit Clarius and open KCon

To exit Clarius and open KCon:

1. Exit Clarius.
2. On the Windows desktop, select the **KCon** icon.

Step 2. Add a test fixture or probe station

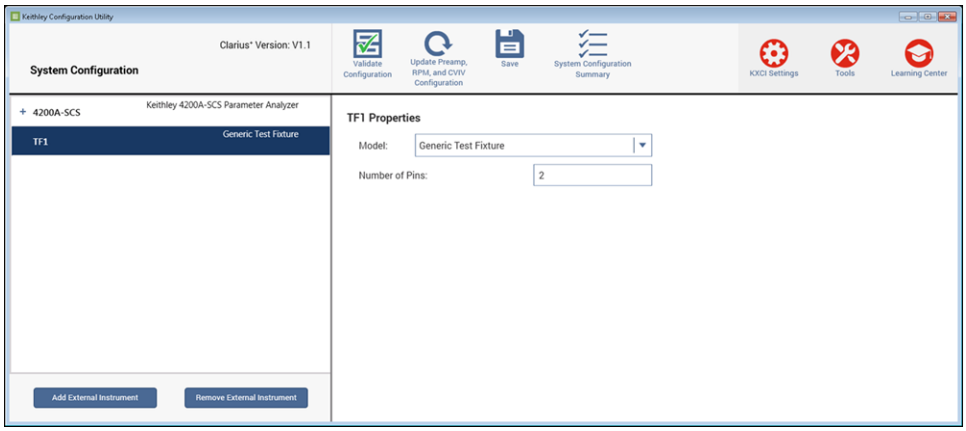
You must use a test fixture or a probe station with the switch matrix. However, both cannot be in the system configuration together. Refer to Remove an external instrument (on page 7-10) for information if you need to remove a component.

Add a test fixture

To add a test fixture to the system configuration:

1. Select **Add External Instrument**.
2. Select **Test Fixture**.
3. Select **OK**.
4. In the Configuration Navigator, select the test fixture (prefix is **TF**).

Figure 637: Add test fixture



5. From the **Model** list, select the appropriate test fixture.
6. Enter the number of pins. You can enter 2 to 72 pins. If you selected the Keithley LR:8028 or 8007 test fixture, the number of pins is automatically set and cannot be changed.

NOTE

The number of pins defined in the test fixture properties determines the pins that are available to assign to a switch matrix card column. Make sure the number of pins assigned is appropriate for your system.

Add a probe station

Supported probe stations include:

- Fake Prober
- Manual Prober
- Micromanipulator 8860 Prober
- Karl Suss PA200 Prober
- Cascade 12000 Prober
- Signatone CM500 (WL250) Prober
- MPI TS2000, TS2000-DP, TS2000-HP, TS2000-SE, TS3000, and TS3000-SE Probers

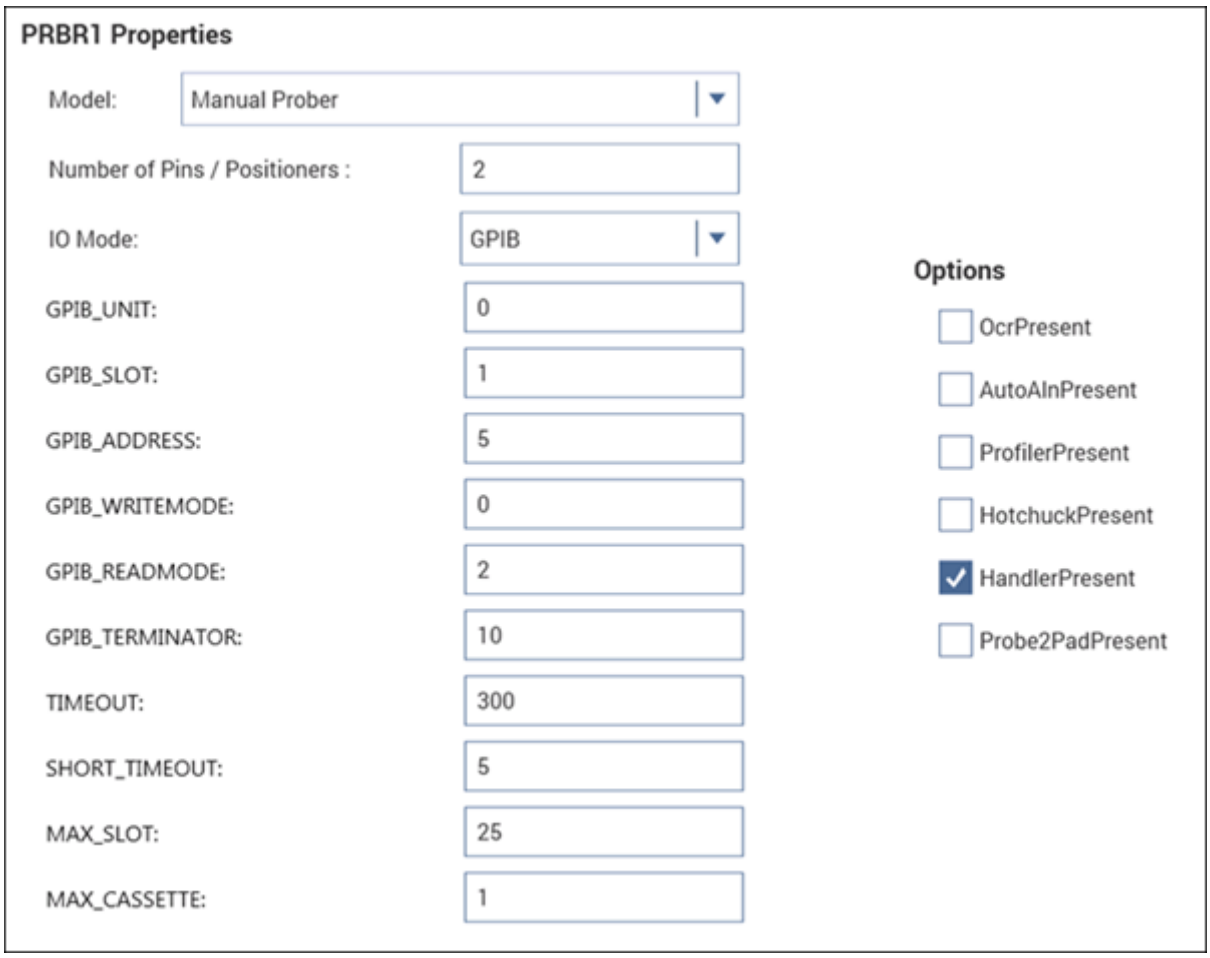
NOTE

Contact Keithley for the most up-to-date list of supported probers. If you are using an unsupported prober, you will have to create a user library and module to control it.

To add a probe station to the system configuration:

1. Select **Add External Instrument**.
2. Select **Probe Station**.
3. Select **OK**.
4. In the Configuration Navigator, select the probe station. The Properties are displayed.

Figure 638: Probe station properties



5. From the **Model** list, select the prober.
6. Enter the **Number of Pins / Positioners**.
7. Select the options that are appropriate for your prober.

NOTE

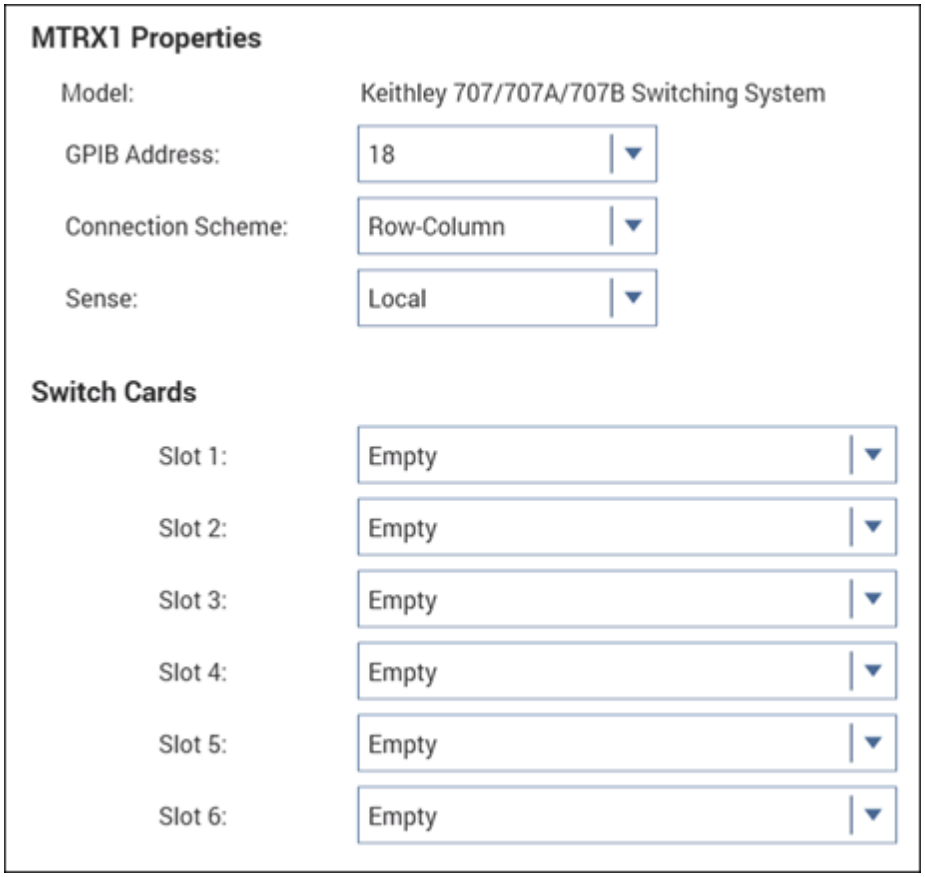
The number of pins defined in the probe station properties determines the pins that are available to assign to a switch matrix card column. Make sure the number of pins assigned is appropriate for your system.

Step 3. Add switching system mainframe

To add a switching system mainframe:

1. Select **Add External Instrument**.
2. Select the **Keithley 707/707A/707B Switching Matrix** or **Keithley 708/708A/708B Switching Matrix**.
3. Select **OK**.
4. In the Configuration Navigator, select the switching matrix. The properties are displayed. The following figure shows the properties for the 707/707A/707B. If the 708/708A/708B mainframe is selected, there is only one switch card slot.

Figure 639: KCon MTRX1 Properties tab



Step 4. Set GPIB address

The GPIB address setting in the properties must match the actual GPIB address of the mainframe. The address for the switch system mainframe is briefly displayed during its power-on sequence.

To set the GPIB address:

1. Select the GPIB Address from the list. Addresses that are in use are displayed with asterisks (*) next to them. The range of addresses is 0 to 30 (GPIB address 31 is reserved as the 4200A-SCS controller address). 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.
2. Select **Save** to save the change.

NOTE

You can programmatically read the GPIB address and other instrument properties from the system configuration using the LPT library `getinstattr` function. Proper use of `getinstattr` allows you to develop user libraries that are independent of the configuration. For more information, refer to Keithley User Library Tool (KULT) (on page 8-1).

Step 5. Configure the instrument connection scheme

To configure the instrument connection scheme:

1. Select the **Connection Scheme** from the list:
   - If you are connecting the instrumentation to matrix rows and the device under test (DUT) to matrix columns, select **Row-Column**.
   - If all connections (instrumentation and DUT) are made to matrix columns only, select **Instrument Card**.
2. Select **Local Sense** or **Remote Sense**:
   - For 2-wire connections to the DUT, select **Local Sense**.
   - For 4-wire connections to the DUT, select **Remote Sense**.

Step 6. Assign switch cards to mainframe slots

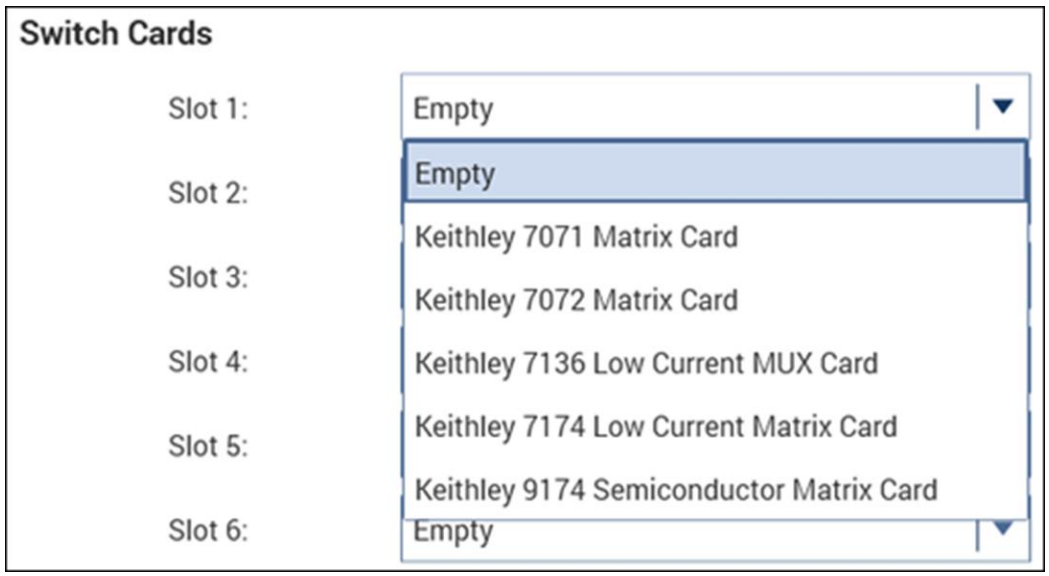
To assign switch cards to mainframe slots:

1. For each slot that contains a matrix card, select the model number of the matrix card.
2. For each slot that is empty, select **Empty**.

NOTE

You cannot mix matrix card models. For example, if you set slot 1 to Keithley 7174 Low Current Matrix Card, all other slots can only be set to the 7174 or Empty. To select a different model, you must set all slots to Empty and then make the new selection.

Figure 640: Assign switch cards to slots



Step 7. Set matrix card properties

The matrix card properties set the connections:

- Between the measurement instrumentation and the matrix card
- Between the matrix card and the test system (prober or test fixture)

NOTE

The number of pins defined in the properties for a probe station or test fixture determines the pins that are available to assign to a switch matrix card column. Make sure the number of pins assigned is appropriate for your system. Refer to Add a test fixture (on page A-32) or Add a probe station (on page A-33) for additional information.

To set matrix card properties:

1. In the Configuration Navigator, expand the switching matrix.
2. Select the card. The properties are displayed. Each row and column has a list to set the card properties. If the row-column connection scheme is selected, instruments are assigned to the rows and 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.

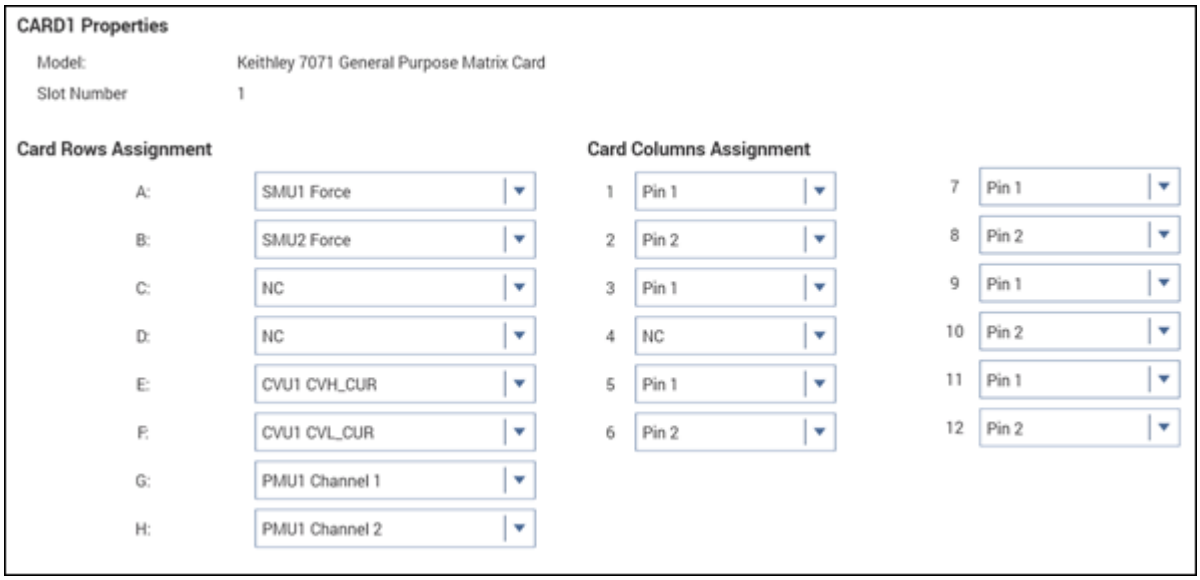
   The following figure shows the KI 7XXX matrix-card **Properties** tab configuration that is required to support the physical connection configuration that is shown in Row-column or instrument card settings (on page A-16).
3. Select from the lists to connect the rows and columns to instrument terminals and prober or test fixture pins. Note that card properties must match the actual physical connections to the matrix card.

   In the figure below, the lists labeled **A** to **H** correspond to the eight rows of all Keithley Instruments Series 700 Switching System-compatible matrix cards. The lists labeled **1** to **12** correspond to the 12 columns of Series 700 Switching System-compatible matrix cards.
4. Select **Validate Configuration**.

NOTE

Prober or test-fixture pins are always connected to matrix-card columns.

Figure 641: KI 7174 Matrix Card Properties tab



Step 8. Save configuration

To save the KCon configuration:

1. Select **Save**.

Step 9. Close KCon and open Clarius

To close KCon and open Clarius:

1. To close KCon, select the close button in the upper right.
2. On the Windows desktop, select the Clarius icon.

Switch matrix control example

This example demonstrates how the `connectpins` action controls a switch matrix. You modify the `connectpins` action to connect SMU2 to a DUT, as shown in the Switch matrix control (on page A-21) figure. 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 Switch matrix control (on page A-21).

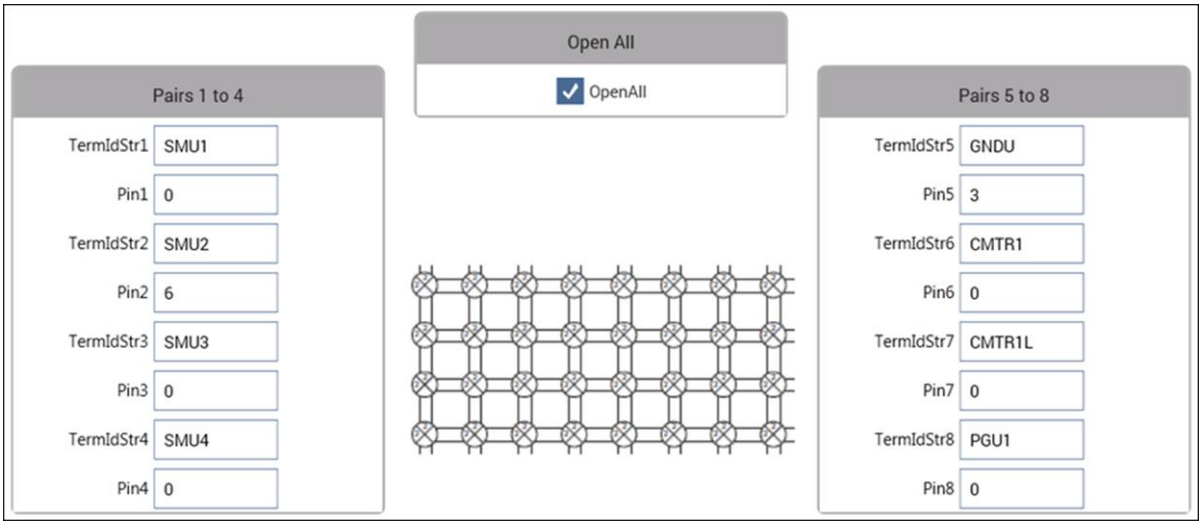
The `connectpins` action is based on the `ConnectPins` user module. Detail on `ConnectPins` is provided in Matrixulib user library (on page 6-390).

Set up and run a switch matrix in Clarius

To set up and run the `connectpins` action:

1. Choose **Select**.
2. Select **Actions**.
3. Search for **connectpins**.
4. Select the `connectpins` action.
5. Select **Add**.
6. In the project tree, select the **connectpins** action.
7. Select **Configure**. The parameter settings are displayed, as shown below.
8. Set Pin2 to **6**. This connects SMU2 to point 6.
9. Select **OpenAll** to open all matrix card switches.
10. Set Pin5 to **3**. This connects GNDU to pin 5.
11. Leave all other pin settings at 0 to indicate that no connection will be made.

Figure 642: connectpins settings



12. Select **Run**. The 4200A-SCS connects to the DUT as shown in Switch matrix control (on page A-21).

Matrixulib user library

The `Matrixulib` connects instrument terminals to output pins using a Keithley Instruments Series 700 Switching System. It is for use with switching systems that are configured as a general purpose, low current, or ultra-low current matrix.

Matrixulib user module

User module	Description
`ConnectPins`	Allows you to control your switch matrix.

ConnectPins user module

The `ConnectPins` module allows you to control your switch matrix.

Usage

```
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);
```

status	Returned values; see **Details**
OpenAll	Controls if the switch matrix is cleared before making any new connections: ■ Clear all previous connections: `1` ■ Leave previous connections intact: `0`
TermIdStr1 *TermIdStr2* *TermIdStr3* *TermIdStr4* *TermIdStr5* *TermIdStr6* *TermIdStr7* *TermIdStr8*	Terminal identification string; refers to an instrument as defined by `TermIdStr8` in KCon; valid inputs (configuration dependent) are: `SMU`*n*, `CMTR`*n*, `CMTR`*n*`L`, `PGU`*n*, `GPI`*n*, `GPI`*n*`L`, `GNDU` (where *n* is a number from 1 through 8)
Pin1 *Pin2* *Pin3* *Pin4* *Pin5* *Pin6* *Pin7* *Pin8*	The DUT pin number (configuration dependent) to which the instrument will be attached; if a number less than 1 is specified, no connection is made; valid inputs: –1 to 72

Details

This user module allows you to control a switch matrix. The default input parameters are shown in the following figure. 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.

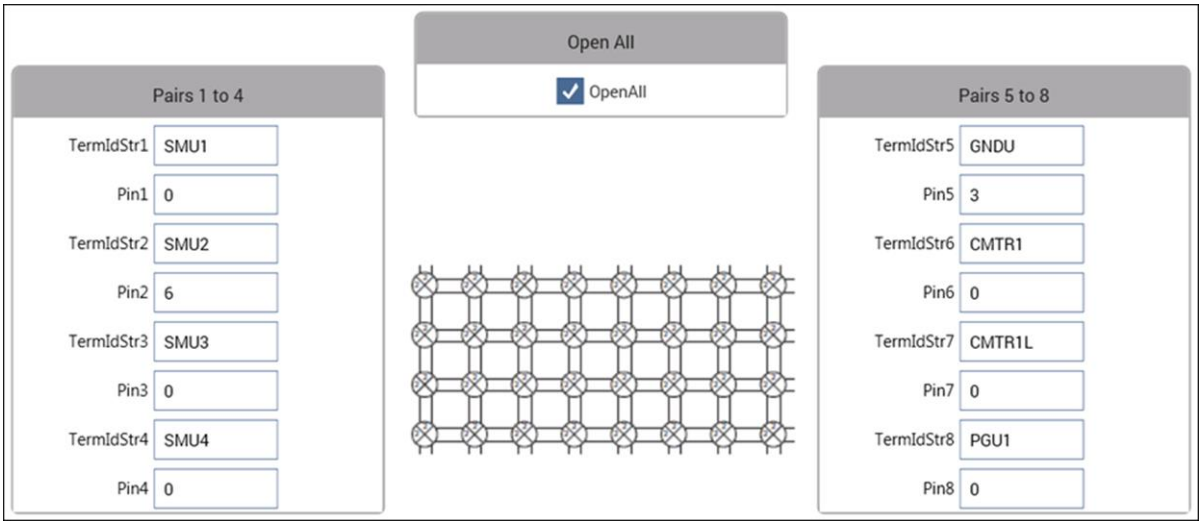
Terminal ID: Terminal identification for the most common components used in the system configuration are as follows:

- SMU1 to SMU4: These are the signal HI terminals for the four SMUs.
- GNDU: This is common terminal for the Ground Unit of the 4200A-SCS.
- CMTR1: This is used for a C-V Analyzer. For the 590, it is the OUTPUT terminal. For the Keysight Model 4980A, it is the HCUR terminal.
- CMTR1L: This is also used for a C-V Analyzer. For the 590, it is the INPUT terminal. For the Keysight Model 4980A, it is the LCUR terminal.
- PGU1: This is output HI for the Keysight Model 8110A Pulse Generator.

NOTE

A test example demonstrates how this user module controls the switch matrix (see Switch matrix control example (on page A-39)).

Figure 643: connectpins settings



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.

Returned values are placed in the Analyze sheet and can be:

- `0 OK`.
- `-10000 (INVAL_INST_ID)`: The specified instrument ID does not exist. 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)
```

Also see

None

Appendix B

Using a Model 590 C-V Analyzer

In this appendix:

Introduction

NOTE

For details on 590 operation, refer to the *Model 590 C-V Analyzer Instruction Manual*.

C-V measurement basics

The Keithley Instruments Model 590 C-V Analyzer measures capacitance versus voltage (C-V) and capacitance versus time (C-t) of semiconductor devices. Typically, C-V measurements are made on capacitor-like devices, such as a metal-oxide-silicon capacitor (MOS capacitor).

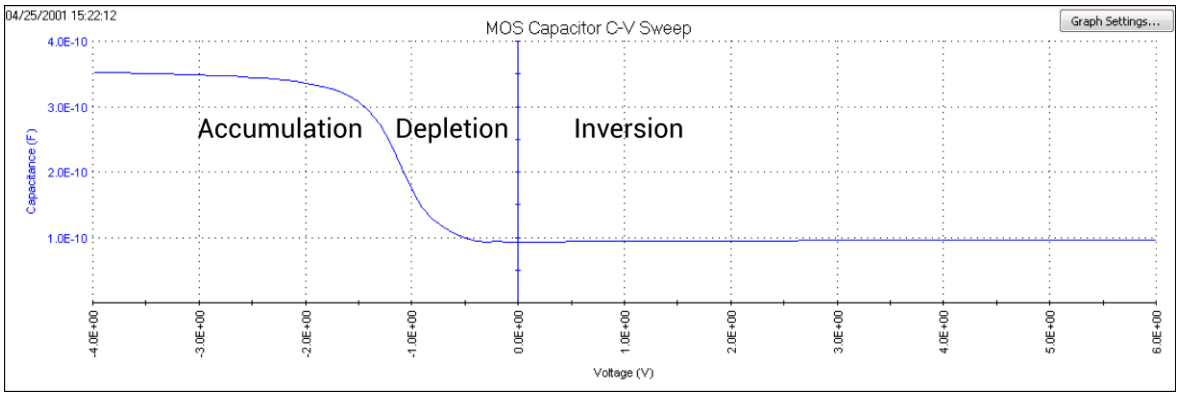
The measurements of MOS capacitors study:

- The integrity of the gate oxide and semiconductor doping profile
- The lifetime of semiconductor material
- The interface quality between the gate oxide and silicon
- Other dielectric materials used in an integrated circuit

The voltage sweeping capability of the 590 makes it easy to make a series of capacitance measurements that span the three regions of a C-V curve: the accumulation region, depletion region, and inversion region.

The following figure shows the three regions of a typical C-V curve for a MOS capacitor.

Figure 644: Typical C-V curve for a MOS capacitor



Capacitance measurement tests

The 4200A-SCS provides the following tests to perform C-V tests using the 590:

- **590 C-V Sweep (590-cvsweep):** Makes a capacitance measurement at each step of a user-configured linear voltage sweep.
- **590 C-V Pulse Sweep (590-cvpulsesweep):** Makes a capacitance measurement at each step of a user-configured pulsed voltage sweep.
- **590 C-t Sweep (590-ctsweep):** Makes a specified number of capacitance measurements at a specified time interval. Voltage is held constant for these capacitance measurements.
- **590 Capacitance Measurements (590-cmeas):** Makes capacitance and conductance measurements at a fixed bias voltage.

NOTE

There are also user modules available for the 590. Refer to KI590ulib user library (on page 6-388).

Connections

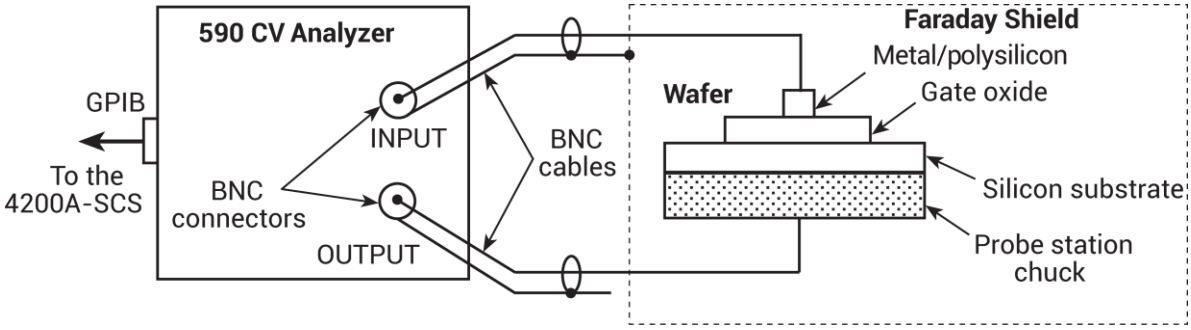
For additional information about 590 connections, see the Model 590 C-V Analyzer Instruction Manual.

Signal connections

Basic signal connections for the 590 are shown in the following figure.

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 capacitor.

Figure 645: Basic 590 connections to the DUT



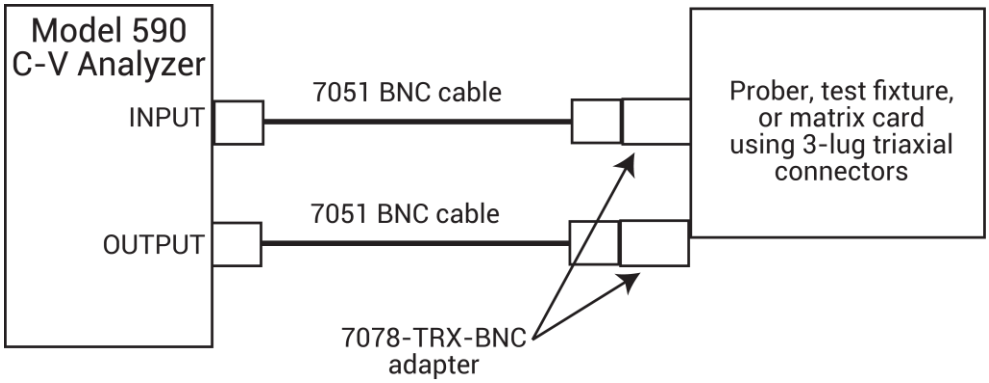
Triaxial connectors

Adapters are required to connect the 590 to equipment (for example, a probe station, test fixture, or matrix card) that uses triaxial connectors. The 7078-TRX-BNC is a 3-lug triaxial to BNC adapter. As shown in the following figure, connect the adapters to the 3-slot triaxial connectors and then use a 7051 BNC cable to make the connections to the 590.

NOTE

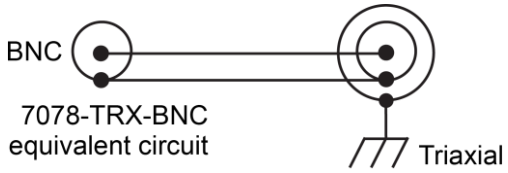
See Using Switch Matrices (on page A-1) for details on using a switch matrix with the 590 C-V Analyzer.

Figure 646: Connecting the 590 to equipment with triaxial connectors



The figure below shows the equivalent circuit for the adapter.

Figure 647: 7078-TRX-BNC equivalent circuit



GPIB connections

The 4200A-SCS controls the 590 through the General Purpose Interface Bus (GPIB). Use the 7007-1 or 7007-2 GPIB cable to connect the GPIB of the 590 to the GPIB of the 4200A-SCS.

Cable compensation

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, you should do cable compensation before measuring the capacitance of DUT. In general, you do cable compensation by connecting precisely known capacitance sources in place of the DUT and then measuring them. The 590 then uses these measured values to make corrections when measuring the DUT.

During cable compensation:

1. The 590 calculates the compensation parameters based on the comparison between the given and measured values.
2. The 590 makes a probe-up offset measurement and suppresses any remaining offset capacitance. This step is done every time a new measurement is made.

Typically, cable compensation is done for all measurement ranges (2 pF, 20 pF, 200 pF, and 2 nF) of the 590. Once cable compensation is done, it does not have to be done again unless the connections to the DUT are changed or power is cycled.

For each measurement range of the 590, you must use a low-capacitance source and a high-capacitance source. The following table lists the Keithley Instruments Model 5909 capacitance sources that can be used for each 590 range.

5909 capacitance sources

590 range	Low capacitance source	High capacitance source
2 pF	0.5 pF	1.5 pF
20 pF	4.7 pF	18 pF
200 pF	47 pF	180 pF
2 nF	470 pF	1.8 nF

Cable compensation user modules

The 4200A-SCS `KI590ulib` user library includes the following user modules for 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 the Analyze spreadsheet:** When this test is executed, the capacitance values saved by `SaveCableCompCaps82` are placed into the Analyze spreadsheet.
- **CableCompensate590: Performs cable compensation:** The user specifies the ranges and test frequencies for cable compensation. When this test is executed, on-screen prompts guide you through the cable compensation process.
- **LoadCableCorrectionConstants:** This function reads the cable compensation parameters for the range and frequency specified from the cable compensation file and sends these parameters to the 590.

NOTE

Details on all user modules for the 590 are provided in KI590ulib user library (on page 6-388).

Using KCon to add 590 C-V Analyzer to system

To use the 4200A-SCS to control an external instrument, that instrument must be added to the system configuration. The 590 C-V Analyzer is added to the test system using the Keithley Configuration Utility (KCon).

Refer to Use KCon to add equipment to the 4200A-SCS (on page 7-7) for instruction.

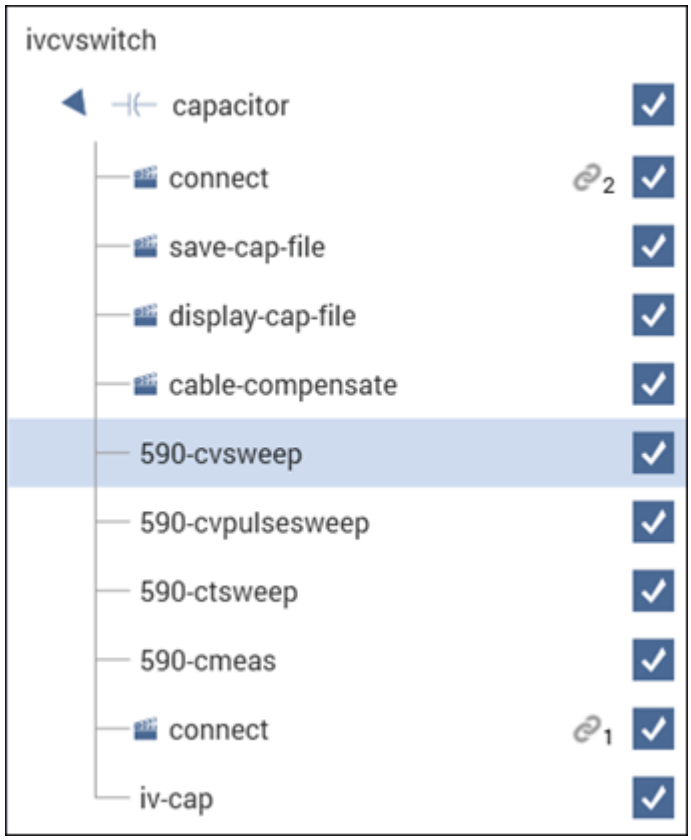
For additional detail on KCon, refer to Keithley Configuration Utility (KCon) (on page 7-1).

Model 590 test examples

The test examples for the Model 590 C-V Analyzer are controlled by user test modules (UTMs) in the `ivcvswitch` project. The following figure shows the tree for the project.

A switch matrix is not used for these examples.

Figure 648: Project tree when ivcvswitch project is selected



Cable compensation example

This example assumes that the 590 is connected directly to the DUT. The DUT could be a device installed in a test fixture or a MOS capacitor on a wafer.

NOTE

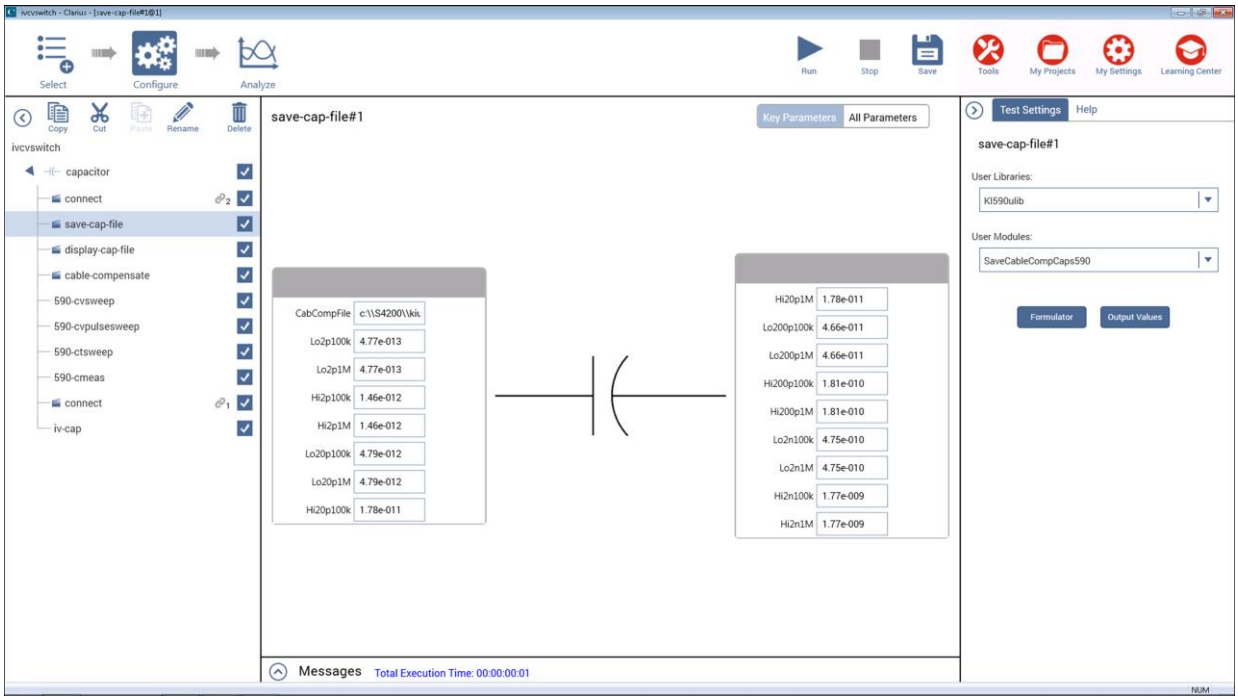
The user modules for cable compensation must share the same file for capacitance source values. Therefore, the same file directory path must be used in all three user modules. For this example, use the default file directory path (see line 1 of the parameter list in the following figures).

Enter and save capacitance source values (save-cap-file)

To enter and save the capacitance source values:

1. Select **Configure**.
2. In the project tree, select `save-cap-file`. The default parameters for the user module are displayed, as shown in the following figure.

Figure 649: save-cap-file action and SaveCableCompCaps590 user module



3. Enter the capacitance source calibration value for each range and frequency. If you are using the 5909, 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:

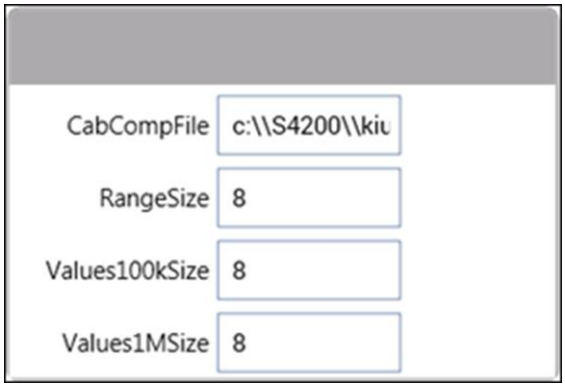
   - Lo2p100k: Enter `0.47773e-12`
   - Lo2p1M: Enter `0.47786e-12`
4. In the project tree, select the action.
5. Click **Run** to execute the action.

Place capacitance source values in a spreadsheet (display-cap-file)

To place the source values in the Analyze sheet:

1. Select **display-cap-file**. The default parameters are displayed, as shown in the following figure.

Figure 650: DisplayCableCompCaps590 default parameters



2. Ensure that the `CabCompFile` field has the same file directory path that is used in `save-cap-file` (Enter and save capacitance source values (save-cap-file) (on page B-8)).
3. Set the array size in the `RangeSize`, `Values100kSize`, and `Values1MSize` fields.
4. Click **Run**. The calibration source values are placed into the Analyze sheet for this action.
5. Select **Analyze** to view the spreadsheet for `display-cap-file`. An example spreadsheet is shown in the following figure.

Figure 651: Display-cap-file spreadsheet showing capacitor source values

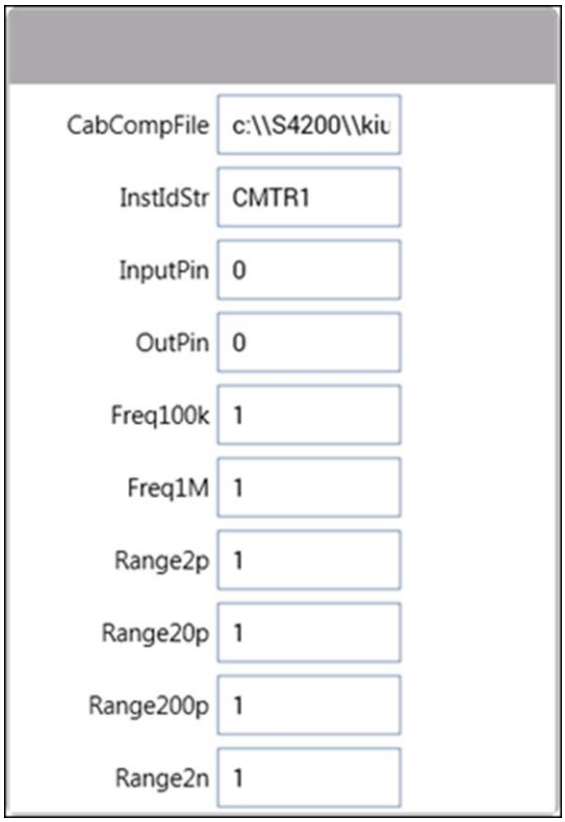
	A	B	C	D
1	DisplayCable	Range	Values100k	Values1M
2	0	2.00000E-12	4.77700E-13	4.77700E-13
3		2.00000E-12	1.46100E-12	1.46100E-12
4		2.00000E-11	4.79600E-12	4.79600E-12
5		2.00000E-11	1.78300E-11	1.78300E-11
6		2.00000E-10	4.66800E-11	4.66800E-11
7		2.00000E-10	1.81100E-10	1.81100E-10
8		2.00000E-09	4.75500E-10	4.75900E-10
9		2.00000E-09	1.77100E-09	1.77600E-09
10				

Perform cable compensation (CableCompensate)

To do cable compensation:

1. Select **Configure**.
2. In the project tree, select **cable-compensate** to open the action. The figure below shows the default parameters for the action.

Figure 652: CableCompensate590 default parameters

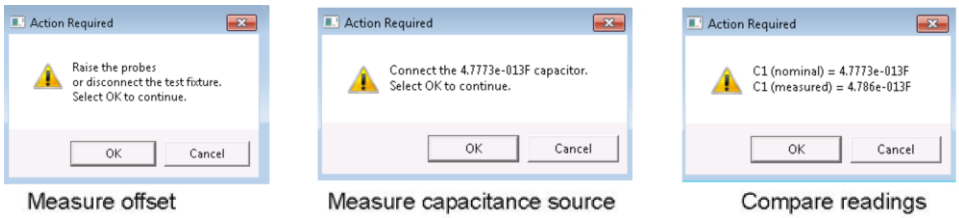


3. Ensure that `CabCompFile` has the same file directory path that is used in `save-cap-file` (Enter and save capacitance source values (SaveCableCompCaps590) (on page B-8)).
4. Enable or disable cable compensation. The `FreqN` and `RangeN` parameters either disable (`0`) or enable (`1`) cable compensation for the frequencies and ranges. The figure above shows cable compensation enabled for all ranges and test frequencies.
5. Click **Run** to execute the action. A series of dialog boxes guides you through the cable compensation process. The basic dialog boxes are shown in the following figure:
   - Raise the probes: This dialog box indicates that an offset (open circuit) measurement is required. Open the circuit as close to the DUT as possible.
   - Connect the capacitor: The value in the dialog box corresponds to a calibration value entered by the user in Enter and save capacitance source values (on page B-8). Connect the capacitance source as close to the DUT as possible.
   - Compare readings: 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 **Run**.

Figure 653: Cable compensation dialog boxes



C-V sweep example

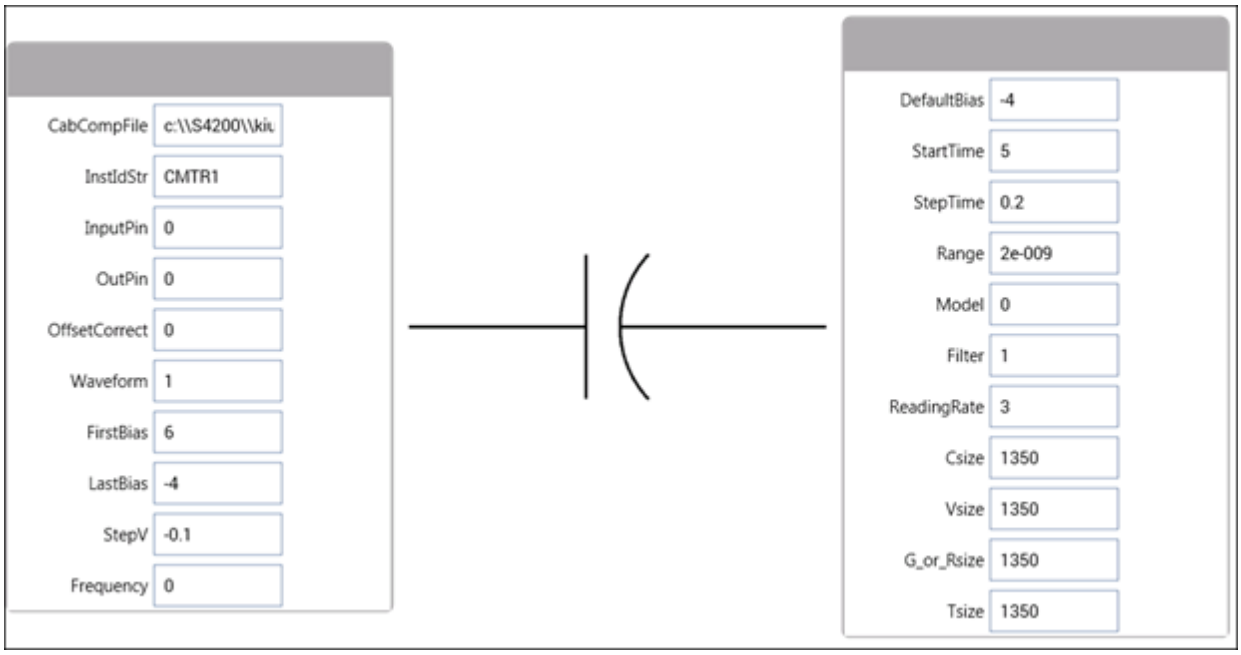
This example demonstrates how to control a Keithley Instruments 590 C-V Analyzer to acquire capacitance verses voltage (CV) data from a MOS capacitor. In this example, the C-V Analyzer applies a linear staircase voltage sweep to a capacitor. A capacitance measurement is made on every voltage step of the sweep.

This example assumes that the 590 is connected directly to the DUT. The DUT could be a device installed in a test fixture or a MOS capacitor on a wafer.

To do a C-V sweep:

1. Choose **Select**.
2. Select **Tests**.
3. Search for **590**.
4. Select **Add** for the 590 C-V Sweep (`590-cvsweep`) test.
5. In the project tree, select **590-cvsweep**.
6. Select **Configure**. The parameters are displayed.

Figure 654: CvSweep590 user module (590-cvsweep UTM)

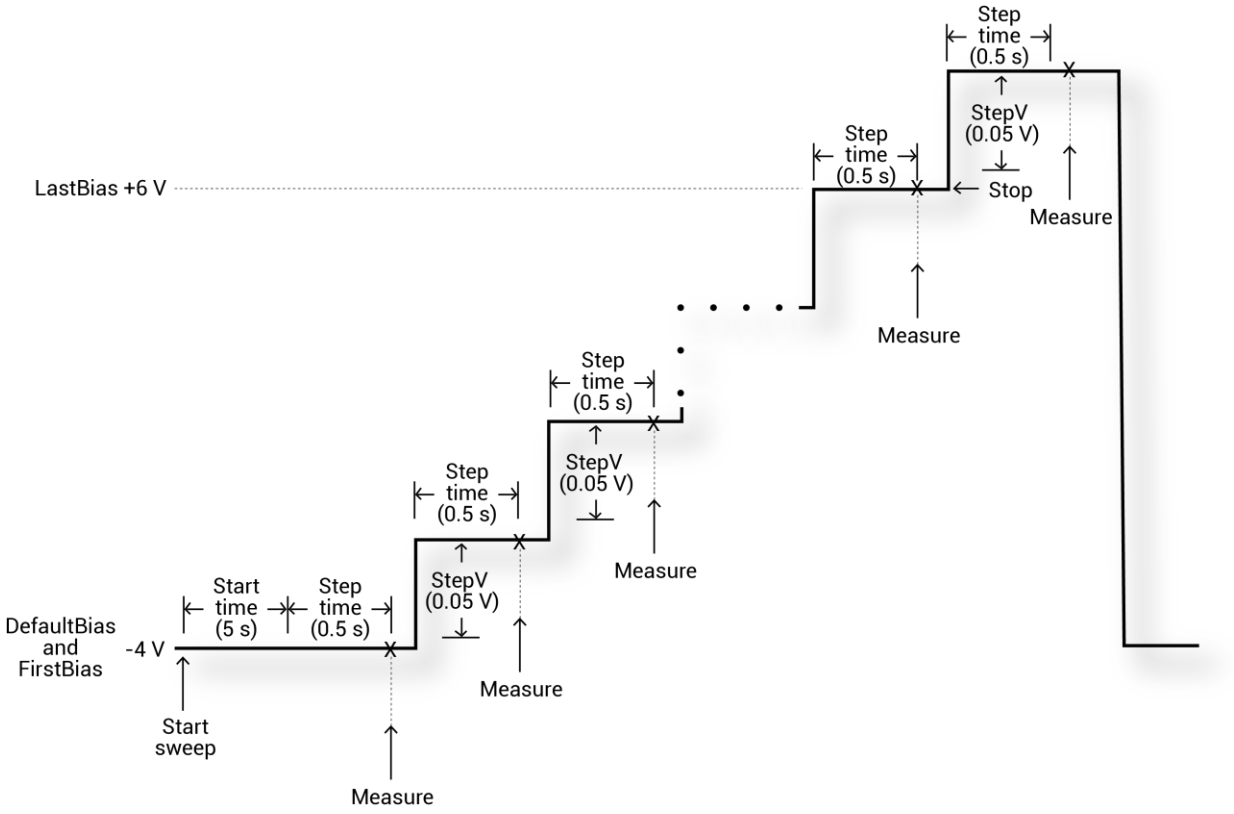


7. Change the test parameters as needed.
8. Execute the test by clicking **Run**.

This test uses the `CvSweep590` user module. For details on this test description, see CvSweep590 user module (on page B-32).

If you use the default parameters, this test causes the 590 to do a –4 V to +6 V staircase sweep using 50 mV steps. The figure below shows the default sweep.

Figure 655: C-V linear staircase sweep



KI590ulib user library

The user modules in the `KI590ulib` user library are used to control the 590 C-V Analyzer. These user modules are summarized in the table below. Also listed in the table are names of the user test modules (UTMs) and actions in Clarius that use the user modules.

KI590ulib user modules

User module	UTM or action name	Description
CableCompensate590 (on page B-15)	`cable-compensate` in the `ivcvswitch` project	Performs cable compensation using known capacitance source values.
Cmeas590 (on page B-18)	`590-cmeas`	Makes a single capacitance measurement.
CtSweep590 (on page B-21)	`590-ctsweep`	Makes a capacitance versus time measurement.
CvPulseSweep590 (on page B-26)	`590-cvpulsesweep`	Makes capacitance versus voltage measurements using a pulse sweep.
CvSweep590 (on page B-32)	`590-cvsweep`	Makes capacitance versus voltage measurements using a staircase sweep.
DisplayCableCompCaps590 (on page B-36)	`display-cap-file` in the `ivcvswitch` project	Places capacitance source values in a spreadsheet.
LoadCableCorrectionConstants (on page B-39)	n/a	Reads the cable compensation parameters for the range and frequency specified from the cable compensation file and sends these parameters to the 590.
SaveCableCompCaps590 (on page B-40)	`save-cap-file` in the `ivcvswitch` project	Saves entered capacitance source values in a file.

CableCompensate590 user module

The `CableCompensate590` routine performs the 590 cable compensation procedure using the capacitor values that are stored in the specified cable compensation file. The resultant compensation values generated by the compensation process are stored in the same file.

Usage

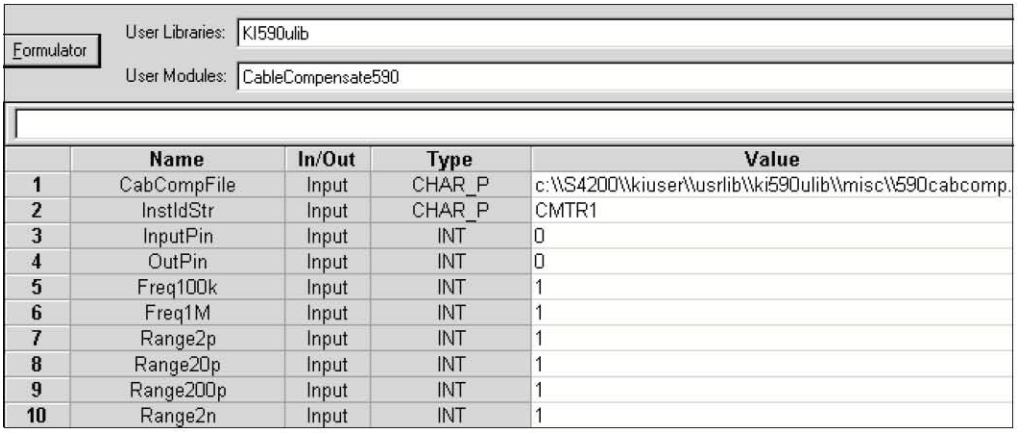
```
status = CableCompensate590(char *CabCompFile, char *InstIdStr, int InputPin, int
   OutPin, int Freq100k, int Freq1M, int Range2p, int Range20p, int Range200p, int
   range2n);
```

status	Returned values; see **Details**
CabCompFile	The complete name and path for the cable compensation file; see **Details**
InstIdStr	The CMTR instrument ID; `CMTR1`, `CMTR2`, `CMTR3`, or `CMTR4`, depending on your system configuration
InputPin	The DUT pin to which the 590 input terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connection is made; see **Details**
OutPin	The DUT pin to which the 590 output terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connection is made; see **Details**
Freq100k	Determines if compensation is done for the 100 kHz frequency: ■ Do not compensate: `0` ■ Compensate: `1`
Freq1M	Determines if compensation is done for the 1 MHz frequency: ■ Do not compensate: `0` ■ Compensate: `1`
Range2p	Determines if compensation is done for the 2 pF range: ■ Do not compensate: `0` ■ Compensate: `1`
Range20p	Determines if compensation is done for the 20 pF range: ■ Do not compensate: `0` ■ Compensate: `1`
Range200p	Determines if compensation is done for the 200 pF range: ■ Do not compensate: `0` ■ Compensate: `1`
range2n	Determines if compensation is done for the 2 nF range: ■ Do not compensate: `0` ■ Compensate: `1`

Details

This user module performs cable compensation for the selected ranges and test frequencies of the 590. The figure below shows the default input parameters for a UTM that uses the `CableCompensate590` user module.

Figure 656: CableCompensate590 default parameters



	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

If the default parameters are used, cable compensation is done for the 2 pF, 20 pF, 200 pF, and 2 nF ranges and for 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 to do cable compensation (see Model 590 test examples (on page B-7)).

If the file defined for *`CabCompFile`* does not exist, or there is no path specified (null string), the default compensation parameters are used. When entering the path, use two backslash (`\\`) characters to separate each directory. For example, if your cable file is in `C:\calfiles\590cal.dat`, you enter the following:

```
C:\\calfiles\\590cal.dat
```

NOTE

If a switch matrix to route signals is being controlled by a connection action (for example, `connect`), there is no need to connect *`InputPin`* and *`OutPin`*. Set these parameters to `0`.

Returned values are placed in the Analyze sheet.

- `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 configuration.

Procedure

For each range and test frequency specified by the input parameters:

1. You are prompted to open the circuit so that an offset capacitance measurement can be made.
2. When the offset capacitance measurement is completed, you are prompted to connect the low value capacitor for the selected range. The system does the low value capacitor compensation.
3. You are prompted to connect the high value capacitor for the selected range. The system does the high value capacitor compensation.
4. You are prompted to reconnect the low capacitor.
5. The nominal and measured values are displayed in a dialog box. You can:
   - Select **Cancel** to abort the procedure if the results are not correct. No changes to the cable compensation file occur.
   - Select **Save** to save the cable compensation values.

Also see

None

Cmeas590 user module

The `Cmeas590` routine measures capacitance and conductance using the Keithley Instruments 590 C-V Analyzer. You can make an offset correction measurement and use cable compensation.

Usage

```
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);
```

status	Returned values; see **Details**
CabCompFile	The complete name and path for the cable compensation file; see **Details**
InstIdStr	The CMTR instrument ID; `CMTR1`, `CMTR2`, `CMTR3`, or `CMTR4`, depending on your system configuration
InputPin	The DUT pin to which the 590 input terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connection is made; see **Details**
OutPin	The DUT pin to which the 590 output terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connection is made; see **Details**
OffsetCorrect	Determines if an offset correction measurement should be made: ■ Do not make offset measurement: `0` ■ Make offset measurement: `1`
Frequency	The measurement frequency to use: ■ 100 kHz: `0` ■ 1 MHz: `1`
DefaultBias	The DC bias to use for the measurement: –20 to +20 V
StartTime	The amount of time to delay after applying the *DefaultBias* voltage step (0.001 s to 65 s)
Range	The measurement range to use: 1 to 4; see **Details**
Model	Measurement model: ■ Series model: `0` ■ Parallel model: `1`
Filter	Enables or disables the analog filter: ■ Disable: `0` ■ Enable (can minimize the amount noise in the readings, but increases measurement time): `1`
ReadingRate	The reading rate used to acquire the measurements (0 to 4); see **Details**
C	Output: The measured capacitance
V	Output: The bias voltage used
G_or_R	Output: ■ Parallel measurement model: *G_or_R* is the measured conductance ■ Series measurement model: *G_or_R* is the measured resistance

Details

This user module is used to make a single, fixed-bias capacitance and conductance measurement. The default parameters for the `590-cmeas` UTM, which uses the `Cmeas590` user module, are shown in the figure below.

Figure 657: Cmeas590 (cmeas UTM parameters)

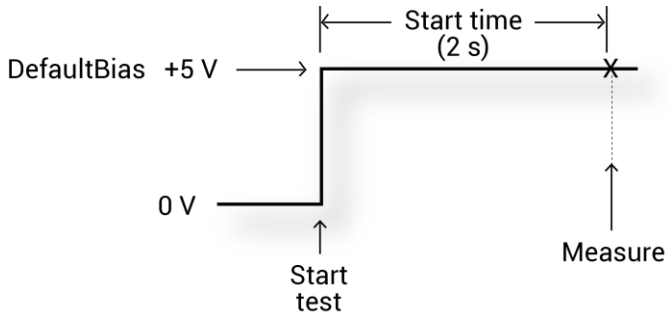
Formulator | User Libraries: KI590ulib

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	

In general, the 590 is set to source 5 V for 2 seconds, then make a measurement, as shown in the following figure.

Figure 658: Cmeas590 measurement



If the file defined for *`CabCompFile`* does not exist, or there is no path specified (null string), the default compensation parameters are used. When entering the path, use two backslash (`\\`) characters to separate each directory. For example, if your cable file is in `C:\calfiles\590cal.dat`, you enter the following:

```
C:\\calfiles\\590cal.dat
```

NOTE

If a switch matrix to route signals is being controlled by a connection action (for example, `connect`), there is no need to connect *`InputPin`* and *`OutPin`*. Set these parameters to `0`.

For *Range*, the measurement range values are shown in the following table.

Range	100 kHz	1 MHz
1	2 pF / 2 µs	20 pF / 200 µs
2	20 pF / 20 µs	20 pF / 200 µs
3	200 pF / 200 µs	200 pF / 2 ms
4	2 nF / 2 ms	2 nF / 20 ms

The reading rates and resolutions for the *ReadingRate* parameter are described in the following table.

Reading rate	Nominal reading rate (per second)	Display readings	Resolution (digits)
0	1000	C	3.5
1	75	C,G,V	3.5
2	18	C,G,V	4.5
3	10	C,G,V	4.5
4	1	C,G,V	4.5

Returned values are placed in the Analyze sheet:

- `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 configuration.
- `-10023 (KI590_MEAS_ERROR)`: A measurement error occurred.
- `-10090 (GPIB_ERROR_OCCURRED)`: A GPIB communications error occurred.
- `-10091 (GPIB_TIMEOUT)`: A timeout occurred during communications.
- `-10102 (ERROR_PARSING)`: There was an error parsing the response from the 590.
- `-10104 (USER_CANCEL)`: The user canceled the correction procedure.

Procedure

1. You are prompted to open the circuit so that an offset capacitance measurement can be made, if needed.
2. If a cable compensation file is specified, the compensation information in that file for the selected range and frequency is loaded. If not, instrument default compensation is used.
3. The capacitance and conductance are measured.

Also see

None

CtSweep590 user module

The `CtSweep590` routine performs a capacitance versus time (C_t) sweep using the Keithley Instruments 590 C-V Analyzer. You can make an offset correction measurement and use cable compensation.

Usage

```
status = 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);
```

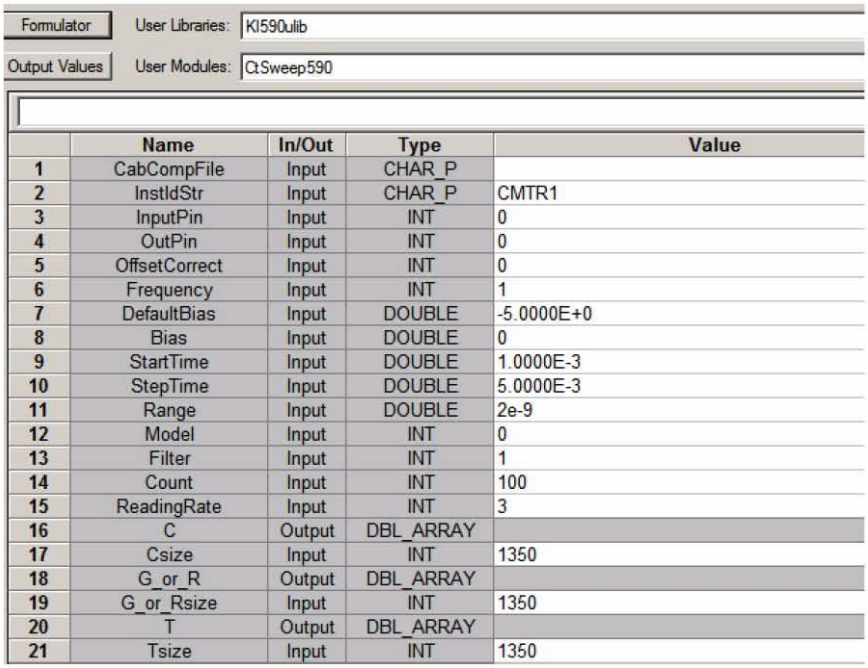
status	Returned values; see **Details**
CabCompFile	The complete name and path for the cable compensation file; see **Details**
InstIdStr	The CMTR instrument ID; `CMTR1`, `CMTR2`, `CMTR3`, or `CMTR4`, depending on your system configuration
InputPin	The DUT pin to which the 590 input terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connection is made; see **Details**
OutPin	The DUT pin to which the 590 output terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connection is made; see **Details**
OffsetCorrect	Determines if an offset correction measurement should be made: ■ Do not make offset measurement: `0` ■ Make offset measurement: `1`
Frequency	The measurement frequency to use: ■ 100 kHz: `0` ■ 1 MHz: `1`
DefaultBias	The DC bias to use for the measurement: –20 V to +20 V
Bias	The DC bias that is applied during a sweep: –20 V to +20 V
StartTime	The time that occurs on the first bias step, from the point the instrument is first triggered until the first step time: 0.001 s to 65 s
StepTime	The time period after a transition to a new bias step and before the instrument begins a measurement: 0.001 s to 65 s
Range	The measurement range to use in F: 2E-12, 20E-12, 200E-12, or 2E-9; see **Details**
Model	Measurement model: ■ Series model: `0` ■ Parallel model: `1`
Filter	Enable or disable the analog filter, which can minimize the amount of noise that appears in the readings; however, it increases the measurement time: ■ Disable the filter: `0` ■ Enable the filter: `1`
Count	The number of readings per sweep: 1 to 1350
ReadingRate	Selects the reading rate used to acquire the measurements: 0 to 4; see **Details**
C	Output: The measured array of capacitance values
Csize	Set to a value that at minimum is equal to the *Count*; if in doubt, set to 1350

G_or_R	Output: ■ When the parallel measurement model (1) is selected, *G_or_R* is the measured conductance ■ When the series measurement model (0) is selected, this is the measured resistance
G_or_Rsize	Set to a value that at minimum is equal to the *Count*; if in doubt, set to 1350
T	The array of time stamps for each measurement step
Tsize	Set to a value that at minimum is equal to the *Count*; if in doubt, set to 1350

Details

This user module performs a capacitance versus time (Ct) sweep. The figure below shows the default parameters for the `ctsweep-590` UTM, which uses the `CtSweep590` user module.

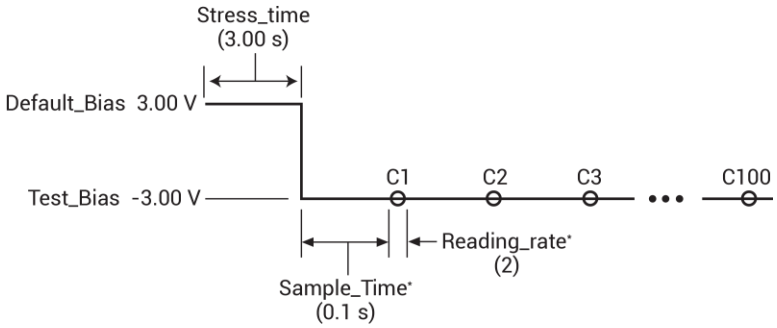
Figure 659: Starting KULT



	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

In this example, the 590 is set to source –5 V and performs 100 capacitance measurements using a 5 ms time interval, as shown in the following figure.

Figure 660: C-t measurements



*The time interval between reading samples is the sum of the set Sample Time and the Reading Rate time. The Reading Rate time for Reading_rate2 is 1/18 s (0.0555 s). Therefore:
Time interval = Sample Time + Reading Rate time
= 0.100 + 0.056
= 0.156 s

If the file defined for *CabCompFile* does not exist, or there is no path specified (null string), the default compensation parameters are used. When entering the path, use two backslash (`\\`) characters to separate each directory. For example, if your cable file is in `C:\calfiles\590cal.dat`, you enter the following:

```
C:\\calfiles\\590cal.dat
```

NOTE

If a switch matrix to route signals is being controlled by a connection action (for example, `connect`), there is no need to connect *InputPin* and *OutPin*. Set these parameters to `0`.

For *Range*, the measurement range values are shown in the following table.

Range	100 kHz	1 MHz
1	2 pF / 2 µs	20 pF / 200 µs
2	20 pF / 20 µs	20 pF / 200 µs
3	200 pF / 200 µs	200 pF / 2 ms
4	2 nF / 2 ms	2 nF / 20 ms

The reading rates and resolutions for the *ReadingRate* parameter are described in the following table.

Reading rate	Nominal reading rate (per second)	Display readings	Resolution (digits)
0	1000	C	3.5
1	75	C,G,V	3.5
2	18	C,G,V	4.5
3	10	C,G,V	4.5
4	1	C,G,V	4.5

Returned values are placed in the Analyze sheet.

- 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 configuration.
- -10023 (KI590_MEAS_ERROR): A measurement error occurred.
- -10090 (GPIB_ERROR_OCCURRED): A GPIB communications error occurred.
- -10091 (GPIB_TIMEOUT): A timeout 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 response from the 590.
- -10104 (USER_CANCEL): The user canceled the correction procedure.

Procedure

1. You are 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 is loaded. If not, instrument default compensation is used.
3. A C-t sweep is performed.

Also see

None

CvPulseSweep590 user module

The `CvPulseSweep590` routine performs a capacitance versus voltage (C-V) sweep using the pulse waveform capability of the Keithley Instruments 590 C-V Analyzer. You can make an offset correction measurement and use cable compensation.

Usage

```
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 StopTime, 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);
```

status	Returned values; see **Details**
CabCompFile	The complete name and path for the cable compensation file; see **Details**
InstIdStr	The CMTR instrument ID; `CMTR1`, `CMTR2`, `CMTR3`, or `CMTR4`, depending on your system configuration
InputPin	The DUT pin to which the 590 input terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connection is made; see **Details**
OutPin	The DUT pin to which the 590 output terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connection is made; see **Details**
OffsetCorrect	Determines if an offset correction measurement should be made: ■ Do not make offset measurement: `0` ■ Make offset measurement: `1`
FirstBias	The starting bias for the sweep: –20 V to +20 V; see **Details**
LastBias	The last voltage used in the sweep: –20 V to +20 V; see **Details**
StepV	The voltage step size: –20 V to +20 V; see **Details**
Frequency	The measurement frequency to use: ■ 100 kHz: `0` ■ 1 MHz: `1`
DefaultBias	The DC bias applied before and after a sweep: –20 V to +20 V
StartTime	The time that occurs on the first bias step, from the point the instrument is first triggered until the first step time: 0.001 s to 65 s
StopTime	The time between pulses with the 590 at the default bias: 0.001 s to 65 s
StepTime	The time after a transition to a new bias step and before the instrument begins a measurement: 0.001 s to 65 s
Range	The measurement range to use in F: 2E-12, 20E-12, 200E-12, or 2E-9; see **Details**
Model	Measurement model: ■ Series model: `0` ■ Parallel model: `1`
Filter	Enable or disable the analog filter, which can minimize the amount of noise that appears in the readings; however, it increases the measurement time: ■ Disable the filter: `0` ■ Enable the filter: `1`
ReadingRate	Selects the reading rate used to acquire the measurements: 0 to 4; see **Details**
C	Output: The measured array of capacitance values

Csize	Set to a value that is equal to or greater than the *G_or_Rsize* or number of voltage steps in the sweep, or is equal to ((*LastBias* – *FirstBias*) /*StepV*) + 1; when this function is called from a Clarius, this parameter is fixed at 1350
V	The array of bias voltages used
Vsize	Set to a value that is equal to or greater than the *G_or_Rsize* or number of voltage steps in the sweep, or is equal to ((*LastBias* – *FirstBias*) /*StepV*) + 1; when this function is called from a Clarius, this parameter is fixed at 1350
G_or_R	Output: ■ When the parallel measurement model (1) is selected, *G_or_R* is the measured conductance ■ When the series measurement model (0) is selected, this is the measured resistance
G_or_Rsize	Set to a value that is equal to or greater than the *G_or_Rsize* or number of voltage steps in the sweep, or is equal to ((*LastBias* – *FirstBias*) /*StepV*) + 1; when this function is called from a Clarius UTM, this parameter is fixed at 1350
T	The array of time stamps for each measurement step
Tsize	Set to a value that is equal to or greater than the *G_or_Rsize* or number of voltage steps in the sweep, or is equal to ((*LastBias* – *FirstBias*) /*StepV*) + 1; when this function is called from a Clarius UTM, this parameter is fixed at 1350

Details

This user module performs a capacitance versus voltage pulse sweep. The figure below shows the default parameters for the `590-cvpulsesweep` UTM, which uses the `CvPulseSweep590` user module.

Figure 661: CvPulseSweep590 (cvpulsesweep UTM parameters)

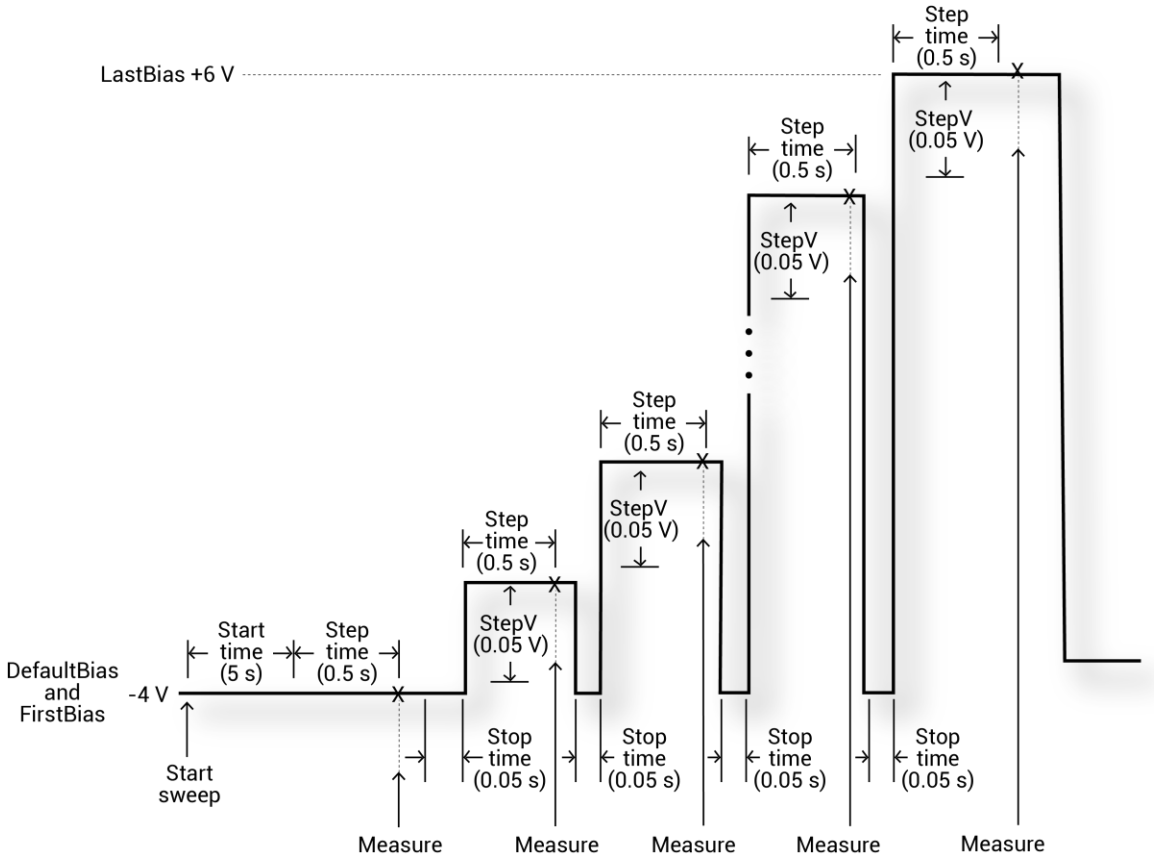
Formulator | User Libraries: KI590ulib

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

In this example, the 590 outputs a series of pulses in 50 mV steps from –4 V to +6 V. As shown in the following figure, a measurement is made on each pulse step.

Figure 662: C-V pulse sweep measurements



If the file defined for *CabCompFile* does not exist, or there is no path specified (null string), the default compensation parameters are used. When entering the path, use two backslash (\\) characters to separate each directory. For example, if your cable file is in C:\calfiles\590cal.dat, you enter the following:

```
C:\\calfiles\\590cal.dat
```

NOTE

If a switch matrix to route signals is being controlled by a connection action (for example, connect), there is no need to connect *InputPin* and *OutPin*. Set these parameters to 0.

The value of ((*LastBias* – *FirstBias*) / *StepV*) +1 must be less than or equal to the *Csize*, *Vsize*, *G_or_Rsize*, and *Tsize* parameters.

For *Range*, the measurement range values are shown in the following table.

Range	100 kHz	1 MHz
1	2 pF / 2 µs	20 pF / 200 µs
2	20 pF / 20 µs	20 pF / 200 µs
3	200 pF / 200 µs	200 pF / 2 ms
4	2 nF / 2 ms	2 nF / 20 ms

The reading rates and resolutions for the *ReadingRate* parameter are described in the following table.

Reading rate	Nominal reading rate (per second)	Display readings	Resolution (digits)
0	1000	C	3.5
1	75	C,G,V	3.5
2	18	C,G,V	4.5
3	10	C,G,V	4.5
4	1	C,G,V	4.5

Returned values are placed in the Analyze sheet.

- 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 configuration.
- -10023 (KI590_MEAS_ERROR): A measurement error occurred.
- -10090 (GPIB_ERROR_OCCURRED): A GPIB communications error occurred.
- -10091 (GPIB_TIMEOUT): A timeout 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 response from the 590.
- -10104 (USER_CANCEL): The user canceled the correction procedure.

Procedure

1. You are prompted to open the circuit so that an offset capacitance measurement can be made if needed.
2. If a cable compensation file is specified, the compensation information in that file for the selected range and frequency is loaded. If not, instrument default compensation is used.
3. A C-V pulse sweep is done.

Also see

None

CvSweep590 user module

The `CvSweep590` routine does a capacitance versus voltage (C-V) sweep using the Keithley Instruments 590 C-V Analyzer. You can make an offset correction measurement and use cable compensation.

Usage

```
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);
```

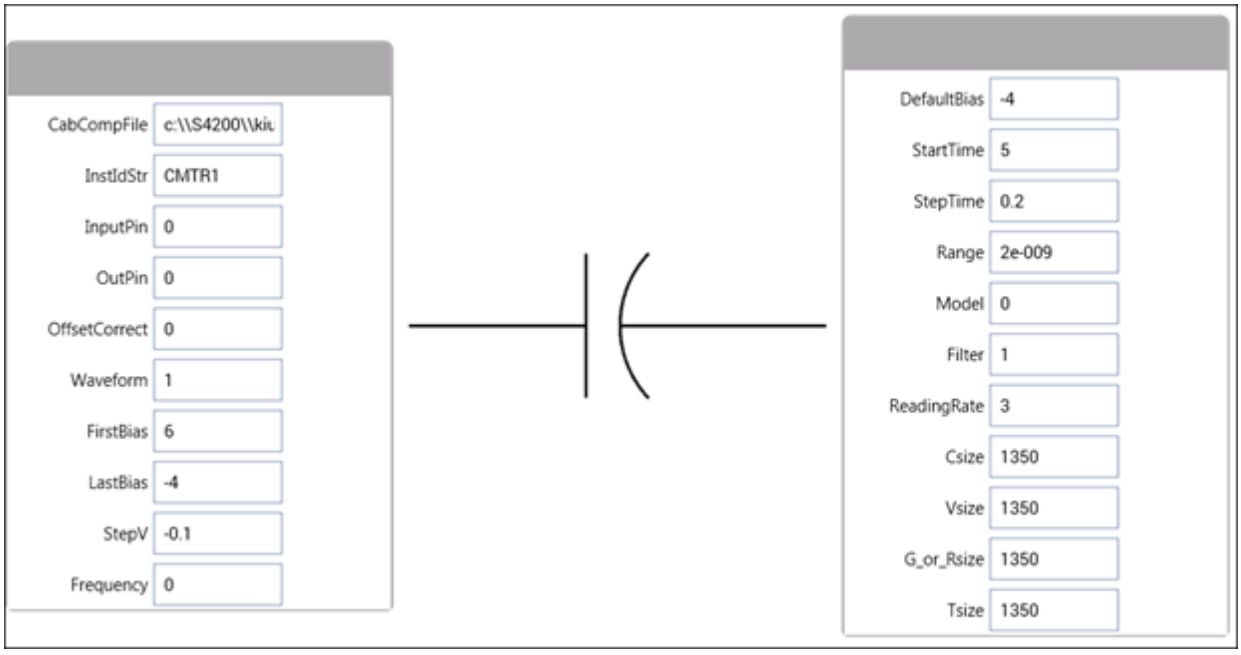
status	Returned values; see **Details**
CabCompFile	The complete name and path for the cable compensation file; see **Details**
InstIdStr	The CMTR instrument ID; `CMTR1`, `CMTR2`, `CMTR3`, or `CMTR4`, depending on your system configuration
InputPin	The DUT pin to which the 590 input terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connection is made; see **Details**
OutPin	The DUT pin to which the 590 output terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connection is made; see **Details**
OffsetCorrect	Determines if an offset correction measurement should be made: ■ Do not make offset measurement: `0` ■ Make offset measurement: `1`
Waveform	Selects either the single staircase or dual staircase waveform: ■ Single: `1` ■ Dual: `2`
FirstBias	The starting voltage for the sweep: –20 V to +20 V
LastBias	The last voltage used in the sweep: –20 V to +20 V
StepV	The voltage step size: –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	The measurement frequency to use: ■ 100 kHz: `0` ■ 1 MHz: `1`
DefaultBias	The DC bias that is applied before and after a sweep: –20 V to +20 V
StartTime	The time that occurs on the first bias step, from the point the instrument is first triggered until the first step time: 0.001 s to 65 s
StepTime	The time period after a transition to a new bias step and before the instrument begins a measurement: 0.001 s to 65 s
Range	The measurement range to use in F: 2E-12, 20E-12, 200E-12, or 2E-9; see **Details**
Model	Measurement model: ■ Series model: `0` ■ Parallel model: `1`

Filter	Enable or disable the analog filter, which can minimize the amount of noise that appears in the readings; however, it increases the measurement time: ■ Disable the filter: 0 ■ Enable the filter: 1
ReadingRate	Selects the reading rate used to acquire the measurements: 0 to 4; see **Details**
C	Output: The measured array of capacitance values
Csize	Set to a value equal to or greater than the number of voltage steps in the sweep or equal to ((*LastBias* – *FirstBias*) / *StepV*) + 1
V	Output: The array of bias voltages used
Vsize	Set to a value equal to or greater than the number of voltage steps in the sweep or equal to ((*LastBias* – *FirstBias*) / *StepV*) + 1
G_or_R	Output: ■ When the parallel measurement model (0) is selected, *G_or_R* is the measured conductance ■ When the series measurement model (1) is selected, this is the measured resistance
G_or_Rsize	When this function is called from a Clarius UTM, this parameter is fixed at 1350
T	The array of time stamps for each measurement step
Tsize	When this function is called from a Clarius UTM, this parameter is fixed at 1350

Details

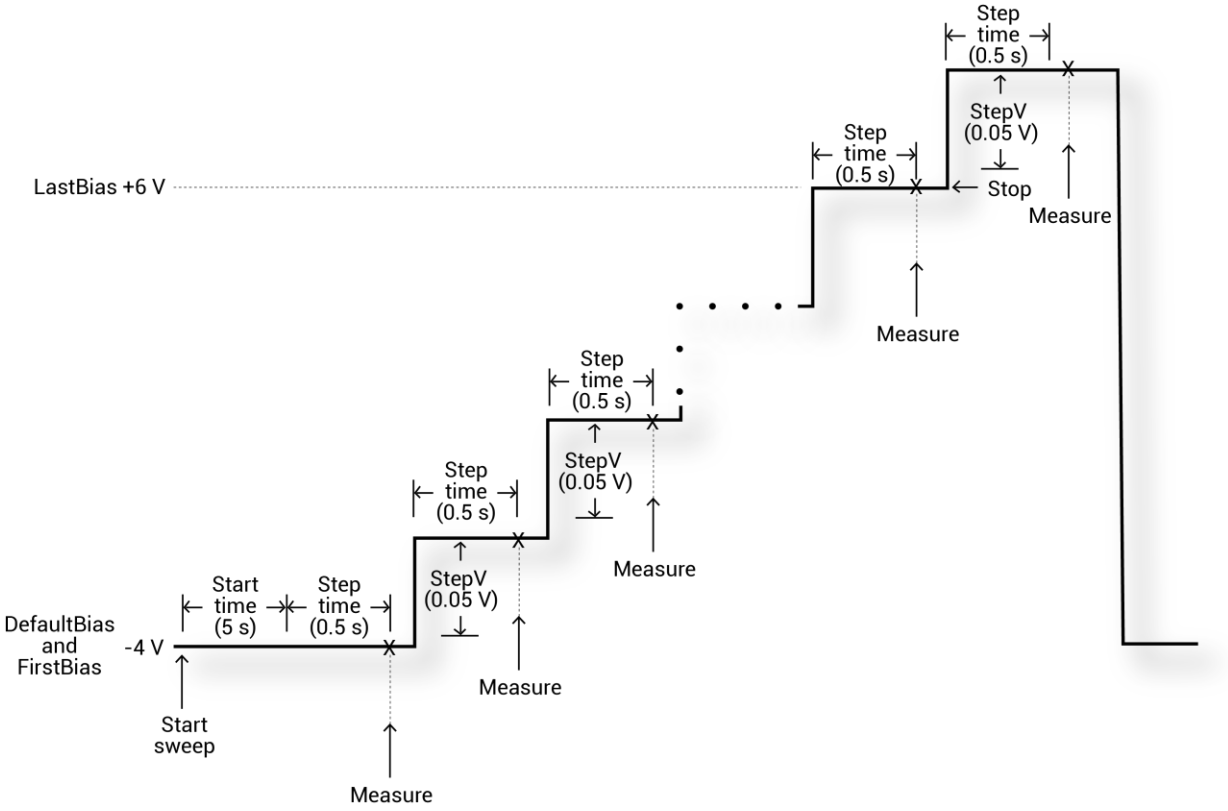
This user module performs a capacitance versus voltage staircase sweep. The figure below shows the default parameters for the 590-cvsweep UTM, which uses the CvSweep590 user module.

Figure 663: CvSweep590 user module (590-cvsweep UTM)



In general, the 590 outputs a linear staircase voltage sweep from –4 V to +6 V in 50 mV steps. As shown in the following figure, a capacitance measurement is made on each step of the sweep. A test example demonstrates how to perform a C-V sweep (see Model 590 test examples (on page B-7)).

Figure 664: C-V linear staircase sweep



If the file defined for *CabCompFile* does not exist, or there is no path specified (null string), the default compensation parameters are used. When entering the path, use two backslash (\\) characters to separate each directory. For example, if your cable file is in C:\calfiles\590cal.dat, you enter the following:

```
C:\\calfiles\\590cal.dat
```

NOTE

If a switch matrix to route signals is being controlled by a connection action (for example, connect), there is no need to connect *InputPin* and *OutPin*. Set these parameters to 0.

For *Range*, the measurement range values are shown in the following table.

Range	100 kHz	1 MHz
1	2 pF / 2 µs	20 pF / 200 µs
2	20 pF / 20 µs	20 pF / 200 µs
3	200 pF / 200 µs	200 pF / 2 ms
4	2 nF / 2 ms	2 nF / 20 ms

The reading rates and resolutions for the *ReadingRate* parameter are described in the following table.

Reading rate	Nominal reading rate (per second)	Display readings	Resolution (digits)
0	1000	C	3.5
1	75	C,G,V	3.5
2	18	C,G,V	4.5
3	10	C,G,V	4.5
4	1	C,G,V	4.5

Returned values are placed in the Analyze sheet:

- 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 configuration.
- -10023 (KI590_MEAS_ERROR): A measurement error occurred.
- -10090 (GPIB_ERROR_OCCURRED): A GPIB communications error occurred.
- -10091 (GPIB_TIMEOUT): A timeout 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 response from the 590.
- -10104 (USER_CANCEL): The user canceled the correction procedure.

Procedure

1. You are prompted to open the circuit so that an offset capacitance measurement can be made if needed.
2. If a cable compensation file is specified, the compensation information in that file for the selected range and frequency is loaded. If not, instrument default compensation is used.
3. A C-V sweep is performed.

Also see

None

DisplayCableCompCaps590 user module

`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 Clarius, to the Analyze sheet.

Usage

```
status = DisplayCableCompCaps590(char *CabCompFile, double *Range, int RangeSize,
    double *Values100k, int Values100kSize, double *Values1M, int Values1MSize);
```

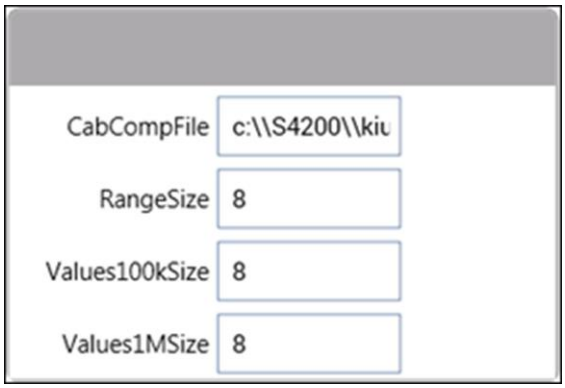
status	Returned values are placed in the Analyze sheet; see **Details**
CabCompFile	The complete name and path for the cable compensation file; see **Details**
Range	Output: An 8-element array that receives the nominal range values
RangeSize	The size of the *Range* array; set to 8
Values100k	Output: An 8-element fixed array that receives the nominal capacitor values used for the cable compensation at the 100 kHz frequency
Values100kSize	The size of the *Values100k* array; set to 8
Values1M	Output: An 8-element fixed array that receives the nominal capacitor values used for the cable compensation at the 1 MHz frequency
Values1MSize	The size of the *Values1M* array; set to 8

Details

This user module is used for 590 cable compensation. When this test is run, the nominal capacitance source values saved by the `SaveCableCompCaps590` user module are placed into a spreadsheet.

The default parameters for this user module are shown in the following figure. 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.

Figure 665: DisplayCableCompCaps590 default parameters



To prevent unpredictable results, the array size values for the *RangeSize*, *Values100kSize*, and *Values1MSize* must be set to 8, as shown in the figure above.

See Example 1: Cable compensation (on page B-7) for a demonstration of how cable compensation is done.

The returned arrays are arranged in the order shown in the following table.

Reading_rate **valid inputs**

Range	100 kHz values	1 MHz values
2E-12	2 pF low comp value	2 pF low comp value
2E-12	2 pF high comp value	2 pF high comp value
20E-12	20 pF low comp value	20 pF low comp value
20E-12	20 pF high comp value	20 pF high comp value
200E-12	200 pF low comp value	200 pF low comp value
200E-12	200 pF high comp value	200 pF high comp value
2E-9	2 nF low comp value	2 nF low comp value
2E-9	2 nF high comp value	2 nF high comp value

If the file defined for *CabCompFile* does not exist, or there is no path specified (null string), the default compensation parameters are used. When entering the path, use two backslash (\\) characters to separate each directory. For example, if your cable file is in C:\calfiles\590cal.dat, you enter the following:

```
C:\\calfiles\\590cal.dat
```

NOTE

If a switch matrix to route signals is being controlled by a connection action (for example, connect), there is no need to connect *InputPin* and *OutPin*. Set these parameters to 0.

The return values from *status* can be:

- 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 configuration.

Also see

SaveCableCompCaps590 user module (on page B-40)

LoadCableCorrectionConstants

`LoadCableCorrectionConstants` reads the cable compensation parameters for the range and frequency specified from the cable compensation file and sends these parameters to the 590.

Usage

```
status = LoadCableCorrectionConstants(char *CabCompFile, char *instr_id, *frequency,
    int range);
```

status	Returned values are placed in the Analyze sheet; see **Details**
CabCompFile	The complete name and path for the cable compensation file; see **Details**
instr_id	The CMTR instrument ID; this can be `CMTR1` through `CMTR4`, depending on the configuration of your system
frequency	The frequency of the correction constant: `0` = 1 MHz; `1` = 100 kHz
range	The range of the correct constant; see **Details**

Details

If the file specified by *CapCompFile* does not exist, it is created. 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 in file `C:\calfiles\590cal.dat`, you would enter `C:\\calfiles\\590cal.dat`.

range **values**

Range	100 kHz values	1 MHz values
1	2 pF/2 µs	20 pF/200 µs
2	20 pF/20 µs	20 pF/200 µs
3	200 pF/200 µs	200 pF/2 ms
4	2 nF/2 ms	2 nF/20 ms

The return values from *status* can be:

- `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.
- `-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 configuration.

Also see

SaveCableCompCaps590 user module (on page B-40)

SaveCableCompCaps590 user module

This function saves the nominal values of the capacitors used to do the 590 cable compensation procedure to the indicated file. If no cable compensation file exists, this module creates one if the user has the proper system permissions.

Usage

```
status = SaveCableCompCaps590(char *CabCompFile, double Lo2p100k, double Lo2p1M, double
    Hi2p100k, double Hi2p1M, double Lo20p100k, double Lo20p1M, double Hi20p100k, double
    Hi20p1M, double Lo200p100k, double Lo200p1M, double Hi200p100k, double 200p1M,
    double Lo2n100k, double Lo2n1M, double Hi2n100k, double Hi2n1M);
```

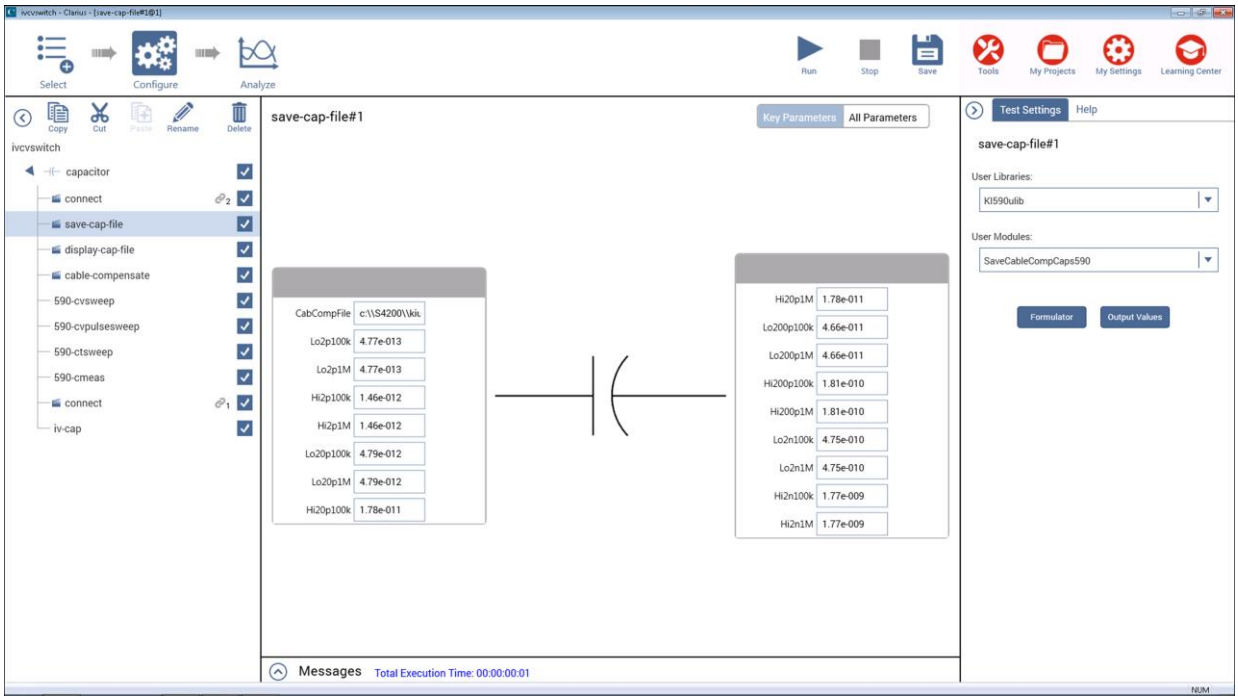
status	Returned values are placed in the Analyze sheet; see **Details**
CabCompFile	The complete name and path for the cable compensation file; see **Details**
Lo2p100k	The nominal value of the low-range capacitor used for cable compensation for the 2 pF range and 100 kHz frequency: 0 F to 0.95E-12 F
Lo2p1M	The nominal value of the low-range capacitor used for cable compensation for the 2 pF range and 1 MHz frequency: 0 F to 0.95E-12 F
Hi2p100k	The nominal value of the high-range capacitor used for cable compensation for the 2 pF range and 100 kHz frequency: 1E-12 F to 2E-12 F
Hi2p1M	The nominal value of the high-range capacitor used for cable compensation for the 2 pF range and 1 MHz frequency: 1E-12 F to 2E-12 F
Lo20p100k	The nominal value of the low-range capacitor used for cable compensation for the 20 pF range and 100 kHz frequency: 0 F to 9.5E-12 F
Lo20p1M	The nominal value of the low-range capacitor used for cable compensation for the 20 pF range and 1 MHz frequency: 0 F to 9.5E-12 F
Hi20p100k	The nominal value of the high-range capacitor used for cable compensation for the 20 pF range and 100 kHz frequency: 10E-12 F to 20E-12 F
Hi20p1M	The nominal value of the high-range capacitor used for cable compensation for the 20 pF range and 1 MHz frequency: 10E-12 F to 20E-12 F
Lo200p100k	The nominal value of the low-range capacitor used for cable compensation for the 200 pF range and 100 kHz frequency: 0 F to 95E-12 F
Lo200p1M	The nominal value of the low-range capacitor used for cable compensation for the 200 pF range and 1 MHz frequency: 0 F to 95E-12 F
Hi200p100k	The nominal value of the high-range capacitor used for cable compensation for the 200 pF range and 100 kHz frequency: 100E-12 F to 200E-12 F
Hi200p1M	The nominal value of the high-range capacitor used for cable compensation for the 200 pF range and 1 MHz frequency: 100E-12 F to 200E-12 F
Lo2n100k	The nominal value of the low-range capacitor used for cable compensation for the 2 nF range and 100 kHz frequency: 0 F to 995E-12 F
Lo2n1M	The nominal value of the low-range capacitor used for cable compensation for the 2 nF range and 1 MHz frequency: 0 F to 995E-12 F
Hi2n100k	The nominal value of the high-range capacitor used for cable compensation for the 2 nF range and 100 kHz frequency: 1000E-12 F to 2000E-12 F
Hi2n1M	The nominal value of the high-range capacitor used for cable compensation for the 2 nF range and 1 MHz frequency: 1000E-12 F to 2000E-12 F

Details

This user module is used for 590 cable compensation. You enter 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 the following figure. These are suggested low and high values that can be used for cable compensation. You must replace these values with the calibration values of the capacitance sources.

Figure 666: save-cap-file action and SaveCableCompCaps590 user module



Example 1: Cable compensation (on page B-7) demonstrates how cable compensation is done.

If the file defined for *`CabCompFile`* does not exist, or there is no path specified (null string), the default compensation parameters are used. When entering the path, use two backslash (`\\`) characters to separate each directory. For example, if your cable file is in `C:\calfiles\590cal.dat`, you enter the following:

```
C:\\calfiles\\590cal.dat
```

NOTE

If a switch matrix to route signals is being controlled by a connection action (for example, `connect`), there is no need to connect *`InputPin`* and *`OutPin`*. Set these parameters to `0`.

The return values from *status* can be:

- 0: OK.
- -10000 (INVAL_INST_ID): The specified instrument ID does not exist. 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 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 the configuration of your system.

Also see

CableCompensate590 user module (on page B-15)

Appendix C

Using a Keysight 4284/4980A LCR Meter

In this appendix:

Introduction

NOTE

This section contains information on using the 4200A-SCS with the Keysight Models 4284A and 4980A.

For details on Keysight Model 4284A operation, refer to the *Keysight Model 4284A Operation Manual*. For details on Keysight Model 4980A operation, refer to the *Keysight Model 4980A Operation Manual*.

C-V measurement basics

The Keithley Instruments 4200A-SCS can control a Keysight 4284A or 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 capacitor).

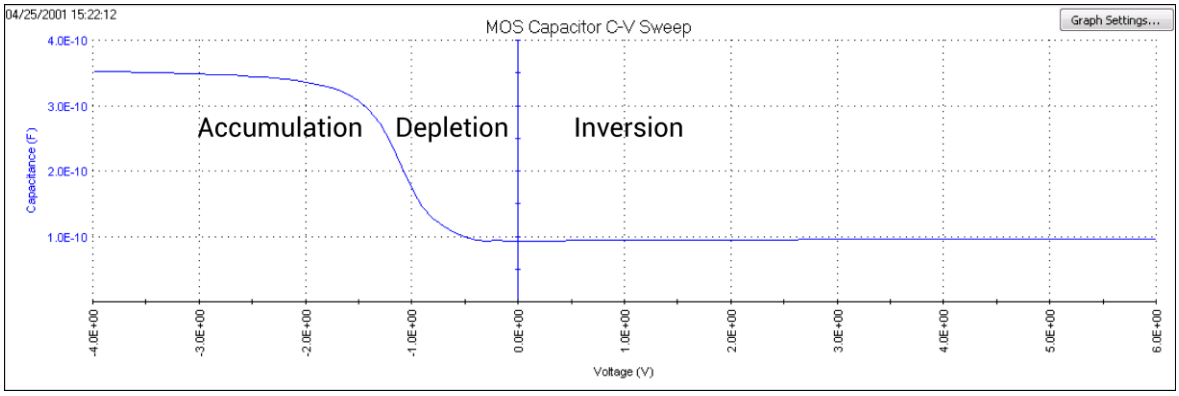
The measurements of MOS capacitors study:

- The integrity of the gate oxide and semiconductor doping profile
- The lifetime of semiconductor material
- The interface quality between the gate oxide and silicon
- Other dielectric materials used in an integrated circuit

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.

The following figure shows the three regions of a typical C-V curve for a MOS capacitor.

Figure 667: Typical C-V curve for a MOS capacitor



Capacitance measurement tests

The 4200A-SCS provides the following user modules to perform C-V tests using the Keysight Models 4284A and 4980A:

- **CvSweep4284.** C-V sweep test: Performs a capacitance and conductance measurement at each step of a user-configured linear voltage sweep.
- **Cmeas4284.** C measurement: Performs a capacitance and conductance measurement at a fixed bias voltage.

Details on the user modules for the Keysight Models 4284A and 4980A are provided in the HP4284ulib User Library Reference (on page 6-385).

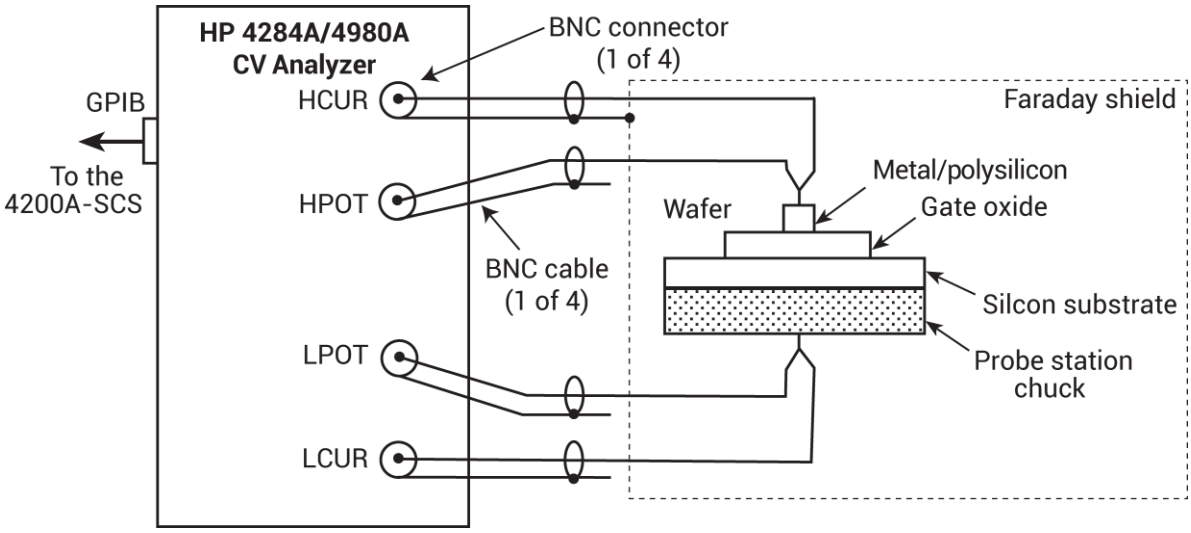
NOTE

If needed, you can initially do an OPEN and SHORT correction on the 4284A or 4980A to achieve the most accurate C-V measurements. See the Keysight 4284A or 4980A *Operation Manual* for details.

Signal connections

Basic 4-wire signal connections for the Model 4284A or 4980A are shown in the following figure. 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 4284A or 4980A output is typically connected to the wafer backside (or well). The input is typically connected to the gate of a MOS capacitor.

Figure 668: Basic Model 4284A and 4980A connections to DUT



Triaxial connections

Adapters are required to connect the Model 4284A or 4980A to equipment (for example, a probe station, test fixture, or matrix card) that uses triaxial connectors.

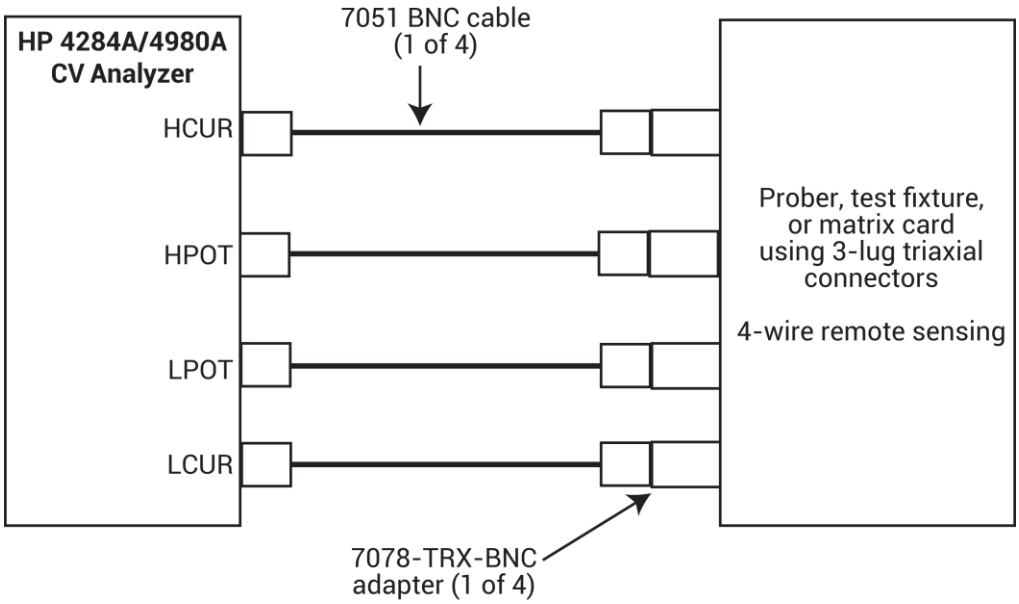
NOTE

See Using Switch Matrices (on page A-1) for details on using a switch matrix with the Model 4284A or 4980A LCR Meter.

4-wire remote sensing

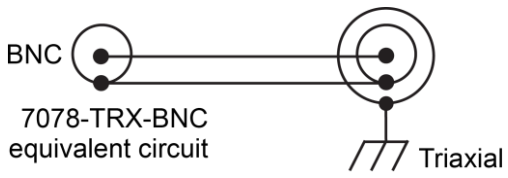
The following figure shows 4-wire remote sense connections. The 7078-TRX-BNC is a 3-lug triaxial-to-BNC adapter. As shown in the figure, connect the adapters to the 3-slot triaxial connectors, and then use a 7051-5 BNC cable to make the connections to the Model 4284A or 4980A.

Figure 669: 4-wire remote sense connections to equipment using triaxial connectors



The figure below shows the equivalent circuit for the adapter.

Figure 670: 7078-TRX-BNC equivalent circuit



2-wire remote sensing

For 2-wire local sense connections, coaxial tees are required to adapt dual BNC cables to single BNC cables, as shown in the following figure.

Figure 671: 2-wire local sense connections to equipment using triaxial connectors

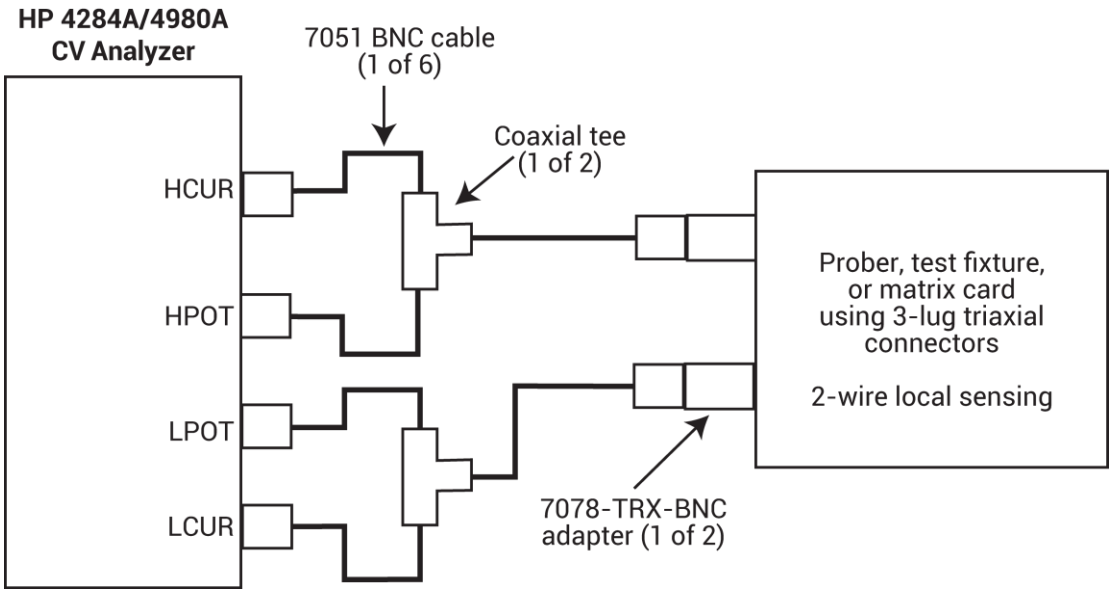
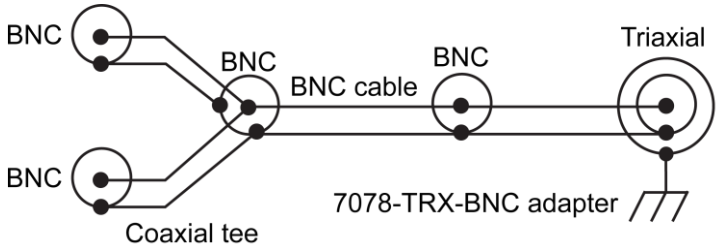


Figure 672: Equivalent circuit



GPIB connections

The 4200A-SCS controls the Model 4284A or 4980A through the General Purpose Interface Bus (GPIB). Use the Keithley 7007-1 or 7007-2 GPIB cable to connect the GPIB port of the Model 4284A or 4980A to the GPIB port of the 4200A-SCS.

Using KCon to add a Keysight LCR Meter to the system

To use the 4200A-SCS to control an external instrument, you must add the instrument to the 4200A-SCS system configuration. You use Keithley Configuration Utility (KCon) to add the Keysight Model 4284A or 4980A to the test system.

Refer to Using KCon to add equipment to the 4200A-SCS (on page 7-7) for instruction.

For additional details on KCon, refer to Keithley Configuration Utility (on page 7-1).

Model 4284A or 4980A test example

The following test example for the Model 4284A or 4980A LCR Meter is controlled by the `ivcvswitch` project with an added user test module (UTM) created from a user module from the `HP4284ulib` user library. A switch matrix is not used for this example.

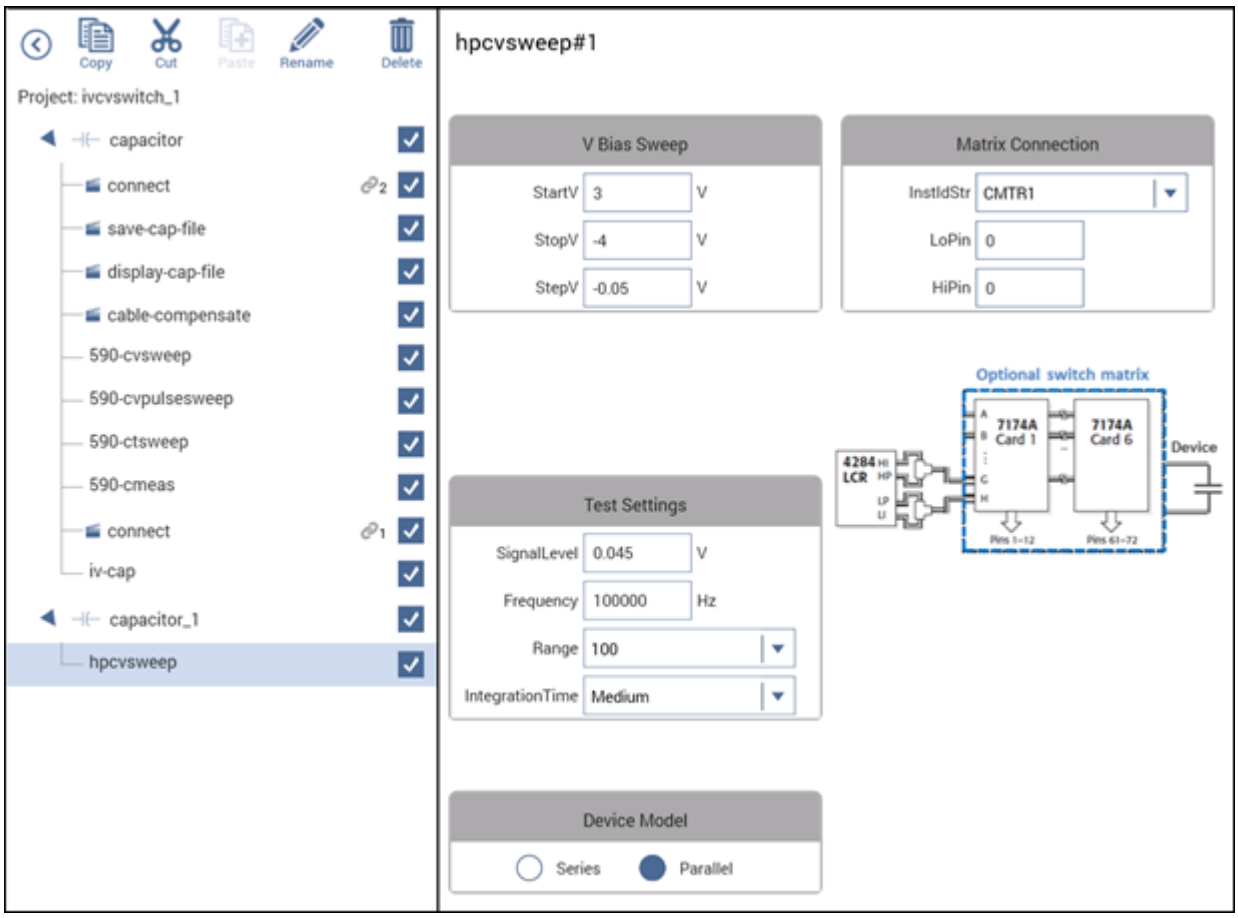
C-V sweep

This example assumes that the Model 4284A or 4980A is already connected directly to the device under test (DUT). The DUT can be a device installed in a test fixture or a MOS capacitor on a wafer. Complete the following steps to do a C-V sweep.

Set up the project:

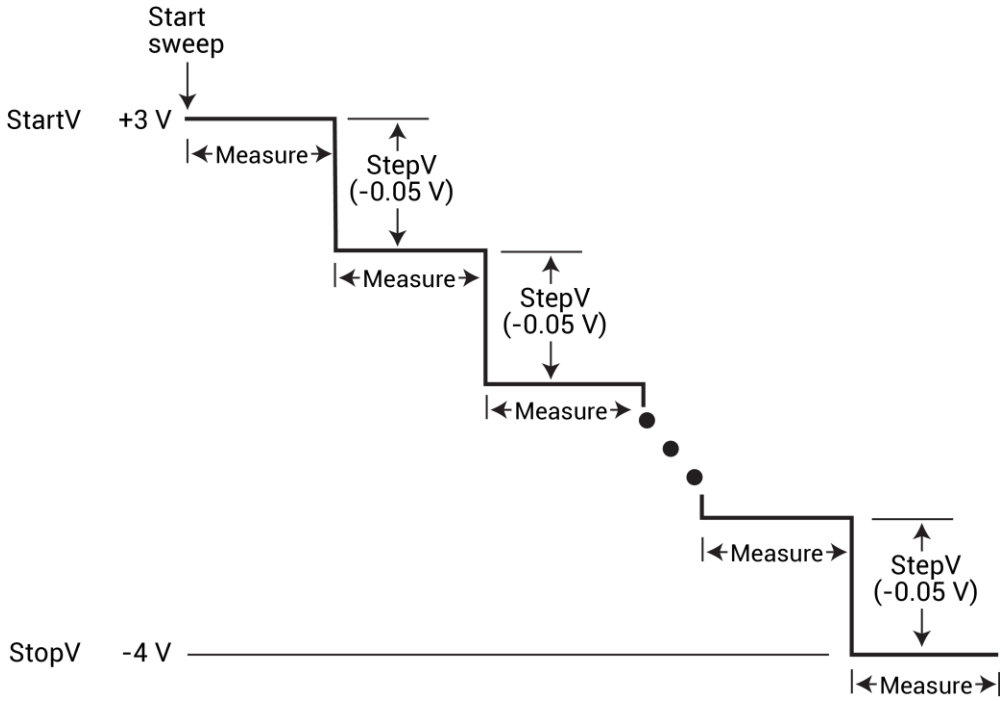
1. From the Project Library, select the **ivcvswitch** project.
2. Select **Create**.
3. At the bottom of the project tree, add another **capacitor** from the Device Library.
4. From the Test Library, select **Custom Test**, **Choose a test from the pre-programmed library (UTM)**.
5. Drag **Custom Test** to the project tree under the capacitor. The test has a red triangle next to it to indicate that it is not configured.
6. Select **Rename**.
7. Enter **hpcvsweep** and press **Enter**.
8. Select **Configure**.
9. In the Test Settings pane, select the user library **HP4284ulib**.
10. Select the user module **CvSweep4284**.
11. On the Configure pane, you can modify the test parameters as needed. With the settings shown in the figure below, the Model 4284A or 4980A does a +3 V to –4 V staircase sweep using 50 mV steps. A measurement is made on each step of the sweep. For details on the test description, see CvSweep4284 User Module (on page C-11).
12. Select **Run** to execute the test.

Figure 673: CvSweep4284 user module example



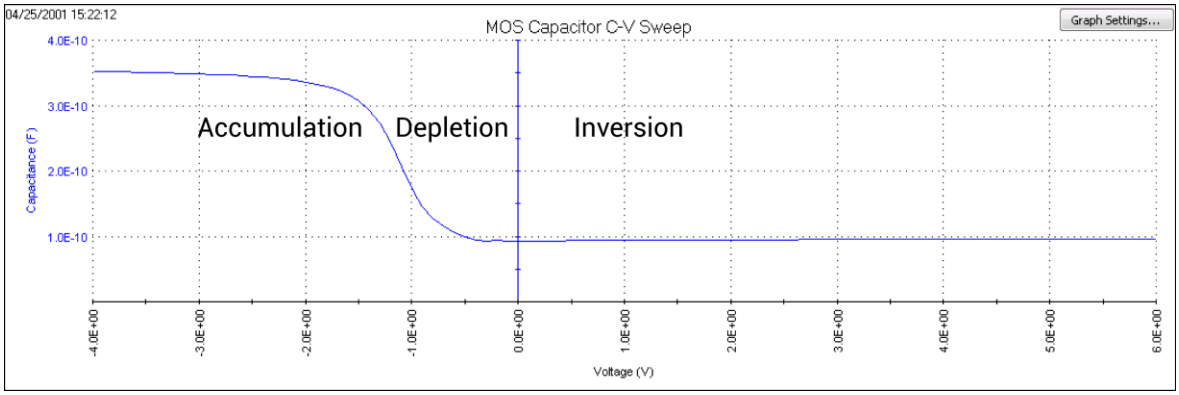
A configured sweep is shown in the following figure.

Figure 674: C-V linear staircase sweep



A typical graph that is generated by this test is shown below.

Figure 675: Typical C-V curve for a MOS capacitor



HP4284ulib user library

You use the user modules in the `HP4284ulib` user library to control the Keysight 4284A or 4980A LCR Meter. These user modules are summarized in the following table.

HP4284ulib user library

User module	Description
Cmeas4284 (on page C-13)	Makes a single capacitance measurement.
CvSweep4284 (on page C-11)	Makes capacitance versus voltage measurements using a staircase sweep.

CvSweep4284 User Module

The `CvSweep4284` routine performs a capacitance versus voltage (C-V) sweep using the Keysight Model 4284A or Model 4980 LCR Meter.

Usage

```
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);
```

status	Returned values; see **Details**
InstIdStr	The CMTR instrument ID; `CMTR1` or `CMTR2`, depending on your system configuration
LoPin	The DUT pin to which the 4284A or 4980 low terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connection is made; see **Details**
HiPin	The DUT pin to which the 4284A or 4980 high terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connection is made; see **Details**
StartV	Starting voltage of the sweep: –40 V to 40 V
StopV	Ending voltage of the sweep: –40 V to 40 V
StepV	The sweep voltage step size: –40 V to +40 V; the value of ((StopV – StartV)/StepV) + 1 must be less than or equal to the values for *Csize*, *Vsize*, and *G_or_Rsize*
SignalLevel	The oscillator output voltage level (5e-3 V to 20 V)
Frequency	Measurement frequency of the sweep: 20 Hz to 1e6 Hz
Range	The measurement range to use (in ohms): `0` (Auto), `100`, `300`, `1000`, `3000`, `10000`, `30000`, or `100000`
Model	Measurement model: `Series` or `Parallel`
IntegrationTime	The integration time to use: ■ Short: `0` ■ Medium: `1` ■ Long: `2`
C	Output: The measured array of capacitance values
Csize	A value equal to or greater than the *G_or_Rsize* number of steps in the sweep or = ((*StopV* – *StartV*)/*StepV*) + 1; when this function is called from a Clarius UTM, the value is fixed at 1350
V	Output: The array of voltage biases used in the sweep
Vsize	A value equal to or greater than the *G_or_Rsize* number of steps in the sweep or = ((*StopV* – *StartV*)/*StepV*) + 1; when this function is called from a Clarius UTM, the value is fixed at 1350
G_or_R	Output: ■ When the parallel measurement model (`1`) is selected, *G_or_R* is the measured conductance ■ When the series measurement model (`0`) is selected, this is the measured resistance
G_or_Rsize	A value equal to or greater than the *G_or_Rsize* number of steps in the sweep or = ((*StopV* – *StartV*)/*StepV*) + 1; when this function is called from a Clarius UTM, the value is fixed at 1350

Details

This user module performs a capacitance versus voltage staircase sweep. An example of its use is shown in C-V sweep (on page C-7). In this example, the Model 4284A or Model 4980A outputs a linear staircase voltage sweep from +3 V to –4 V in 50 mV steps. As shown in the C-V linear staircase sweep figure in C-V sweep (on page C-7), a capacitance measurement is made on each step of the sweep. For an example of how to run a C-V sweep, see Model 4284A or 4980A Test Example (on page C-6).

NOTE

If a switch matrix to route signals is being controlled by a connection action UTM (for example, `connect`), there is no need to connect *`LoPin`* and *`HiPin`*. Set these parameters to 0.

Returned values are placed in the Analyze spreadsheet.

- `0`: OK.
- `-10030 (HP4284_NOT_IN_KCON)`: No Keysight 4284A or Keysight 4980 LCR is defined in your system configuration.
- `-10031 (HP4284_MEAS_ERROR)`: A measurement error occurred.
- `-10090 (GPIB_ERROR_OCCURRED)`: A GPIB communications error occurred.
- `-10091 (GPIB_TIMEOUT)`: A timeout occurred during communications.
- `-10100 (INVAL_PARAM)`: An invalid input parameter is specified.
- `-10102 (ERROR_PARSING)`: There was an error parsing the response from the Model 4284A or 4980.
- `-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.

Also see

None

Cmeas4284 User Module

The `Cmeas4284` routine measures capacitance and conductance using the Keysight Model 4284A or 4980 LCR Meter.

Usage

```
status = Cmeas4284( char *InstIdStr, int LoPin, int HiPin, double SignalLevel, double
    Frequency, double BiasV, double Range, int Model, int IntegrationTime, double *C,
    double *V, double *G_or_R);
```

status	Returned values; see **Details**
InstIdStr	The CMTR instrument ID; `CMTR1` or `CMTR2` (default), depending on your system configuration
LoPin	The DUT pin to which the Model 4284A or 4980 low terminal is attached (–1 to 72; default 0); if a value of less than 1 is specified, no switch matrix connection is made; see **Details**
HiPin	The DUT pin to which the Model 4284A or 4980 high terminal is attached (–1 to 72; default 0); if a value of less than 1 is specified, no switch matrix connection is made; see **Details**
SignalLevel	The oscillator output voltage level: 5 mV to 20 V; default 0.045 V
Frequency	Measurement frequency of the sweep: 20 Hz to 1e6 Hz; default 100e3 Hz
BiasV	The DC bias to use for the measurement: –40 V to +40 V; default 1.0 V
Range	The measurement range to use (in ohms): 0 (Auto, the default), 100, 300, 1000, 3000, 10000, 30000, or 100000
Model	Measurement model: `Series` or `Parallel`
IntegrationTime	The integration time to use: ■ Short: `0` ■ Medium: `1` (default) ■ Long: `2`
C	Output: The measured capacitance
V	Output: The bias voltage used
G_or_R	Output: ■ Parallel measurement model (*G_or_R* is the measured conductance): `1` ■ Series measurement model (*G_or_R* is the measured resistance): `0`

Details

This user module makes a single, fixed-bias capacitance and conductance measurement.

NOTE

If a switch matrix to route signals is being controlled by a connection action UTM (for example, `connect`), there is no need to connect *LoPin* and *HiPin*. Set these parameters to `0`.

Returned values are placed in the Analyze spreadsheet.

- `0`: OK.
- `-10000 (INVAL_INST_ID)`: The specified instrument ID does not exist.
- `-10030 (HP4284_NOT_IN_KCON)`: No Keysight 4284A or Keysight 4980 LCR is defined in your system configuration.
- `-10031 (HP4284_MEAS_ERROR)`: A measurement error occurred.
- `-10090 (GPIB_ERROR_OCCURRED)`: A GPIB communications error occurred.
- `-10091 (GPIB_TIMEOUT)`: A timeout occurred during communications.
- `-10102 (ERROR_PARSING)`: There was an error parsing the response from the Model 4284A or 4980.

Also see

None

Appendix D

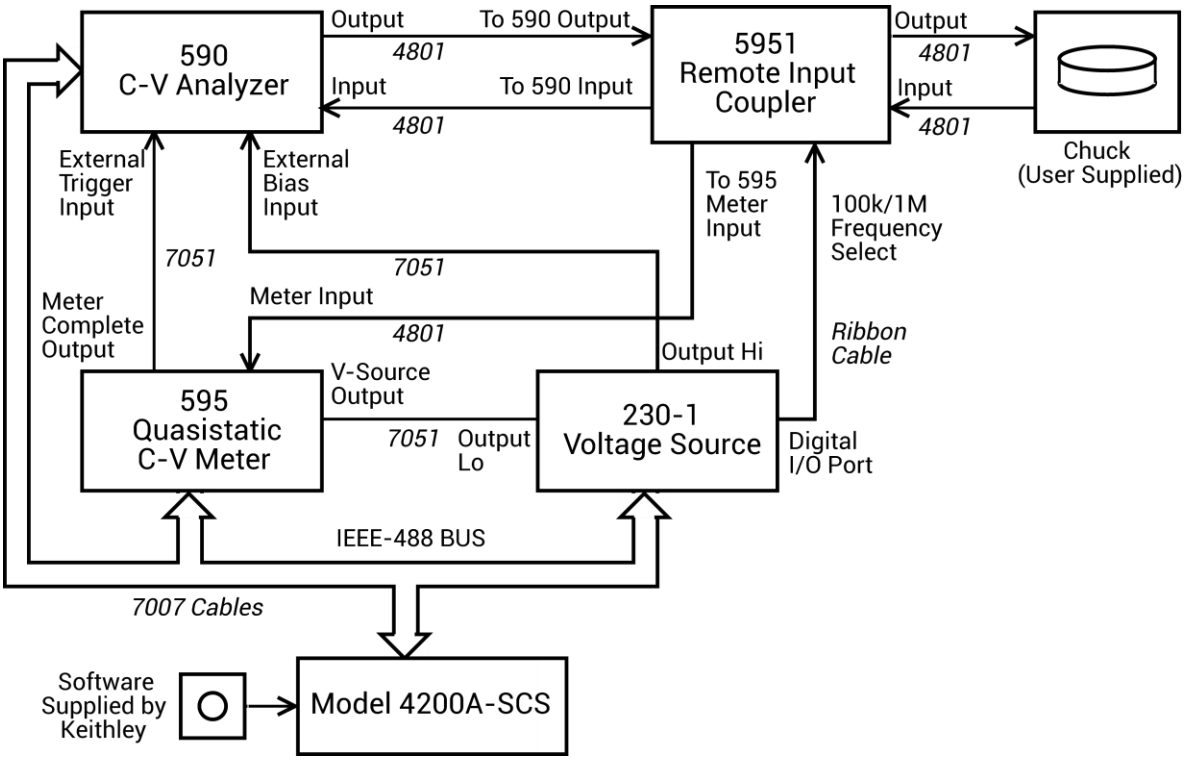
Using a Model 82 C-V System

In this appendix:

Introduction

The 82 C-V System uses a Keithley Instruments 590 C-V Analyzer and a Keithley Instruments 595 Quasistatic C-V Meter to make simultaneous C-V measurements. The complete system is shown in the figure below. Projects for the 4200A-SCS are provided to make simultaneous C-V measurements, STVS measurements for mobile ion extraction, and minority carrier generation lifetime measurements.

Figure 676: 82 C-V System block diagram



Capacitance measurement tests

The 4200A-SCS provides the following user modules for capacitance testing 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.

Details on all user modules for the Model 82 are provided in ki82ulib user library (on page 6-389).

C-t measurements

A C-t sweep performs a specified number of capacitance measurements at a specified time interval with voltage held constant. An example of a C-t waveform is shown in the figure below.

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 projects (on page D-11) for details on the test to perform C-t measurements.

Details on all parameters for the test using the CtSweep82 user module are in the ki82ulib user library (on page 6-389).

Figure 677: C-t waveform

Definitions:

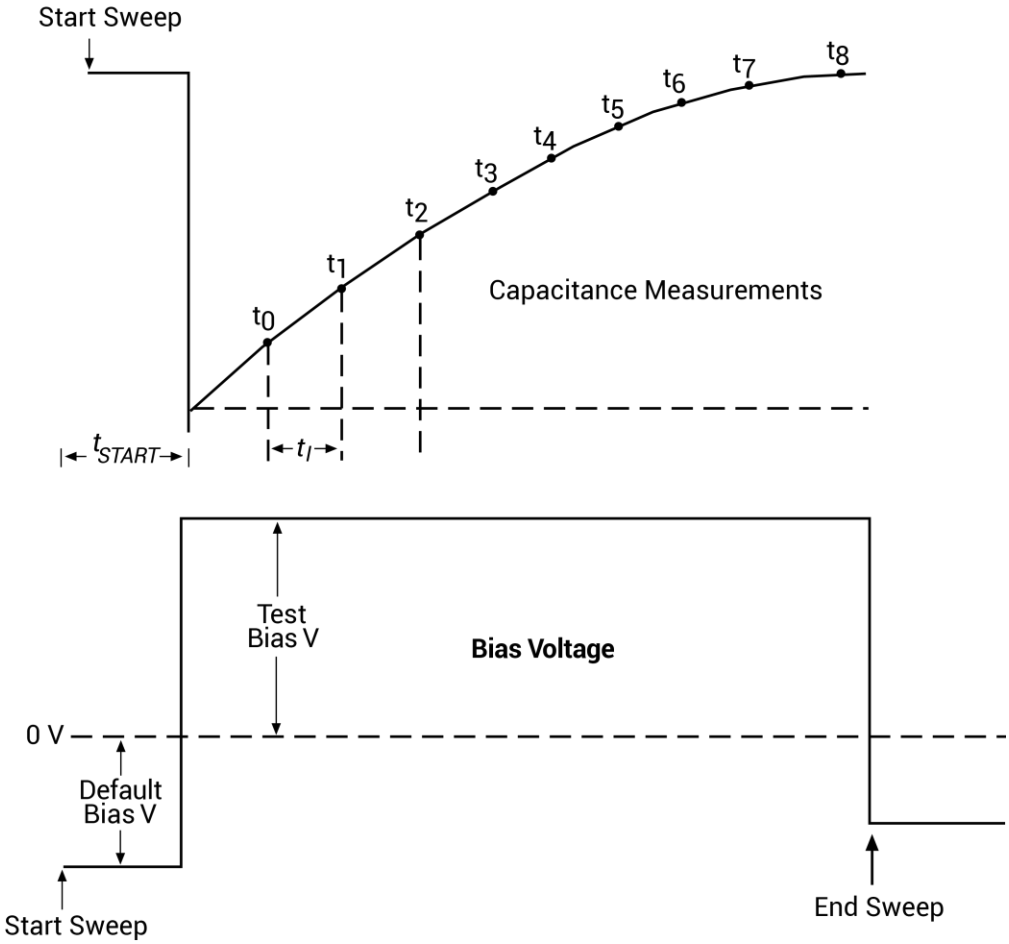
t_{START} = Start delay

t_{SAMPLE} = Sample time

t_I = Reading Interval = $(t_{SAMPLE} + I/R)$
where R = reading rate

Time computation:

t_s = Time at S
= $t_{START} + (t_{SAMPLE} + I/R)S$
where:
S = Sample #



Simultaneous C-V measurements

For simultaneous C-V measurements, the 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.

The figure below shows a simplified representation of the stepped bias voltage supplied by the 595 during a measurement sweep. Each vertical voltage step size is determined by the programmed 595 bias step, while each horizontal time step is determined by the programmed delay time.

A quasistatic measurement is a two-step process that requires 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 590 is triggered one delay time after the completion of each 595 reading. As a result, high-frequency measurements are made on only every other step (as represented by the small rectangles in the waveform figure). Also, 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 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 $VQ_m = (V_n - 0.5*V_{step})$.

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

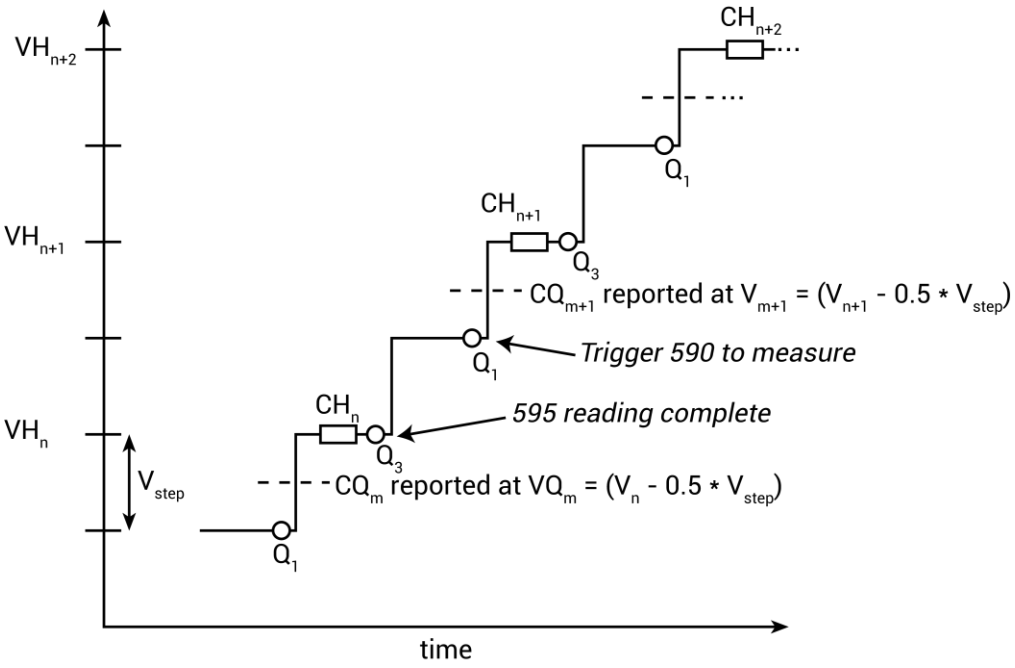
See Model 82 projects (on page D-11) for details on the test to perform simultaneous C-V measurements.

Details on all parameters for the test are provided in the ki82ulib user library (on page 6-389) for the CVsweep82 user module (on page D-41).

NOTE

As shown in the following figure, the first high frequency measurement (CH1) is made during the second phase of the voltage sweep. Only quasistatic capacitance (C1) is measured during the first phase and is disregarded.

Figure 678: Simultaneous C-V waveform



In the figure:

- CQ represents 595 quasistatic capacitance readings
- VQ represents voltage reported by the 595 corresponding to CQ
- CH represents Model 590 high frequency capacitance readings
- VH represents the voltage reported by the 590 corresponding to CH

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 done before measuring the capacitance of the DUT. In general, compensate for cables by connecting precisely known capacitance sources in place of the DUT and then measuring them. The Model 590 then uses these measured values to correct for unwanted capacitance when measuring the DUT.

Cable compensation involves these 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 done every time a new measurement is made.

Typically, cable compensation is done for all four measurement ranges (2 pF, 20 pF, 200 pF, and 2 nF) of the Model 590. Once cable compensation is done, it does not have to be done again unless the connection scheme to the DUT is changed or power is cycled.

Cable compensation user modules

The Model 82 user modules for cable compensation are:

- **SaveCableCompCaps82 (on page D-50): 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 (on page D-45): Places capacitance values into the Analyze spreadsheet:** When this test is executed, the capacitance values saved by `SaveCableCompCaps82` are placed into the Analyze spreadsheet.
- **CableCompensate82 (on page D-38): Performs cable compensation:** The user specifies the ranges and test frequencies for cable compensation. When this test is executed, on-screen prompts guide you through the cable compensation process.
- **CabCompFile**: Each of the user modules for cable compensation uses a cable compensation file to save and load capacitor source values. Therefore, these user modules must use the same file directory path.

Connections

The system block diagram in Introduction (on page D-2) shows the overall system configuration for the Model 82. Connect all cables as shown in the diagram.

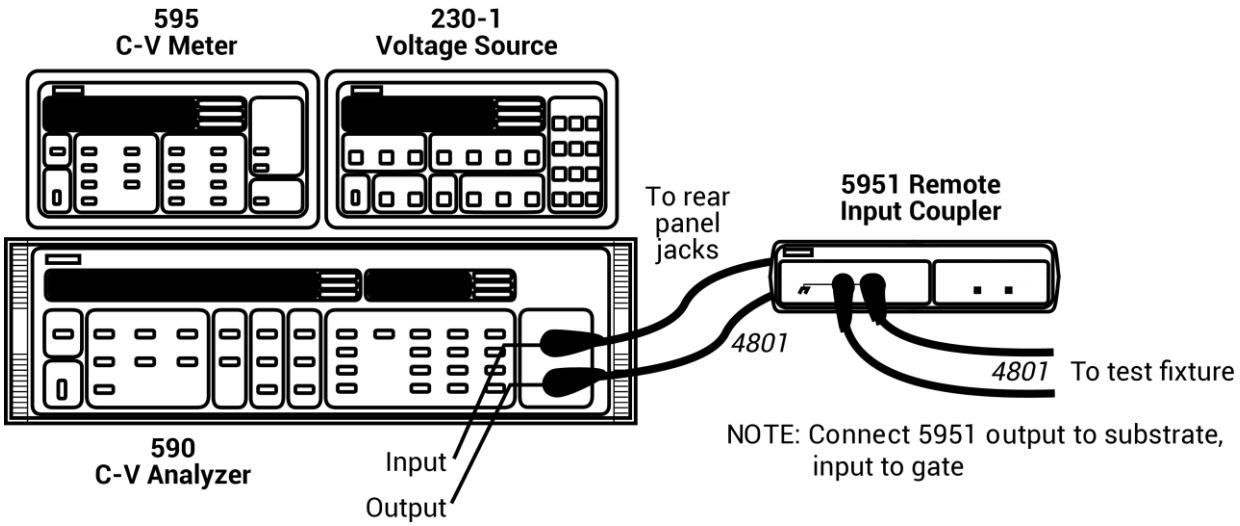
Front-panel connections

The front-panel connections diagram below shows the connections from the Model 5951 Remote Input Coupler to the Model 590.

To make front-panel connections:

1. 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.
2. 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.
3. Connect two more low-noise cables to the front of the Model 5951, where the input and output to the device are located.
4. 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 679: System 82 C-V system front-panel connections



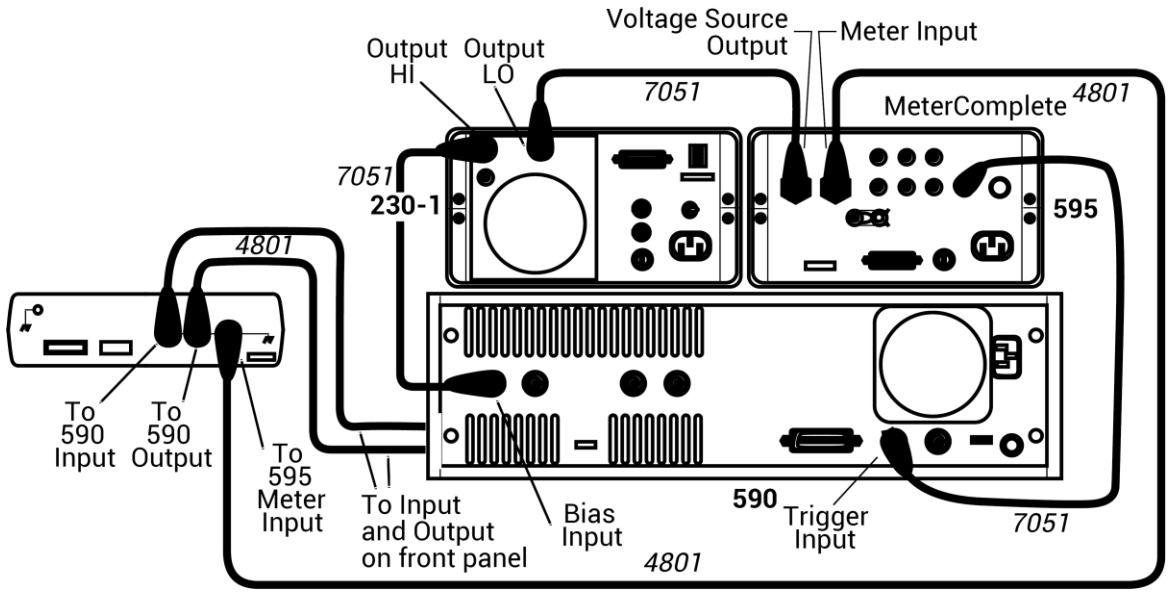
Rear-panel connections

The rear-panel connections diagram below shows the rest of the main cabling configuration.

To make rear-panel connections:

1. Use a Model 4801 cable to connect the METER INPUT on the back of the Model 595 to the TO 595 INPUT on the Model 5951.
2. Use a Model 7051-2 BNC cable to connect the METER COMPLETE port on the back of the Model 595 to the TRIGGER INPUT on the back of the Model 590
3. Use a 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.
4. Use a 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 680: System 82 C-V system rear-panel connections

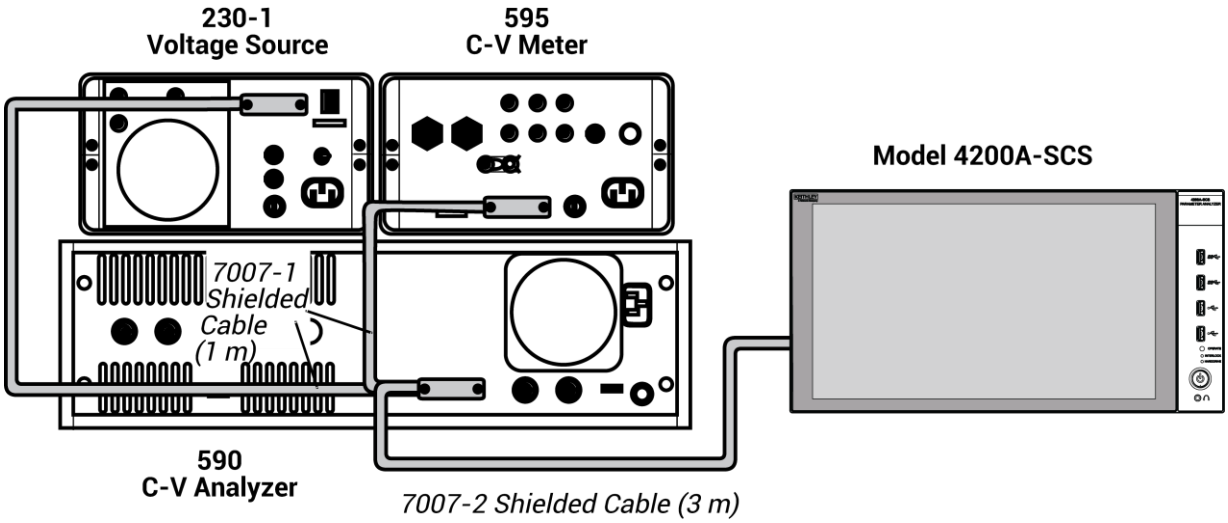


Make power and GPIB connections

To attach the power and GPIB connections:

1. 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.
2. Use the power cables to plug in the units.
3. The following figure shows the connections for the GPIB bus cables. Use the GPIB bus cables and connect the Model 590, the Model 595, and the Model 230-1 to the 4200A-SCS through the GPIB card.

Figure 681: System 82 IEEE-488 connections



Using KCon to add Model 82 C-V System

To use the 4200A-SCS to control instruments in the C-V system, you must add the system to the 4200A-SCS system configuration. You use the Keithley Configuration Utility (KCon) to add the 82 C-V system to the test system.

Refer to Using KCon to add equipment to the 4200A-SCS (on page 7-7) for instruction.

For additional detail on KCon, refer to Keithley Configuration Utility (on page 7-1).

Model 82 projects

The Model 82 projects are:

- `simcv`: This project uses the `qtsweep` test (`QTsweep82` user module) to make quasistatic capacitance measurements. This test optimizes delay time for quasistatic measurements so that the entire simultaneous C-V test is done at DUT equilibrium. Then the `system82-cvsweep` test (`SIMCVsweep82` user module) makes 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 make simultaneous C-V measurements.
- `lifetime`: This project uses the `cvsweep` test (`SIMCVsweep82` user module) to make simultaneous C-V measurements, and then uses the `ctsweep` test (`CTsweep82` user module) to make C-t measurements at the condition determined by the `cvsweep` test.

Each project begins by performing cable compensation.

The project trees for these projects are shown in the following figure.

NOTE

Details on all parameters for the compensation and capacitance tests are provided in the ki82ulib user library (on page 6-389).

Cable compensation tests

Complete the following steps to do cable compensation.

These tests assume that the calibration capacitors are installed as close to the wafer-chuck end of the cable as possible.

NOTE

The 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 lists for the user modules).

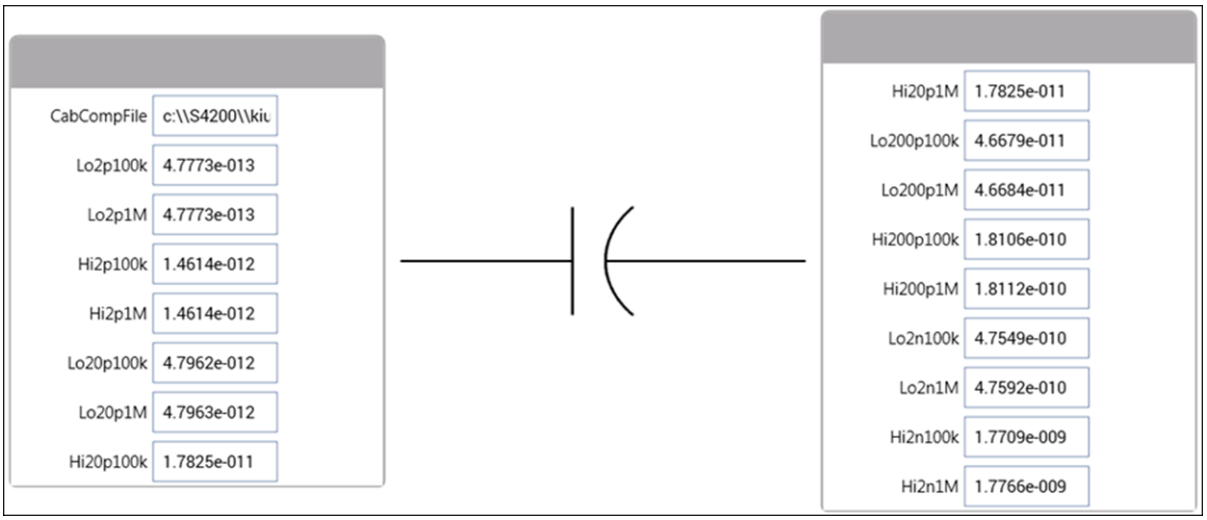
Enter and save capacitance source values (SaveCableCompCaps82)

To enter and save capacitance source values:

1. Open the project.
2. Select **save-cap-file** or **savecablecompfile**. These actions use the `SaveCableCompCaps82` user module
3. Select **Configure**. The Configure pane for these actions is shown in the figure below.
4. In the parameter list, enter the capacitance source calibration value for each range and frequency. For example, assume the low capacitance source for the 2 pA range is 0.47773 pF (100 kHz) and 0.47786 pA (1 MHz). Enter these values, using scientific notation:

   Lo2p100k: Enter **0.47773e-12**
   Lo2p1M: Enter **0.47786e-12**
5. Click **Run** to execute the test. The capacitor source values entered into the action are saved in the file using the directory path specified in the `CapCompFile` field.

Figure 682: SaveCableCompCaps82 user module

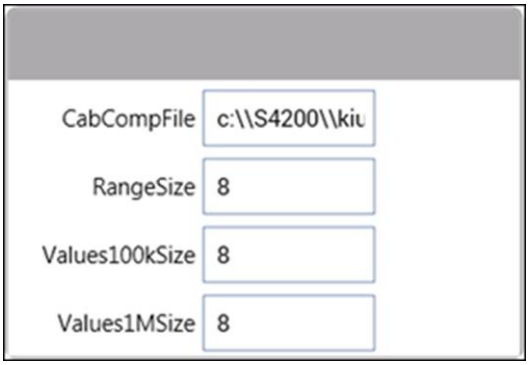


Place capacitance source values in a spreadsheet (DisplayCableCompCaps82)

To place capacitance source values in a spreadsheet:

1. In the project tree, select **display-cap-file** or **displaycablecomp**. The parameter list for the `DisplayCableCompCaps82` user module is shown in the figure below.

Figure 683: DisplayCableCompCaps82 user module



2. Ensure that the CabCompFile field has the same file directory path that is used in Enter and save capacitance source values (SaveCableCompCaps82) (on page D-12).
3. Set the other fields to **8**.
4. Click **Run**. The calibration source values entered into the action are placed into its spreadsheet.
5. Select **Analyze**. The sheet displays the values. An example spreadsheet is shown here.

Figure 684: display-cap-file spreadsheet with capacitor source values

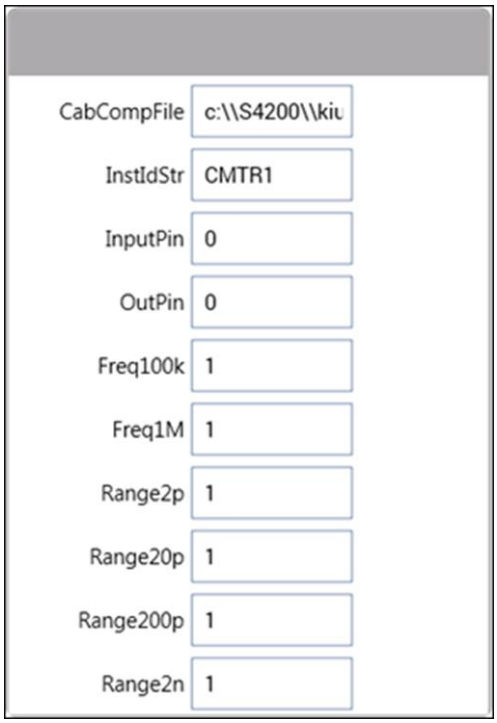
	A	B	C	D
1	DisplayCable	Range	Values100k	Values1M
2	0	2.00000E-12	4.77700E-13	4.77700E-13
3		2.00000E-12	1.46100E-12	1.46100E-12
4		2.00000E-11	4.79600E-12	4.79600E-12
5		2.00000E-11	1.78300E-11	1.78300E-11
6		2.00000E-10	4.66800E-11	4.66800E-11
7		2.00000E-10	1.81100E-10	1.81100E-10
8		2.00000E-09	4.75500E-10	4.75900E-10
9		2.00000E-09	1.77100E-09	1.77600E-09
10				

Compensate for cable capacitance (CableCompensate82)

To compensate for cable capacitance:

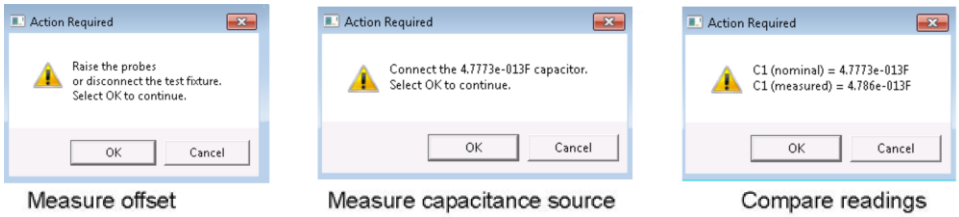
1. In the project tree, select **cable-compensate** or **cablecomp**.
2. Select **Configure**.
3. Ensure that the `CabCompFile` field of the parameter list has the same file directory path that is used in Enter and save capacitance source values (on page D-12).
4. Enable or disable cable compensation: Use the frequency and range fields to either disable (`0`) or enable (`1`) cable compensation for the test frequencies and ranges. The following figure shows cable compensation enabled for all ranges and test frequencies.

Figure 685: CableCompensate82 user module



5. Select **Run** to execute the test.
6. Follow the instructions on the dialog boxes, which will guide you through the cable compensation process. The three basic dialog boxes are shown below.
   - **Measure offset:** An open circuit measurement is required. Open the circuit as close to the DUT as possible.
   - **Measure capacitance source:** Connect a capacitance source in place of the DUT. Note that the value in the dialog box corresponds to a calibration value you entered in Enter and save capacitance source values (on page D-12). Connect the capacitance source as close to the DUT as possible.
   - **Compare readings:** Compares the measured value to the calibration (nominal) value you entered. The two readings should be fairly close. If they are not, the wrong capacitance source may have been connected or an open circuit condition occurred. In that case, click **Cancel** to abort the cable compensation process.

Figure 686: Cable compensation dialog boxes



NOTE

Clicking **Cancel** in a cable compensation dialog box aborts the cable compensation process. To start over, click **Run**.

Capacitance tests

The following topics describe the user modules that can be made into tests Clarius.

QTsweep (equilibrium test)

To achieve accurate simultaneous C-V test results, measurements must be made 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.

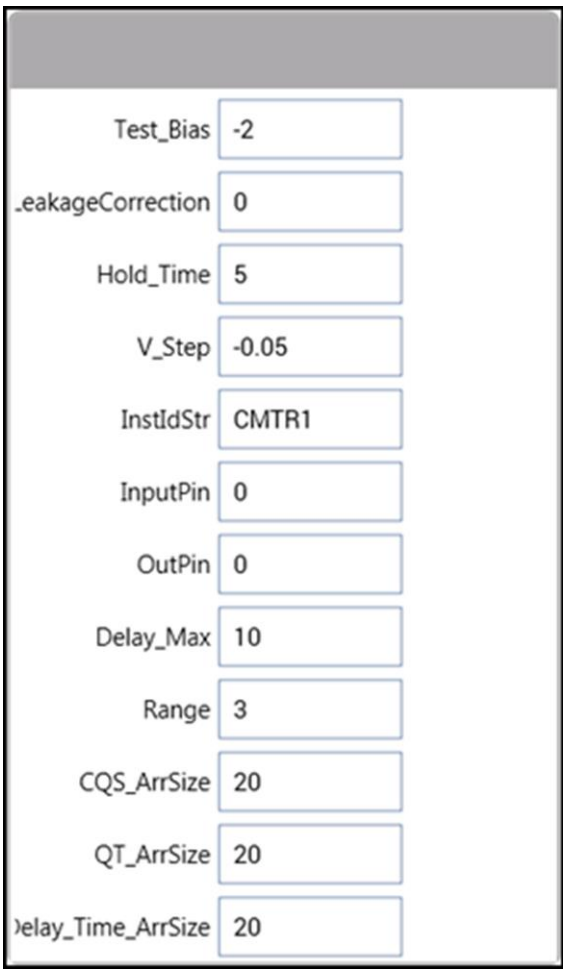
This example assumes that the Model 82 is connected directly to the DUT. The DUT could be a device installed in a test fixture, or a substrate on a wafer.

To open and execute the QTsweep UTM:

1. Choose **Select**.
2. In the Test Library, select Custom Test, **Choose a test from the pre-programmed library (UTM)**.
3. Drag the test to the project tree.
4. Select **Configure**.
5. In the Test Settings pane, under User Libraries, select **ki82ulib**.
6. Under User Modules, select **QTsweep82**.
7. Modify the test parameters as needed. Refer to QTsweep82 user module (on page D-47) for parameter definitions.

   If you use the parameters shown in the figure below, the Model 82 makes 20 quasistatic capacitance measurements using 20 mV pulses (`V_Step`) ranging from 0.07 seconds to 1 second (`Delay_Max`).
8. Click **Run** to execute the test.

Figure 687: QTsweep82 user module

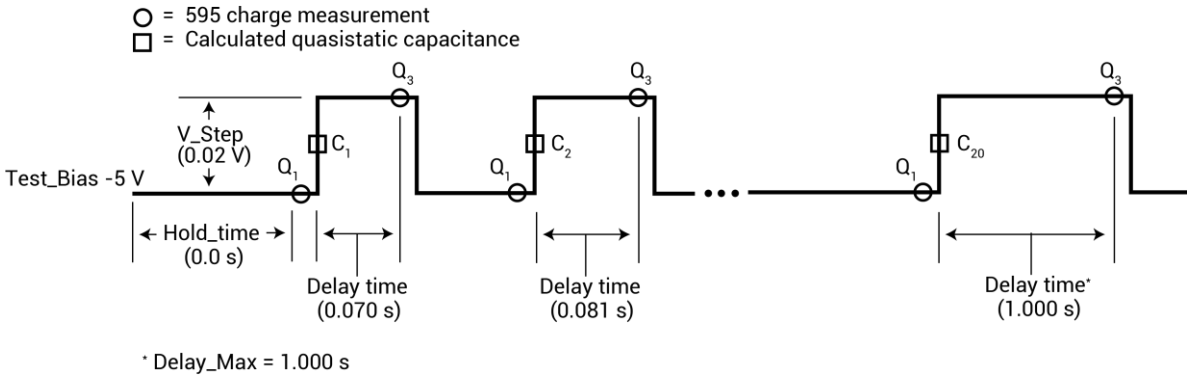


Equilibrium test (QTsweep) description

This test performs a quasistatic capacitance measurement (CQS) 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 = ΔQ/Δt).

The figure below shows the pulse stream for the equilibrium test using the parameters shown in QTsweep (equilibrium test) (on page D-16). 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 ms (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 688: Equilibrium test

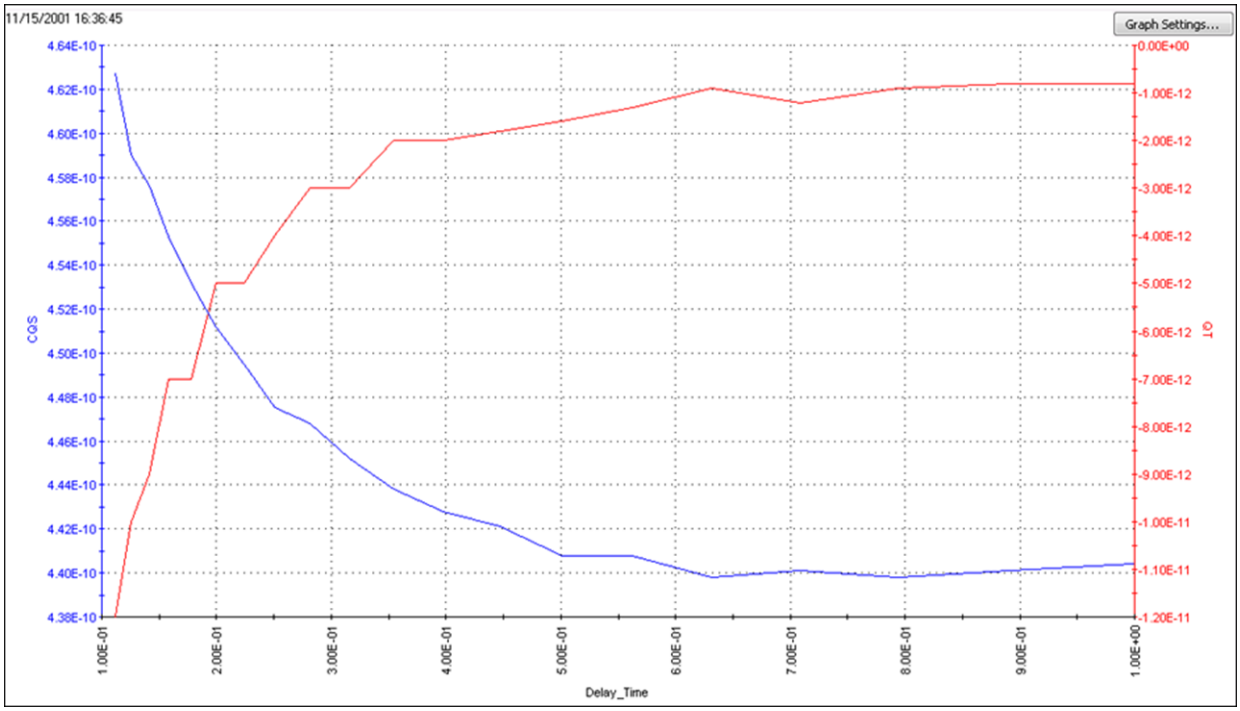


The generated graph for this test plots:

- Quasistatic capacitance (CQS) vs. delay time
- Leakage current (Q / t) vs. delay time

A typical graph for the equilibrium test is shown here. 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 689: Equilibrium test graph



Simultaneous C-V sweep

The Model 82 uses the Models 595 and 590 to perform simultaneous C-V measurements. Refer to Simultaneous C-V measurements (on page D-5) for detail on simultaneous C-V measurements.

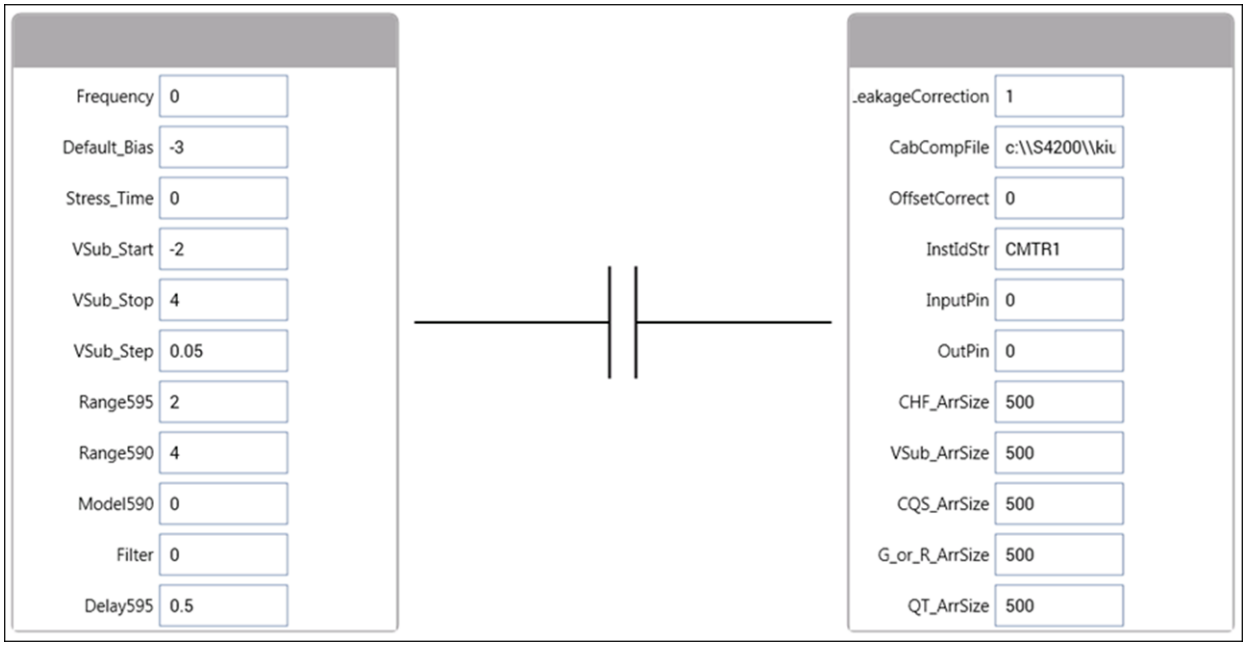
This example assumes that the Model 82 is connected directly to the DUT. The DUT could be a device installed in a test fixture, or a substrate on a wafer.

To do a simultaneous C-V sweep:

1. In the project tree, select **cvsweep**.
2. Select **Configure**.
3. Modify the test parameters as needed. Refer to SIMCVsweep82 user module (on page D-53) for definitions.
4. Select **Run**.

If you use the parameters shown in the figure below, the Model 82 performs a –3 C to +3 V staircase sweep using 20 mV steps, delaying 70 ms on each step.

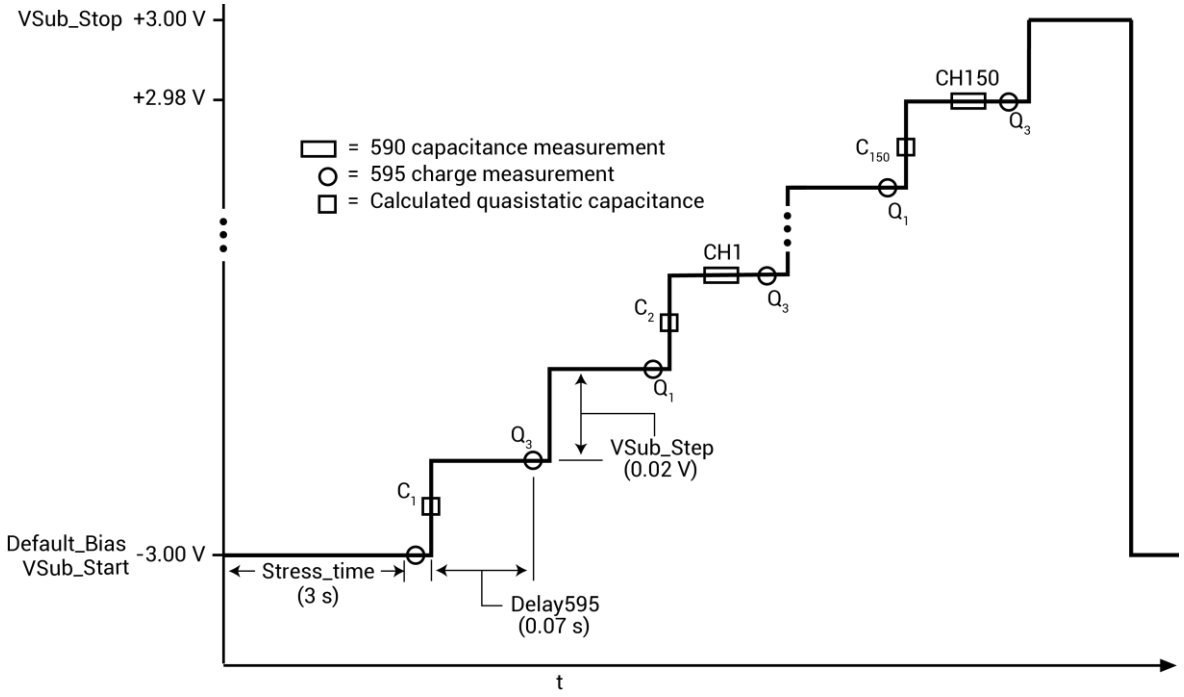
Figure 690: SIMCVsweep82 user module



cvsweep test description

As described in Simultaneous C-V sweep (on page D-20), the cvsweep UTM uses the `SIMCVsweep82` user module to make simultaneous C-V measurements. A 595 quasistatic measurement is a two-step process that requires at least two charge measurements. As shown in the figure below, charge measurements on two steps are made to yield a single quasistatic reading. The 590 makes a capacitance measurement on every second step of the staircase sweep.

Figure 691: Simultaneous C-V linear staircase sweep

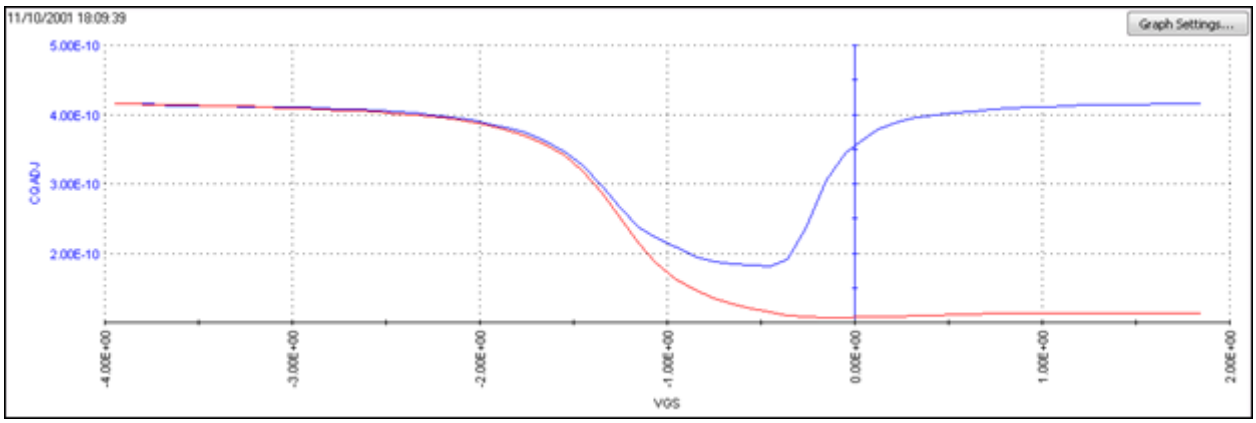


The graph for this test plots 590 capacitance (red trace) and 595 quasistatic capacitance (blue trace) versus bias voltage. The figure below shows a typical graph that is generated by this test.

NOTE

The shape of the curves in the following figure indicate that measurements were made with the device in equilibrium. If the curves for your test deviate significantly, the device was probably not in equilibrium. Do the equilibrium test (`QTsweep82`) to determine the optimum delay time (`Delay595` parameter) to use for the `simcv` test (`SIMCVsweep82` user module).

Figure 692: cvsweep graph



C-t sweep

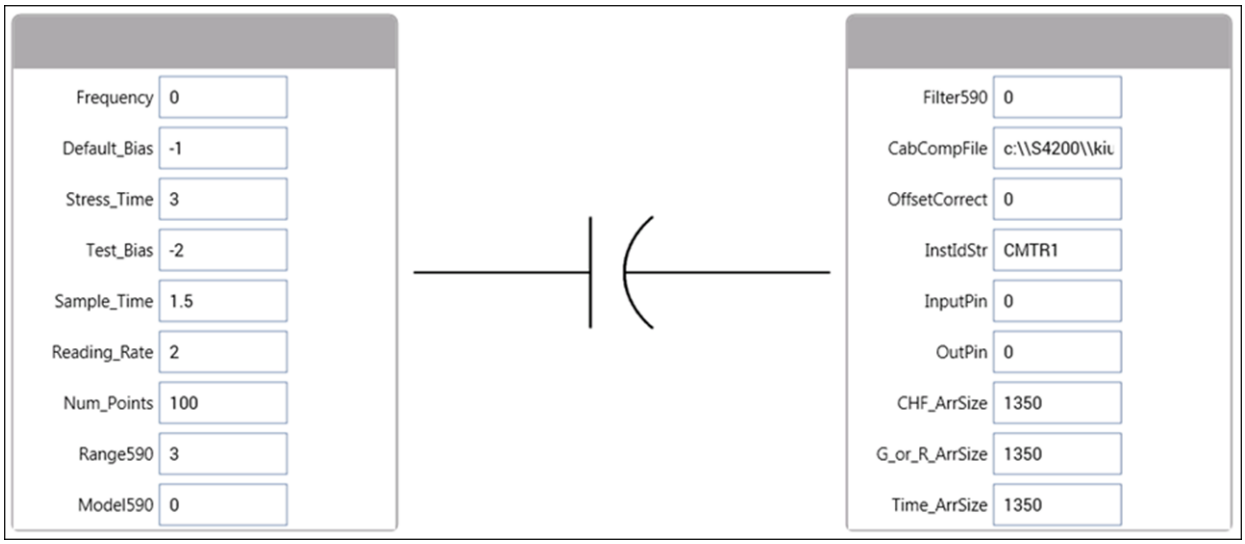
The Model 82 uses the Model 590 to make a specified number of capacitance measurements using a specified time interval between reading samples. The specified voltage bias is held constant for this test. Details on simultaneous C-t measurements are provided in C-t measurements (on page D-3).

This example assumes that the Model 82 is connected directly to the DUT. The DUT can be a device installed in a test fixture or a substrate on a wafer.

To perform a CtSweep:

1. In the project tree, select **ctsweep**.
2. Select **Configure**.
3. Modify the test parameters as needed. If you use the parameters shown in the figure below, the Model 82 makes 100 capacitance measurements.
4. Select **Run**.

Figure 693: CtSweep82 user module

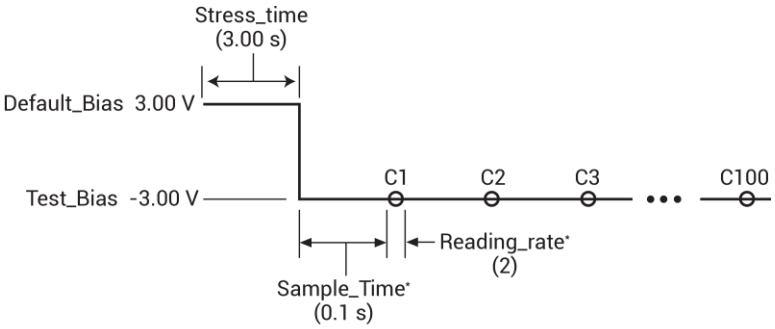


CtSweep test description

As shown in C-t sweep (on page D-23), the `CtSweep` UTM uses the **CTsweep82** user module to make C-t measurements. Refer to CtSweep82 user module (on page D-41) for definitions of the input parameters.

If using the parameters shown in C-t sweep (on page D-23), the 590 performs 100 capacitance measurements using a 100 ms sample time between reading samples. The time interval between reading samples is determined by the set sample time and the selected reading rate.

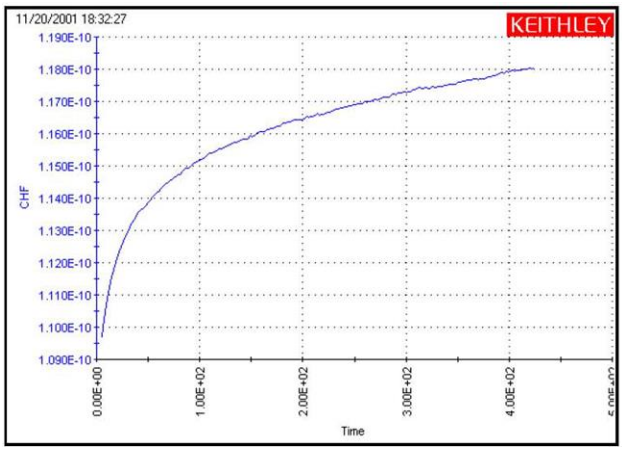
Figure 694: C-t measurements



*The time interval between reading samples is the sum of the set Sample Time and the Reading Rate time. The Reading Rate time for Reading_rate2 is 1/18 s (0.0555 s). Therefore:
Time interval = Sample Time + Reading Rate time
= 0.100 + 0.056
= 0.156 s

The graph below shows for this test plots capacitance versus time.

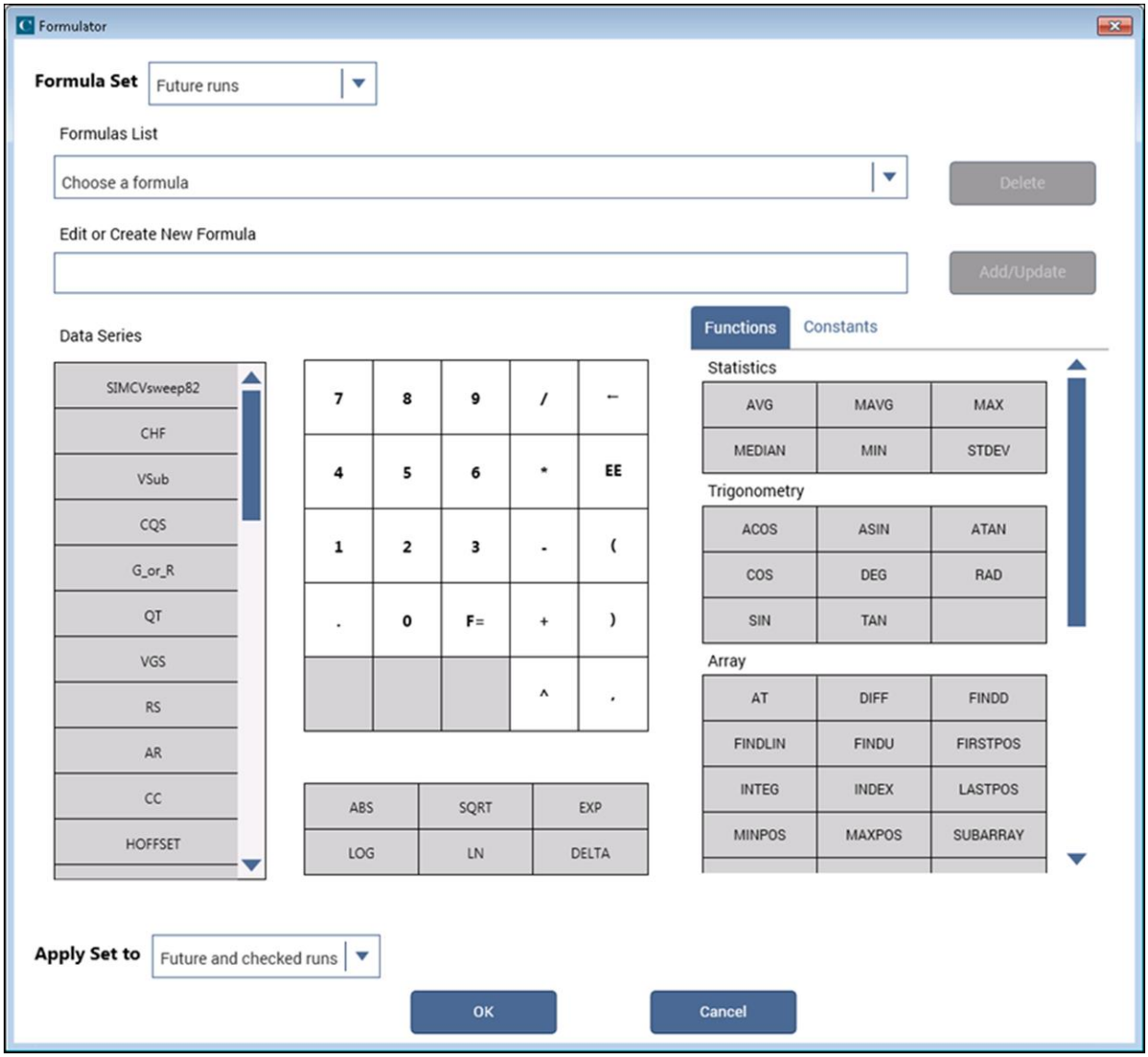
Figure 695: CtSweep graph



Formulas for capacitance tests

Formulas to calculate data for graphs are in the Formulator for each test. To open the Formulator dialog box, click **Formulator** in the Test Settings pane for the selected test. The following figure shows the Formulator for the `system82-cvsweep` test used in the `simcv` project.

Figure 696: Formulator for system82-cvsweep test (simcv project)



Formulas for the `system82-cvsweep` test (`simcv` project) are shown in Formulas for system82-cvsweep test (simcv project) (on page D-27).

Formulas for `ctsweep` test (`lifetime` project) are shown in Formulas for ctsweep test (lifetime project) (on page D-28)

Formulas for `cvsweep` test (`stvs` project) are shown in Formulas for cvsweep test (stvs project) (on page D-29).

The values for constants used in the formulas are in the Constants area in the Formulator. The constants include:

- Area, with a value of 0.012 mm^2
- eOX, with a value of 3.4e-013 F/cm
- eS, with a value of 1.04e-012 F/cm

NOTE

Refer to Simultaneous C-V analysis (on page D-57) for details on simultaneous C-V theory and the formulas.

Formulas for system82-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: 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))

Formula name	Description and formula
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

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

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

This section describes how to choose the correct parameters for:

- Simultaneous C-V measurement
- The delay time to ensure that the device remains in equilibrium in the inversion region during a sweep
- Controlling errors at the source

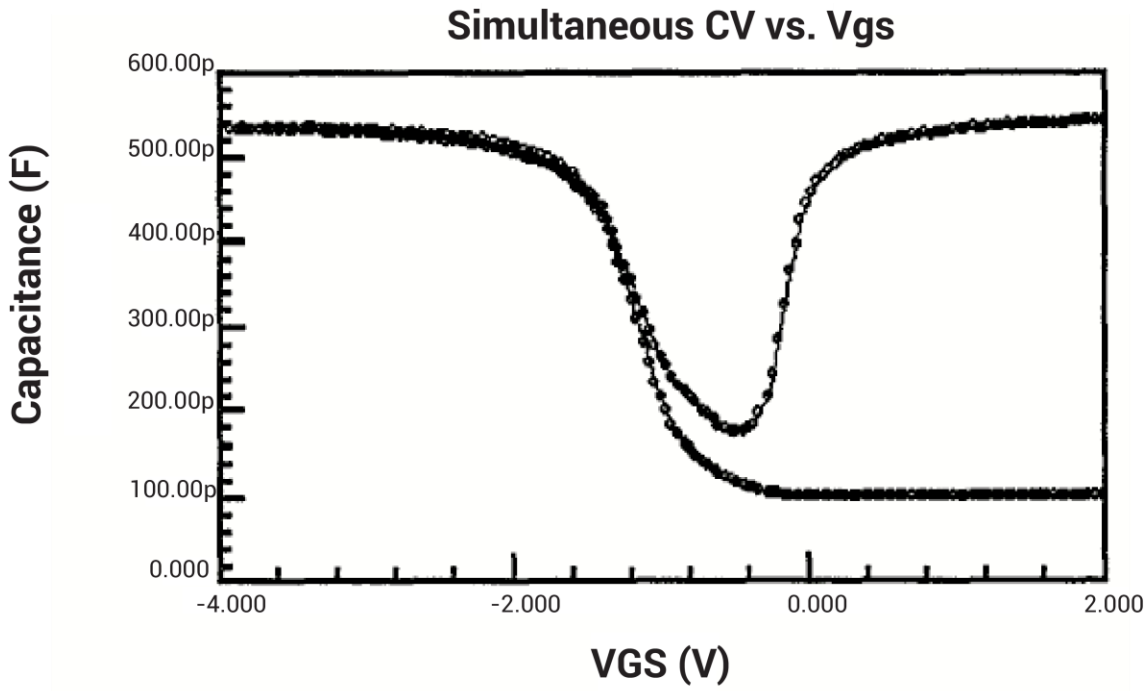
Optimal C-V measurement parameters

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.

Figure 697: Typical simultaneous C-V curve



The upper flat, or accumulation region of the high frequency C-V curve defines the oxide capacitance, COX. Since most analysis relies on the ratio C/COX, 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.

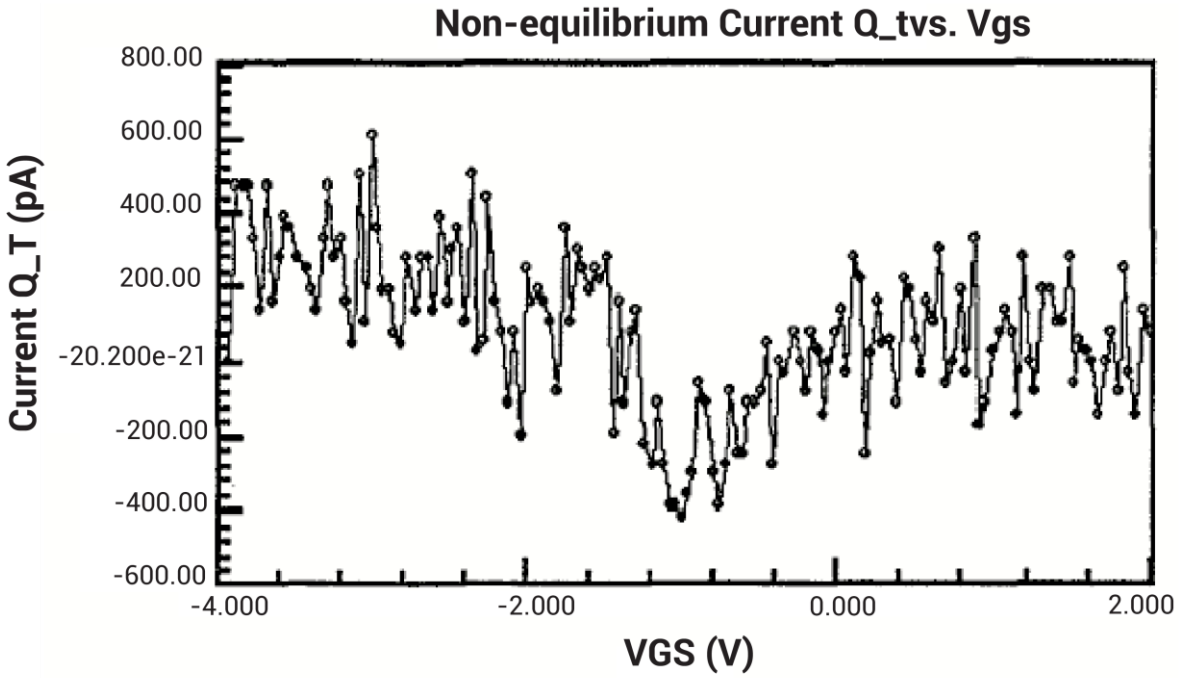
Sweep Direction

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, shown in the following figure, and also resulting in skewed C-V curves.

Figure 698: 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.

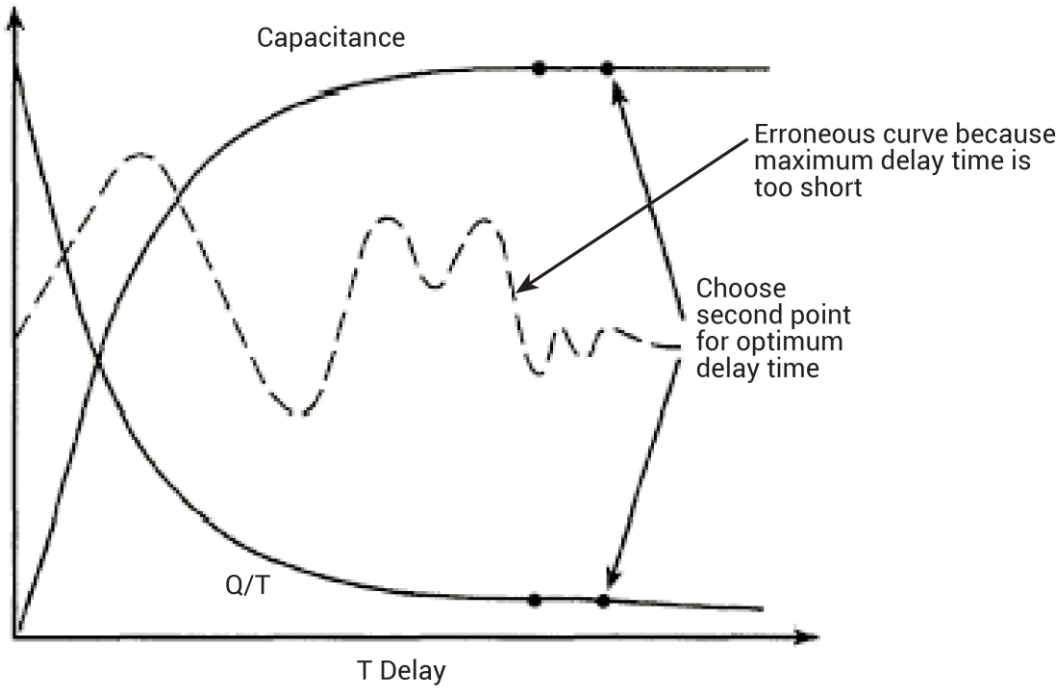
The equilibrium test uses the Model 595 to perform a series of quasistatic capacitance and Q/t current measurements using different delay times. The figure below shows the typical capacitance and Q/t curves generated for this test. As shown, the optimal delay is the second TDelay point after both curves 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 because it is difficult to quantify the amount of accuracy degradation in any given situation.

NOTE

See QTsweep (equilibrium test) (on page D-16) for details on the equilibrium test.

Figure 699: Choosing optimal delay time



Determining delay time with leaky devices

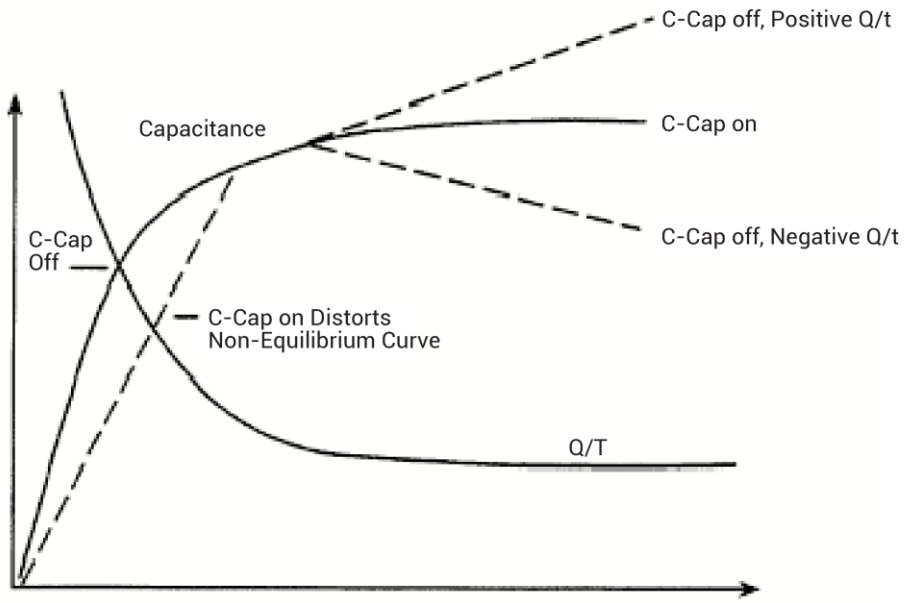
When testing for delay time on devices with relatively large leakage currents, it is recommended that you use the corrected capacitance feature, which is designed to compensate for leakage current. The reason for doing so is illustrated in the figure below. When large leakage currents are present, the 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 (on page D-53)).

Figure 700: Capacitance and leakage current curves of leaky device



Testing slow devices

A decaying noise curve, such as the dotted line shown in the figure in Determining the optimal delay time (on page D-33), 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. Methods of correcting 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 before 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 (on page D-53)).

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 (on page D-53)). 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 CQ to dip below CH resulting in erroneous DIT calculations. If this situation occurs, turn off the filter or add more data points.

ki82ulib user library

The user modules in the `ki82ulib` user library control the Model 82 C-V System. They perform simultaneous C-V, C-t, and Q/t measurements and cable compensation. The next table lists the user modules. It also provides the name of tests and actions in Clarius that are based on these user modules.

ki82ulib user modules

User module	Test and action names	Description
Abortmodule82 (on page D-37)	n/a	Puts the three System 82 instruments into a known state when a test is aborted. This function is used by other library modules in the `atexit()` function.
CableCompensate82 (on page D-38)	`cable-compensate` `cablecomp`	Performs cable compensation using known capacitance source values.
CTsweep82 (on page D-41)	`ctsweep`	Performs C-t measurements.
DisplayCableCompCaps82 (on page D-45)	`display-cap-file`	Places capacitance source values in a spreadsheet.
`LoadCableCorrectionConstants82`	n/a	Read the cable compensation parameters and sends them to the 590. This module is for internal use by the `SIMCVsweep82` and `CTsweep82` modules. It is not normally used as a stand-alone module.
QTsweep82 (on page D-47)	`qtsweep`	Performs quasistatic measurement sweep.
SaveCableCompCaps82 (on page D-50)	`save-cap-file` `savecablecompfile`	Saves entered capacitance source values in a file.
SIMCVsweep82 (on page D-53)	`system82-cvsweep` `cvsweep`	Performs simultaneous C-V sweep.

Abortmodule82

The `Abortmodule82()` function puts the three System 82 instruments into a known state when a test is aborted. This function is used by other library modules in the `atexit()` function.

Usage

```
atexit(Abortcleanup);
```

CableCompensate82 user module

The `CableCompensate82` routine performs 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.

Usage

```
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);
```

status	Returned values; see **Details**
CabCompFile	The complete name and path for the cable compensation file; see **Details**
InstIdStr	KCon instrument ID; default is `CMTR1`; can be `CMTR1` to `CMTR4`, depending on your system configuration
InputPin	The DUT pin to which the 5951 input terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connections are made; see **Details**
OutPin	The DUT pin to which the 5951 output terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connections are made; see **Details**
Freq100 k	Use compensation for the 100 kHz frequency: ■ Skip compensation for this frequency: `0` ■ Do compensation for this frequency: `1`
Freq1M	Use compensation for the 1 MHz frequency: ■ Skip compensation for this frequency: `0` ■ Do compensation for this frequency: `1`
Range2p	Use compensation for the 2 pF range: ■ Skip compensation for this range: `0` ■ Do compensation for this range: `1`
Range20 p	Use compensation for the 20 pF range: ■ Skip compensation for this range: `0` ■ Do compensation for this range: `1`
Range200 p	Use compensation for the 200 pF range: ■ Skip compensation for this range: `0` ■ Do compensation for this range: `1`
range2n	Use compensation for the 2 nF range: ■ Skip compensation for this range: `0` ■ Do compensation for this range: `1`

Details

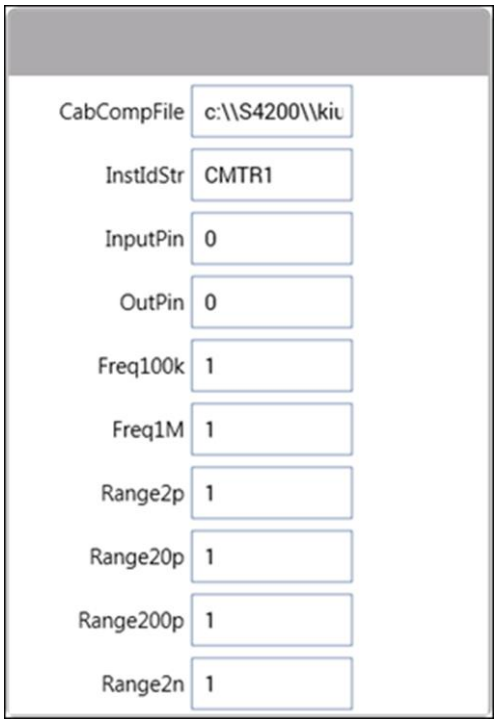
This user module, shown below, is used to do cable compensation for the selected ranges and test frequencies of the 590. For the input parameters shown in the figure, cable compensation for the 590 is done for the 2 pF, 20 pF, 200 pF, and 2 nF ranges and for both the 100 kHz and 1 MHz test frequencies. 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 (on page D-50).

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 Cable compensation tests (on page D-11).

Figure 701: CableCompensate82 user module



The return values from *status* can be:

- 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 configuration
- -10090 (GPIB_ERROR_OCCURRED): A GPIB communications error occurred
- -10100 (INVAL_PARAM): An invalid input parameter is specified

If *CabCompFile* does not exist, or if there is no path specified (null string), the default compensation parameters are used. When entering the path, be sure to use two \ characters to separate each directory. For example, if your cable file is in:

```
C:\calfiles\82cal.dat
```

You would enter:

```
C:\\calfiles\\82cal.dat
```

If you are controlling a switch matrix to route signals using a connection UTM (for example, “connect”), you do not need connect *InputPin* and *OutputPin*. Set these parameters to 0.

Procedure

For each range and test frequency specified by the input parameters:

1. You are prompted to open the circuit so that an offset capacitance measurement can be made.
2. Once the offset capacitance measurement is completed, you are prompted to connect the low value capacitor for the selected range. The system performs the low capacitor compensation.
3. You are prompted to connect the high value capacitor for the selected range. The system does the high value capacitor compensation.
4. You are prompted to reconnect the low capacitor.
5. The nominal and measured values are displayed in a dialog box.
6. Verify the values. If you are unsatisfied with the measurement, select Cancel to abort the procedure. If you select Cancel, the cable compensation file is not affected.
7. When all selected ranges and frequencies have been compensated successfully, the cable compensation values are saved.

Also see

None

CtSweep82 user module

The CtSweep82 user module measures capacitance as a function of time at a certain bias.

Usage

```
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 *instr_id, int
   InputPin, int OutPin, double *CHF, int CHF_ArrSize, double *G_or_R, int
   G_or_R_ArrSize, double *Time, int Time_ArrSize);
```

status	Returned values are placed in the Analyze sheet; see **Details**
Frequency	The measurement frequency: ■ 100 kHz: `0` ■ 1 MHz: `1`
Default_Bias	DC bias applied before and after a C-t sweep (–20 V to +20 V)
Stress_Time	Duration of the default bias before test bias is applied (0.001 s to 65 s)
Test_Bias	Voltage bias for capacitance measurements (–20 V to +20 V)
Sample_Time	Time delay between each sampling measurement (0.001 s to 65 s)
Reading_rate	The reading rate used to acquire the measurements (1 to 4; see **Details**)
Num_Points	Number of sampling points (1 to 1350)
Range590	The measurement range for the 590 (1 to 4; see **Details** for valid range values)
Model590	The measurement model to use for high frequency measurement: ■ Parallel mode: `0` ■ Series model: `1`
Filter590	Enable or disable the analog filter; see **Details**: ■ Disable the filter: `0` ■ Enable the filter: `1`
CabCompFile	The complete name and path for the cable compensation file; see **Details**
OffsetCorrect	Enable or disable an offset correction measurement: ■ Disable offset correction: `0` ■ Enable offset correction: `1`
instr_id	KCon instrument ID; default is `CMTR1`; can be `CMTR1` to `CMTR4`, depending on your system configuration
InputPin	The DUT pin to which the 5951 input terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connections are made; see **Details**
OutPin	The DUT pin to which the 5951 output terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connections are made; see **Details**
CHF	Output; the measured array of high frequency capacitance values
CHF_ArrSize	Set to 1350
G_or_R	Output; the array of measured conductance (G) or resistance (R) values
G_or_R_ArrSize	Set to 1350
Time	Output; the array of time from the 595 output for each measurement step

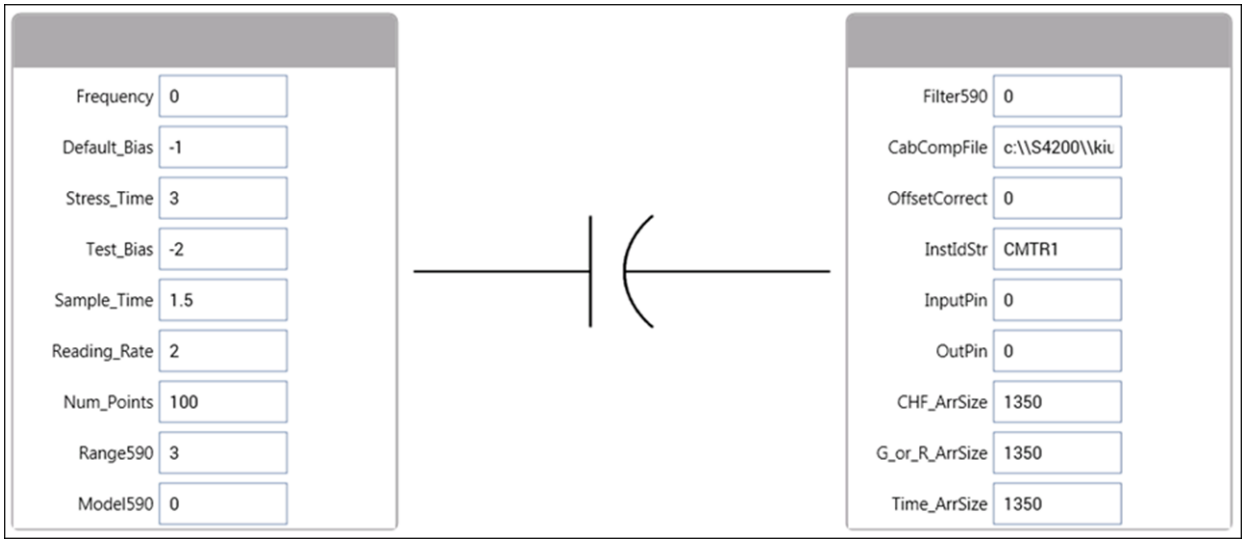
Time_ArrSize	Set to 1350

Details

This method can be used for minority carrier lifetime measurements using Zerbst plot.

The figure below 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 +3 V for three seconds, and then perform 100 capacitance measurements at –3 V using a 0.1 s time interval (see CtSweep test description (on page D-24)). For details on C-t measurements, refer to C-t sweep (on page D-23).

Figure 702: CtSweep82 user module



The analog filter, enabled with *Filter590*, can minimize the amount of noise that appears in the readings. It does, however, increase the measurement time.

Reading_rate **valid inputs**

Reading rate	Nominal reading rate (per second)	Readings	Display resolution (digits)
1	75	C,G,V	3.5
2	18	C,G,V	4.5
3	10	C,G,V	4.5
4	1	C,G,V	4.5

Range590 **valid range values**

Range	100 kHz	1 MHz
1	2 pF / 2 µs	20 pF / 200 µs
2	20 pF / 20 µs	20 pF / 200 µs
3	200 pF / 200 µs	200 pF / 2 ms
4	2 nF / 2 ms	2 nF / 20 ms

If *CabCompFile* does not exist, or if there is no path specified (null string), the default compensation parameters are used. When entering the path, be sure to use two \ characters to separate each directory. For example, if your cable file is in:

```
C:\calfiles\82cal.dat
```

You would enter:

```
C:\\calfiles\\82cal.dat
```

If you are controlling a switch matrix to route signals using a connection UTM (for example, “connect”), you do not need connect *InputPin* and *OutputPin*. Set these parameters to 0.

The return values from *status* can be:

- 0: OK.
- -10000 (INVAL_INST_ID): The specified instrument ID does not exist (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 configuration
- -10023 (KI590_MEAS_ERROR): A measurement error occurred
- -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 response of the 590.
- -10104 (USER_CANCEL): The user canceled the correction procedure.
- -10090 (GPIB_ERROR_OCCURRED): A GPIB communications error occurred
- -10100 (INVAL_PARAM): An invalid input parameter is specified

Procedure

1. If set, you are 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 is used.
3. A C-t sweep is performed.

Also see

None

DisplayCableCompCaps82 user module

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 spreadsheet for viewing.

Usage

```
status = DisplayCableCompCaps82(char *CabCompFile, double *Range, int RangeSize, double
    *Values100 k, int Values100 kSize, double *Values1M, int Values1MSize);
```

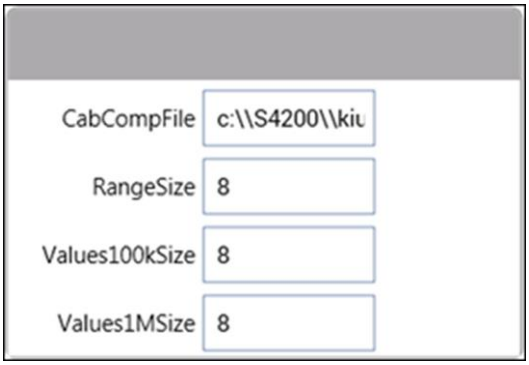
status	Returned values are placed in the Analyze sheet; see **Details**
CabCompFile	The complete name and path for the cable compensation file; see **Details**
Range	Output; an 8-element array that receives the nominal range values
RangeSize	The size of the *Range* array; set to 8
Values100k	Output; an 8-element (fixed) array that receives the nominal capacitor values used for the cable compensation at the 100 kHz frequency
Values100kSize	The size of the *Values100k* array; set to 8
Values1M	Output; an 8-element (fixed) array that receives the nominal capacitor values used for the cable compensation at the 1 MHz frequency
Values1MSize	The size of the *Values1M* array; set to 8

Details

The `DisplayCableCompCaps82` user module reads the nominal cable compensation values that are stored in the compensation file and returns them to the calling function. In the case of Clarius, it returns the values to the UTM data sheet.

The default parameters for this user module are shown in the following figure. 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.

Figure 703: DisplayCableCompCaps82 user module



To prevent unpredictable results, the array size values for the `RangeSize`, `Values100kSize`, and `Values1MSize` arrays must be set to 8.

NOTE

For details on the procedure to perform cable compensation, refer to Cable compensation tests (on page D-11).

The returned arrays are arranged in the order shown in the following table.

Reading_rate **valid inputs**

Range	100 kHz values	1 MHz values
2E-12	2 pF low comp value	2 pF low comp value
2E-12	2 pF high comp value	2 pF high comp value
20E-12	20 pF low comp value	20 pF low comp value
20E-12	20 pF high comp value	20 pF high comp value
200E-12	200 pF low comp value	200 pF low comp value
200E-12	200 pF high comp value	200 pF high comp value
2E-9	2 nF low comp value	2 nF low comp value
2E-9	2 nF high comp value	2 nF high comp value

If *CabCompFile* does not exist, or if there is no path specified (null string), the default compensation parameters are used. When entering the path, be sure to use two \ characters to separate each directory. For example, if your cable file is in:

```
C:\calfiles\82cal.dat
```

You would enter:

```
C:\\calfiles\\82cal.dat
```

The return values from *status* can be:

- `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 configuration
- `-10090 (GPIB_ERROR_OCCURRED)`: A GPIB communications error occurred
- `-10100 (INVAL_PARAM)`: An invalid input parameter is specified

Also see

SaveCableCompCaps82 user module (on page D-50)

QTsweep82 user module

This user module uses the 595 to determine the equilibrium point for a device by measuring quasistatic capacitance using different delay times.

Usage

```
status = QTsweep82(double Test_Bias, int LeakageCorrection, double Hold_time, double
    V-Step, char *InstldStr, 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);
```

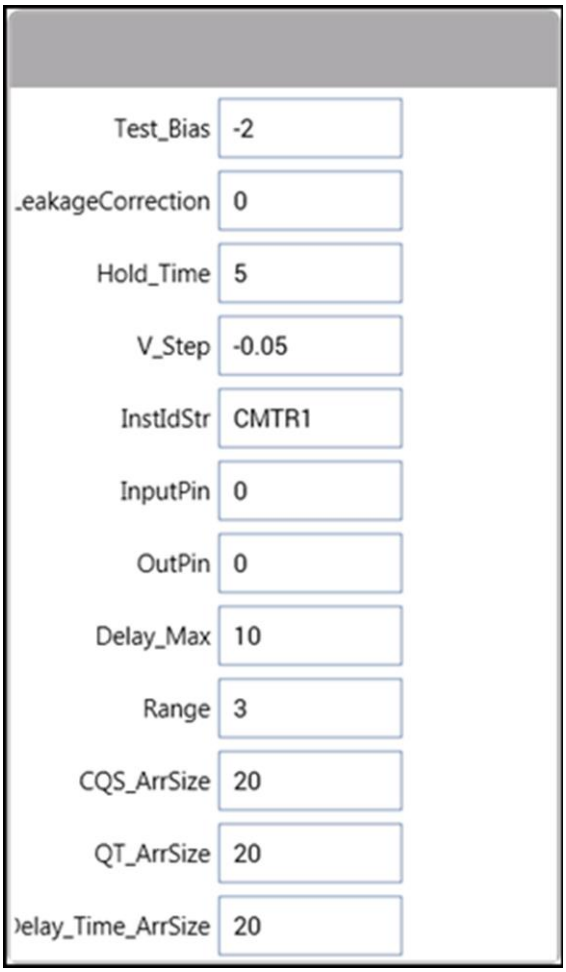
status	Returned values are placed in the Analyze sheet; see **Details**
Test_Bias	Voltage bias for capacitance measurements (–120 V to +120 V)
LeakageCorrection	Disable: `0` Enable: `1`
Hold_Time	Hold time at the beginning of the sweep (0 s to 200 s; default 5)
V-Step	Step voltage size: ±0 V, ±0.01 V, ±0.02 V, ±0.05 V, ±0.1 V
InstldStr	KCon instrument ID; default is `CMTR1`; can be `CMTR1` to `CMTR4`, depending on your system configuration
InputPin	The DUT pin to which the 5951 input terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connections are made
OutPin	The DUT pin to which the 5951 output terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connections are made
Delay_Max	Maximum delay time: 1 s to 199.99 s (default 10 s)
Range	The measurement range for the 595 to use: 1 to 3; see **Details**
CQS	Output; the measured array of quasistatic capacitance values
CQS_ArrSize	Set to 20
QT	Output; the measured array of leakage current Q/T
QT_ArrSize	Set to 20
Delay_time	Output; the array of *Delay_Time* used up to *Delay_Max* in logarithm scale
Delay_time_ArrSize	Set to 20

Details

The module measures quasistatic capacitance and leakage current as a function of delay time using the 595. It is used to determine the equilibrium condition. 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 ($i = \Delta Q/\Delta t$).

The following figure shows the default parameters for the `QTsweep82` user module.

Figure 704: QTsweep82 user module



The Q/T sweep in Equilibrium test (QTsweep) description (on page D-18) acquires 20 quasistatic capacitance readings. After the graph for quasistatic capacitance and leakage current versus time is plotted, the optimum delay time for equilibrium can be determined.

NOTE

For details on quasistatic measurements, see QTsweep (on page D-16). For details on C-t measurements, see C-t sweep (on page D-23).

The Range values are shown in the following table.

Range **values**

Value	595 range
1	200 pF
2	2 nF
3	20 nF

The return values from *status* can be:

- 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 configuration
- -10090 (GPIB_ERROR_OCCURRED): A GPIB communications error occurred
- -10091 (GPIB_TIMEOUT): A timeout occurred during communications
- -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
- -10100 (INVAL_PARAM): An invalid input parameter is 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.

Procedure

1. If set, you are 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 is used.
3. A Q/T sweep is performed.

Also see

None

SaveCableCompCaps82 user module

The user modules saves the nominal values of the capacitors used with the 590 cable compensation procedure to a file.

Usage

```
status = SaveCableCompCaps82(char *CabCompFile, double Lo2p100k, double Lo2p1M, double
    Hi2p100k, double Hi2p1M, double Lo20p100k, double Lo20p1M, double Hi20p100k,
    double Hi20p1M, double Lo200p100k, double Lo200p1M, double Hi200p100k, double
    200p1M, double Lo2n100k, double Lo2n1M, double Hi2n100k, double Lo2n1M);
```

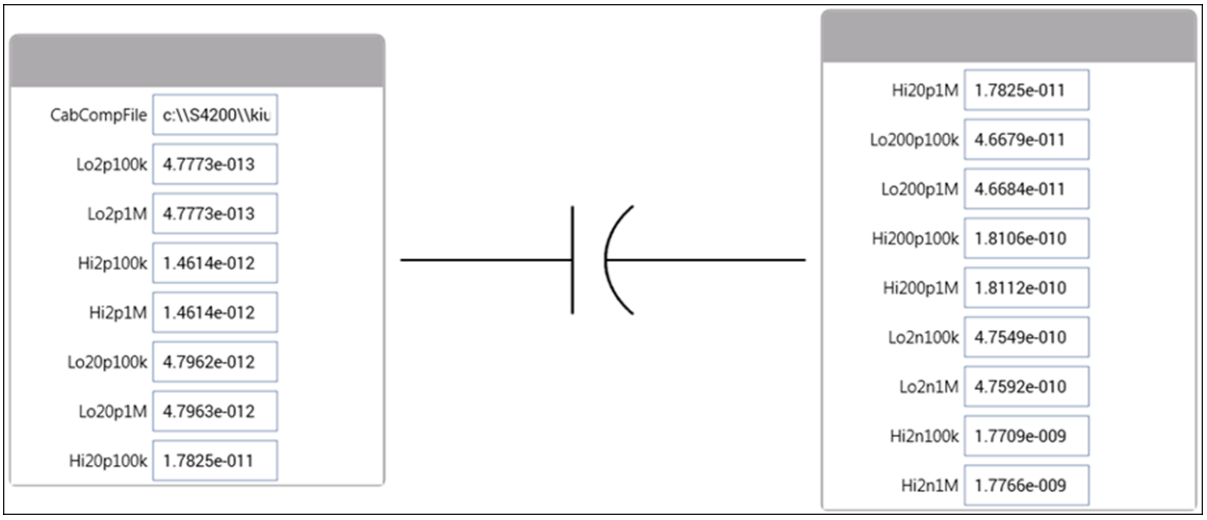
CabCompFile	The complete name and path for the cable compensation file; see **Details**
Lo2p100k	The nominal value of the low range capacitor used to perform cable compensation for the 2 pF range and 100 kHz frequency: 0 F to 0.95E-12 F
Lo2p1M	The nominal value of the low range capacitor used to perform cable compensation for the 2 pF range and 1 MHz frequency: 0 F to 0.95E-12 F
Hi2p100k	The nominal value of the high range capacitor used to perform cable compensation for the 2 pF range and 100 kHz frequency: 1E-12 F to 2E-12 F
Hi2p1M	The nominal value of the high range capacitor used to perform cable compensation for the 2 pF range and 1 MHz frequency: 1E-12 F to 2E-12 F
Lo20p100k	The nominal value of the low range capacitor used to perform cable compensation for the 20 pF range and 100 kHz frequency: 0 F to 9.5E-12 F
Lo20p1M	The nominal value of the low range capacitor used to perform cable compensation for the 20 pF range and 1 MHz frequency: 0 F to 9.5E-12 F
Hi20p100k	The nominal value of the high range capacitor used to perform cable compensation for the 20 pF range and 100 kHz frequency: 10E-12 F to 20E-12 F
Hi20p1M	The nominal value of the high range capacitor used to perform cable compensation for the 20 pF range and 1 MHz frequency: 10E-12 F to 20E-12 F
Lo200p100k	The nominal value of the low range capacitor used to perform cable compensation for the 200 pF range and 100 kHz frequency: 0 F to 95E-12 F
Lo200p1M	The nominal value of the low range capacitor used to perform cable compensation for the 200 pF range and 1 MHz frequency: 0 F to 95E-12 F
Hi200p100k	The nominal value of the high range capacitor used to perform cable compensation for the 200 pF range and 100 kHz frequency: 100E-12 F to 200E-12 F
Hi200p1M	The nominal value of the high range capacitor used to perform cable compensation for the 200 pF range and 1 MHz frequency: 100E-12 F to 200E-12 F
Lo2n100k	The nominal value of the low range capacitor used to perform cable compensation for the 2 nF range and 100 kHz frequency: 0 F to 995E-12 F
Lo2n1M	The nominal value of the low range capacitor used to perform cable compensation for the 2 nF range and 1 MHz frequency: 0 F to 995E-12 F
Hi2n100k	The nominal value of the high range capacitor used to perform cable compensation for the 2 nF range and 100 kHz frequency: 1000E-12 F to 2000E-12 F
Hi2n1M	The nominal value of the high range capacitor used to perform cable compensation for the 2 nF range and 1 MHz frequency: 1000E-12 F to 2000E-12 F

Details

This user module is used for 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. If no cable compensation file exists, this module creates one. The user module to perform cable compensation (CableCompensate82) can then access the capacitance source values from this file. The user must have the proper system permissions in order for this user module to create a file.

The default parameter values for this user module are shown in the following figure. These are example low and high values that can be used for cable compensation. You must replace these values with the calibration values of the actual capacitance sources.

Figure 705: SaveCableCompCaps82 user module



NOTE

For details on the procedure to perform cable compensation, see Cable compensation tests (on page D-11).

The return values from *status* can be:

- 0: OK.
- -10000 (INVAL_INST_ID): The specified instrument ID does not exist
- -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 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 configuration
- -10090 (GPIB_ERROR_OCCURRED): A GPIB communications error occurred
- -10100 (INVAL_PARAM): An invalid input parameter is specified

If *CabCompFile* does not exist, or if there is no path specified (null string), the default compensation parameters are used. When entering the path, be sure to use two \ characters to separate each directory. For example, if your cable file is in:

```
C:\calfiles\590cal.dat
```

You would enter:

```
C:\\calfiles\\590cal.dat
```

If you are controlling a switch matrix to route signals using a connection UTM (for example, “connect”), you do not need connect *InputPin* and *OutputPin*. Set these parameters to 0.

Also see

None

SIMCVsweep82 user module

The `SIMCVsweep82` routine performs a simultaneous capacitance versus voltage (C-V) sweep using the Keithley Instruments 82 C-V System.

Usage

```
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 *InstldStr, 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);
```

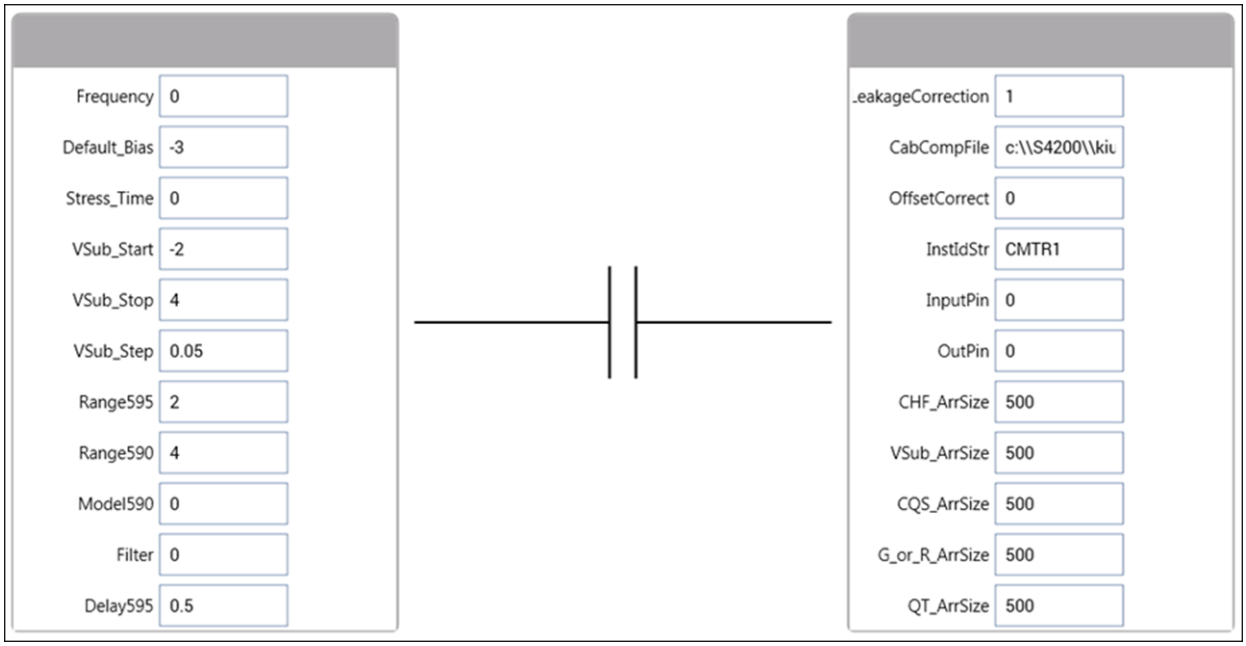
status	Returned values are placed in the Analyze sheet; see **Details**
Frequency	The measurement frequency to use for the 590: ■ 100 kHz: `0` ■ 1 MHz: `1`
Default_Bias	DC bias applied before and after a voltage sweep (–100 V to +100 V)
Stress_Time	Time for which default bias is stressed on the device before voltage sweep: 0 s to 999 s
VSub_Start	Start voltage on substrate: –120 V to +120 V
VSub_Stop	Stop voltage on substrate: –120 V to +120 V
VSub_Step	Voltage step size: ±0 V , ±0.01 V, ±0.02 V, ±0.05 V, or ±0.1 V
Range595	The measurement range for the 595 to use: ■ 200 pF: `1` ■ 2 nF: `2` ■ 20 nF: `3`
Range590	The measurement range for the 590 to use: `1` to `4`; refer to **Details**
Model590	The measurement model to use for high frequency measurement: ■ Parallel mode: `0` ■ Series model: `1`
Filter	Enable or disable the digital filter: ■ 1 reading: `0` ■ 3 readings: `1` ■ 9 readings: `2` ■ 24 readings: `3`
Delay595	Delay time for 595; maximum 199 s (default 0.07 s)
LeakageCorrection	Enable or disable the leakage current correction of the 595: ■ Disable: `0` ■ Enable: `1`
CabCompFile	The complete name and path for the cable compensation file; see **Details**

OffsetCorrect	Enable or disable an offset correction measurement: ■ Disable offset correction: `0` ■ Enable offset correction: `1`
InstIdStr	CMTR 82 instrument ID; default is `CMTR1`; can be `CMTR1` to `CMTR4`, depending on your system configuration
InputPin	The DUT pin to which the 5951 input terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connections are made
OutPin	The DUT pin to which the 5951 output terminal is attached (–1 to 72); if a value of less than 1 is specified, no switch matrix connections are made
CHF	Output; the measured array of high frequency capacitance values
CHF_ArrSize	This must be set to a value equal to the number of voltage steps in the sweep or value = ((VSub_Stop – VSub_Start) / VSub_Step – 1)
VSub	Output; the array of bias voltages used
VSub_ArrSize	This must be set to a value equal to the number of voltage steps in the sweep or value = ((VSub_Stop – VSub_Start) / VSub_Step – 1)
CQS	Output; the measured array of quasistatic capacitance values
CQS_ArrSize	This must be set to a value equal to the number of voltage steps in the sweep or value = ((VSub_Stop – VSub_Start) / VSub_Step – 1)
G_or_R	Output; the array of measured conductance (G) or resistance (R) values
G_or_R_ArrSize	This must be set to a value equal to the number of voltage steps in the sweep or value = ((VSub_Stop – VSub_Start) / VSub_Step – 1)
QT	Output; the array of Q/T from 595 output for each measurement step
QT_ArrSize	This must be set to a value equal to the number of voltage steps in the sweep or value = ((VSub_Stop – VSub_Start) / VSub_Step – 1)

Details

This user module uses the 590 and 595 to perform simultaneous C-V measurements. The following figure shows the default parameters for the `SIMCVsweep82` user module.

Figure 706: SIMCVsweep82 user module



It performs a staircase sweep from –3 V to +3 V in 20 mV steps, as shown in cvsweep test description (on page D-21).

You can make an offset correction measurement and use the cable compensation.

NOTE

For details on quasistatic measurements, see Simultaneous C-V sweep (on page D-20).

The following table lists the valid range values for *`Range590`*.

`Range590` **valid range values**

Range	100 kHz	1 MHz
1	2 pF / 2 µs	20 pF / 200 µs
2	20 pF / 20 µs	20 pF / 200 µs
3	200 pF / 200 µs	200 pF / 2 ms
4	2 nF / 2 ms	2 nF / 20 ms

If *CabCompFile* does not exist, or if there is no path specified (null string), the default compensation parameters are used. When entering the path, be sure to use two \ characters to separate each directory. For example, if your cable file is in:

```
C:\calfiles\590cal.dat
```

You would enter:

```
C:\\calfiles\\590cal.dat
```

The return values from *status* can be:

- 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_OCCURRED): A GPIB communications error occurred
- -10091 (GPIB_TIMEOUT): A timeout occurred during communications
- -10100 (INVAL_PARAM): An invalid input parameter is 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 590 response
- -10104 (USER_CANCEL): The user canceled the correction procedure
- -10045 (KI82_NOT_IN_KCON): KI82 is not in KCon

Procedure

1. If set, you are 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 is used.
3. A simultaneous C-V sweep is made.

Also see

None

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 and bibliography of C-V measurements (on page D-78).

Analysis methods

The following figures show fundamental C-V curves for p-type and n-type materials. 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 shown below.

Figure 707: C-V characteristics of p-type material

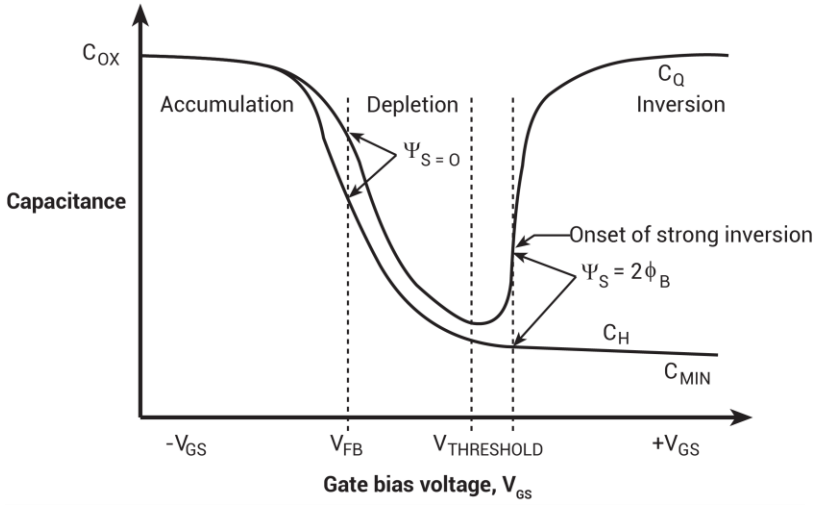
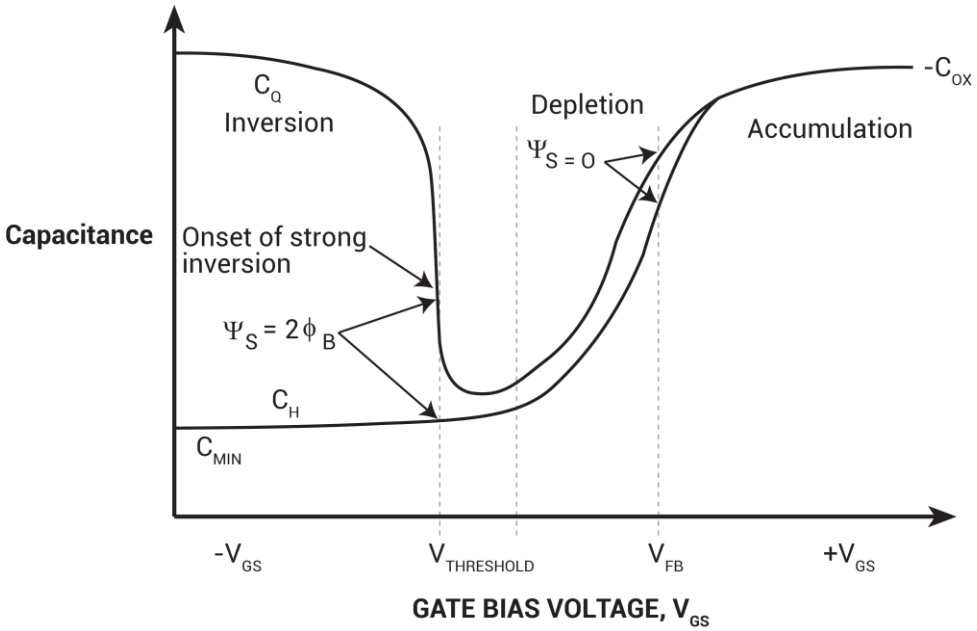


Figure 708: C-V characteristics of n-type material



Basic device parameters

The following topics provide additional detail on device parameters and how they are calculated.

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 Analysis methods (on page D-58)). 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 V_{GS} when poitive, the substrate material is p-type.
- If C_H is greater with positive V_{GS} than 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:

$$t_{ox} = \frac{A\,\varepsilon_{ox}}{(1\times10^{-19})C_{ox}}$$

Where:

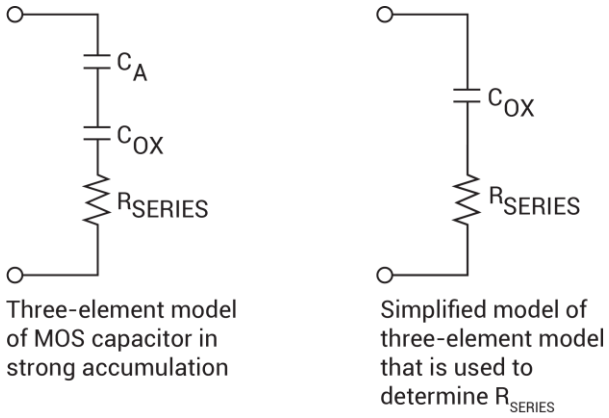
- t_{ox} = oxide thickness (nm)
- A = gate area (cm^2)
- ε_{ox} = permittivity of oxide material (F/cm)
- C_{ox} = oxide capacitance (pF)

You can rearrange the above equation 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.

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 the simplified model below. In this model, C_{OX} is the oxide capacitance. CA is the capacitance of the accumulation layer. The series resistance is represented by R_{SERIES}.

Figure 709: Simplified model to determine series resistance



Three-element model of MOS capacitor in strong accumulation

Simplified model of three-element model that is used to determine R_{SERIES}

From Nicollian and Brews 224, the correction capacitance, C_C, and corrected conductance, G_C, are calculated as follows:

$$C_C = \frac{(G_M^2+\omega^2C_M^2)C_M}{a^2+\omega^2C_M^2}$$

and:

$$G_C = \frac{(G_M^2+\omega^2C_M^2)a}{a^2+\omega^2C_M^2}$$

Where:

- $a = G_M - (G^2_M + \omega^2C^2_M) R_{SERIES}$
- 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 2.3 pF less than Cn. If you then add an offset of +2.3 pF 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 are stored in the data file so compensated curves can be easily 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:

$$C_{FB} = \frac{C_{ox}\ \varepsilon_S A /(1\times10^{-4})(\lambda)}{(1\times10^{-12})(C_{ox}) + \varepsilon_S A / (1\times10^{-4})(\lambda)}$$

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 =

$$(1x10^4)\left(\frac{\varepsilon_S kT}{q^2 N_x}\right)^{1/2}$$

Where:

- kT = thermal energy at room temperature ($4{,}046 \times 10^{-21}$ J)
- q = electron charge (1.60219×10^{-19} coulombs)
- 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:

$$V_{TH} = \left[\pm\frac{A}{10^{12}C_{OX}}\sqrt{4\,\varepsilon_S\, q|N_{BULK}||\phi_B|}+2|\phi_B|\right]+V_{FB}$$

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} coulombs)
- 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:

$$W_{MS} = W_M - \left[W_S + \frac{E_G}{2} - \phi_B \right]$$

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)

In tests, the values for W_M, W_S, and E_G are listed in the Formulator as constants. You can change the values depending on the type of materials.

For silicon, silicon dioxide, and aluminum:

$$W_{MS} = 4.1 - \left[4.15 + \frac{1.12}{2} - \phi_B \right]$$

$$W_{MS} = -0.61 + \phi_B$$

$$W_{MS} = -0.61 - \left(\frac{kT}{q} \right) \ln\left(\frac{N_{BULK}}{n_i} \right) \quad (DopeType)$$

Where:

- 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 = is +1 for p-type materials and -1 for n-type materials; the value for DopeType is changed in the Constants area of the Formulator

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.

For the same gate and n-type silicon ($N_{BULK} = 10^{16} cm^{-3}$), $W_{MS} = -0.27$ V.

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 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 in a sheet at the silicon-silicon dioxide interface. From Nicollian and Brews, Eq. 10. 10, we have:

$$V_{FB} - W_{MS} = -\frac{Q_{EFF}}{C_{OX}}$$

Note that C_{OX} here is per unit of area. So that,

$$Q_{EFF} = \frac{C_{OX}\left(W_{MS} - V_{FB}\right)}{A}$$

However, since C_{OX} is in F, we must convert to pF by multiplying by 10^{-12} as follows:

$$Q_{EFF} = 10^{-12}\frac{C_{OX}(W_{MS}-V_{FB})}{A}$$

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:

$$N_{EFF} = \frac{Q_{EFF}}{q}$$

Where:

- N_{EFF} = effective concentration of oxide charge (Units of charge/cm^2)
- Q_{EFF} = effective oxide charge (coulombs/cm^2)
- q = electron charge (1.60219×10^{-19} coulombs)

For example, with an effective oxide charge of 2.5×10^{-8} coul/cm^2, the effective oxide charge concentration is:

$$N_{EFF} = \frac{2.5\times 10^{-8}}{1.60219\times 10^{-19}}$$

$$N_{EFF} = 1.56\times 10^{11} units/cm^2$$

Doping profile

The doping profile of the device is derived from the C-V curve based on the definition of the differential capacitance (measured by the 590 and 595) as the differential change in depletion region charge produced by a differential change in gate voltage (Nicollian and Brews 380-389).

Depletion depth versus 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:

$$w = A\,\varepsilon_S\left(\frac{1}{C_H} - \frac{1}{C_{OX}}\right)$$

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/C² versus 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 $1/C^2$ versus 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 versus depth

The standard N versus 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 $(I\text{-}C_Q/C_{OX})/1\text{-}C_H/C_{OX})$ is included in the calculations as follows:

$$N = \frac{(-2\times10^{-24})[(1-C_Q/C_{OX})/(1-C_H/C_{OX})]}{A^2 q\,\varepsilon_S}\left[\frac{d}{dV_{GS}}\left(\frac{1}{C_H^2}\right)\right]^{-1}$$

Where:

- N = doping concentration (cm^{-3})
- C_Q = quasistatic capacitance (pF)
- C_{OX} = oxide capacitance (pF)
- $(1-C_Q/C_{OX})/1-C_H/C_{OX})$ = voltage stretchout term
- C_H = high-frequency capacitance (pF)
- A = gate area (cm^2)
- q = electron charge (1.60219×10^{-19} coulombs)
- ε_S = permittivity of substrate material
- 1×10^{-24} = units conversion factor

Interface trap density

Interface trapped charges (Q_{it}) are electrons or holes trapped in localized surface states near the Si-SiO_2 interface. These charges are one of four general types associated with the Si-SiO_2 iterface. Interface charges interact electrically with the silicon substrate, which affects MOSFET channel carrier mobility.

Band bending versus 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:

$$(\Psi_S - \Psi_0) = \sum_{V_{GS}\#1}^{V_{GS}Last} (1 - C_Q / C_{OX})(2V_{STEP})$$

Where:

- $(\psi_S - \psi_0)$ = 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 $(\psi_S - \psi_0)$ value is accumulated as the column is built, from the first row of the array (V_{GS} #1) 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 $(\psi_S - \psi_0)$ values are stored in the array, the value of $(\psi_S - \psi_0)$ at the flatband voltage is used as a reference point and is set to 0 by subtracting that value from each entry in the $(\psi_S - \psi_0)$ column, changing each element in the column to ψ_S.

Interface trap capacitance CIT and density DIT

The density of interface traps (D_{it}) is a function of the silicon orientation and the fabrication process. It is determined by performing simultaneous high frequency and quasistatic C-V sweeps. The measurements are extracted mostly from the depletion and inversion regions near mid-band.

Interface trap density is calculated from C_{IT} as shown below (from Nicollian and Brews 332, see References (on page 4-88)).

$$C_{IT} = \left(\frac{1}{C_Q} - \frac{1}{C_{ox}}\right)^{-1} - \left(\frac{1}{C_H} - \frac{1}{C_{ox}}\right)^{-1}$$

$$D_{IT} = \frac{C_{IT}}{Aq}$$

Where:

- C_{IT} = interface trap capacitance (F)
- D_{IT} = interface trap density ($cm^{-2}\ eV^{-1}$)
- C_Q = quasistatic capacitance (F)
- C_H = high-frequency capacitance (F)
- C_{OX} = oxide capacitance (F)
- A = gate area (cm^2)
- q = electron charge (1.60219×10^{-19} coulombs)

Mobile ion charge concentration

Mobile ion contaminants in an oxide layer can cause problems in the manufacture and performance of integrated circuits. To measure the concentration of mobile ions in the oxide layer, you can use the triangular voltage sweep (STVS) method, developed by Keithley Instruments to monitor mobile ion charge in MOS structures.

You can also use the flatband voltage shift or temperature-bias stress method to measure oxide charge density.

Mobile ion monitoring with triangular voltage sweep (STVS) method

STVS is a 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:

$$N_M = \frac{\sum_{-V_{GS}}^{+V_{GS}} (C_Q - C_H)\Delta V_{GS}}{q}$$

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 (coulombs)

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:

$$V_{FB} - W_{MS} = \frac{\bar{x} Q_O}{\varepsilon_{OX}} = \frac{\bar{x} Q_O}{X_O C_{OX}}$$

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:

$$\Delta V_{FB} = \Delta(V_{FB} - W_{MS})$$

$$\Delta V_{FB} = \Delta \frac{\bar{x} Q_O}{\varepsilon_{OX}}$$

$$\Delta V_{FB} = \frac{Q_O}{C_{OX}} \Delta \frac{\bar{x}}{X_O}$$

For the common case of thermally grown oxide, x (before) = X_O and x (after) = 0, so that

$$\Delta V_{FB} = \frac{-Q_O}{C_{OX}}$$

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 VGS as shown in the following:

$$N_m = \frac{+V_{GS}}{-V_{GS}} (C_q - C_b)\, V_{GS} / (qA)$$

Where:

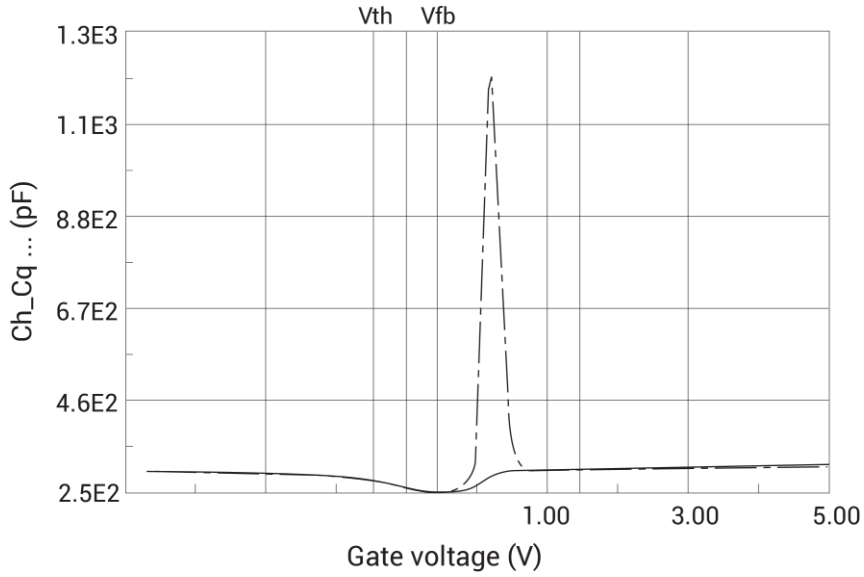
- N_m = mobile ion concentration (cm^{-2})
- $+V_{GS}$ = gate-substance voltage (V)
- $-V_{GS}$ = change in gate-substrate voltage (V)
- C_q = quasistatic capacitance at given VGS (pF)
- C_b = high frequency capacitance at given V_{GS} (capacitance without mobile charges)(pF)
- q = electron charge = $1.60219X10^{-19}C$
- A = area of gate capacitor (cm^2)

The following figure 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 710: 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 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/nI vs. w-w_F as outlined in the following computation information.

G/nI computation

$$G / n_I = -\varepsilon_S A N_{AVG} C_{OX} \bullet \left[\frac{\frac{1}{C^2_{t(i+1)}} - \frac{1}{C^2_{t(i-1)}}}{n_I t_{int}} \right] \bullet \left(\frac{1 \times 10^{12}}{2} \right)$$

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 - wF computation

$$w - w_F = 1 \times 10^{12} \varepsilon_S A \left(\frac{1}{C_{ti}} - \frac{1}{C_{OX}} \right) - w_F$$

$$w_F = 1 \times 10^{12} \varepsilon_S A \left(\frac{1}{C_{ti}} - \frac{1}{C_{OX}} \right)$$

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.

Default material constants

The following table lists default material constants, values, descriptions, and symbols.

Default material constants

Symbol	Description	Default value
q	Electron charge (coulombs)	1.60218e-019 coulombs
k	Boltzmann's constant (J/°K)	1.38065e-023 J/°K
T	Test temperature (°K)	297.13 °K
εOX	Permittivity of oxide (F/cm)	3.4e-013 F/cm
εS	Semiconductor permittivity (F/cm)	1.04e-012 F/cm
EG	Semiconductor energy gap (eV)	1.12 eV
n_I	Intrinsic carrier concentration (I/cm^3)	1.45e010 cm^{-3}
W_{MS}	Metal work function (V)	4.1 V
W_M	Electron affinity (V)	4.15 V

Data symbols

The following table summarizes data symbols in the library, including a description of each symbol.

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.	Ω
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, ψS, is equal to twice the bulk potential, ϕB.	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 ψS due to calculation method and V_0.	V
ϕ_B	Silicon bulk potential.	V

Symbol	Description	Units
l	Extrinsic Debye length.	m

Summary of analysis equations

The analysis equations used by the Model 82 software are summarized in the following.

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

$$Q_{EFF} = \frac{C_{OX}\left(W_{MS} - V_{FB}\right)}{A}$$

Effective charge concentration

$$N_{EFF} = \frac{Q_{EFF}}{q}$$

Flatband capacitance

$$C_{FB} = \frac{C_{OX}\,\varepsilon_S A / (1\times10^{-4})(\lambda)}{(1\times10^{-12})(C_{OX}) + \varepsilon_S A / (1\times10^{-4})(\lambda)}$$

Where λ = extrinsic DeBye length =

$$(1\text{x}10^4)\left(\frac{\varepsilon_S kT}{q^2 N_X} \right)^{1/2}$$

N_x = N at 90% W_{MAX}, or N_A, or N_D when input by the user

Flatband voltage shift

$$V_{FB} - W_{MS} = \frac{\bar{\chi} Q_O}{\varepsilon_{OX}} = \frac{\bar{\chi} Q_O}{X_O C_{OX}}$$

$$\Delta V_{FB} = \frac{-Q_O}{C_{OX}}$$

Interface trap capacitance and Interface trap density

$$C_{IT} = \left(\frac{1}{C_Q} - \frac{1}{C_{OX}}\right)^{-1} - \left(\frac{1}{C_H} - \frac{1}{C_{OX}}\right)^{-1}$$

$$D_{IT} = \frac{C_{IT}}{Aq}$$

Mobile ion charge concentration – TVS method

$$\sum_{-V_{GS}}^{+V_{GS}} (C_{MEAS} - C_{OX}) \Delta V_{GS} = Q_O$$

Mobile ion charge concentration – STVS method

$$N_M = \frac{\sum_{-V_{GS}}^{+V_{GS}} (C_Q - C_H) \Delta V_{GS}}{q}$$

Oxide thickness / gate area

$$t_{OX} = \frac{A\,\varepsilon_{OX}}{(1 \times 10^{-19}) C_{OX}}$$

Series resistance compensation

$$C_C = \frac{(G_M^2 + \omega^2 C_M^2) C_M}{a^2 + \omega^2 C_M^2}$$

$$G_C = \frac{(G_M^2 + \omega^2 C_M^2) a}{a^2 + \omega^2 C_M^2}$$

$a = G_M - (G^2_M + \omega^2 C^2_M) R_{SERIES}$

Threshold voltage

$$V_{TH} = \left[\pm \frac{A}{10^{12} C_{OX}} \sqrt{4 \varepsilon_S q |N_{BULK}| |\phi_B|} + 2|\phi_B| \right] + V_{FB}$$

Work function

$$W_{MS} = W_M - \left[W_S + \frac{E_G}{2} - \phi_B \right]$$

Zerbst plot (generation lifetime and velocity)

$$G / n_i = - \varepsilon_S A N_{AVG} C_{OX} \bullet \left[\frac{\frac{1}{C^2_{t(i+1)}} - \frac{1}{C^2_{t(i-1)}}}{n_i t_{int}} \right] \bullet \left(\frac{1 \times 10^{12}}{2} \right)$$

$$w - w_F = 1 \times 10^{12} \varepsilon_S A \left(\frac{1}{C_{ti}} - \frac{1}{C_{OX}} \right) - w_F$$

$$w_F = 1 \times 10^{12} \varepsilon_S A \left(\frac{1}{C_{ti}} - \frac{1}{C_{OX}} \right)$$

References

The references below are cited in this chapter:

Nicollian, E.H. and Brews, J.R., *MOS Physics and Technology*. Wiley, New York (2003).

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 SiO2-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 E

Using a Keysight 8110A/8111A Pulse Generator

In this appendix:

Introduction

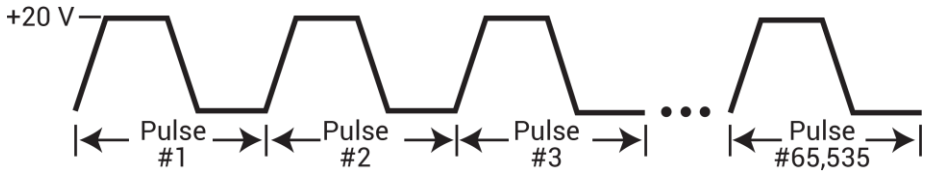
NOTE

For details on all aspects of the HP pulse generator operation, refer to the *Keysight Model 8110A User's Manual*.

The 4200A-SCS can control a Keysight Model 8110A Pulse Generator to output from 1 to 65,535 pulses. The figure below 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 (on page 6-385).

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 4200A-SCS performs before-stress and after-stress characterization tests on the device.

Figure 711: Pulse generator output example



Pulse generator tests

The 4200A-SCS includes the following user modules to run tests using a Keysight pulse generator:

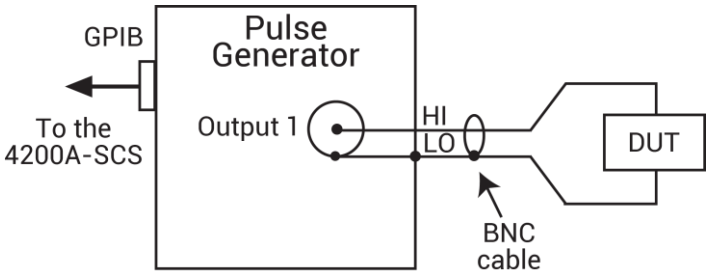
- **PguInit8110: Initialization:** Disables the pulse generator output and returns it to a default setup configuration.
- **PguSetup8110: Set up pulse:** Used to define the output pulse.
- **PguTrigger8110: Trigger output:** Used to specify the number of pulses and trigger the pulse output process.

Details on the user modules for the Keysight pulse generator library are in HP8110ulib user library (on page 6-385).

Signal connections

Basic signal connections for an output of the pulse generator is shown in the following figure. The output LO is connected to the chassis of the pulse generator.

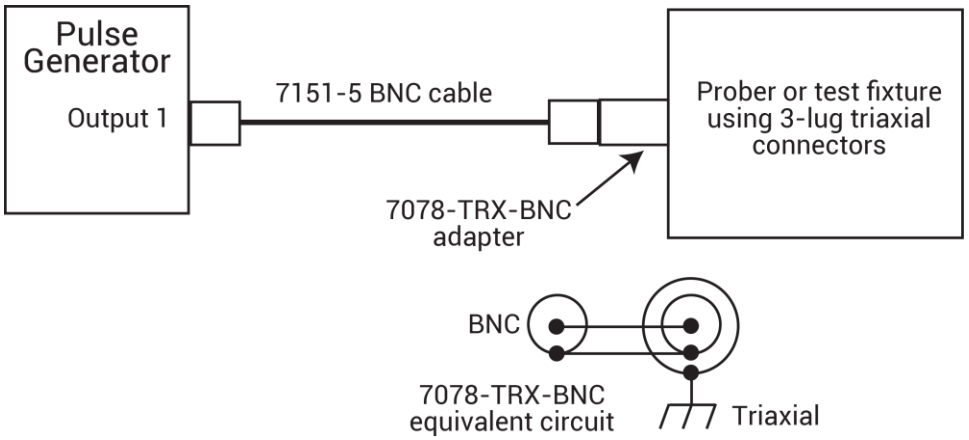
Figure 712: Basic pulse generator connections to DUT



Triaxial connections: Adapters are required to connect the pulse generator to equipment that uses triaxial connectors (for example, the probe station, test fixture, and matrix card).

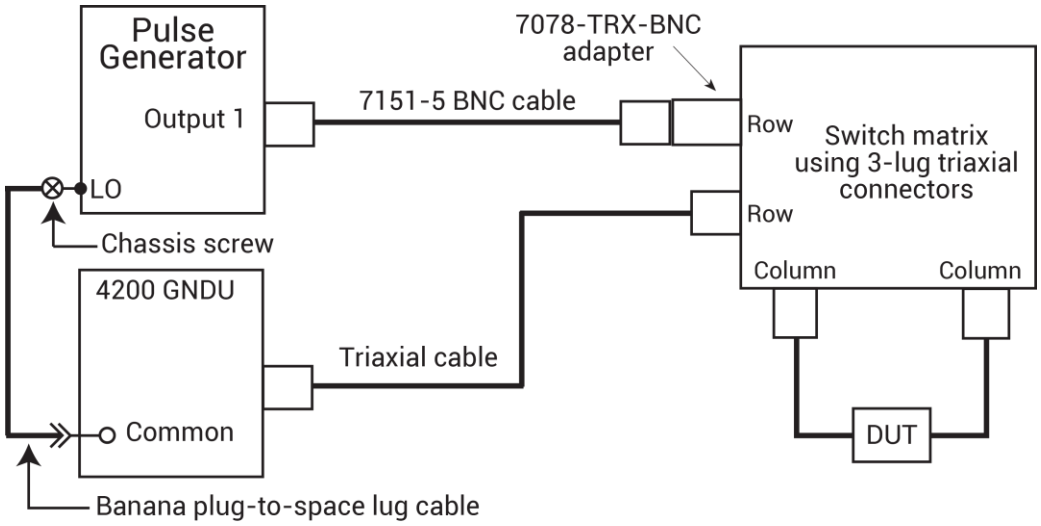
Probe station and test fixture connections: The following figure shows connections to a probe station or a test fixture that is equipped with 3-slot triaxial connectors. The 7078-TRX-BNC is a 3-lug triaxial to BNC adapter. As shown, connect the adapter to the 3-slot triaxial connector and then use a 7051-5 BNC cable to make the connection to the pulse generator. This figure also shows the equivalent circuit for the adapter.

Figure 713: Connections to prober or test fixture equipped with triaxial connectors



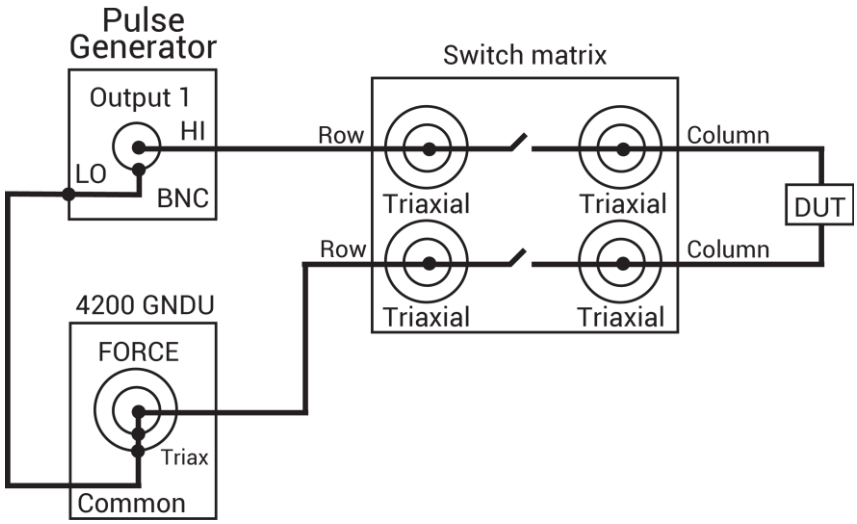
Switch matrix connections: When using a switch matrix that is equipped with triax connectors, separate HI-to-LO matrix paths are required for the pulse generator. A typical connection scheme for this type of switch matrix is shown below. As shown, OUTPUT 1(HI) is connected to a matrix row, and the return path (LO) from the switch matrix is connected to the ground unit (GNDU). 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 714: Connections to switch matrix equipped with triaxial connectors



The following figure 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 (on page A-1).

Figure 715: Pulse output signal path



GPIB connections

The 4200A-SCS controls the pulse generator through the General Purpose Interface Bus (GPIB). Use the 7007-1 or 7007-2 GPIB cable to connect the GPIB port of the pulse generator to the GPIB port of the 4200A-SCS.

Using KCon to add a Keysight pulse generator to the system

In order for the 4200A-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 the Keithley Configuration Utility (KCon).

Refer to Using KCon to add equipment to the 4200A-SCS (on page 7-7) for instruction.

For additional detail on KCon, refer to Keithley Configuration Utility (on page 7-1).

HP8110ulib user library

Use the user modules in the `HP8110ulib` user library to control a Keysight Model 8110A Pulse Generator. These user modules are summarized in the following table. The table also lists the user test modules (UTM) created by Keithley Instruments that use the user modules.

HP8110ulib user modules

User Module	UTM Name	Description
PguInit8110 (on page E-6)	`pgu1-init`	Initializes the pulse generator to the default setup.
PguSetup8110 (on page E-7)	`pgu1-setup`	Sets the output pulse parameters.
PguTrigger8110 (on page E-9)	`pgu-trigger`	Specifies pulse count and trigger start of output.

PguInit8110 user module

This user module initializes the pulse generator to a default setup.

Usage

```
status = PguInit8110(char *instr_id);
```

status	Returned values are placed in the Analyze sheet: ■ 0: OK ■ -10000 (INVAL_INST_ID): The specified instrument ID does not exist ■ -10040 (HP8110_NOT_IN_KCON): No PGU was found in the system 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 input parameter is specified
instr_id	The PGU (pulse generator) instrument ID: PGU*X*, where *X* is a number from 1 through 8 (configuration dependent); the PGU instrument ID effectively corresponds to a single pulse generator channel

Details

The user module used by the pgu1-init UTM.

The PguInit8110 user module initializes the Keysight 8110A pulse generator as follows:

- Disables the output of the specified channel.
- Resets (*RST) to ensure that all errors are cleared.
- Sets the output polarity to NORMAL.
- Sets the trigger count to 1.
- Sets the trigger source to MANUAL.
- Enables SINGLE PULSE mode.
- Allows the rise/fall to be independently programmable.
- Sets the pulse height to 0.2 V and base to 0 V.
- Sets the rise/fall to 100e-9 s.
- Sets the width to 300e-9 s.
- Disables error checking.

Also see

None

PguSetup8110 user module

This user module defines the output pulse of the pulse generator (PGU).

Usage

```
status = PguSetup8110(char *instr_id, double DelayTime, double RiseTime, double
   FallTime, double Width, double Period, double BaseValue, double Amplitude, double
   OutImpedance, double LoadImpedance, double OutpEnable);
```

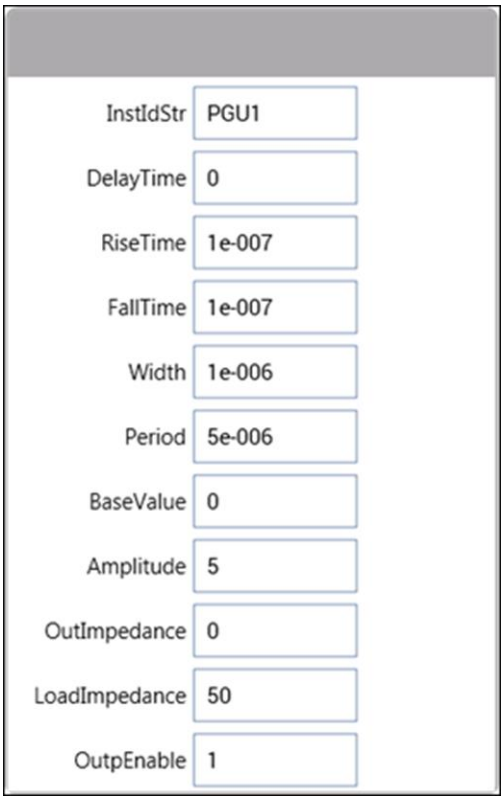
status	Returned values are placed in the Analyze sheet: ■ 0: OK ■ -10000 (INVAL_INST_ID): The specified instrument ID does not exist ■ -10040 (HP8110_NOT_IN_KCON): No PGU was found in the system configuration ■ -10041 (HP8110_NOT_INITED): The PGU was never initialized ■ -10042 (HP8110_PULSE_ERROR): There 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 input parameter is specified
instr_id	The PGU instrument ID: PGU*X*, where *X* is a number from 1 through 8 (configuration dependent); the PGU instrument ID effectively corresponds to a single pulse generator channel
DelayTime	The amount of time to delay after receiving the trigger (0 s to 0.999 s)
RiseTime	Sets the pulse rise time (2e-09 s to 0.2 s)
FallTime	Sets the pulse fall time (2e-09 s to 0.2 s)
Width	Sets the pulse width (3.3e-09 s to 0.999 s)
Period	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; (6.65e-09 to 999; 6.65e-09 to 0.999 if there is no PLL option installed in the pulse generator)
BaseValue	The base value of the pulse (–20 V to +20 V); for a pulse with no DC offset, set this parameter to 0
Amplitude	The amplitude of the pulse as measured from the base value (–20 V to +20 V)
OutImpedance	Sets the output impedance of the PGU: ■ 0: 50 Ω ■ 1: 1000 Ω
LoadImpedance	The expected impedance of the load (DUT) (0 to 999 kΩ); if unsure, enter the maximum value
OutpEnable	A flag that determines whether to enable or disable the output relay of the PGU: ■ 0: Disable the output ■ 1: Enable the output

Details

The `PguSetup8110` user module defines the pulse timing and voltage settings. Once defined, the pulse can be triggered using the `PguTrigger8110` user module.

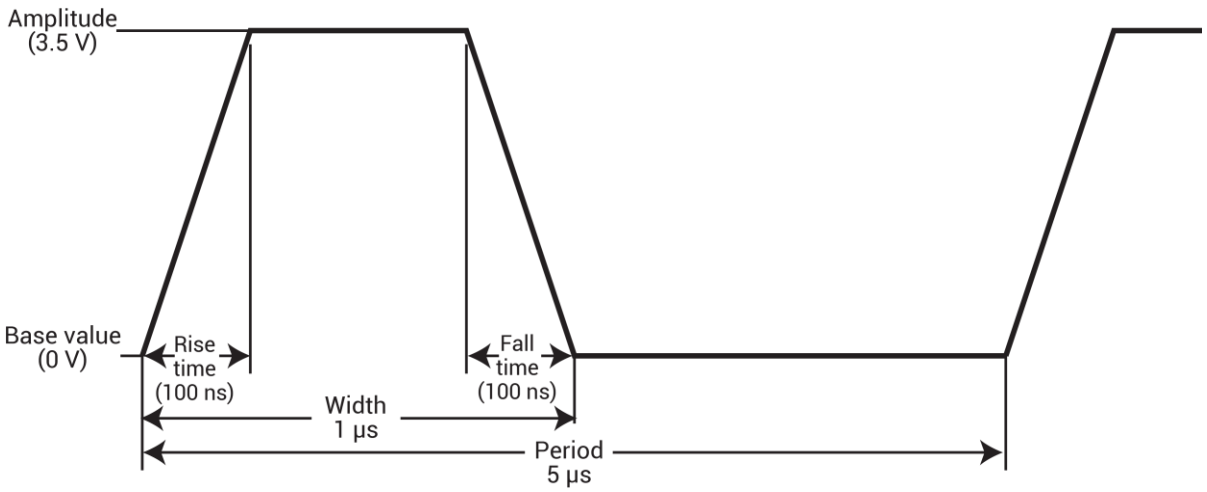
The following figure shows the default parameters for `pgu1-setup` UTM.

Figure 716: PguSetup8110 (pgu1-setup UTM)



The following figure shows the output pulse for the default UTM setup.

Figure 717: pgu1-setup UTM pulse specifications



Also see

PguTrigger8110 (on page E-9)

PguTrigger8110 user module

This user module specifies number of pulses to output and triggers the start of the pulse output process.

Usage

```
status = PguTrigger8110(char *InstIdStr, double Count);
```

status	Returned values are placed in the Analyze sheet: ■ 0: OK ■ -10000 (INVAL_INST_ID): The specified instrument ID does not exist ■ -10040 (HP8110_NOT_IN_KCON): No PGU was found in the system 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 timeout occurred during communications ■ -10100 (INVAL_PARAM): An invalid input parameter is specified
InstIDStr	The PGU (pulse generator) instrument ID: PGU*X*, where *X* is a number from 1 through 8 (configuration dependent); the PGU instrument ID effectively corresponds to a single pulse generator channel
Count	The number of pulses to output; if *Count* is > 1, a burst of pulses with a period as defined in the PguSetup8110 function is output; if *Count* is 1, a single pulse is output

Details

The PguTrigger8110 function will trigger the pulse (or pulses) defined using the PguSetup8110 function.

Also see

PguSetup8110 user module (on page E-7)

Appendix F

Set up a probe station

In this appendix:

Prober control overview

Semi-automatic and fully-automatic probe stations are typically controlled programmatically through a GPIB or RS-232 communications interface. In this situation, the 4200A-SCS acts as the system controller and is connected to the probe station using the appropriate communications interface.

The 4200A-SCS facilitates automated wafer-level testing through various prober control mechanisms. Standard prober drivers are included with the 4200A-SCS, and a number of commercially available automated probe stations are supported. The 4200A-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 4200A-SCS software allows you to integrate prober control into the test flow by creating a user library.

A probe station is controlled by the 4200A-SCS with user modules. User modules are created in the Keithley User Library Tool. Refer to Keithley User Library Tool (KULT) (on page 8-1) for more information regarding user libraries.

The PRBGEN library of prober user modules is provided with the 4200A-SCS to simplify prober control. This generic prober user library, developed and maintained by Keithley Instruments, allows Clarius to control all supported probers in the same manner. Therefore, Clarius projects that use PRBGEN work with any prober supported by Keithley Instruments. Refer to Supported probers (on page F-4) for a list of supported probers and links to additional information.

Many of the PRBGEN user modules have already been built into Clarius as actions. You can add these actions to the project tree in any location that makes sense for your system. The position of the action in the project tree determines when the action is run during a test. For example, in a device with multiple tests, the device level can be run directly, which executes each test under the device sequentially. If an action also exists in the device level, the action runs in sequence with the tests. Similarly, actions under the subsite, site, or project levels execute automatically when the subsite, site, or project is run.

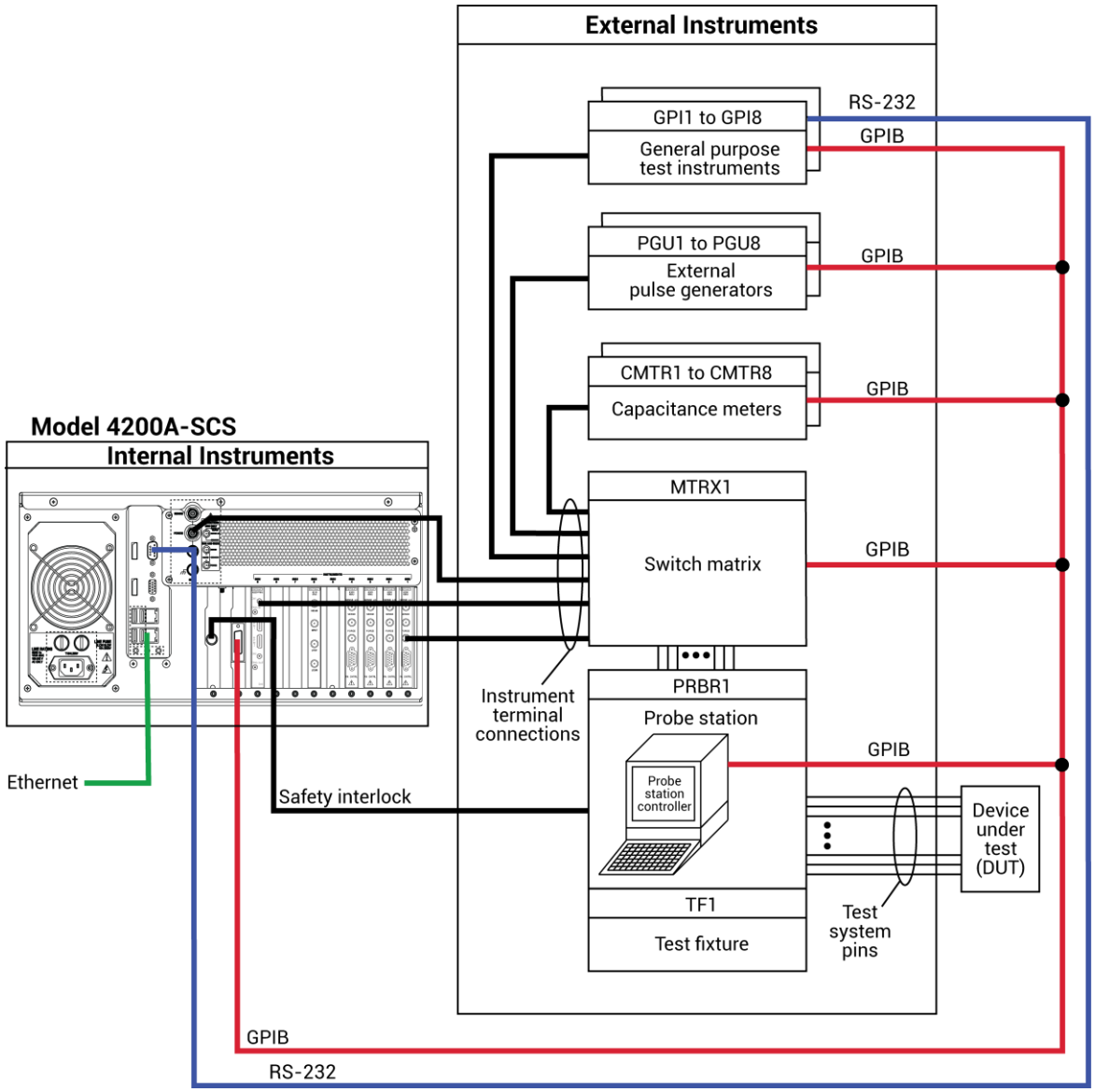
You can connect 4200A-SCS measurement signals to most commercially available wafer probers. Probers that provide triaxial connections to their probes and chuck are the easiest because of the triaxial connections on the 4200-SMU, 4210-SMU, 4200-PA, and GNDU. For other connections, you can get adapters and cable kits from Keithley Instruments that allow the 4200A-SCS to be adapted to any connection environment.

⚠ WARNING

Turning the 4200A-SCS output off does not place the instrument in a safe state (an interlock is provided for this function). Hazardous voltages may be present on all output and guard terminals. To prevent electrical shock that could cause injury or death, never make or break connections to the 4200A-SCS while the instrument is powered on. Turn off the equipment from the front panel or disconnect the main power cord from the rear of the 4200A-SCS before handling cables. Putting the equipment into an output-off state does not guarantee that the outputs are powered off if a hardware or software fault occurs. Precautions must be taken to prevent a shock hazard by surrounding the test device and any unprotected leads (wiring) with double insulation for 250 V, Category O.

Basic system connections are illustrated in the figure below.

Figure 718: Example system connections



Supported probers

Supported probe station	Additional information
Cascade Microtech Model PA200	Cascade Microtech PA200 Prober (on page G-1)
Micromanipulator Model 8860	Micromanipulator 8860 Prober (on page H-1)
Manual or Fake	Using a Manual or Fake Prober (on page I-1)
Cascade Summit-12000	Cascade Summit-12000 Prober (on page J-1)
Signatone CM500	Signatone CM500 Prober (on page K-1)
MPI TS2000, TS2000-DP, TS2000-HP, TS2000-SE, TS3000, and TS3000-SE	Using an MPI Probe Station (on page M-1)

NOTE

Contact Keithley Instruments for the most up-to-date list of supported probe stations.
Use KCon (on page 7-1) to add the prober to the instrument list. See the information for the specific prober for details.

PRBGEN user modules

Prober-control software provided by supported prober vendors gives access to the full feature set of each prober. You use the prober-control software to define a list of wafer locations to be probed. The 4200A-SCS relies on the prober controller and associated software to maintain this probe list. The `PRBGEN` user modules communicate with the prober controller, normally through the GPIB bus or COM1 (serial bus) port, 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 the physical location of each probe location. The following table summarizes the user modules included in the `PRBGEN` prober control user library.

User module	Description
`PrInit`	Initializes the prober driver and establishes the reference site or die.
`PrChuck`	Instructs the prober to move the probe station into contact or to 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 you can execute a Clarius project that uses the `PRBGEN` user library, you must create the probe list using the appropriate vendor-specific prober-control software. Instructions for creating the probe list for each supported prober are included in this manual (refer to Supported probers (on page F-4)).

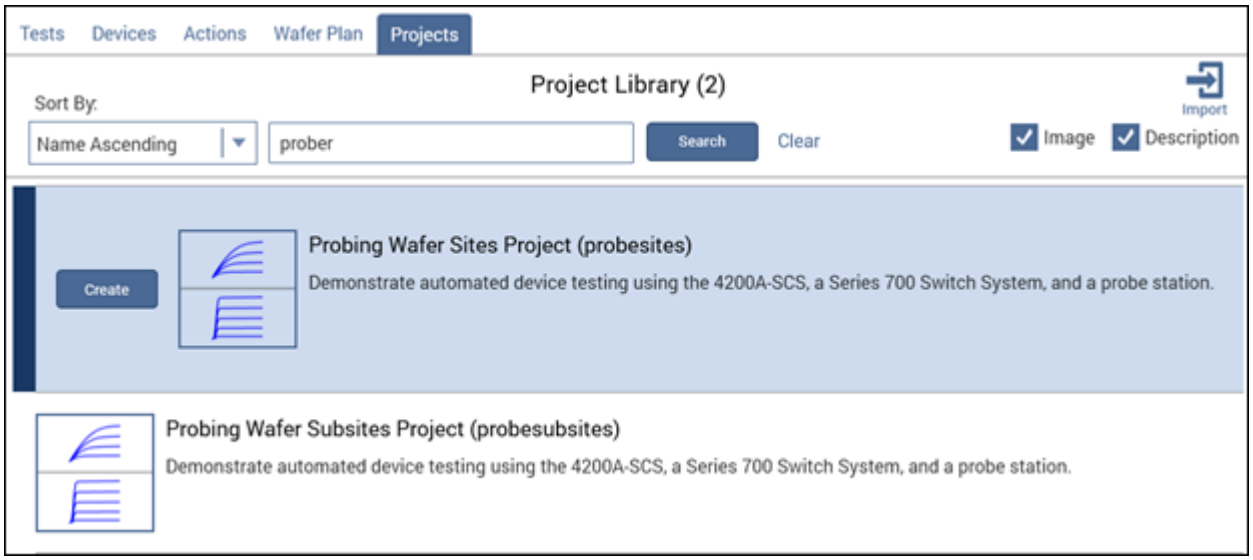
The example projects in the following topics describe a typical project setup. However, you can add and arrange sites, subsites, and prober actions in the project in any order that is appropriate for your system. The 4200A-SCS runs the items in the project in the order in which they are presented, from top to bottom. You can also select the starting point for each run. For example, if you highlight a device, only the tests and actions that are selected and under that device will be used when you select Run.

If you use a semi-automatic prober, understand that a Clarius probe action only triggers movements that are already programmed in the prober controller. Each execution of the action advances the probe to the next site in this programmed sequence. Site numbers are not communicated between the prober and Clarius. Therefore, if you evaluate multiple sites, the range of site numbers that you specify in the Clarius Project window must agree with the sequence of site numbers in the prober controller program.

Set up a Clarius project that controls a prober:

1. Choose **Select**.
2. Select **Projects**.
3. Search for **probesites** or **probesubsites**.
4. Select **Create** to add the prober project to the project tree.

Figure 719: probesites project



Example test execution sequence: probesites project

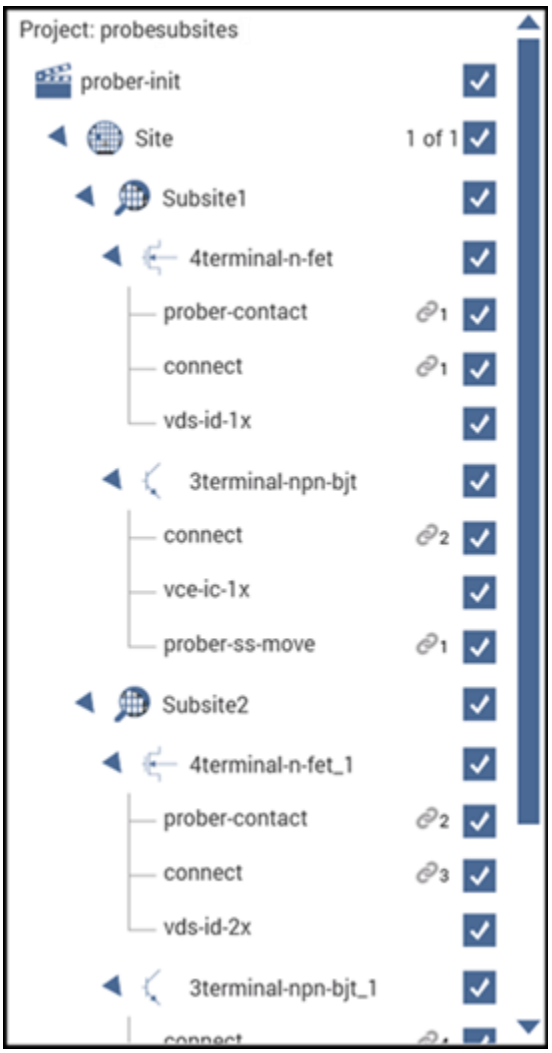
Configure the probesites project:

1. In Clarius, select **Configure**.
2. In the project tree, select **probesites**.
3. Set the Project Execution Loop Settings as needed for your project.
4. Select **Run**.
5. Select **Analyze** to review the data.

Example test execution sequence: probesubsites project

In this example, the `probesubsites` project is selected. When you run the test for the site, tests are run for each of the subsites.

Figure 720: probesubsites project tree



Understanding site coordinate information

The next topics describe the reference site, probe sites, and chuck movement.

Reference site (die)

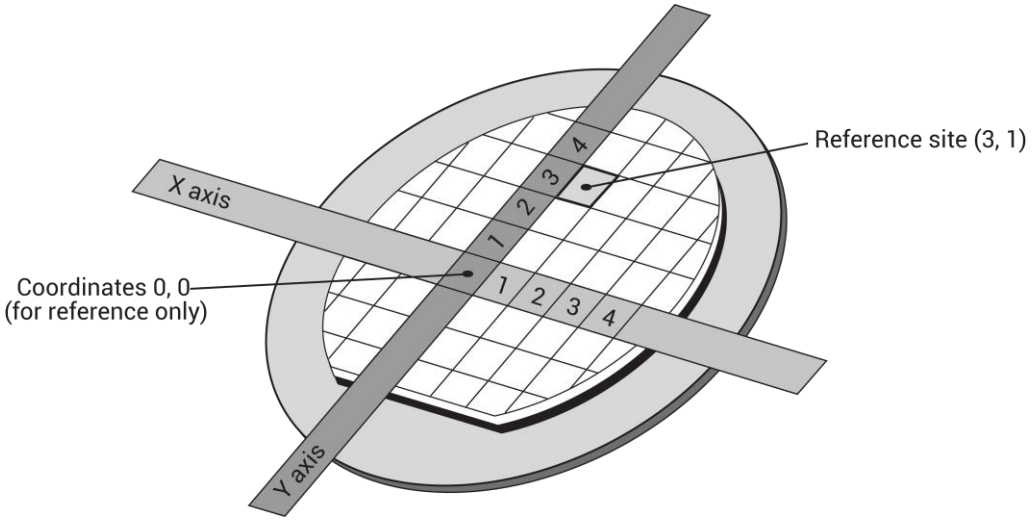
The designated reference site is defined in the `prober-init` action by selecting **Configure** and entering the parameters. This is the first stopping point of the prober once aligned. The physical location of the reference site may be any coordinate that is selected on the wafer and is selected for probing or marked for probing through the prober software. The coordinate system of the wafer is also defined through the prober software. For example, the coordinates of the reference site shown in the following figure are (3, 1).

For parameter descriptions, refer to the Help pane.

NOTE

The defined reference site must match the physical location of the wafer. This is the location on the wafer directly below the probe pins after the wafer has been loaded.

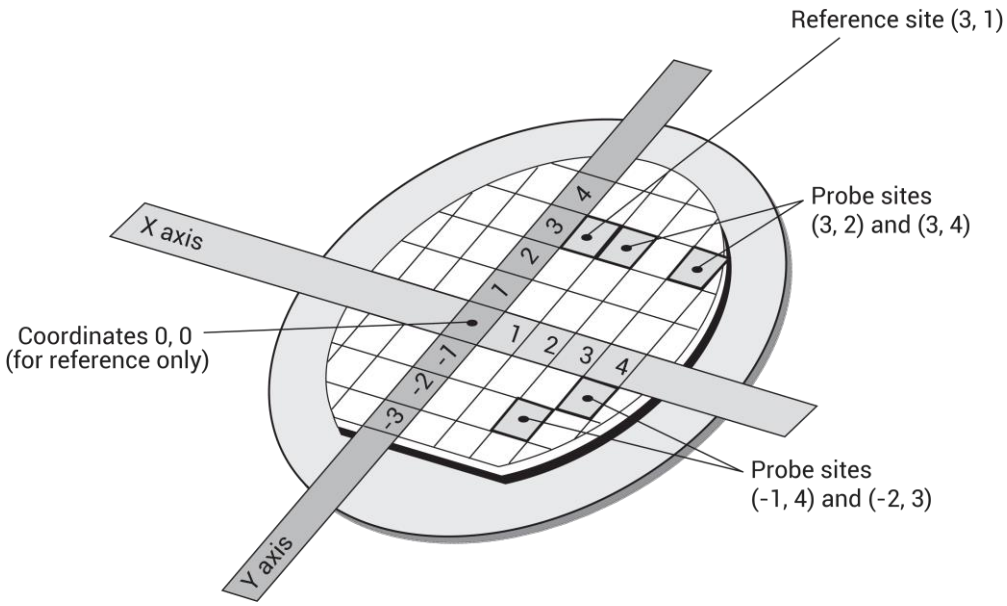
Figure 721: Sample reference site location



Probe sites (die)

Dies marked as probe sites in the prober software define the areas to be tested. The physical location of the probe site can be any coordinates selected on the wafer. Marking a die as a probe site also selects the site for probing. The coordinates of each probe site are referenced with respect to the coordinates of the reference site. For example, with the reference site of (3, 1), the coordinates of the five probe sites shown in the following figure are (3,1), (3, 2), (3, 4), (-1, 4), and (-2, 3).

Figure 722: Sample probe site location



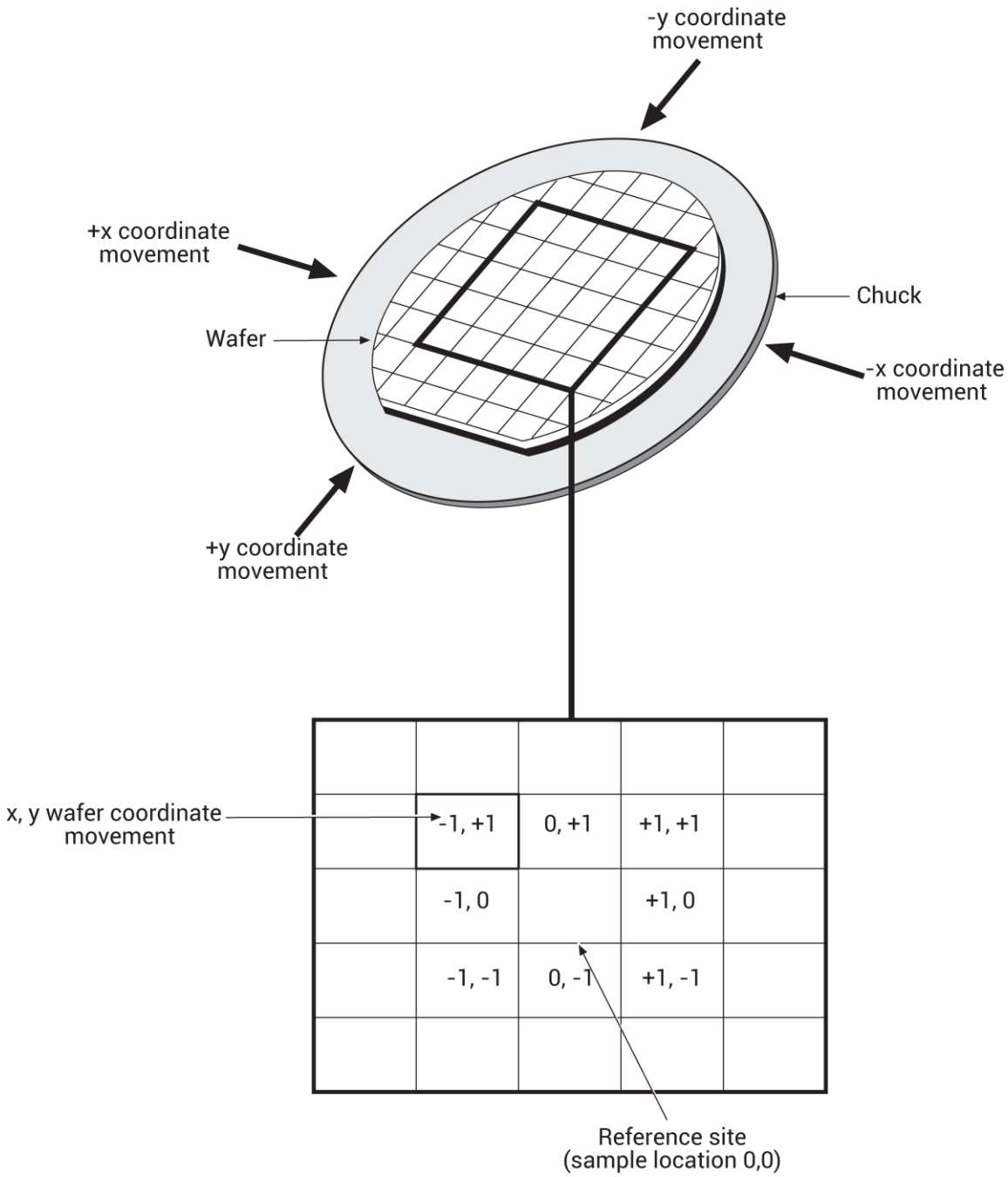
Chuck movement

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, you must configure the correct quadrant (prober dependent). Applicable quadrant setup instructions are in the appendix for your prober. When you specify chuck movements, use the coordinates of the 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 the following figure for a representation of the relationship between chuck movement and (x, y) coordinates.

NOTE

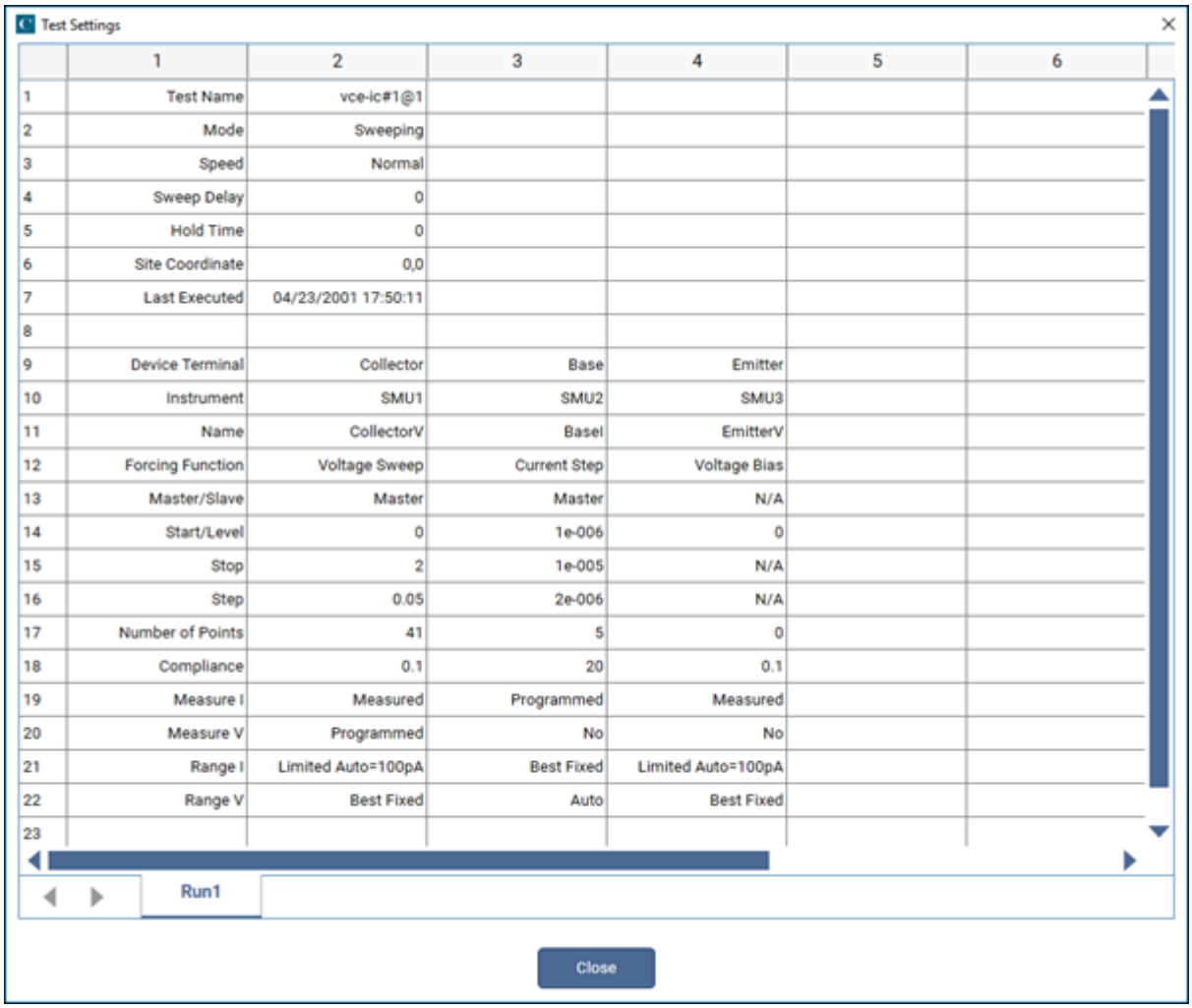
The chuck moves and the probe pins remain stationary. Notice that the chuck movement is opposite of the coordinate system of the probe pins.

Figure 723: Chuck movement



At the conclusion of a test, the site coordinates are recorded in the sheet settings. These coordinates are only valid if a project uses the remote prober control (real prober). The coordinate system is based on the `xstart_position` and `ystart_position` parameters of the `prober-init` action. The site coordinates change only after a site movement is performed; the coordinates change when the bottom of the project loop is reached. At the top of each iteration, the site coordinates remain the same until the site movement is done. Refer to the following figure for an example of a table that contains site coordinates (see column 2, row 6).

Figure 724: Clarius: Example of site coordinates: Analyze sheet



	1	2	3	4	5	6
1	Test Name	vce-ic#1@1				
2	Mode	Sweeping				
3	Speed	Normal				
4	Sweep Delay	0				
5	Hold Time	0				
6	Site Coordinate	0,0				
7	Last Executed	04/23/2001 17:50:11				
8						
9	Device Terminal	Collector	Base	Emitter		
10	Instrument	SMU1	SMU2	SMU3		
11	Name	CollectorV	Basel	EmitterV		
12	Forcing Function	Voltage Sweep	Current Step	Voltage Bias		
13	Master/Slave	Master	Master	N/A		
14	Start/Level	0	1e-006	0		
15	Stop	2	1e-005	N/A		
16	Step	0.05	2e-006	N/A		
17	Number of Points	41	5	0		
18	Compliance	0.1	20	0.1		
19	Measure I	Measured	Programmed	Measured		
20	Measure V	Programmed	No	No		
21	Range I	Limited Auto=100pA	Best Fixed	Limited Auto=100pA		
22	Range V	Best Fixed	Auto	Best Fixed		
23						

PRBGEN user library

The `PRBGEN` user library provides test modules to initialize the prober, move to the next site or subsite in the wafer map of the prober, make or break contact between the probes and the wafer, and get the X position and Y position of the prober. It allows Clarius to control all supported probers in the same manner. Clarius projects that use `PRBGEN` work with any prober supported by Keithley Instruments.

The user modules in the `PRBGEN` user library are provided as actions in Clarius.

PRBGEN user modules

User module	Clarius action	Description
PrChuck (on page F-14)	`prober-contact`	Directs the prober to have the probe pins make contact with the wafer or separate the pins from the wafer.
PrInit (on page F-13)	`prober-init`	Initializes the prober with die size, first coordinate (X and Y), units (mm or mils), and mode information.
PrMovNxt (on page F-17)	`prober-move`	In learn mode, the `PrMovNxt` command causes the prober to move to the next site after inking.
PrSSMovNxt (on page F-15)	`prober-ss-move`	In learn mode, the `PrSSMovNxt` command causes the prober to move to the next subsite after inking.

PrInit

This command initializes the prober with die size, first coordinate (X and Y), units (mm or mils), and mode information.

Usage

```
status = PrInit(int mode, double x_die_size, double y_die_size, int x_start_position,
    int y_start_position, int units, int subprobtype);
```

status	Returned values; see **Details**
mode	The mode to be used with the prober (see **Details**): ■ 1: Manual prober ■ 2: External automatic prober ■ 6: Learn (typically used with semi-automatic probers)
x_die_size	The x die size (units are set by the *units* parameter)
y_die_size	The y die size (units are set by the *units* parameter)
x_start_position	The x location of the prober position at alignment
y_start_position	The y location of the prober position at alignment
units	The units: ■ 0: Mils ■ 1: Millimeters
subprobtype	Not supported for 4200A-SCS

Library

Dependency: PRBCOM

Details

The mode defines the capabilities of the prober. Select External automatic mode when the tester explicitly directs all the prober actions. Use Learn mode when the prober is configured with all the wafer stepping information. When learn is selected, the tester commands the prober to do the next operation. Please confirm the correct mode of operation for each specific application. Supported modes vary from prober to prober.

The PrInit function returns the values:

- 1: Success (OK)
- -1005: Failure setting units
- -1006: Failure setting mode
- -1009: Failure setting die size
- -1011: Operation invalid in mode
- -1013: Unintelligible response

- `-1015`: Unexpected error
- `-1017`: Bad chuck position
- `-1027`: Invalid parameter

Example

```
status = PrInit(6,2,2,1,1,1,0);
```

Also see

None

PrChuck

This command directs the prober to have the probe pins make contact with the wafer or separate the pins from the wafer.

Usage

status = PrChuck(int *chuck_position*);

status	Returned values; see **Details**
chuck_position	The chuck position: ■ `0`: Separate from the chuck ■ `1`: Contact the chuck

Library

Dependency: PRBCOM

Details

The `PrChuck` function returns the values:

- `1`: Success (OK)
- `-1008`: Invalid mode
- `-1013`: Unintelligible response
- `-1015`: Unexpected error
- `-1017`: Bad chuck position

Example

```
status = PrChuck(1);
```

Also see

None

PrSSMovNxt

In learn mode, the `PrSSMovNxt` command causes the prober to move to the next subsite. If needed, you can specify the inker to fire before the move.

Usage

```
status = PrSSMovNxt(int ink_number);
```

status	Returned values; refer to **Details**
ink_number	The inkers to fire: ■ 0: No inker; move only ■ 1: 1 ■ 2: 2 ■ 3: 1, 2 ■ 4: 3 ■ 5: 1, 3 ■ 6: 2, 3 ■ 7: 1, 2, 3 ■ 8: 4 ■ 9: 1, 4 ■ 10: 2, 4 ■ 11: 1, 2, 4 ■ 12: 3, 4 ■ 13: 1, 3, 4 ■ 14: 2, 3, 4 ■ 15: All 4

Library

Dependencies: PRBCOM

Details

The `PrMovNxt` function returns the values:

- `1`: Success (OK)
- `2`: Prober moved to next die (confirmed)
- `4`: Next wafer loaded (confirmed)
- `-1008`: Invalid mode
- `-1011`: Operation invalid in mode
- `-1013`: Unintelligible response

- `-1014`: Movement failure
- `-1015`: Unexpected error
- `-1027`: Invalid parameter

Example

```
status = PrSSMovNxt(0);
```

Also see

None

PrMovNxt

In learn mode, the `PrMovNxt` command causes the prober to move to the next site. If needed, you can specify the inker to fire before the move.

Usage

```
status = PrMovNxt(int ink_number);
```

status	Returned values; refer to **Details**
ink_number	The inkers to fire: ■ 0: No inker; move only ■ 1: 1 ■ 2: 2 ■ 3: 1, 2 ■ 4: 3 ■ 5: 1, 3 ■ 6: 2, 3 ■ 7: 1, 2, 3 ■ 8: 4 ■ 9: 1, 4 ■ 10: 2, 4 ■ 11: 1, 2, 4 ■ 12: 3, 4 ■ 13: 1, 3, 4 ■ 14: 2, 3, 4 ■ 15: All 4

Library

Dependencies: PRBCOM

Details

The `PrMovNxt` function returns the values:

- 1: Success (OK)
- 4: Next wafer loaded (confirmed)
- -1008: Invalid mode
- -1011: Operation invalid in mode
- -1013: Unintelligible response
- -1014: Movement failure

- `-1015`: Unexpected error
- `-1027`: Invalid parameter

Example

```
status = PrMovNxt(0);
```

Also see

None

Tutorial: Control a probe station

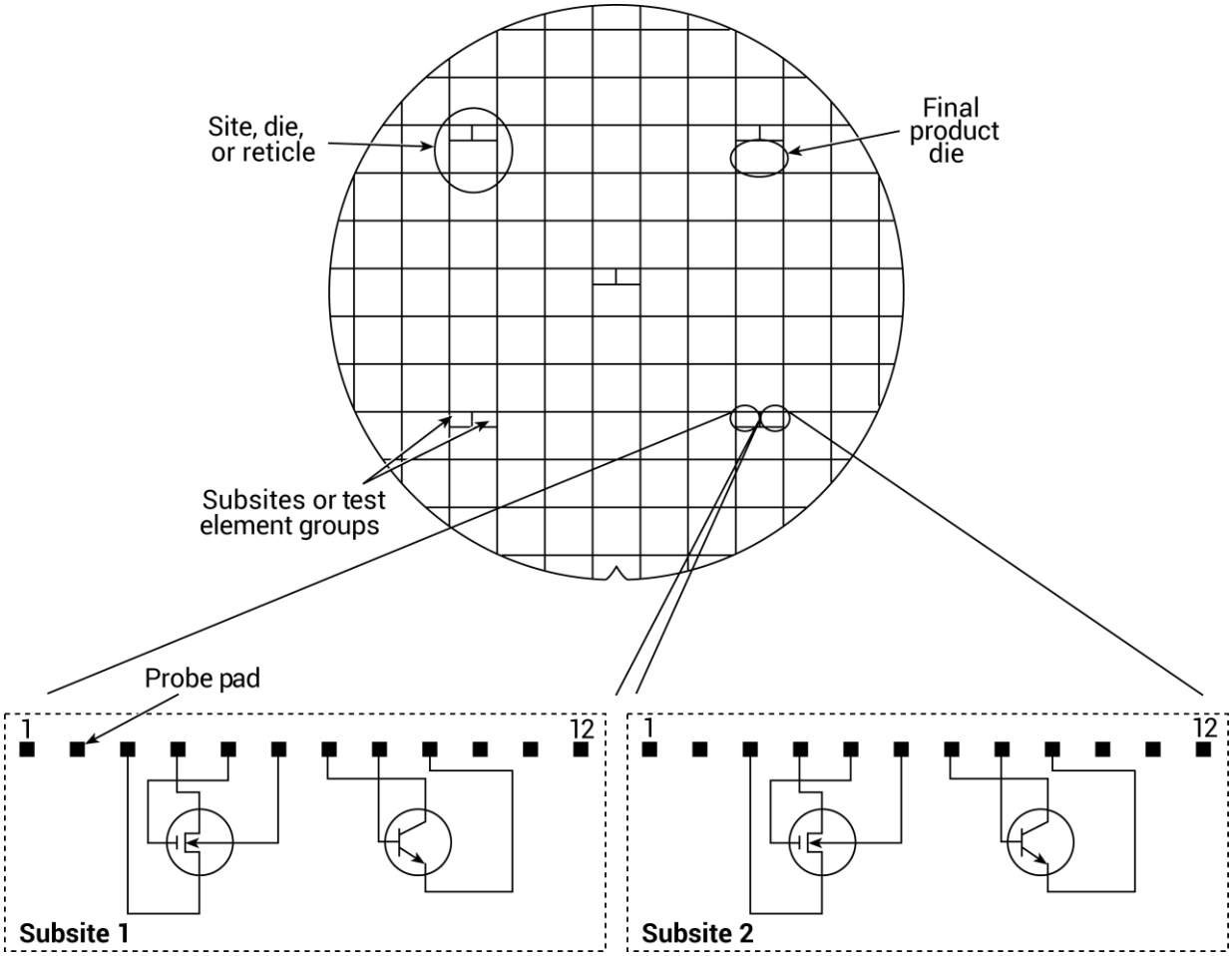
This tutorial demonstrates how to control a probe station to test five identical sites (or die or reticles) on a sample wafer.

Each wafer site has two subsites (or test element groups). At each subsite there are two devices (or test elements) to be tested:

- 4-terminal N-channel MOSFET
- 3-terminal NPN transistor

The subsites do not need to be identical, but for simplicity they are assumed to be the same. This is illustrated in the following figure.

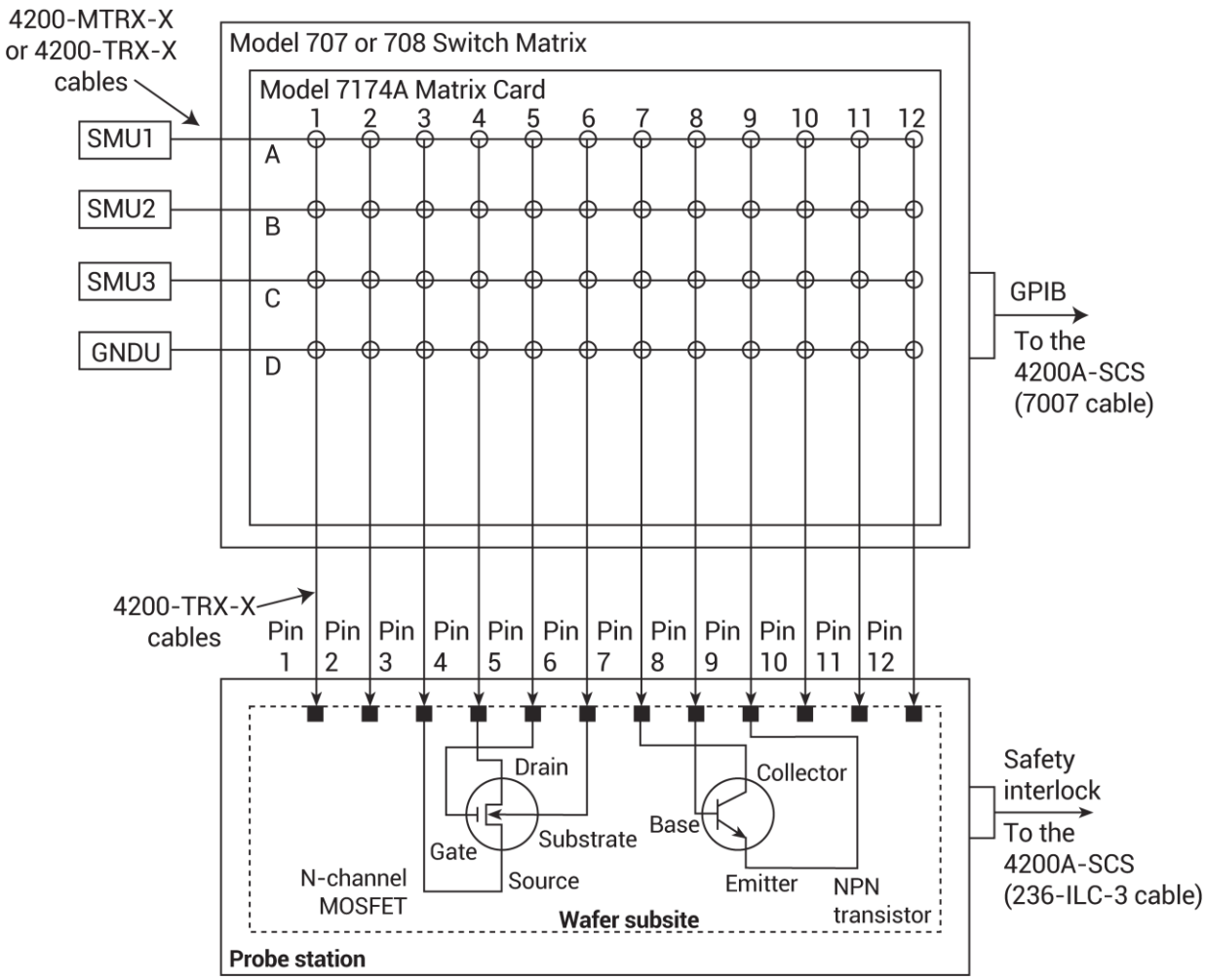
Figure 725: Sample wafer organization



Test system connections

A typical test system for this tutorial is shown in the following figure. The 4200A-SCS and probe station are connected to a 7174A matrix card. The matrix card is installed in the switch matrix, and the switch matrix and probe station are controlled through the GPIB bus.

Figure 726: System configuration for the probesubsites project



KCon setup

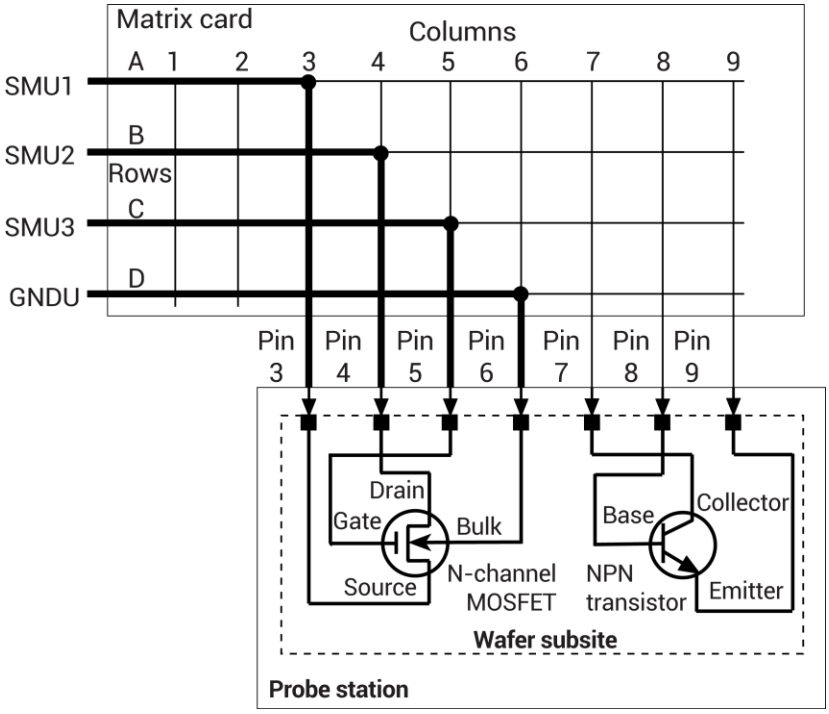
Refer to Use KCon to add a switch matrix to the system (on page A-31).

Test flow

When you run the `probesubsites` project, the following occurs:

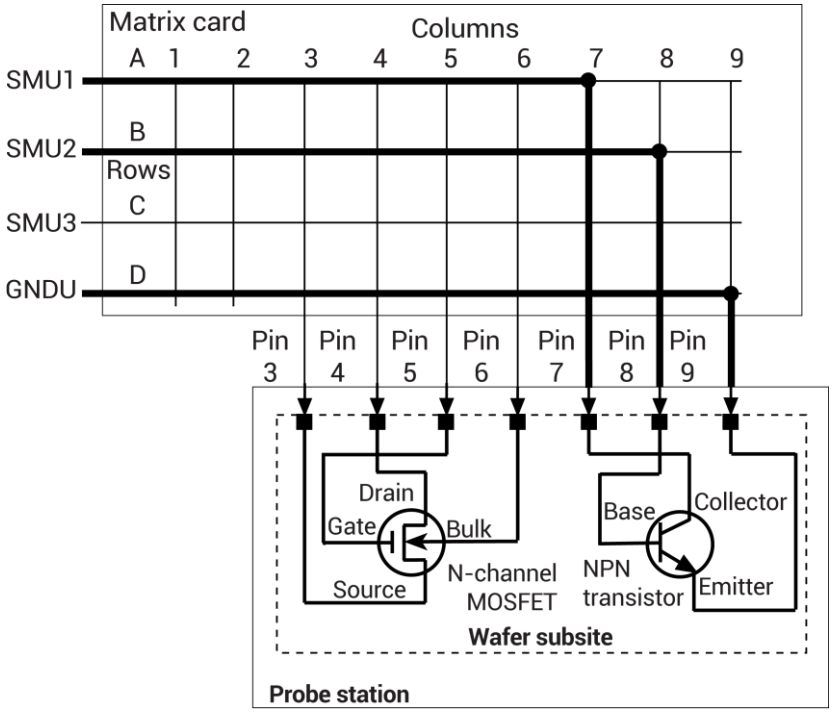
1. The action `prober-init` initializes the prober driver.
2. The test moves to `subsite1`, `4terminal-n-fet`.
3. The action `prober-contact` moves the chuck to the wafer.
4. The action `connect` connects the SMUs to the probes for the n-channel MOSFET as shown in the following figure.

Figure 727: Connect SMUs to N-channel MOSFET



5. The test runs vds-id-1x, which generates a family of curves (I_D vs. V_D) for the MOSFET.
6. The test moves to `3terminal-npn-bjt`.
7. The action `connect` connects the SMUs to the probes for the npn transistor as shown in the following figure.

Figure 728: Connect SMUs to NPN transistor



8. The test runs `vce-ic-1x`, which generates a collector family of curves (I_C vs. V_C) for the transistor.
9. The action `prober-ss-move` moves the prober to the next subsite.
10. The tests continue with `subsite2` and `subsite3`.
11. After all the subsites have run, the action `prober-separate` separates the prober pins from the wafer.
12. The action `prober-prompt` displays the message “Wafer Test Complete” at the end of the test.

Appendix G

Using a Cascade Microtech PA200 Prober

In this appendix:

Cascade Microtech PA200 prober software

To configure and operate the PA200 prober with the Keithley Instruments 4200A-SCS, you need the following applications:

- 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 subsites.
- PB-GPIB: Use to configure the GPIB interface.
- PB-RS-232: Use to configure the serial interface.
- Prober Setup (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 PA200 prober with the 4200A-SCS.

Product Name:	WaferMap for ProberBench NT
Product Version:	3.1 (Feb 12, 1999)
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 for ProberBench NT

Product Version:	3.10 (Feb 12, 1999)
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:	© 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 12, 1999)
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 12, 1999)
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:	© 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

Make sure that you are familiar with the Cascade MicroTech® PA200 Prober and its supporting documentation before you attempt setup, configuration, or operation.

To set up and configure the PA200 prober for use with the 4200A-SCS, you will:

- Set up communications (on page G-8)
- Set up wafer geometry (on page G-13)
- Create a site definition and define a probe list (on page G-17)
- Load, align, and contact the wafer (on page G-18)

Set up communications

You need to set communications between the 4200A-SCS and the prober.

To make the connections:

1. Connect the ProberBench NT computer's COM2 port to the 4200A-SCS COM1 port using a DB9 female to DB9 female cable (shielded null modem cable). See the following figure.
2. Connect the ProberBench NT computer serial port (COM1) to the PA200 Prober Electronics Rack serial port.
3. Connect the 4200A-SCS GPIB port and the ProberBench NT computer's GPIB port using a GPIB cable (Model 7007). Refer to the following two graphics and table for a connection diagram, connector diagram, and connector pinout definitions.

NOTE

Do not use the GPIB port on the Prober Electronics Rack. Make sure to connect the cable between the 4200A-SCS and the ProberBench NT computer's GPIB ports as shown.

Figure 729: 4200A-SCS and PA-200 serial port connection

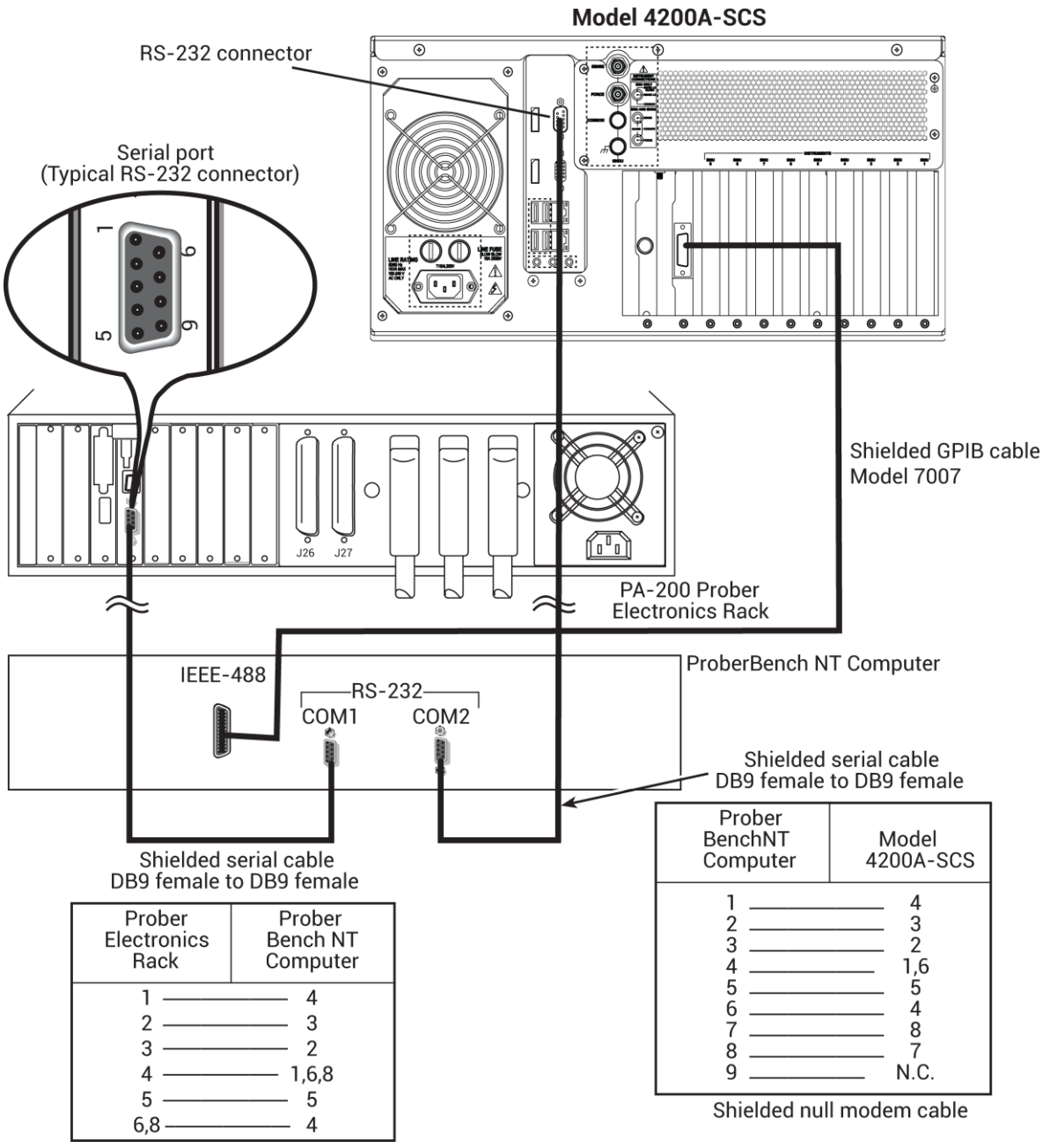
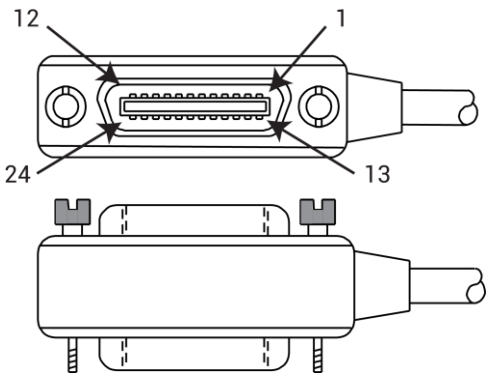


Figure 730: IEEE-488 connector



GPIB control connector terminals

Contact number	GPIB 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

Set up communications on the 4200A-SCS

On the 4200A-SCS, you need to set the communications through the prober configuration file.

The configuration file for use with serial communications is shown below. To configure the prober for use with a GPIB communications setup, use a text editor to comment out (#) the lines after "Configuration for PA200 probers" and activate the lines (remove the #) after "Configuration for direct GPIB probers." Be sure to only activate the lines that start with `PROBER_1_...`.

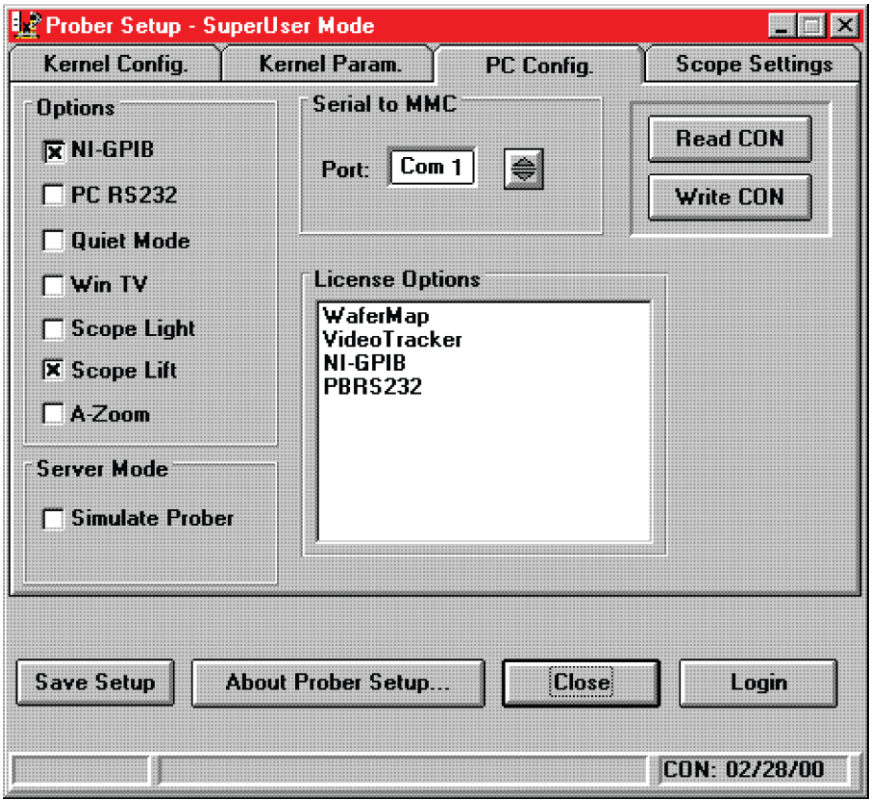
Configuration file location: `C:\s4200\sys\dat\prbcnfg_PA200.dat`

```
# 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
#
#
```

Set up communications on the prober

You can configure the PA-200 prober for serial or GPIB communication. Ensure that the prober is set up for the type of communications interface that is defined on the 4200A-SCS (see Set up communications on the 4200A-SCS (on page G-6)).

Figure 731: Prober setup: PC Config tab



To set up communications for RS232:

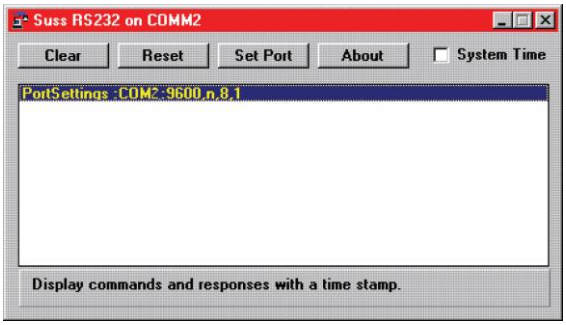
1. On the prober computer, double-click the ProberBench NT icon.
2. Double-click the Service Programs file.
3. Double-click the **Prober Setup** file in the Service Programs directory.
4. Select the PC **RS232** option.
5. Clear the Simulate Prober box in the Server Mode section of the dialog box.
6. Click **Save Setup**.
7. Click **Close**.
8. From the ProberBench NT window, double-click the PBRS232 file.

NOTE

COM2 is used for communications between the 4200A-SCS and the ProberBench NT. COM1 is used for communications between the ProberBench NT and the electronics rack.

9. From the Suss RS232 on COMM2 dialog box, click **Set Port**. See the following figure.

Figure 732: Suss RS232 on COMM2 dialog box



10. Set communications protocol to 9600, n, 8, 1 for serial port COM2.
11. Make sure Disable COM Port is not selected.

NOTE

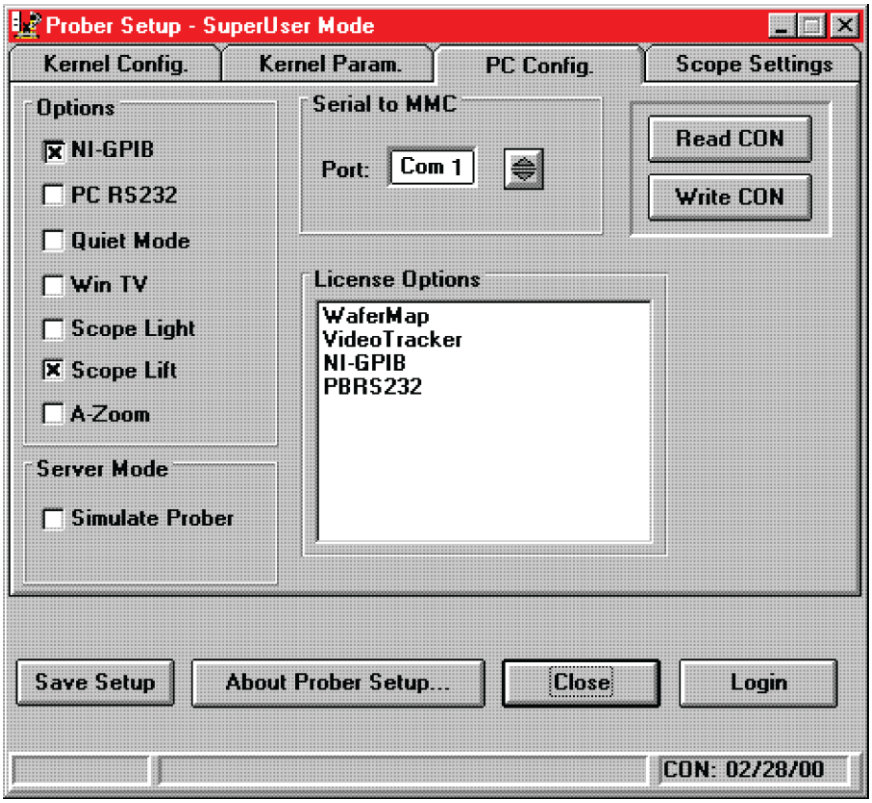
Leave the **Suss RS232 on COMM2** dialog box open. This ensures its services are available for the WaferMap program.

12. Click **Save** and **Exit**.
13. From the **Suss RS232 on COMM2** dialog box, click **Reset**.

To set up GPIB communications:

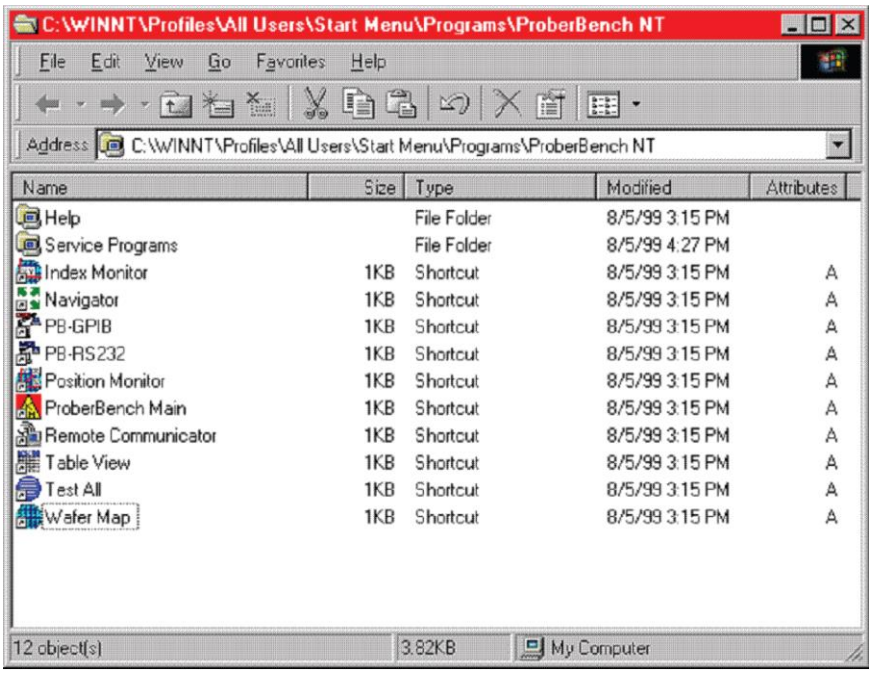
1. Double-click the ProberBench NT icon (shortcut) on desktop.
2. Double-click the Service Programs file.
3. Double-click the Prober Setup file in the Service Programs directory. The Prober Setup window appears (see the below graphic).

Figure 733: Prober setup: PC Config tab



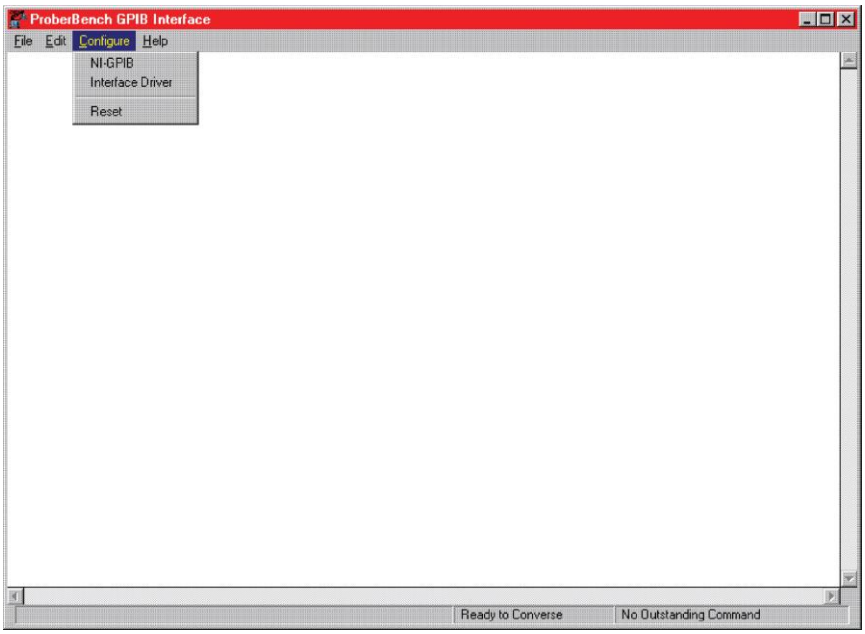
4. Select the NI-GPIB Option.
5. Clear the Simulate Prober box in the Server Mode section of the dialog box.
6. Click **Save Setup**.
7. From the ProberBench NT window, double-click the PB-GPIB file.

Figure 734: ProberBench NT window



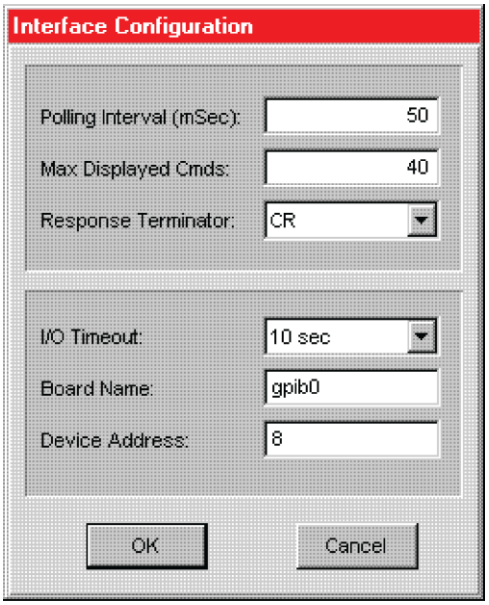
8. From the ProberBench GPIB Interface, from the Configure menu, select **Interface Driver.**

Figure 735: ProberBench: GPIB interface



9. From the Interface Configuration dialog box, change Response Terminator to CR.

Figure 736: Interface Configuration dialog box



10. GPIB only: Ensure that the GPIB address matches the address in the configuration file.
11. Click **OK**.

NOTE

Leave the **Suss RS232 on COMM2** dialog box open. This ensures its services are available for the WaferMap program.

Set up wafer geometry

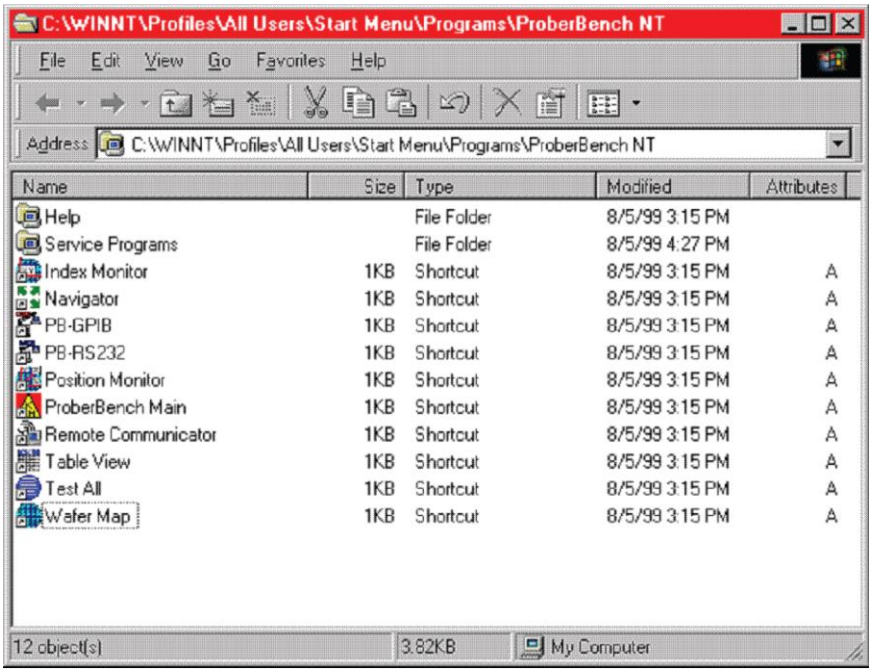
On the ProberBench NT computer:

1. Select the ProberBench **NT** icon (shortcut) on the desktop.

Figure 737: ProberBench NT icon

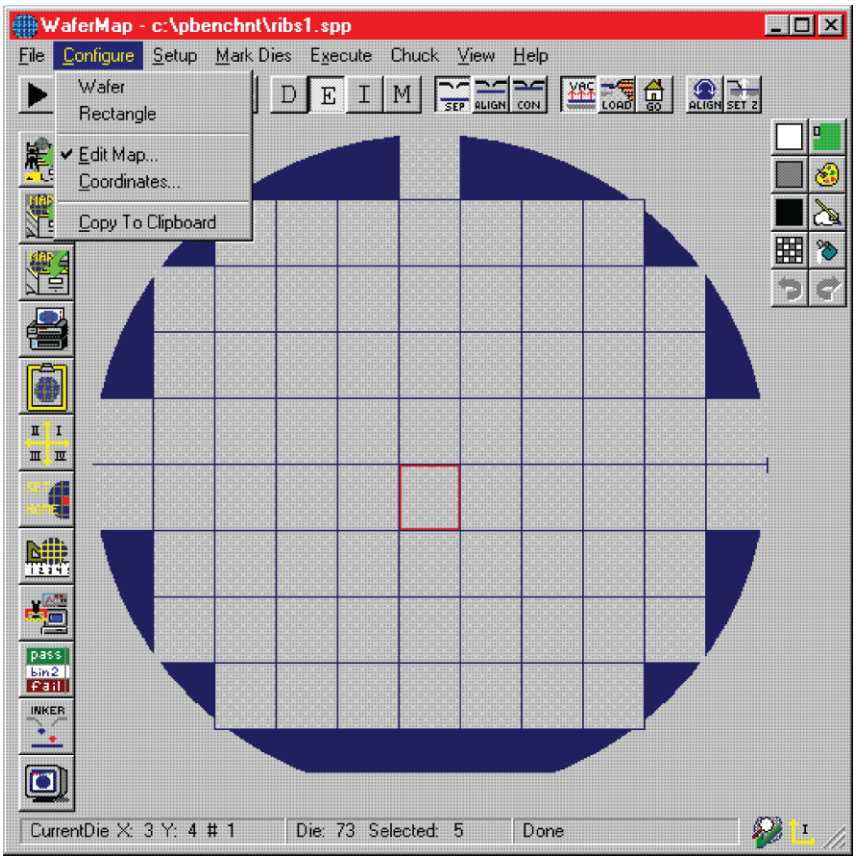
2. From the ProberBench NT window, select the **Wafer Map** file.

Figure 738: ProberBench NT window



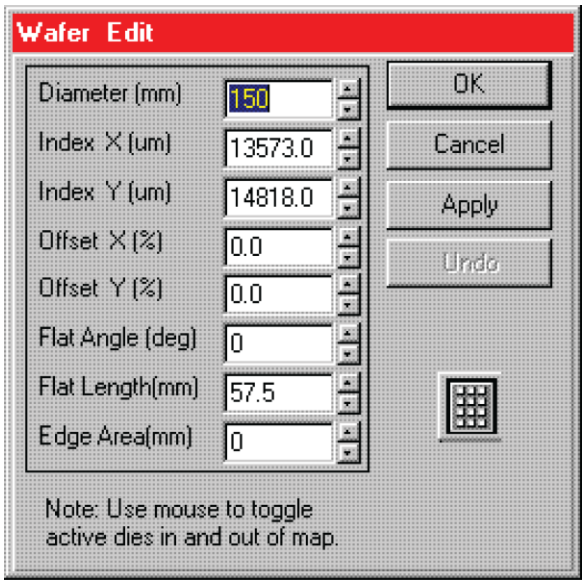
3. From the WaferMap dialog box, create or open a WaferMap.

Figure 739: WaferMap dialog box



4. From the Configure menu, select **Edit Map**.
5. Enter the wafer geometry values and click **Apply**.
6. Click **OK**.

Figure 740: Wafer Edit dialog

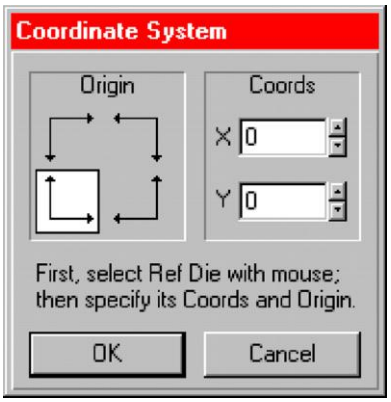


7. From the Configure menu, select **Coordinates**.

Figure 741: Configure pull-down

8. From the Coordinate System dialog box, set **Origin**, as shown. You can set any initial X and Y coordinates.

Figure 742: Coordinate System dialog box



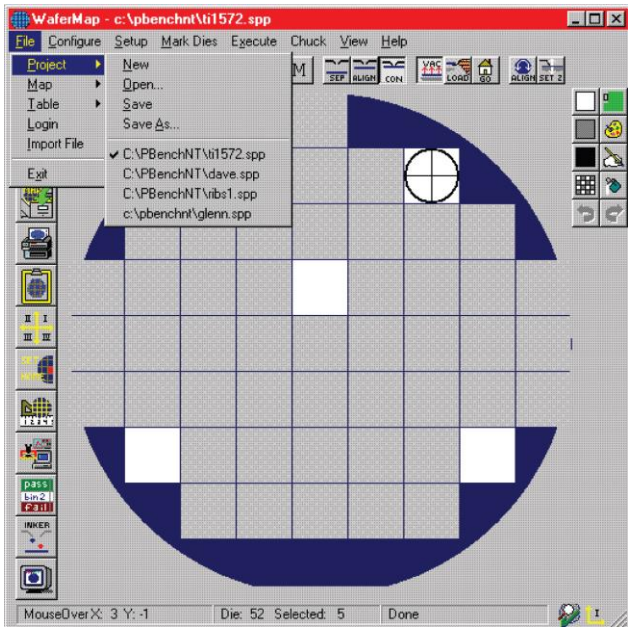
9. Click **OK**.

NOTE

Refer to Clarius `probesites` and `probesubsites` examples for specifics on selecting sites to probe.

10. Select **File > Project > Save** to save the WaferMap settings.

Figure 743: pa200 WaferMap: Save



Create a site definition and define a probe list

Creating a site definition for single subsites for each 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 Clarius project example (on page H-20). Creating a site definition for multiple subsites for each die 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 for each die will be probed, refer to Probesubsites Clarius project example (on page J-27).

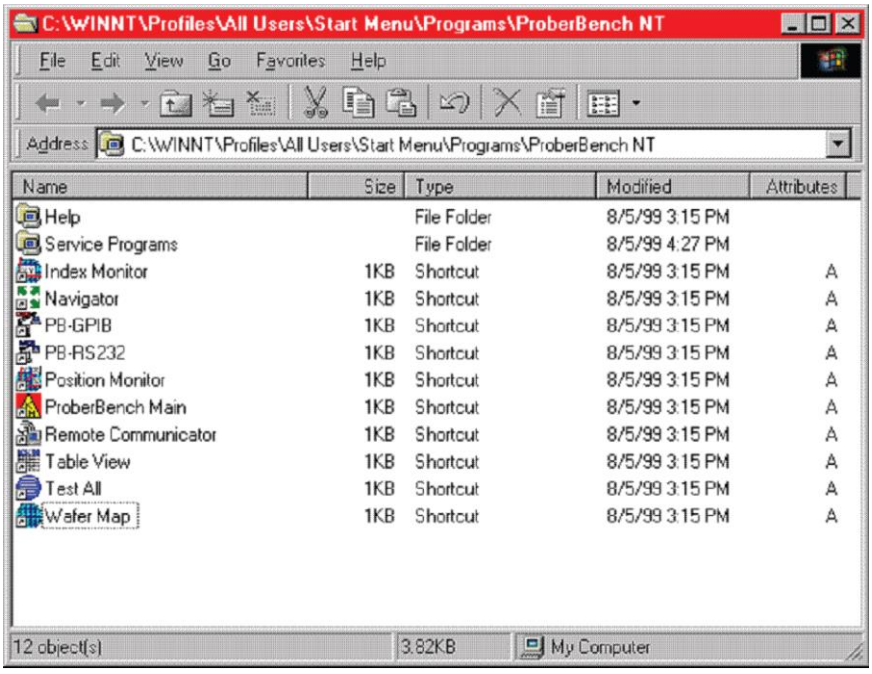
To load a previously defined and saved site definition and a probe list:

1. Select the ProberBench **NT** icon on the desktop.

Figure 744: ProberBench NT icon

2. From the ProberBench NT window, select the **WaferMap** file.

Figure 745: ProberBench NT window



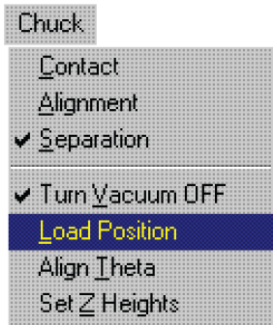
3. From the WaferMap window, select and open the appropriate file.

Load, align, and contact the wafer

Using the ProberBench NT computer:

1. From the WaferMap Chuck menu, select **Load Position**. This brings the chuck to the front of the prober.

Figure 746: Chuck menu

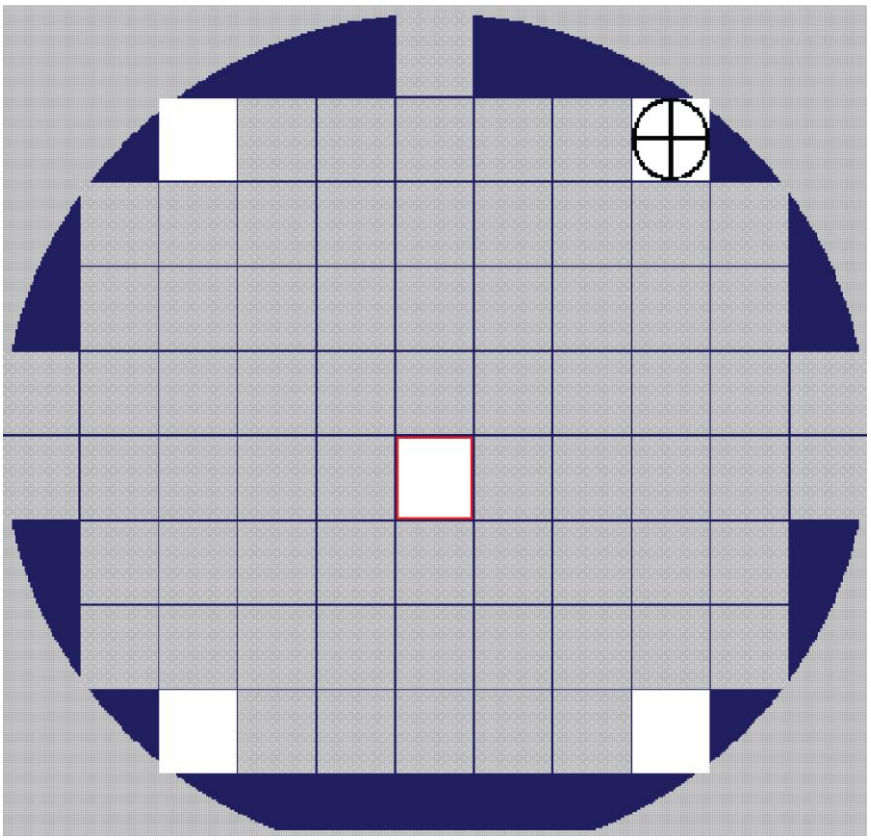


2. Place the wafer on the chuck.
3. From the Chuck menu, select **Turn Vacuum ON**.
4. Manually move the wafer to the Home Die.
5. From the Setup menu, select **Home Die**.

Figure 747: Setup menu

6. Choose the home die on the WaferMap. 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 UI that will be the home die.
   - A cross-hair appears when a die has been selected as the home die.

Figure 748: WaferMap home die selection

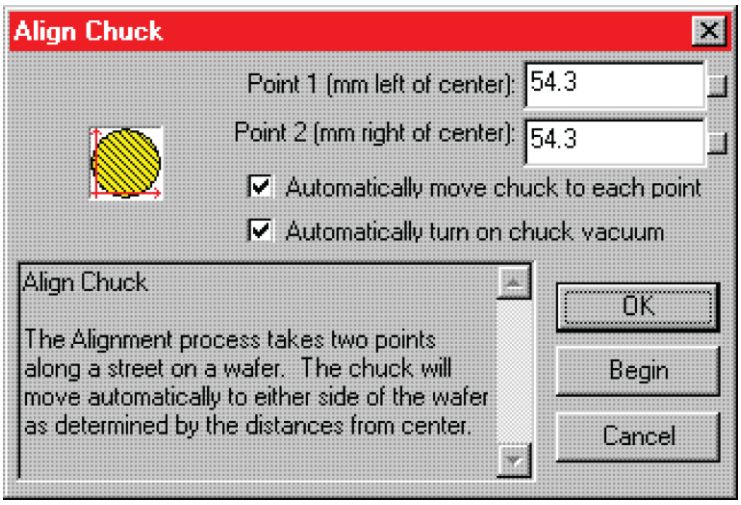


7. From the Chuck menu, select **Align Theta**.
8. Align wafer using the following steps.

Aligning the wafer

1. Enter Point 1 and Point 2 distances from the center using specific X die size multiples. See the following figure. In other words, if the die size is: X = 13.573 mm, and Y = 14.818 mm, set up to move four die to the left and also the right at 54.292 mm (4 • 13.573 mm = 54.292 mm).

Figure 749: Aligning the wafer Step a



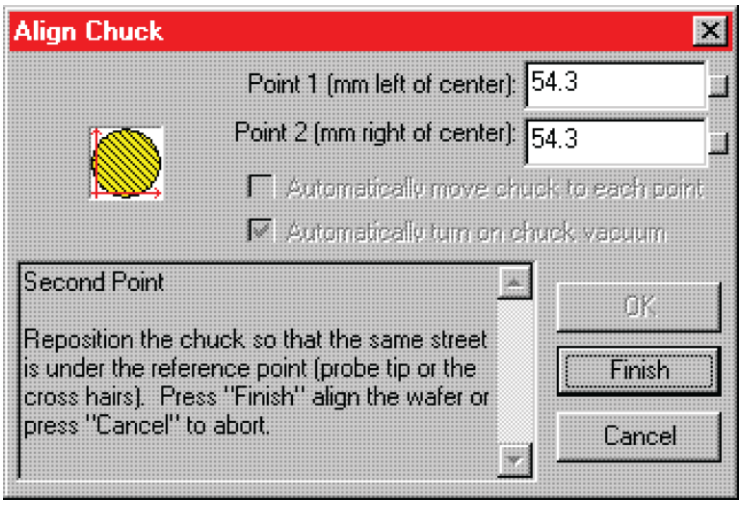
2. Select **Automatically move chuck to each point**.
3. Select **Automatically turn on chuck vacuum**.
4. Click **Begin**.

Start the Alignment Wizard

1. Move to Point 1 (left of center die align pad and pins).
2. Click **Continue** to start the Alignment Wizard.
3. Manually align pins and pads (POINT 1).
4. Click **Continue** (from POINT 1) and move 8 die (for this example) to the right.

5. Manually align pins and pads (POINT 2) and select **Finish**.

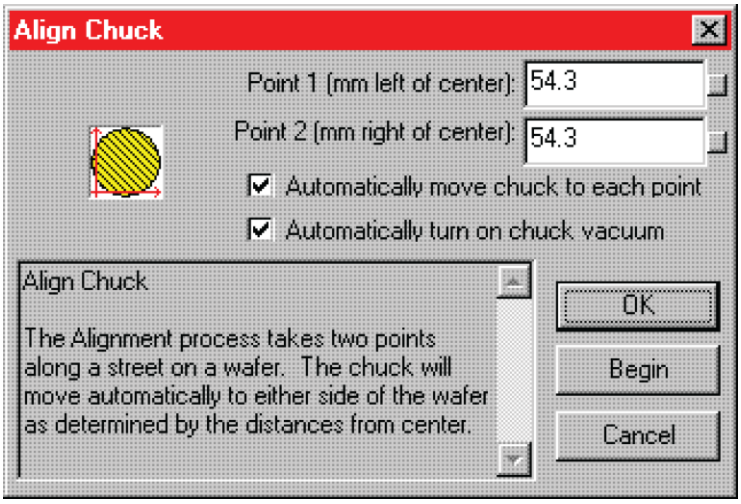
Figure 750: Aligning the wafer: Point 2



Verify wafer alignment

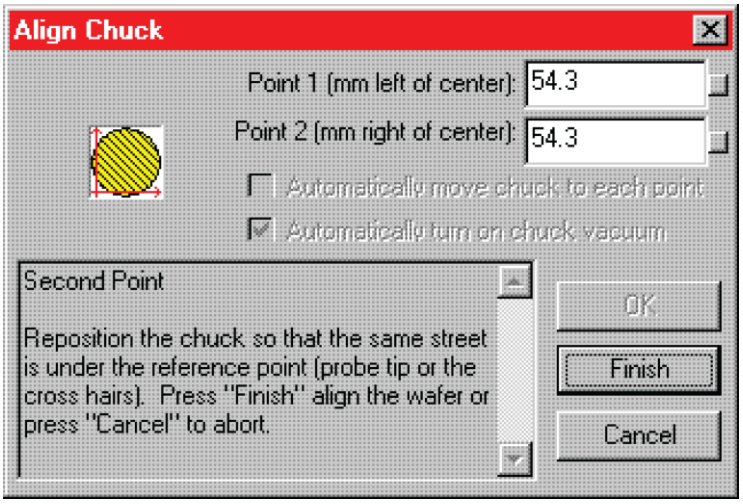
Confirm that the alignment is correct (the alignment procedure is repeated). To check, manually use the joystick to move the chuck in index moves and confirm that the pins and pads are aligned.

Figure 751: Verify wafer alignment



If the alignment is not correct, repeat the alignment. If the alignment is correct, click **Finish**.

Figure 752: Aligning the wafer: Point 2

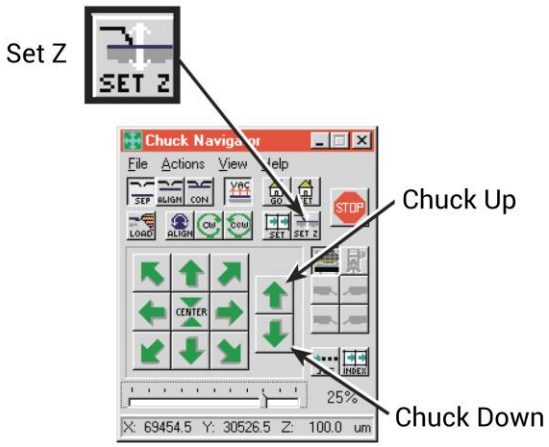


Set the chuck heights

To set the chuck heights:

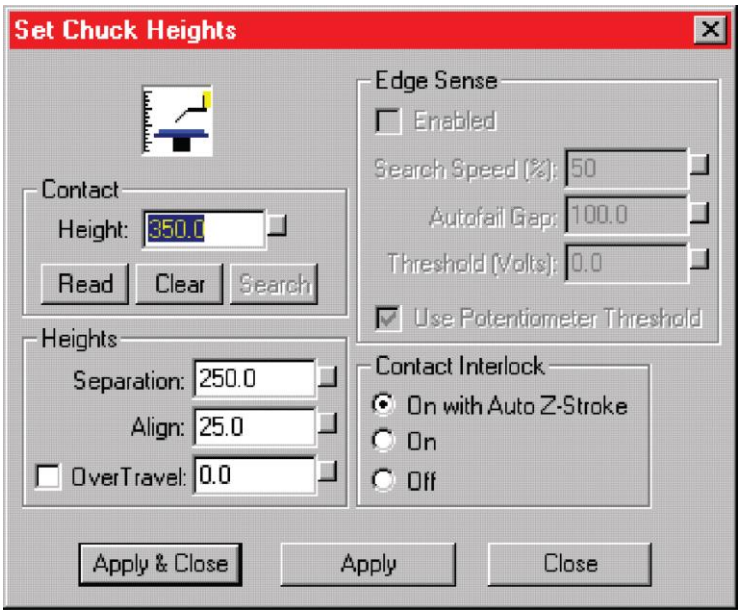
1. Launch the navigator from the ProberBench NT window icon.
2. In the Chuck Navigator dialog box, use the chuck up and down arrows to make contact with the wafer on the home die and home subsite.

Figure 753: Chuck navigator dialog box wafer height



3. Click **Set-Z**. The Set Chuck Heights dialog box is displayed.

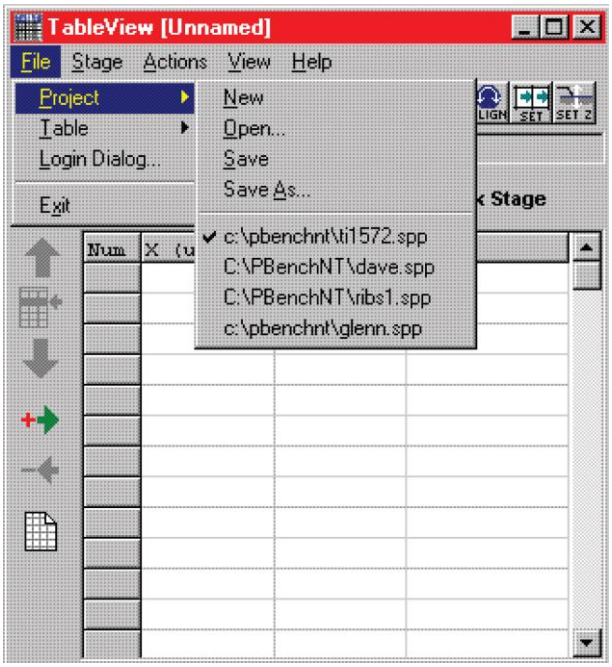
Figure 754: Set chuck heights



4. Click **Read**. The contact height value changes to the present height.
5. Click **Apply**.
6. Click **Close**.

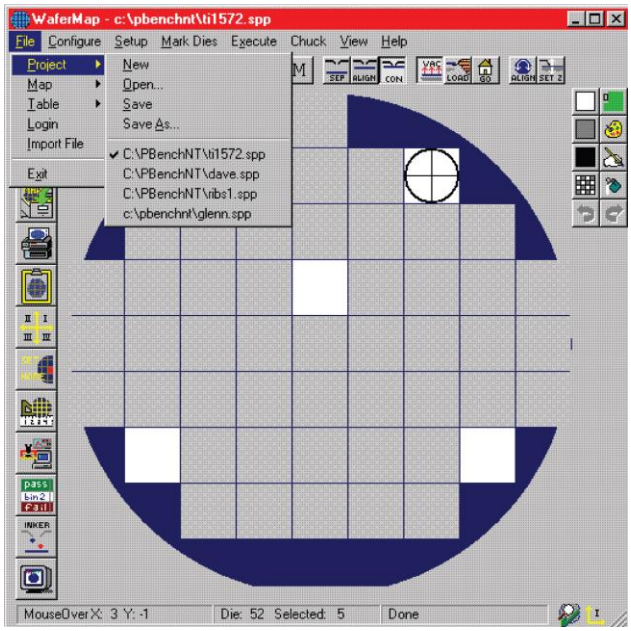
7. Select **File > Project > Save** to save the Chuck Navigator settings.

Figure 755: PA200 Chuck Navigator: Save



8. Select **File > Project > Save** to save the WaferMap configuration.

Figure 756: PA200 WaferMap: Save



Clarius probesubsites project example

The following is a step-by-step procedure to configure the PA-200 so the `probesubsites` Clarius project executes successfully.

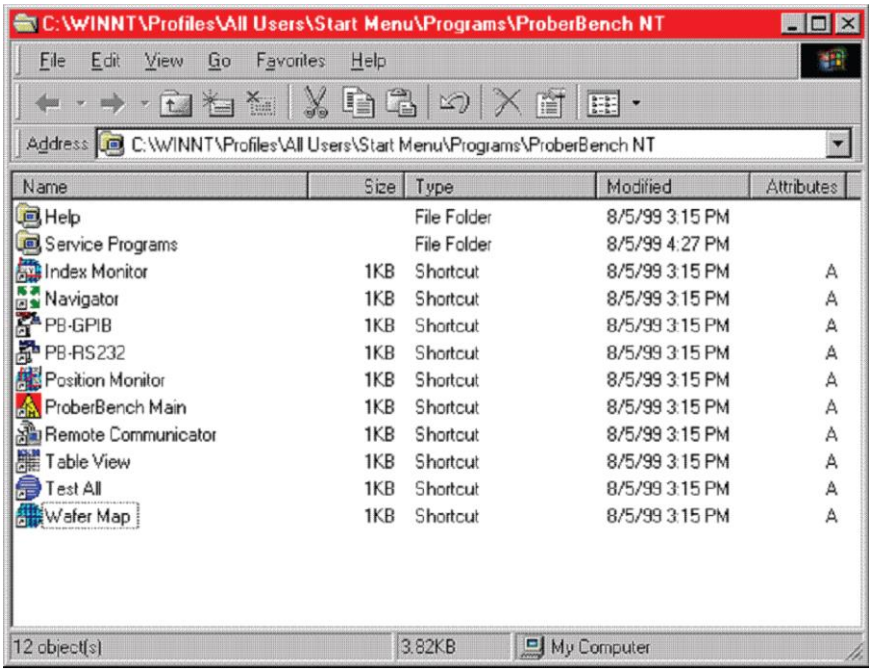
On the ProberBench NT computer:

1. Select the ProberBench NT icon (shortcut) on the desktop.

Figure 757: ProberBench NT icon

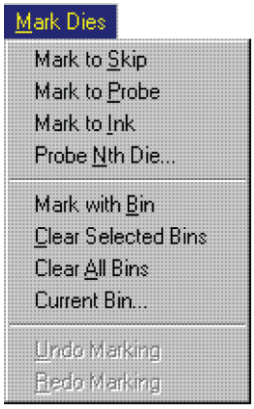
2. From the ProberBench NT window, select **WaferMap** file.

Figure 758: ProberBench NT window



3. From the WaferMap window, from the Mark Dies menu, select **Mark to Skip**.

Figure 759: Mark Dies menu

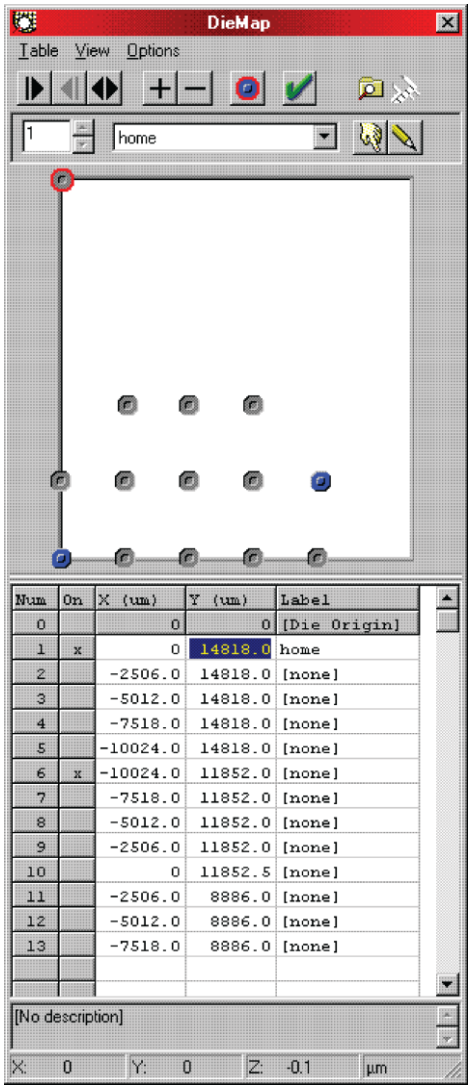


4. Use Mark to Skip and Mark to Probe to set dies. Click a die in the WaferMap window to either set or clear the die. The color of the die indicates status (either probe or skip). With Mark to Probe selected, drag to select multiple dies. With Mark to Skip selected, drag to clear multiple dies. When done, clear **Mark to Skip** or **Mark to Probe**. Otherwise, the Chuck menu remains grayed.
5. From the View menu, select **Die Map**.

Figure 760: View menu

6. Set up the die map.
7. From the View menu, select the Table editor. The spreadsheet portion of the Die Map is displayed.

Figure 761: DieMap dialog



8. From the Options menu, select units (Microns or Mils).
9. Edit the table with the coordinates of the subsites.
10. From the Table menu, select **Save** or **Save As**.

NOTE

An x in the On column defines the subsites that will be probed when using the subsite probing project (in other words, when using **PrSSMovNxt**). Other subsites may be defined in the list.

Set the wafer map

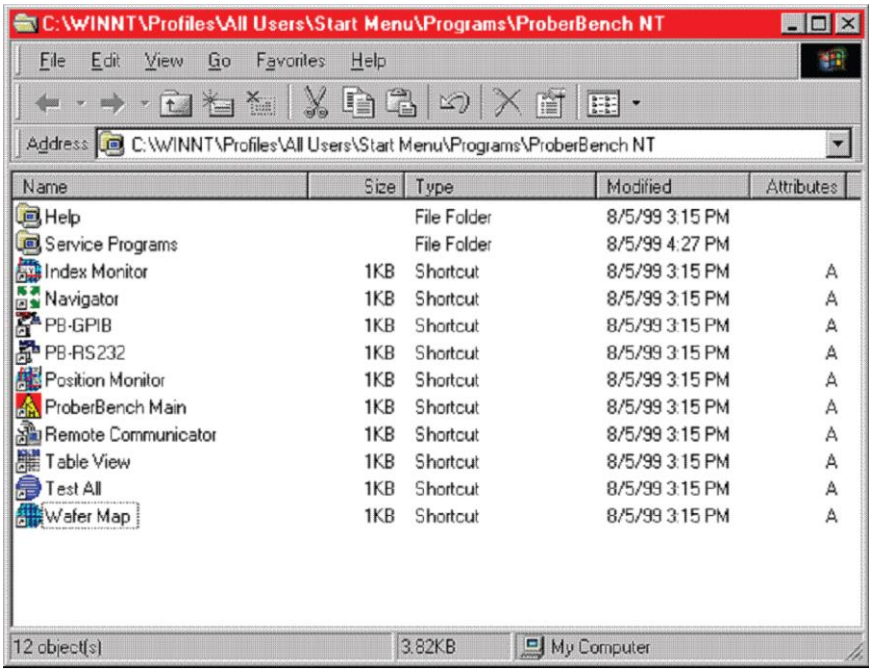
On the ProberBench NT computer:

1. Select the **ProberBench NT** icon on the desktop.

Figure 762: ProberBench NT icon

2. From the ProberBench NT window, select the **Wafer Map** file.

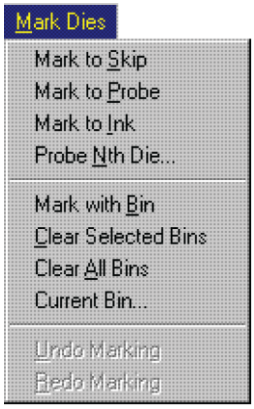
Figure 763: ProberBench NT window



3. From the Mark Dies menu, use **Mark to Skip** and **Mark to Probe** to set dies. Click a die in the WaferMap window to either set or clear the die. The color of the die indicates status (probes white dies, skips blue dies).

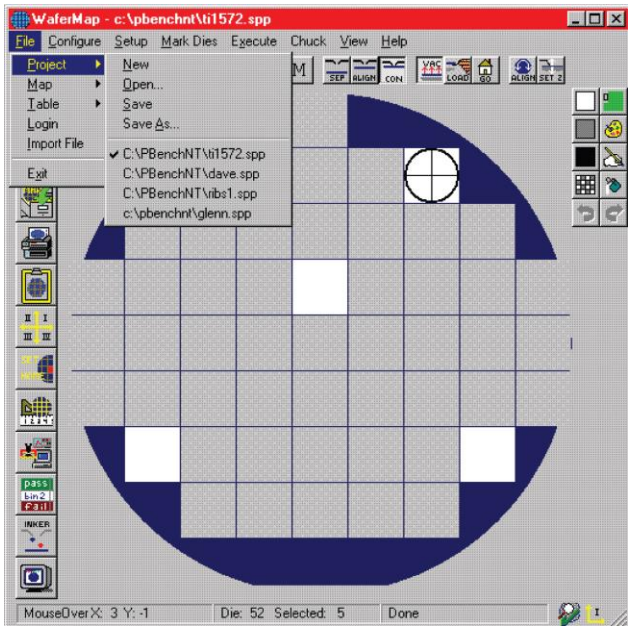
   With **Mark to Probe** selected, drag to select multiple dies. With **Mark to Skip** selected, drag to clear multiple dies. When done, clear **Mark to Skip** or **Mark to Probe**. Otherwise, the Chuck menu remains grayed.

Figure 764: Mark Dies menu



4. Select **File > Project> Save** to save the WaferMap configuration.

Figure 765: PA200 WaferMap: Save

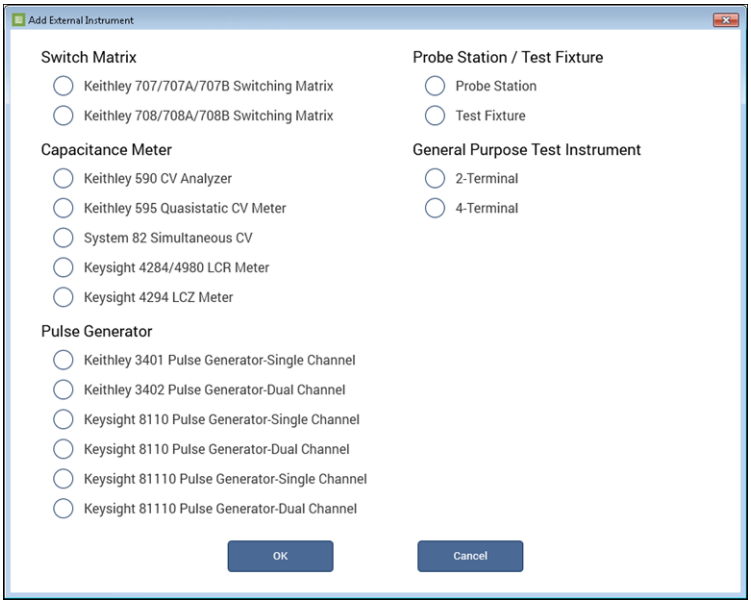


Use KCon to add a prober

On the 4200A-SCS, use KCon to add the prober to the configuration:

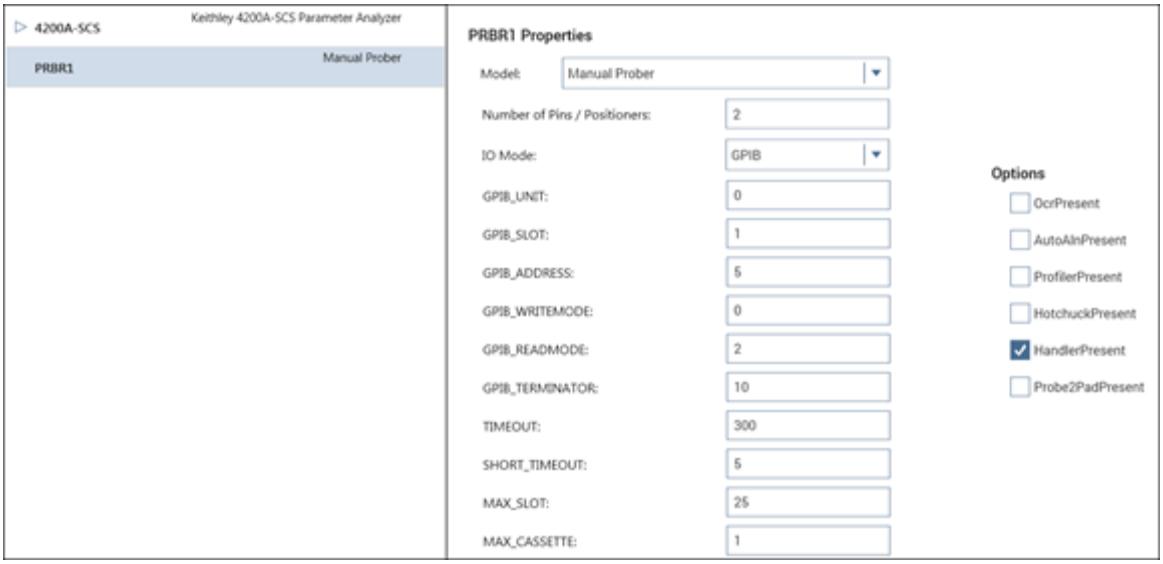
1. Open KCon.
2. At the bottom of the Configuration Navigator, select **Add External Instrument**. The Add External Instrument dialog box is displayed.

Figure 766: Add a prober in KCon



3. Select **Probe Station**.
4. Select **OK**. The KCon navigation displays PRBR1 in the Configuration Navigator and the properties.

Figure 767: Use KCon to select a prober



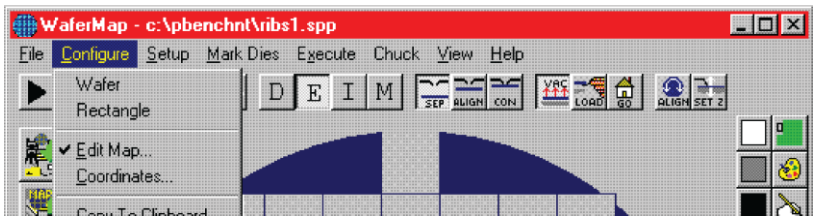
5. Select the **Karl Suss PA200 Prober** as the model.
6. Ensure that the **Number of Pins / Positioners** is correct. The number of pins defined here determines the pins that are available to assign to a switch matrix card column.
7. Select **Save**.
8. Exit KCon.

Running projects

On the ProberBench NT computer:

1. After the wafer is set up and alignment is complete, select the **File > Project > Save**.
2. Select **File > Map > Save**.
3. Select **File > Table > Save**. The wafer is ready to probe.
4. Place the prober in Run mode.
5. Ensure that the "E" in the **WaferMap** toolbar is selected.

Figure 768: WaferMap toolbar



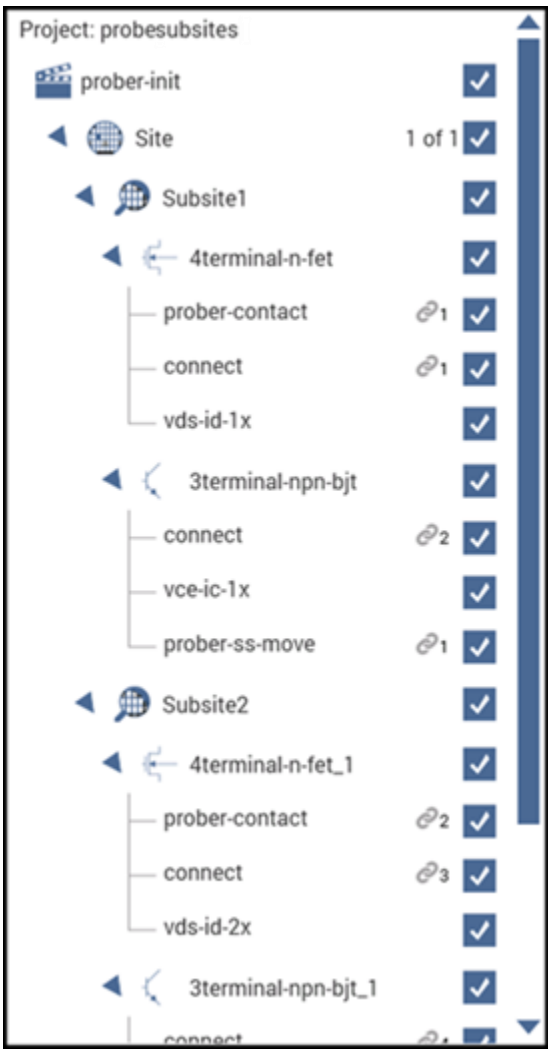
Clarius

Use Clarius to load and run the `probesites` or `probesubsites` project using the new KCon configuration file, which allows you to execute the project for this prober.

On the 4200A-SCS:

1. Open Clarius.
2. Choose **Select**.
3. Select **Projects**.
4. Search for **probesubsites**.
5. Drag the **probesubsites** project to the project tree.

Figure 769: probesubsites project tree



6. Click **Run**.

Commands and error symbols

The following table contains error and status symbols listed by command.

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

Information and error code return values and descriptions

Value	Constant	Explanation
1	PR_OK	Success (OK)
4	PR_WAFERCOMPLETE	Next wafer loaded (confirmed)
-1008	INVAL_MODE	Invalid mode
-1011	BAD_MODE	Operation invalid in mode
-1013	UNINTEL_RESP	Unintelligible response
-1015	UNEXPE_ERROR	Unexpected error
-1017	BAD_CHUCK	Bad chuck position
-1027	INVAL_PARAM	Invalid parameter

Appendix H

Using a Micromanipulator 8860 Prober

In this appendix:

Micromanipulator 8860 prober software

You need to have the following software programs on the Micromanipulator 8860 to configure and operate the 8860 prober with the Keithley Instruments 4200A-SCS:

- **pcBridge**: Used to configure the communications setup (icon on the desktop)
- **pcLaunch**: Used to launch various wafer controls and utilities (icon on the desktop)
- **pcIndie**: Used to probe multi-subsites per die (button in pcLaunch window)
- **pcWafer**: Used to probe single subsites per die (button in pcLaunch window)
- **pcNav**
- **pcRouter**

NOTE

pcIndie and pcWafer, which are not included with standard prober software, are required. Refer to the prober manufacturer, Micromanipulator, for availability.

Software versions

The following list contains the software versions used to verify the configuration of the 8860 prober with the 4200A-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:	2.0.9
Product Name:	pcWafer
Product Version:	2.0.8
Product Name:	pcRouter
Product Version:	2.0.9

Probe station configuration

CAUTION

Ensure that you are familiar with the Micromanipulator 8860 prober and its supporting documentation before attempting setup, configuration, or operation.

To set up and configure the 8860 prober for use with the 4200A-SCS, you will:

- Set up communications (on page H-3)
- Set up wafer geometry (on page H-8)
- Create a site definition and define a probe list (on page H-9)
- Load, align, and contact the wafer (on page H-11)

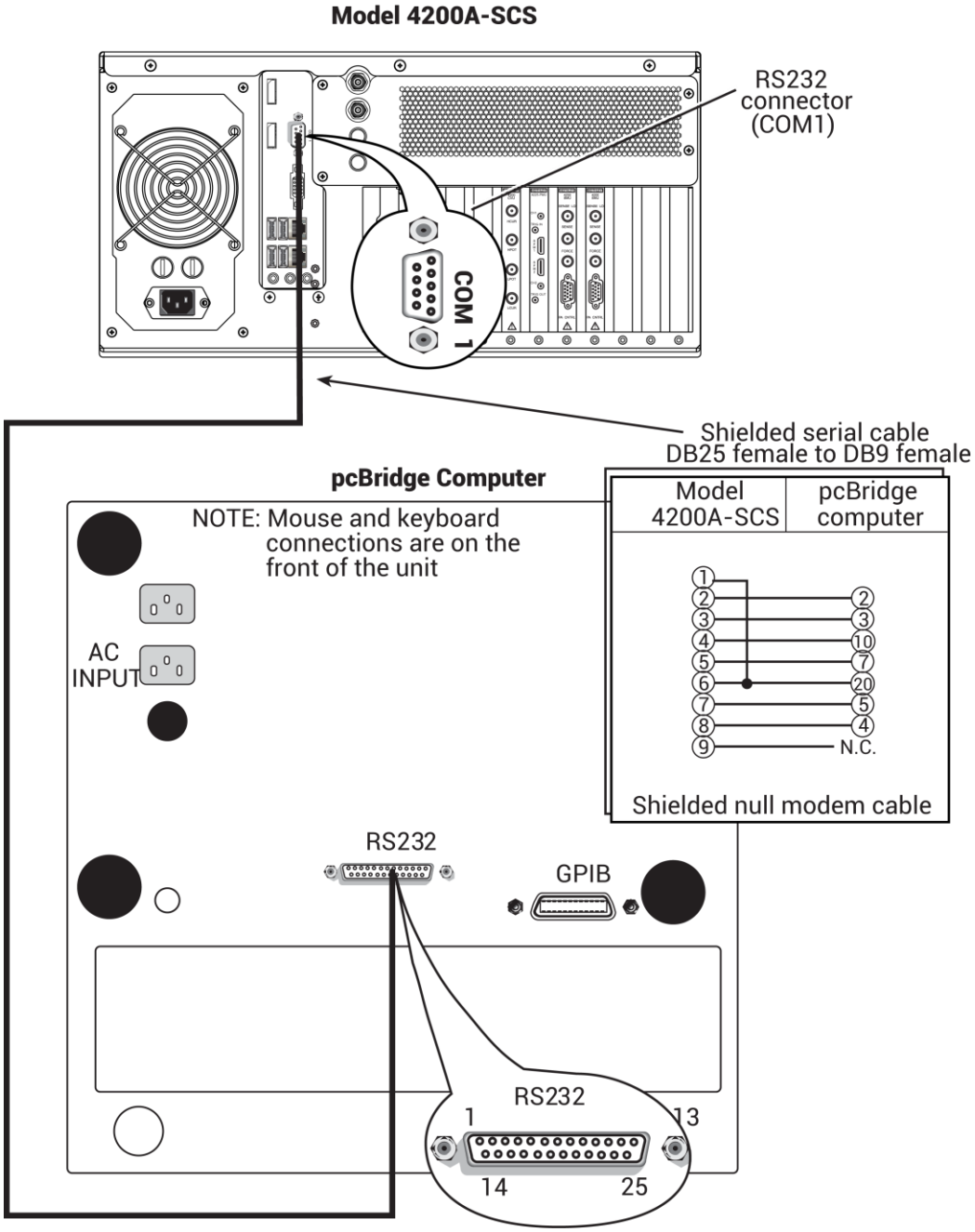
Each step is detailed in the following topics.

Set up communications

To set up communications:

1. Turn on power to the 4200A-SCS.
2. Turn on power to the prober.
3. Ensure that the vacuum has been properly connected.
4. On the pcBridge computer, connect the pcBridge computer's RS232 port (on the rear panel of the pcBridge computer) to the 4200A-SCS COM1 port. Use a DB25 female to DB9 female cable (shielded null modem cable). See the figure below for details.

Figure 770: Prober setup: Serial connections



5. Double-click the **pcBridge** icon on the desktop to open the main pcBridge window.

Figure 771: pcBridge icon

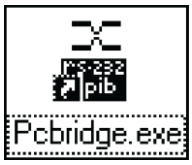
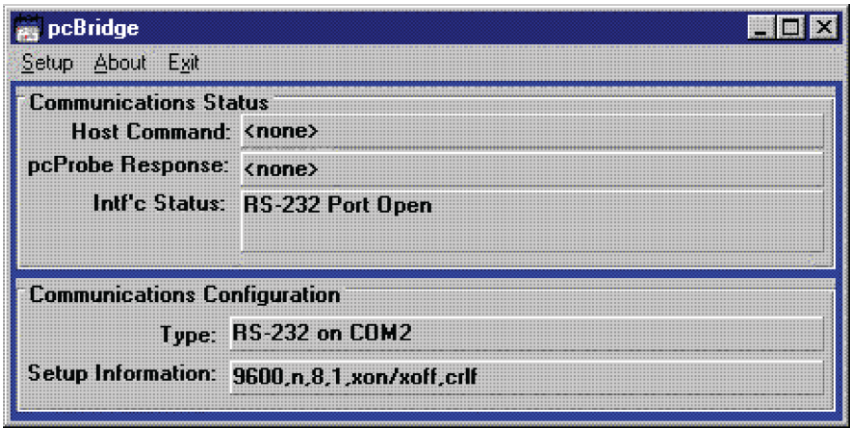
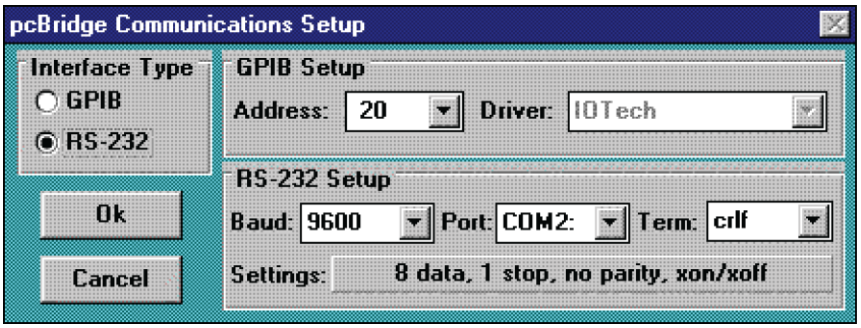


Figure 772: Prober setup: Main pcBridge window



6. Select the **Setup** menu. The pcBridge Communications Setup window is displayed.

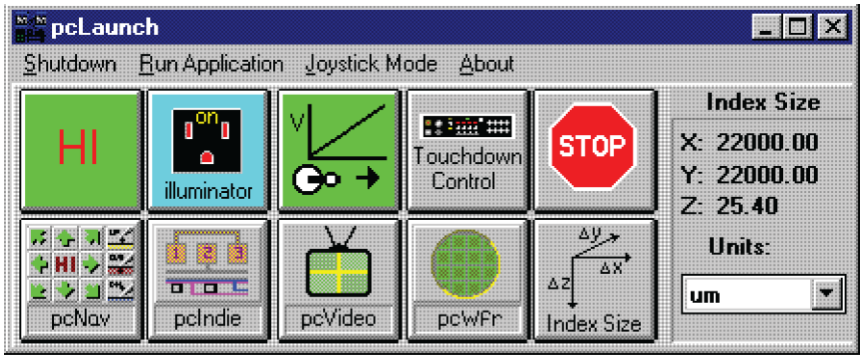
Figure 773: pcBridge window



7. Use the pcBridge Communications Setup to configure the communications settings. These settings should be 8 data, 1 stop, no parity, xon/xoff.
   - Interface Type: RS232
   - Baud: 9600
   - Port: COM2
   - Term: cr and lf (termination character of carriage-return and line-feed)
8. Click **OK**.
9. Click the **pcLaunch** icon to open the main pcLaunch window.

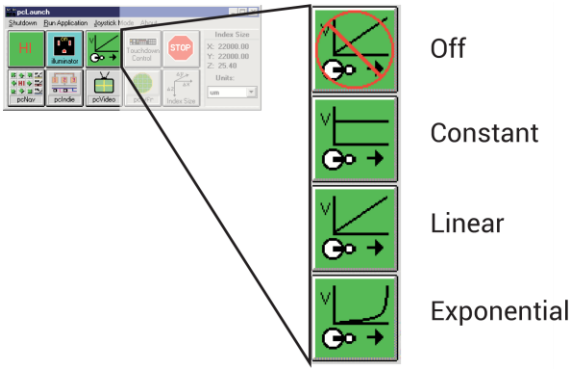
Figure 774: pcLaunch icon

Figure 775: pcLaunch window



10. From the pcLaunch window, set the Joystick Mode for Linear.

Figure 776: Joystick modes



Modify the prober configuration file

The default prober configuration file is shown below. As shown, the file is configured for use with serial communications.

Configuration file location: `C:\s4200\sys\dat\prbcnfg_MM40.dat`

Use the 4200A-SCS to modify the file if needed.

```
# prbcnfg.dat - EXAMPLE Prober Configuration File for MM40 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:
#
#  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
#
#
```

Set up wafer geometry

On the pcBridge computer:

1. From the pcLaunch window, click the **pcNav** button to open the pcNav window.

Figure 777: pcNav button

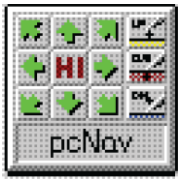
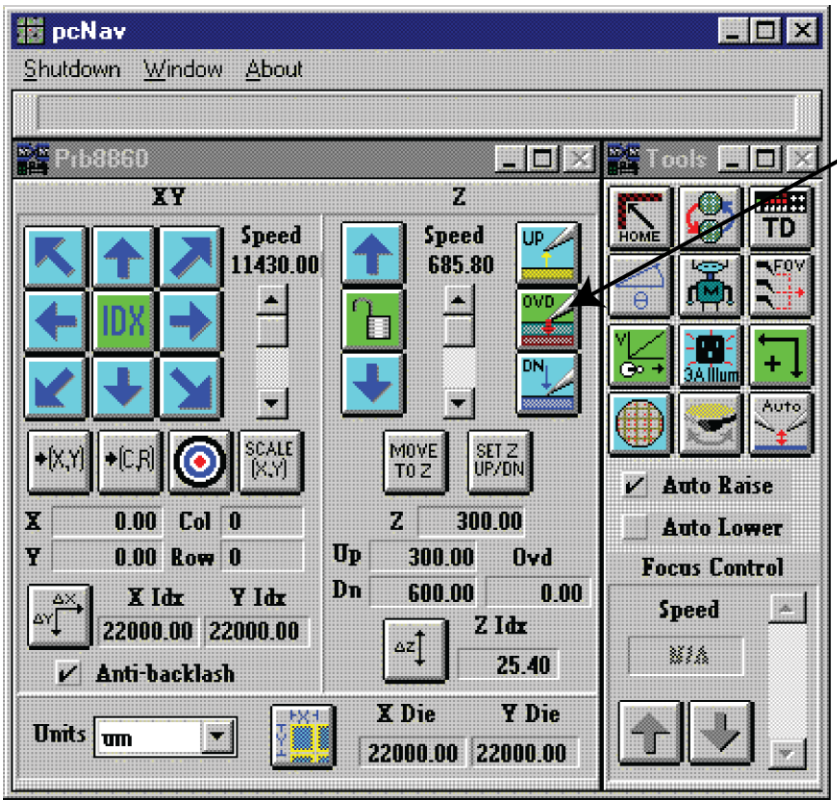


Figure 778: pcNav window



OVD toggle

NOTE

When starting pcNav for the first time, the warning in the following figure is displayed. Click **OK** and continue the configuration (the device will be initialized when the chuck is homed).

Figure 779: pcNav Boot warning

NOTE

Since the platen moves to make or break contact between the pins and pad, selecting **Auto Raise** will automatically separate the pins from the pads. **Auto Lower** will allow automatic contact.

2. Select Auto Raise on the pcNav Tools window.
3. Select Anti-backlash on the pcNav Prb8860 window.

Create a site definition and define a probe list

On the pcBridge computer, create a site definition for a single subsite for each die. To do this, use the software to create a selection of dies to probe. If a single subsite for each die is to be probed, refer to Probesites Clarius project example (on page H-20). Creating a site definition for multiple subsites for each die also uses the software to create a selection of dies to probe and create a selection of the subsites on each die that will be probed. If multiple subsites for each die will be probed, refer to the Probesubsites Clarius project example (on page H-27).

Use the following information to load a previously-defined and saved site definition.

Single subsite per die

To open the file:

1. Start pcWafer by clicking the **pcWfr** button in the pcLaunch window. The pcWfr window is displayed. See the following two figures.

Figure 780: pcWfr button

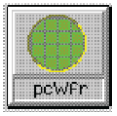
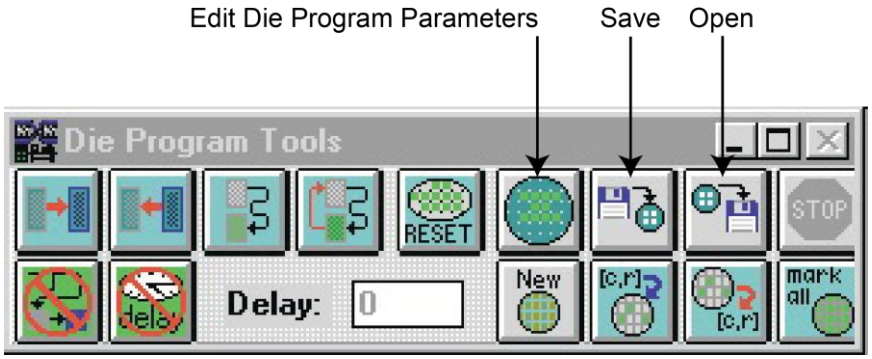


Figure 781: Die Program Tools window



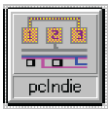
2. Click **Open** on the Die Program Tools window to open an existing file or **New** to create a new file.
3. Select the file and click **OK**.

Multiple subsites per die

To open the file:

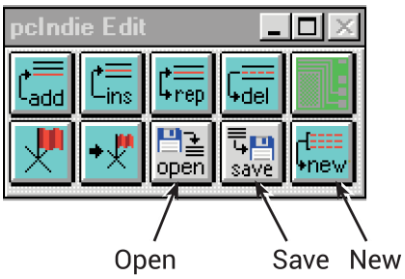
1. Click the **pcIndie** button in the pcLaunch window. The pcIndie window will appear.

Figure 782: pcIndie button



2. Click the **Open** button on the pcIndie Edit window to open an existing file or New to create a new file.

Figure 783: pcIndie Edit window



3. Select the file and click **OK**.

Load, align, and contact the wafer

The following topics describe how to contact the wafer.

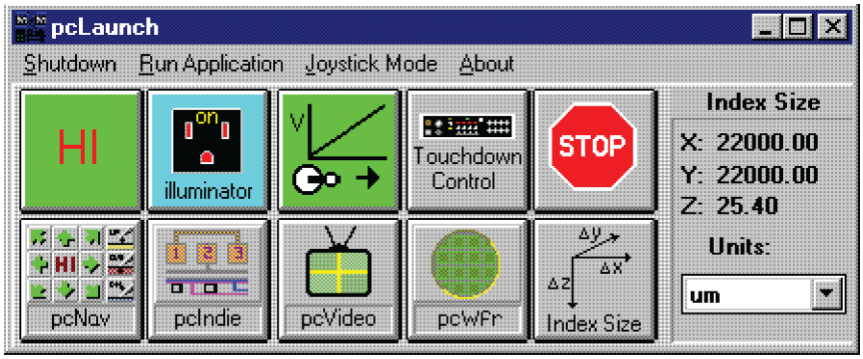
Home the chuck

To home the chuck:

1. On the pcBridge computer, click the pcLaunch icon. The pcLaunch window is displayed. See the following two figures.

Figure 784: pcLaunch icon

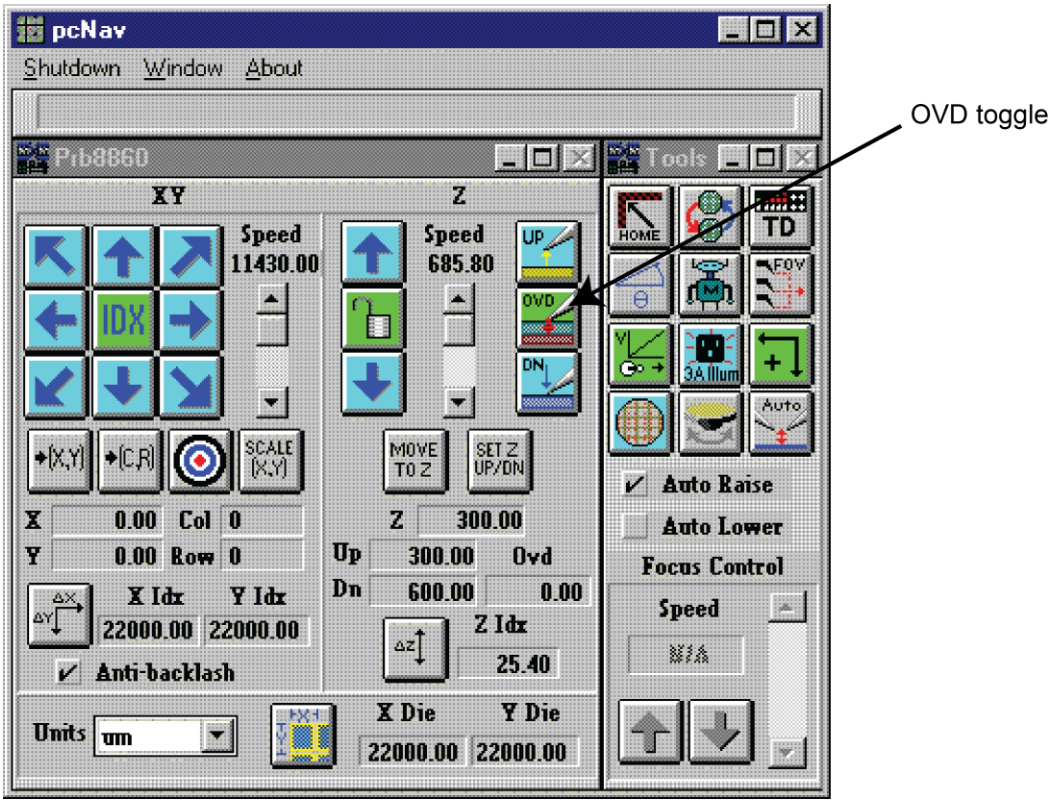
Figure 785: pcLaunch window



2. To home the chuck, from the pcLaunch window, click the **pcNav** button. The pcNav window opens. See the following two figures.

Figure 786: pcNav button

Figure 787: pcNav window



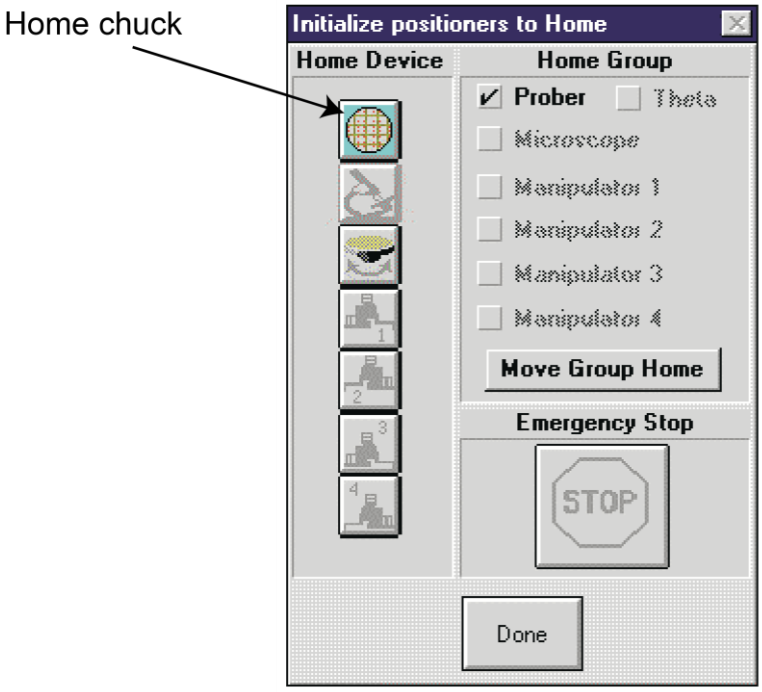
NOTE

The OVD button toggles the state of the overdrive (on or off).

3. Click the **Home** button on the Tools panel of the pcNav window. The Initialize positioners to Home window opens. See the following two figures.

Figure 788: Home button

Figure 789: Initialize positioners to Home window



4. From the Initialize positioners to Home window, click **Home chuck**. The chuck moves to the back left corner and then to the middle.
5. Click **Done** when the chuck is home. The **Done** button turns from grayed to active when the chuck is home.

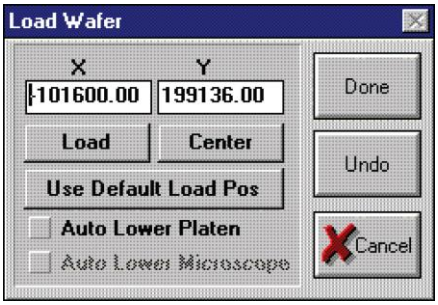
Load the wafer

Load the wafer:

1. Make sure that the vacuum is off.
2. Click the **Load wafer** button on the Tools panel of the pcNav window. The Load Wafer dialog box appears. See the following two figures.

Figure 790: Load Wafer button

Figure 791: Load Wafer window



3. In the Load Wafer dialog box, click **Load**.
4. After the chuck moves to the front, place wafer on the chuck aligning the flat or notch in the proper orientation.
5. Apply vacuum.
6. Click **Center**.
7. Click **Done**.

Set the Z-height

To set the Z-height:

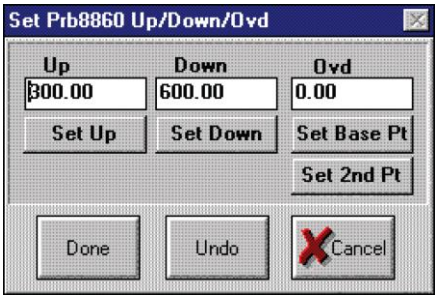
NOTE

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).

1. Use the joystick to manually move the wafer (pads) underneath the pins.
2. Click the **SET Z UP/DN** button on the pcNav dialog box. The SET Prb8860 Up/Down/Ovd dialog box opens. See the following two figures.

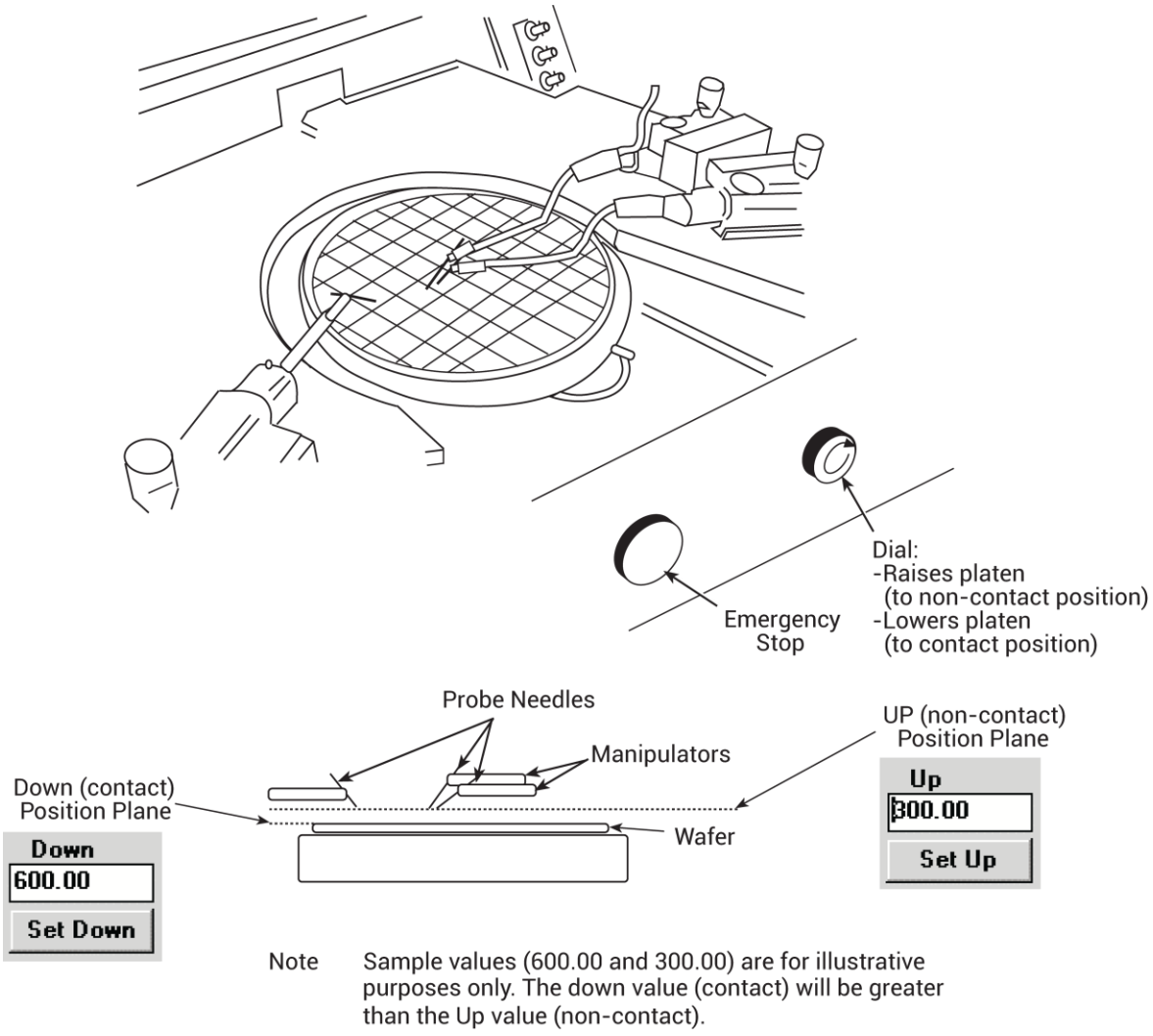
Figure 792: Set Z UP-DN button

Figure 793: SET Prb8860 Up-Down-Ovd window



3. 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 794: Set Z-height



Note Sample values (600.00 and 300.00) are for illustrative purposes only. The down value (contact) will be greater than the Up value (non-contact).

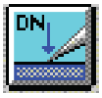
4. Use the manual Z-dial to lower the platen to make initial contact with pads (this assumes that the pins are planar).
5. Click the **Set Down** button when all pins are in contact with their respective pad.
6. Use the Dial to move the pins to a non-contact position (this height in the example is 300).
7. 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.

8. Set overdrive (user preference).
9. In the pcNav window, click the **DN** button, then press the **Set Z UP/DN** button.
10. When the SET Prb8860 Up/Down/Ovd window opens, press the **Set Base Pt** button.
11. Lower the platen to the point where you see good clean probe marks.
12. Click the **Set 2nd Pt** button.
13. Click the **OVD** button in pcNav to ensure that the overdrive will be used. Test the settings by pressing the UP and DN buttons in pcNav.

Figure 795: Down pushbutton



14. Click **Done**.

Align the wafer

To align the wafer:

1. Click the **Align Wafer** button on the Tools panel of the pcNav window. The Prb8860 Alignment window opens.

Figure 796: Align wafer button

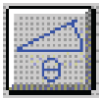
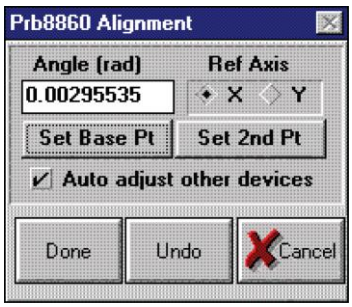


Figure 797: Prb8860 Alignment window



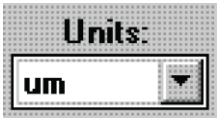
2. Select **Ref Axis X** in the Prb8860 Alignment dialog box.
3. Select **Auto adjust other devices**.
4. Move prober chuck to extreme left of the wafer. Look through the microscope and ensure the pins are over the pads.
5. Click **Set Base Pt**.
6. Move to a die on the extreme right of the wafer.
7. 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).
8. Click **Set 2nd Pt**.
9. Repeat this process until the Angle (rad) is as close to zero as possible.
10. When the alignment is complete, click **Done**.

Set the units and die size

To set the units and die size:

1. Set **Units** to either microns or mils from the Prb8860 window (lower left corner) of pcNav.

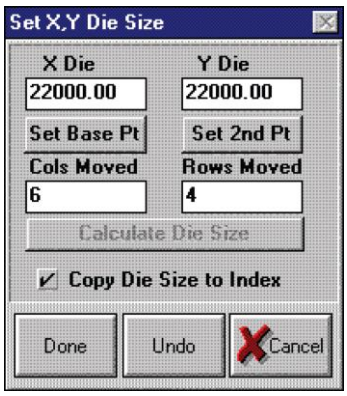
Figure 798: Unit of measure list



2. Click **Set X, Y die size** button 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 (on page H-19) for more information.

Figure 799: Set X, Y die size button

Figure 800: 4200-901_Set X, Y Die Size dialog box



Calculate die sizes

To calculate die sizes:

1. 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).
2. Click **Set Base Pt**.
3. Move over and down a known number of dies. Enter these values into Column moved (columns) and Row moved (rows).
4. Click the **Set 2nd Pt** button.
5. Check the **Copy Die Size to Index** button.
6. Click the **Calculate Die Size** button.

   This determines the accurate die size.
7. To complete, click **Done** (the grayed-out button will turn active after the calculation is completed).
8. Click the **Set Reference Die** button in the Setup Options window to open the Set Reference dialog box.

Figure 801: Set Reference Die button

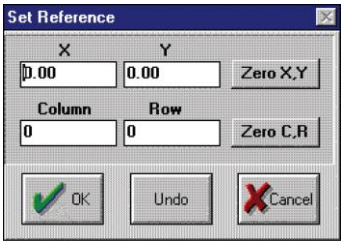
9. Move the pins over the pads of the die to be set as the reference die.
10. Zero out the X/Y, Column, and Row (click Zero X,Y button and Zero C,R).

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 needed before clicking **Done**.

11. Click **Done**.

Figure 802: Set Reference dialog box



Probesites Clarius project example

The following is a step-by-step procedure to configure the 8860 so the `probesites` Clarius project executes successfully.

On the pcBridge computer:

1. Use the pcWafer program to probe a single subsite on multiple dies.
2. Start pcWafer by clicking the **pcWfr** button in the pcLaunch window. The pcWfr window will appear.

Figure 803: pcWfr button

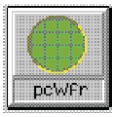
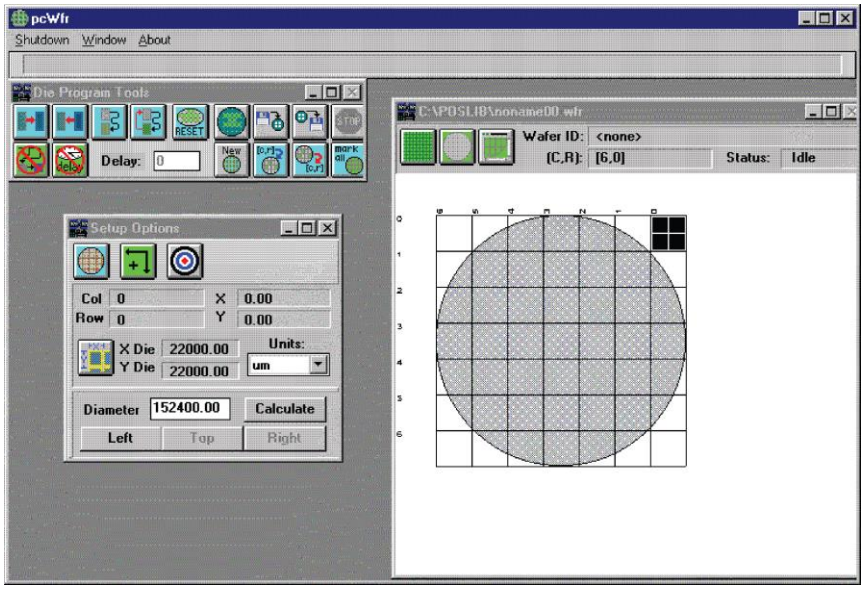


Figure 804: PcWfr window



3. Set units of measure (microns or mils).
4. Calculate the wafer diameter:
    a. Move the pins to the left edge of the wafer.
    b. Click **Left** 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.
5. Set units to either microns or mils from the units of measure Units list in Setup Options window. See the following two figures.

Figure 805: Units of measure list

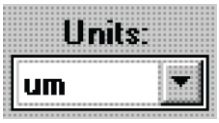
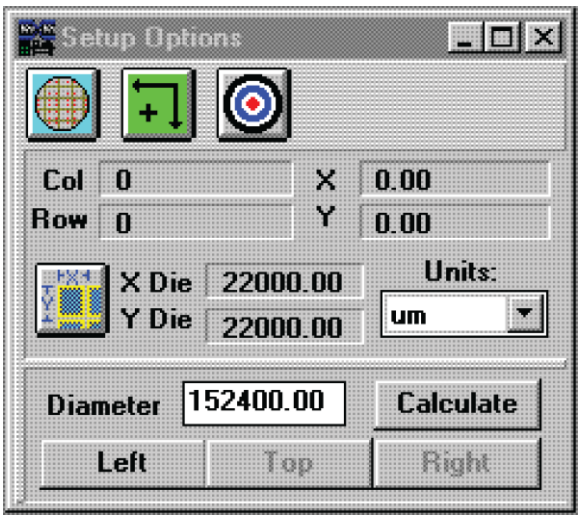


Figure 806: Setup Options window



6. Click Set X, Y die size button in the Setup Options window. The Set X, Y Die size dialog box opens.
7. Set X, Y Die size button.
8. If die size is known, enter it. If not known, calculate (see Calculating die sizes (on page H-19)).

Set spline pattern (optional)

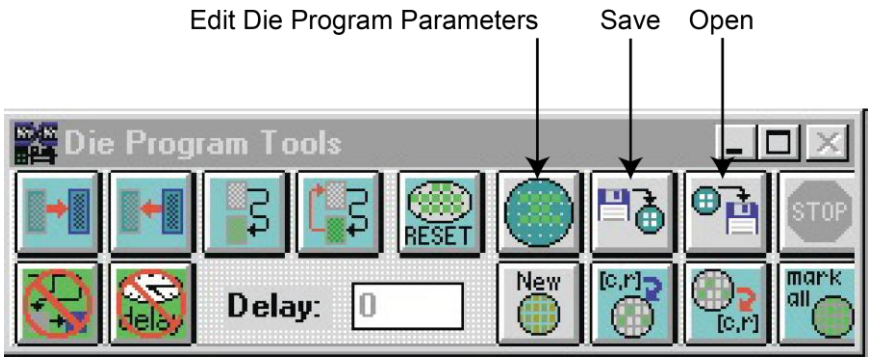
NOTE

The order of selection of the die, the spline pattern (change using edit die program), and the reference die location determine test order sequence.

To set spline pattern:

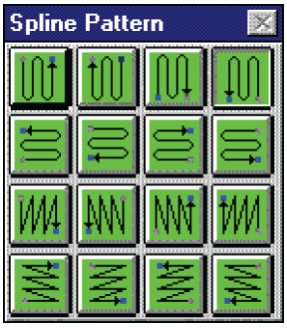
1. Click the **Edit Die Program Parameters** button on the Die Program Tools window of pcWfr. The Edit Die Program Parameters dialog box opens.

Figure 807: Die Program Tools window



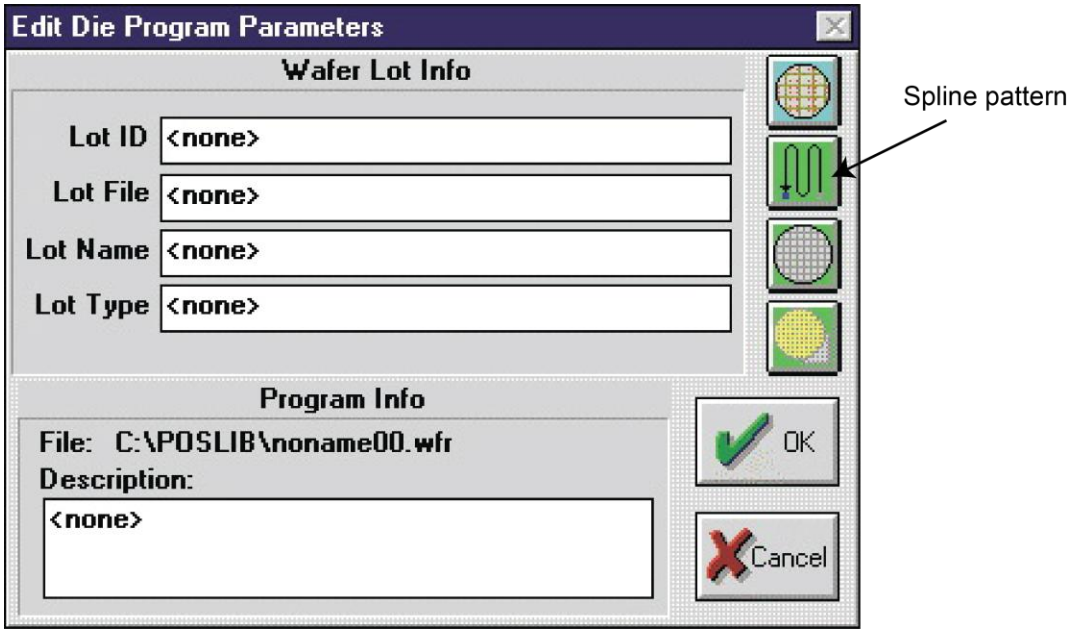
2. Click the **Spline Pattern** button on the Edit Die Program Parameters dialog box. The Spline Pattern window will open.

Figure 808: Spline Pattern window



3. Select the spline pattern. The icon of the active spline pattern is transferred to the Edit Die Program Parameters dialog box, as shown in the following figure.

Figure 809: Edit Die Program Parameters window



4. Click **Save** on the Die Program Tools window.
5. To open an existing program listing file, click the **pcWfr Open** button on the Die Program Tools window. Select the file and click **OK**.

NOTE

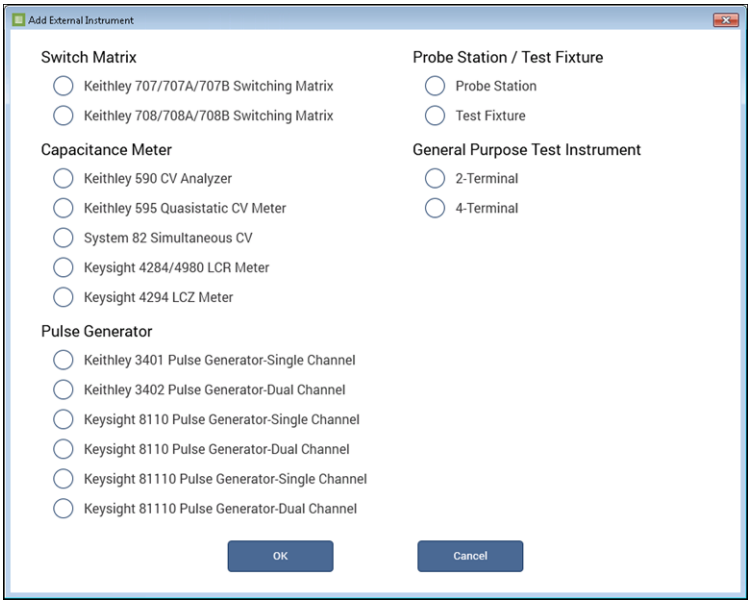
Before starting testing, physically align the pins over the reference die.

Use KCon to add a prober

On the 4200A-SCS, use KCon to add the prober to the configuration:

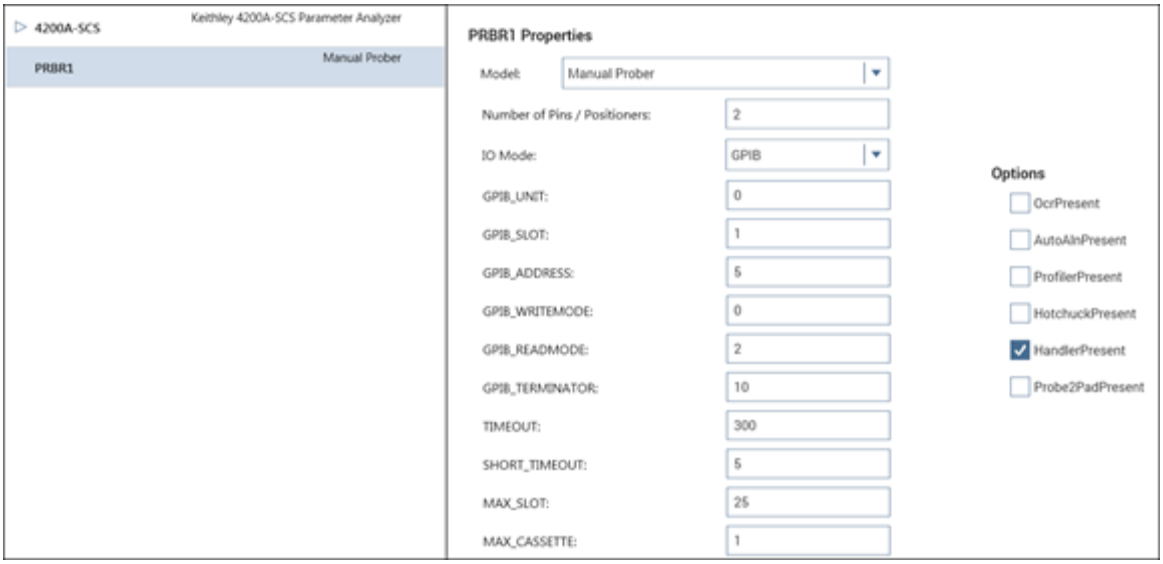
1. Open KCon.
2. At the bottom of the Configuration Navigator, select **Add External Instrument**. The Add External Instrument dialog box is displayed.

Figure 810: Add a prober in KCon



3. Select **Probe Station**.
4. Select **OK**. The KCon navigation displays PRBR1 in the Configuration Navigator and the properties.

Figure 811: Use KCon to select a prober



5. Select the **Micromanipulator 8860 Prober** as the model.
6. Ensure that the **Number of Pins / Positioners** is correct. The number of pins defined here determines the pins that are available to assign to a switch matrix card column.
7. Select **Save**.

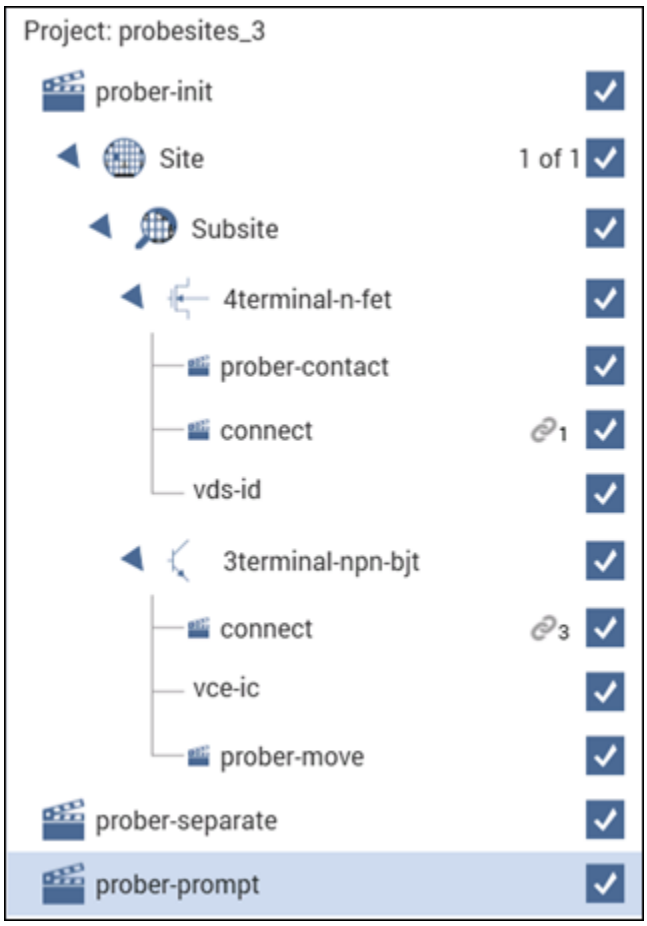
Clarius

Use Clarius to load and run the `probesites` project using the new KCon configuration file, which allows you to execute the project for this prober.

On the 4200A-SCS:

1. Open Clarius.
2. Choose **Select**.
3. Select **Projects**.
4. Search for **probe**.
5. Drag the **probesites** project to the project tree.

Figure 812: probesites project tree



6. Click **Run**.

Probesubsites Clarius project example

The following is a step-by-step procedure to configure the 8860 so the `probesubsites` Clarius project executes successfully.

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.

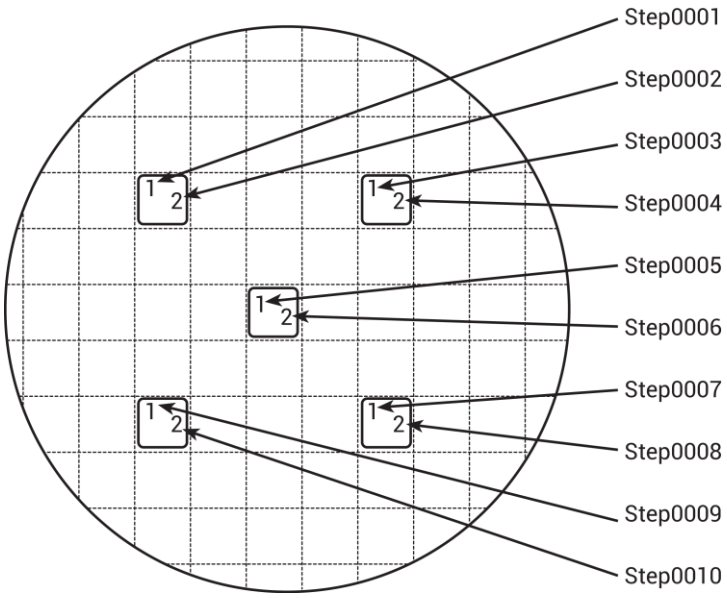
In this example, five dies have been selected for probing. On each die, two subsites have been selected.

Use the pcBridge 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 in the loop for each die to be probed. Refer to the next figure for an illustration of a repeatable pattern.

Figure 813: Multiple subsites per die



NOTE

Ensure that all steps of Setup have been completed before starting pcIndie.

To start pcIndie:

1. Clicking the pcIndie button in the pcLaunch window. The pcIndie window is displayed. See the following two figures.

Figure 814: pcIndie button

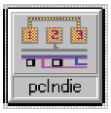
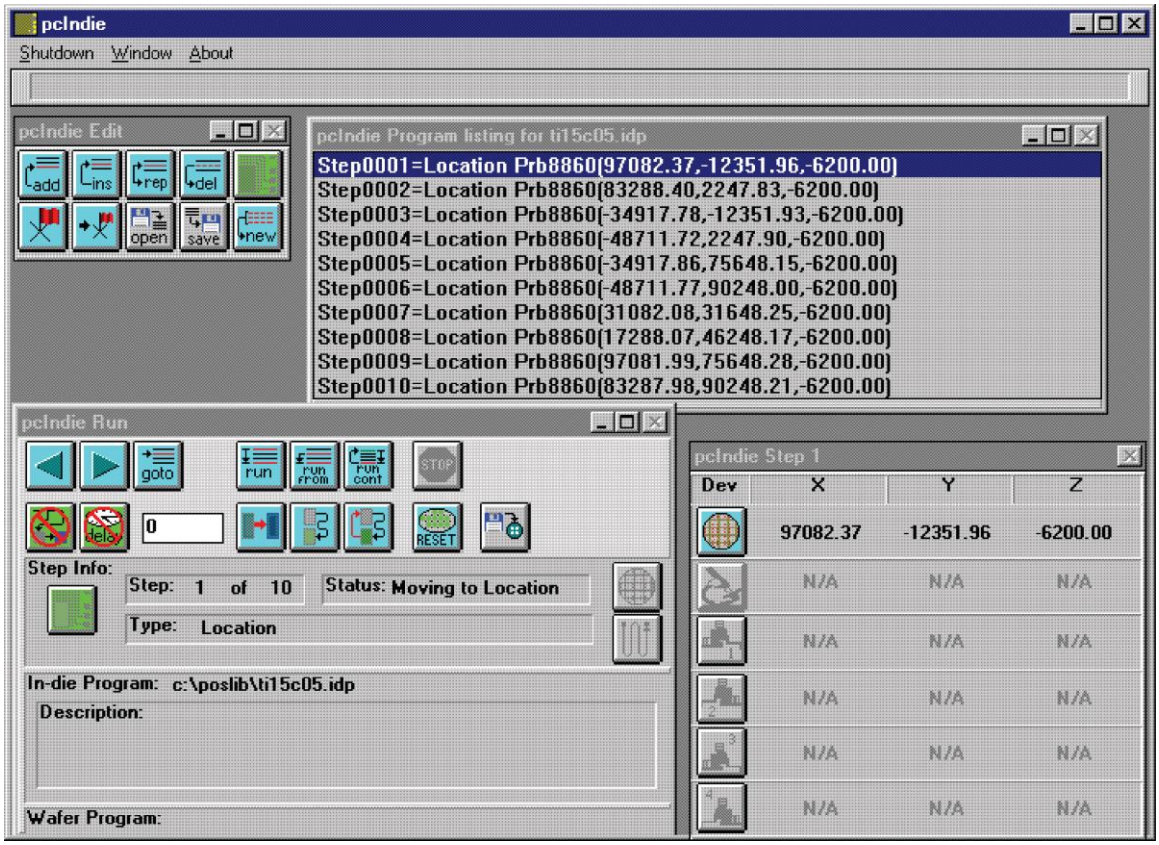


Figure 815: pcIndie window



2. Use the joystick and microscope to move to the first subsite to be tested.
3. Click **add** or **ins** (insert) 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 816: pcIndie save button

6. To open an existing program listing file, click the pcIndie open button in the pcIndie window. Select the file and click **OK**.

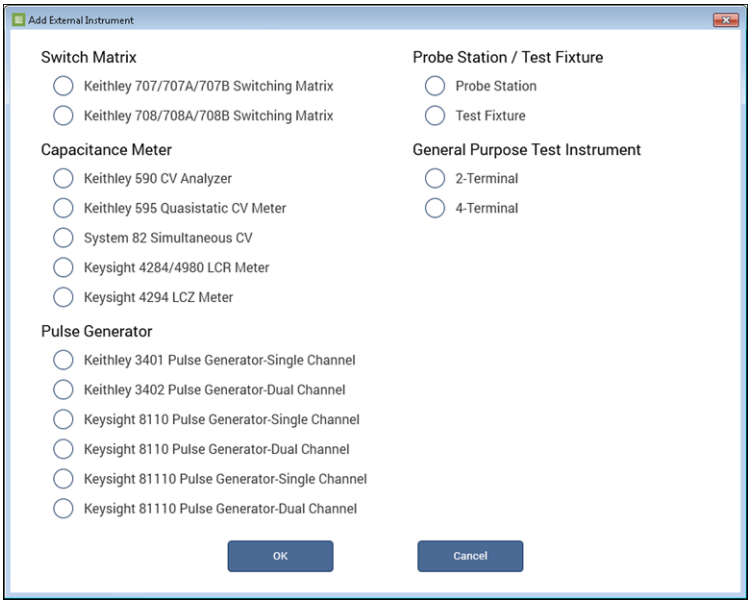
Figure 817: pcIndie open button

Use KCon to add a prober

On the 4200A-SCS, use KCon to add the prober to the configuration:

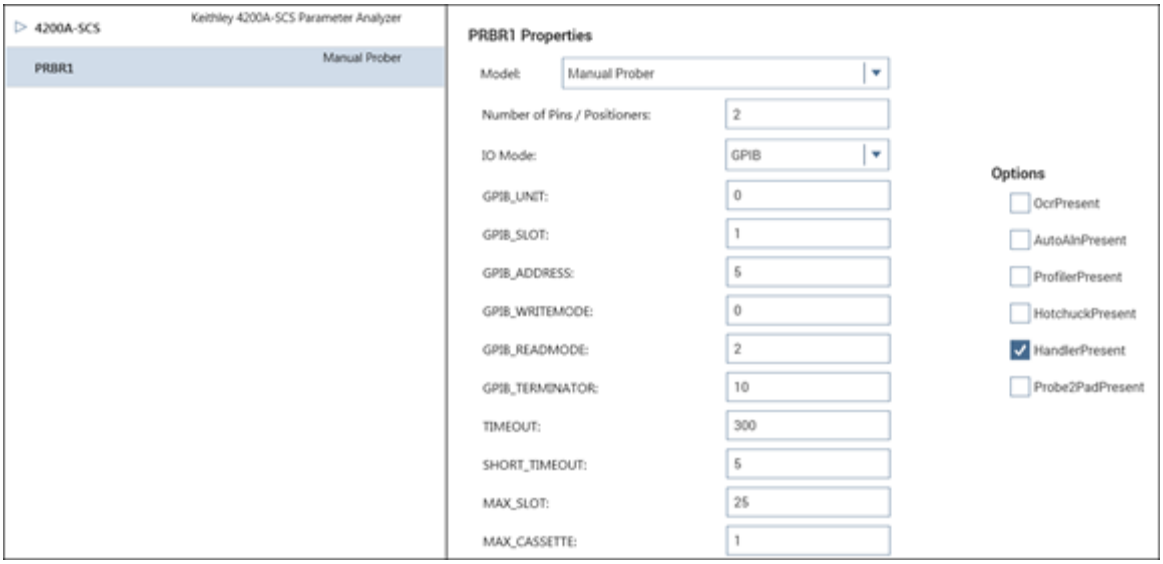
1. Open KCon.
2. At the bottom of the Configuration Navigator, select **Add External Instrument**. The Add External Instrument dialog box is displayed.

Figure 818: Add a prober in KCon



3. Select **Probe Station**.
4. Select **OK**. The KCon navigation displays PRBR1 in the Configuration Navigator and the properties.

Figure 819: Use KCon to select a prober



5. Select the **Micromanipulator 8860 Prober** as the model.
6. Ensure that the **Number of Pins / Positioners** is correct. The number of pins defined here determines the pins that are available to assign to a switch matrix card column.
7. Select **Save**.

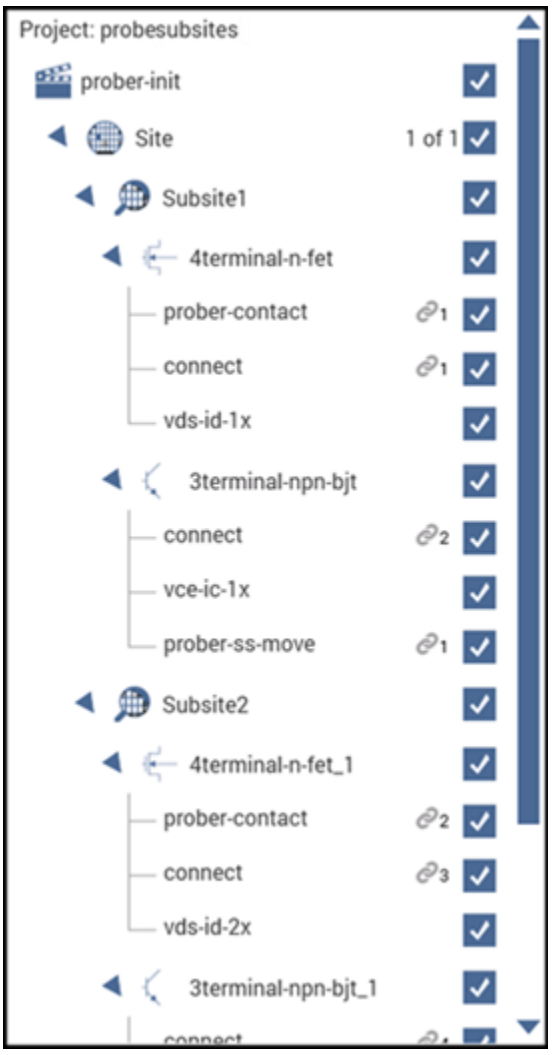
Clarius

Use Clarius to load and run the `probesites` or `probesubsites` project using the new KCon configuration file, which allows you to execute the project for this prober.

On the 4200A-SCS:

1. Open Clarius.
2. Choose **Select**.
3. Select **Projects**.
4. Search for **probesubsites**.
5. Drag the **probesubsites** project to the project tree.

Figure 820: probesubsites project tree



6. Click **Run**.

Commands and error symbols

The following list contains error and status symbols listed by command.

Available commands and responses

	PrChuck	PrInit	PrMovNxt	PrSSMovNxt

PR_OK	X	X	X	X
BAD_CHUCK	X	X		
UNINTEL_RESP	X	X	X	X
UNEXPE_ERROR		X		
SET_MODE_FAIL		X		
INVAL_PARAM		X		
SET_UNITS_FAIL		X		
SET_DIE_FAIL		X		
BAD_MODE			X	X
PR_WAFERCOMPLETE			X	
MOVE_FAIL			X	X
PR_MOVECOMPLETE				X

Information and error code return values and descriptions

Value	Constant	Explanation
1	PR_OK	Success (OK)
2	PR_MOVECOMPLETE	Prober moved to next die (confirmed)
4	PR_WAFERCOMPLETE	Next wafer loaded (confirmed)
-1005	SET_UNITS_FAIL	Failure setting units
-1006	SET_MODE_FAIL	Failure setting mode
-1009	SET_DIE_FAIL	Failure setting die size
-1011	BAD_MODE	Operation invalid in mode
-1013	UNINTEL_RESP	Unintelligible response
-1014	MOVE_FAIL	Movement failure
-1015	UNEXPE_ERROR	Unexpected error
-1017	BAD_CHUCK	Bad chuck position
-1027	INVAL_PARAM	Invalid parameter

Appendix I

Using a manual or fake prober

In this appendix:

Using a manual or fake prober software

The Keithley Instruments 4200A-SCS provides all software required for both manual and fake prober operation; no additional software is needed.

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 using 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 appears that instructs the operator.

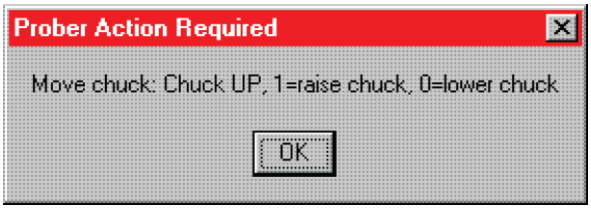
The probing sequence using the MANL prober:

1. Start a project.
2. Issue a `PrInit` command to tell the user to initialize the prober. The `PrInit` dialog box opens.
3. The user continues by clicking **OK**. The project sets up the measurement system to test the first site.

Figure 821: Prober Action Required dialog box: OK initialization

4. Issue a `PrChuck` command to tell the user to ensure that the first test site is ready for testing. The `PrChuck` dialog box opens and the user continues by clicking **OK**. Tests on the site are executed.

Figure 822: Prober Action Required dialog box: Move chuck



5. Issue a `PrMovNxt` command to tell the user to move to the next site to be tested on the wafer. The `PrMovNxt` dialog box opens and the user continues by clicking **OK**.

Figure 823: Prober Action Required dialog: Move probes to next site

NOTE

Subsite probing uses the `PrssMovNxt` command to move to the next subsite.

6. Issue `PRChuck` and `PRMovNxt` commands until all sites are tested.

Fake prober overview

Use the FAKE prober to test without probing. You can use this to take the prober offline when you want to run the test without modifying your project. Configuring the environment for a FAKE prober stops all prober actions.

When using the FAKE prober, you can execute tests individually or in a loop. This allows you to debug projects without removing 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

You can modify these files using the 4200A-SCS.

The default prober configuration file for a manual prober (MANL) is listed below. The only relevant line for this prober type is `PROBER_1_PROBTYPE=MANL`.

```
# 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
```

The default prober configuration file for a fake prober is represented below. The only relevant line for this prober type is `PROBER_1_PROBTYPE=FAKE`.

```
# 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
```

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 a text editor such as Microsoft® Notepad.

Configuration file locations:

- Fake: `C:\s4200\sys\dat\prbcnfg_FAKE.dat`
- Manual: `C:\s4200\sys\dat\prbcnfg_MANL.dat`

Probesites Clarius project example

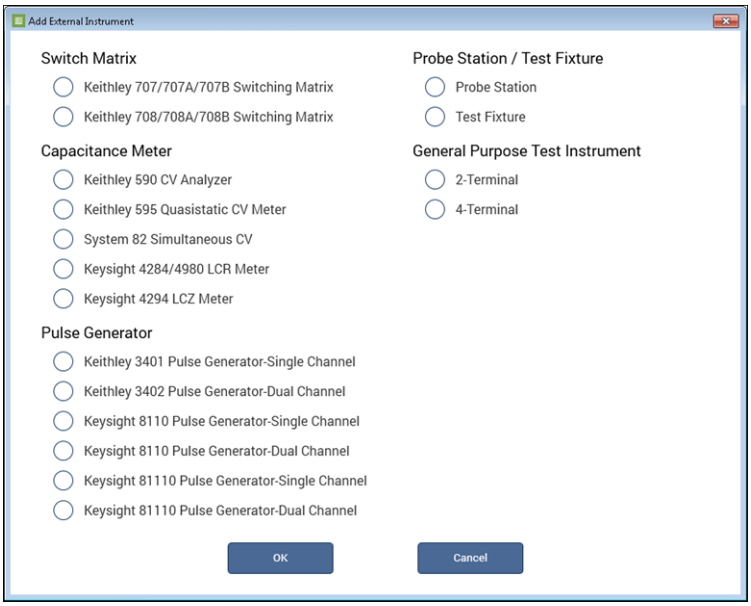
The following is a step-by-step procedure to configure the manual prober so the `probesites` Clarius project executes successfully. The user is responsible for the probe station setup.

Use KCon to add a prober

On the 4200A-SCS, use KCon to add the prober to the configuration:

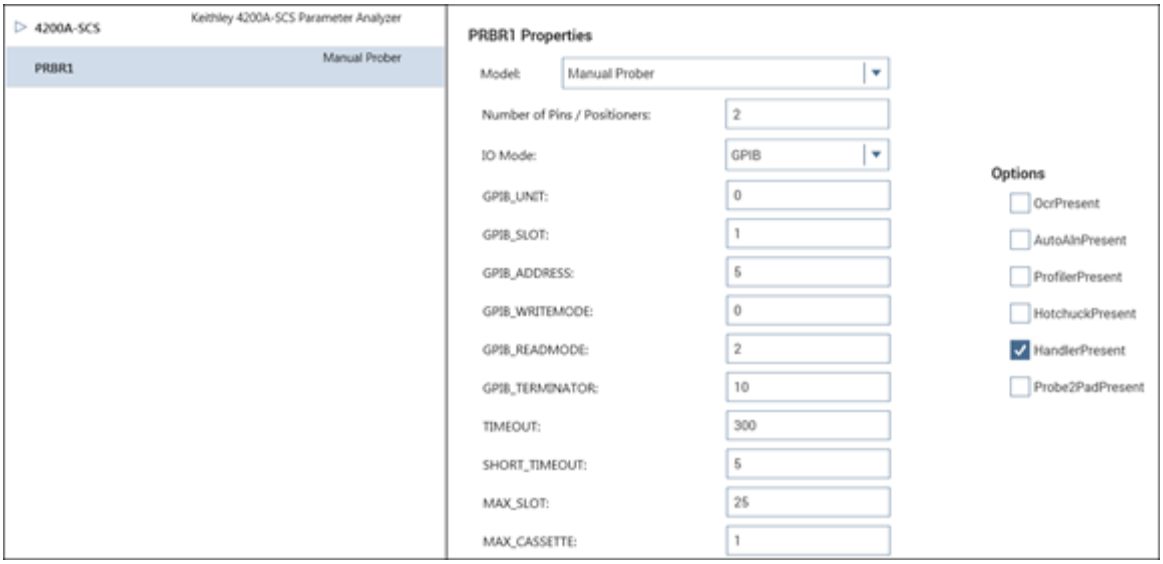
1. Open KCon.
2. At the bottom of the Configuration Navigator, select **Add External Instrument**. The Add External Instrument dialog box is displayed.

Figure 824: Add a prober in KCon



3. Select **Probe Station**.
4. Select **OK**. The KCon navigation displays PRBR1 in the Configuration Navigator and the properties.

Figure 825: Use KCon to select a prober



5. Select the **Manual Prober** or the **Fake Prober** as the model.
6. Ensure that the **Number of Pins / Positioners** is correct. The number of pins defined here determines the pins that are available to assign to a switch matrix card column.
7. Select **Save**.

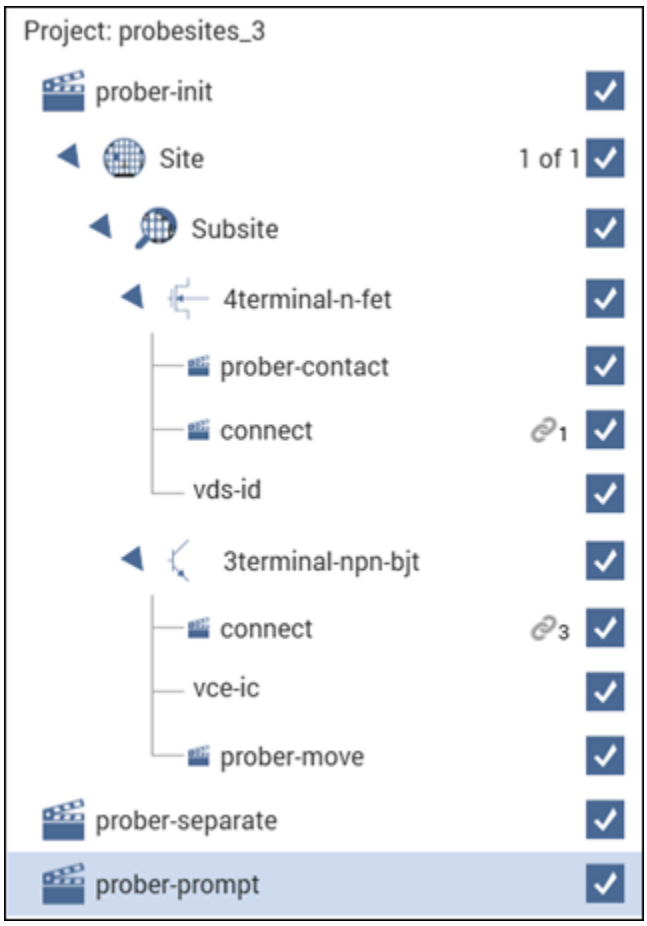
Clarius

Use Clarius to load and run the `probesites` project using the new KCon configuration file, which allows you to execute the project for this prober.

On the 4200A-SCS:

1. Open Clarius.
2. Choose **Select**.
3. Select **Projects**.
4. Search for **probe**.
5. Drag the **probesites** project to the project tree.

Figure 826: probesites project tree



6. Click **Run**.

Probesubsites Clarius project example

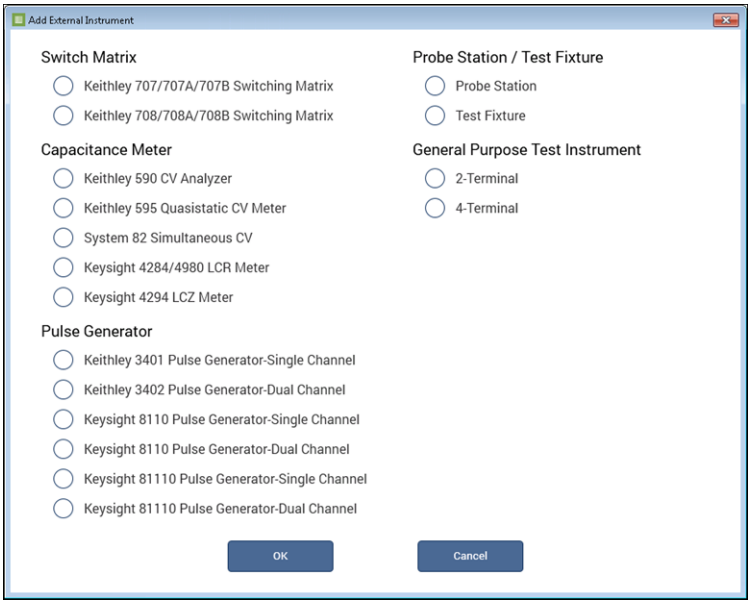
The following is a step-by-step procedure to configure the manual prober so the `probesubsites` Clarius project executes successfully. The user is responsible for the probe station set-up.

Use KCon to add a prober

On the 4200A-SCS, use KCon to add the prober to the configuration:

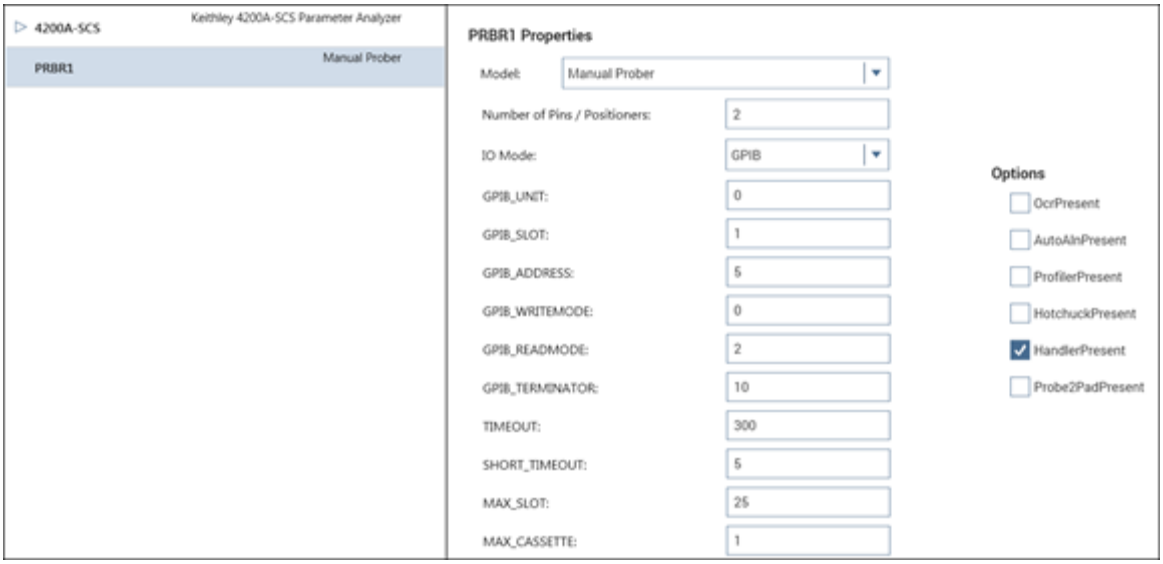
1. Open KCon.
2. At the bottom of the Configuration Navigator, select **Add External Instrument**. The Add External Instrument dialog box is displayed.

Figure 827: Add a prober in KCon



3. Select **Probe Station**.
4. Select **OK**. The KCon navigation displays PRBR1 in the Configuration Navigator and the properties.

Figure 828: Use KCon to select a prober



5. Select the **Manual Prober** or the **Fake Prober** as the model.
6. Ensure that the **Number of Pins / Positioners** is correct. The number of pins defined here determines the pins that are available to assign to a switch matrix card column.
7. Select **Save**.

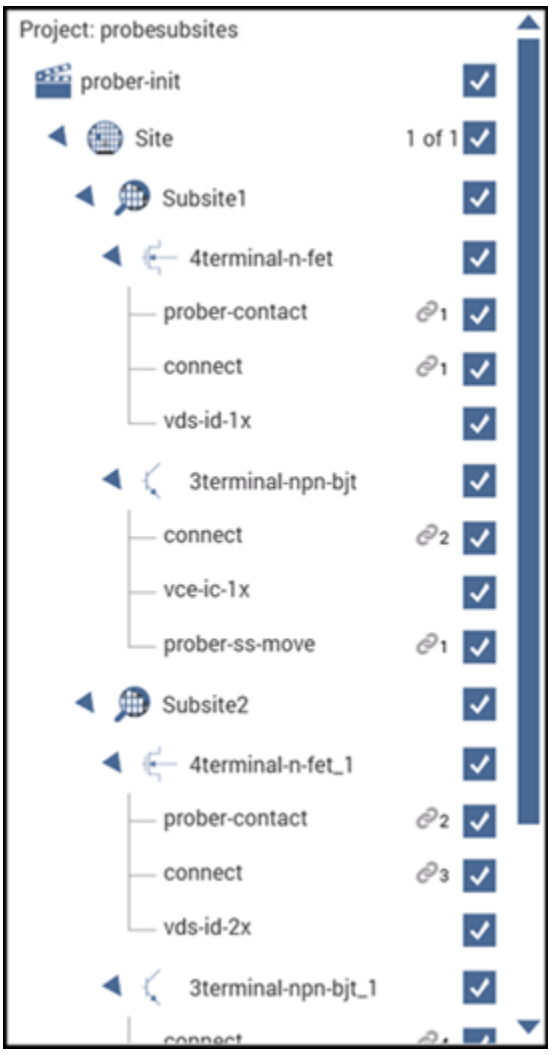
Clarius

Use Clarius to load and run the `probesites` or `probesubsites` project using the new KCon configuration file, which allows you to execute the project for this prober.

On the 4200A-SCS:

1. Open Clarius.
2. Choose **Select**.
3. Select **Projects**.
4. Search for **probesubsites**.
5. Drag the **probesubsites** project to the project tree.

Figure 829: probesubsites project tree



6. Click **Run**.

Appendix J

Using a Cascade Summit-12000 Prober

In this appendix:

Cascade Summit 12000 prober software

The Summit 12000 prober will have one of the following software products installed. Use either of the following software products to configure and operate the Summit 12000 prober with the Keithley Instruments 4200A-SCS:

- **Velox prober control software**: Provides acces to configuration and help programs

  or

- **Nucleus UI prober control software**: Provides access to configuration and help programs.

Software version

The following software versions were used to verify the configuration and remote command set of the Summit-12000 prober with the 4200A-SCS:

- Velox ver. 2.3

  or

- Nucleus UI ver. 2.0

Probe station configuration

CAUTION

Make sure that you are familiar with the Summit-12000 Prober and its supporting documentation before attempting setup, configuration, or operation.

The general steps required to set up and configure the Summit-120000 prober for use with the 4200A-SCS:

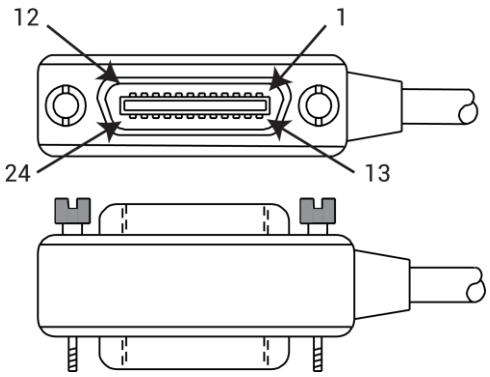
- Set up communications (on page J-2)
- Set up wafer geometry (on page J-9)
- Create a site definition and define a probe list (on page J-14)
- Load, align, and contact the wafer (on page J-17)

Set up communications

The Summit-12000 prober is configured for GPIB communications only. Make sure that the prober configuration file is set up properly for the GPIB communications interface. See Modifying the prober configuration file for information.

To connect the equipment, connect the 4200A-SCS GPIB port and the probe station PC GPIB port using a GPIB cable (Model 7007). See the following figure and table for pinouts.

Figure 830: IEEE-488 connector



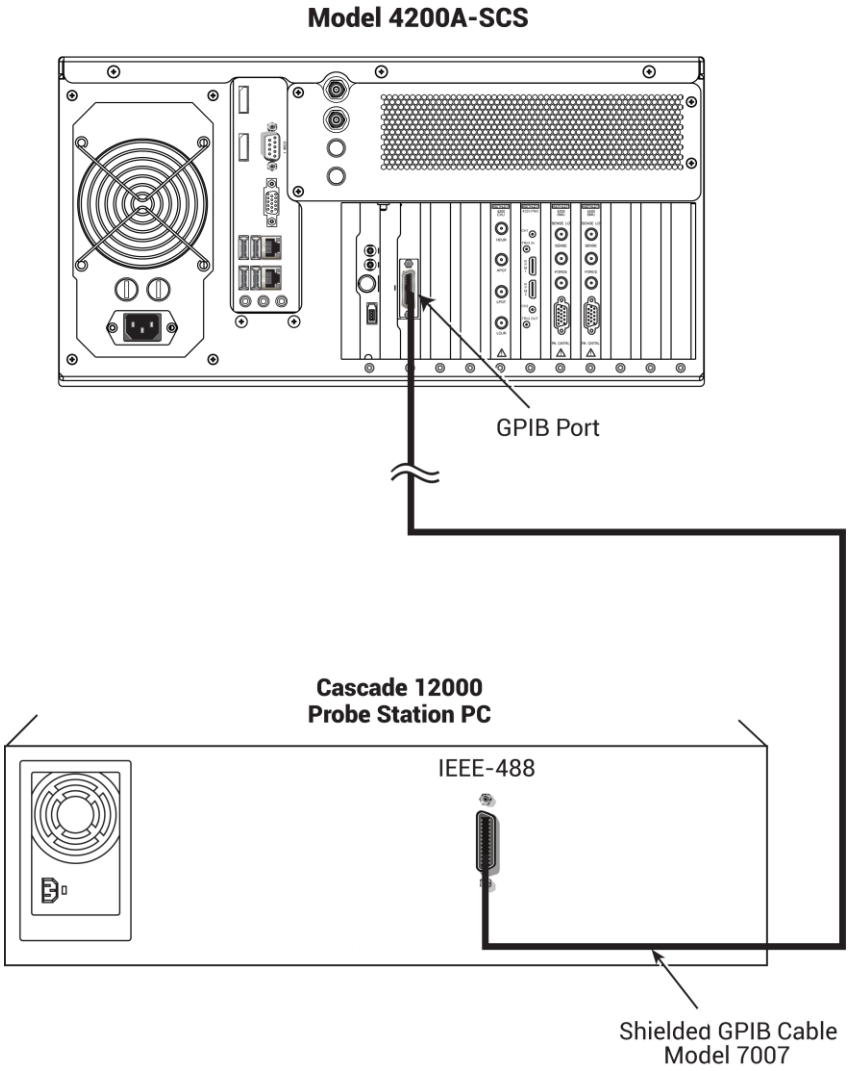
GPIB control connector terminals

Contact number	GPIB 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.

The following figure shows connections between the Cascade Summit-12000 prober to the Keithley Instruments 4200A-SCS.

Figure 831: Connection diagram



For probers using the Velox prober control software:

On the probe station computer, refer to the Velox user manual and help content for information on how to setup and configure the GPIB interface. The following settings should be made in Velox to the GPIB configuration for operation with the 4200A:

- Device Address: 28 - Recommend using 28, as this is the default address used by the 4200A
- Response Terminator: CR - Carriage Return
- Service Request: OFF
- TSK emulation: OFF
- SCPI string responses: ON
- Legacy SCPI status byte: ON

For probers using the Nucleus UI prober control software:

1. On the probe station computer, double-click the **Nucleus** icon.

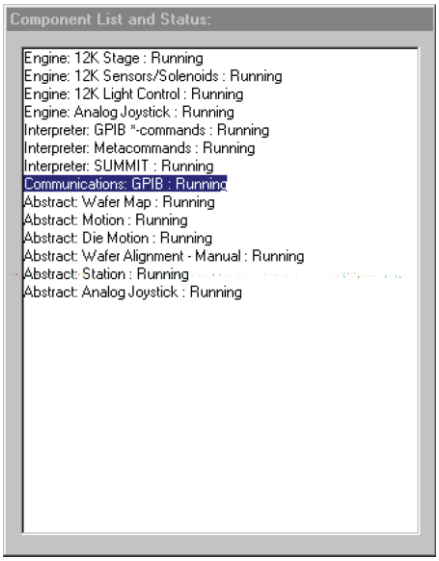
Figure 832: Nucleus icon

2. Log in using the **Nucleus System Login**.

Figure 833: Login window

3. After login is complete, the prober initializes the stage. Click **Proceed** when the prober has completed initialization.
4. Maximize the system manager **Component List and Status** program (right-click the system manager label on the taskbar and choose **Maximize**).
5. Select **Communications: GPIB** on the component list.

Figure 834: Component list and status window



NOTE

If the **Communications: GPIB** component is not on the list, you must add it. To add: Click **Add** from the Add component dialog box, then select **Communications: GPIB.**

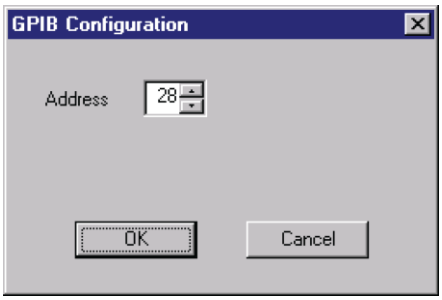
6. If the **Communications: GPIB** component is running, click the **Stop** button, or proceed to the next step (setup).

Figure 835: Stop button

7. Click the **Setup** button to open the GPIB configuration window.

Figure 836: 4200-901_Setup button.png

Figure 837: GPIB Configuration window



8. Change the address as needed. The default value is 28.
9. Save the configuration file by clicking **Save**.

Figure 838: Save button

10. Start the component by clicking **GO**.

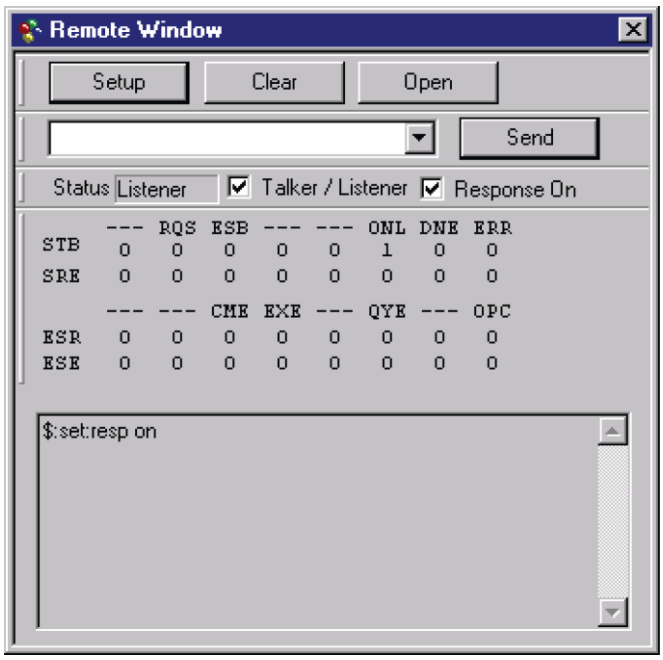
Figure 839: GO button

11. Minimize, but do not close, the system manager window.
12. Click **Remote** on the Nucleus UI toolbar to display the Remote Window. See the following three figures.

Figure 840: Remote button

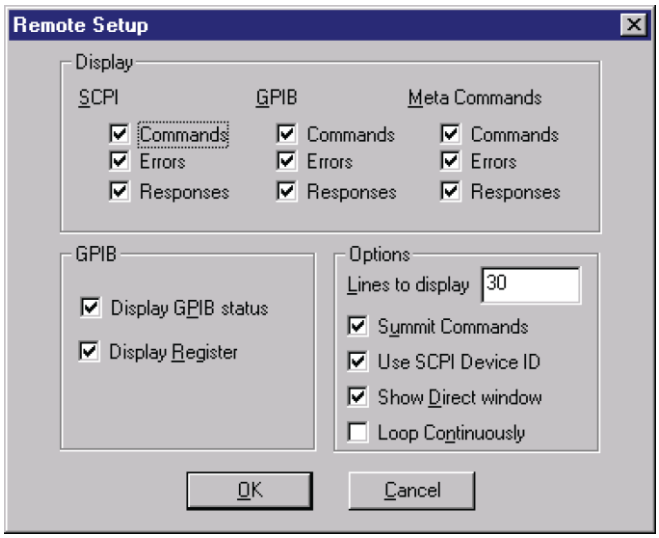
Figure 841: Nucleus UI toolbar

Figure 842: Remote window



13. Select the **Talker / Listener** and **Response On** boxes in the Remote Window.
14. Click **Setup** on the Remote Window to display the Remote Setup dialog box.

Figure 843: Remote setup window



15. Select the items to be displayed.
16. Click **OK**.

NOTE

Selecting boxes on the setup window only affects the DISPLAY properties. It will not change the GPIB physical setting. Use the dialog box in the GPIB configuration window to make changes to the GPIB address.

Set up wafer geometry

For probers using the Velox prober control software:

On the probe station computer, refer to the Velox user manual and help content for information on how to create wafer maps. Note a wafer map wizard is available in the Velox prober control software to assist with this. The die size, reference position, and testing sequence should be specified.

For probers using the Nuceus UI prober control software:

1. On the probe-station computer, if the Nucleus toolbar is not already open, double-click the **Nucleus** icon on the Windows desktop.

Figure 844: Nucleus icon

2. Log in.
3. From the Window menu of the Nucleus toolbar, select **WaferMap** to display the wafer map window. See the following two figures.

Figure 845: Nucleus toolbar

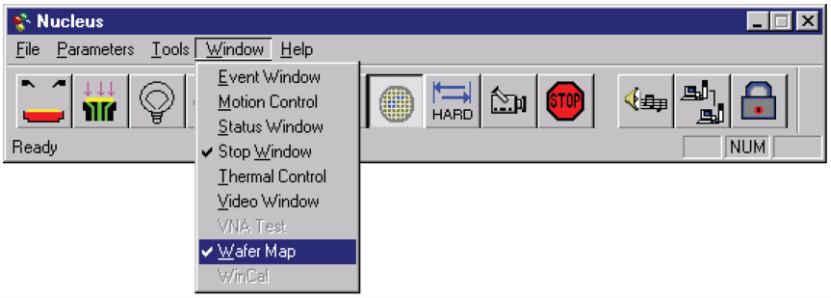
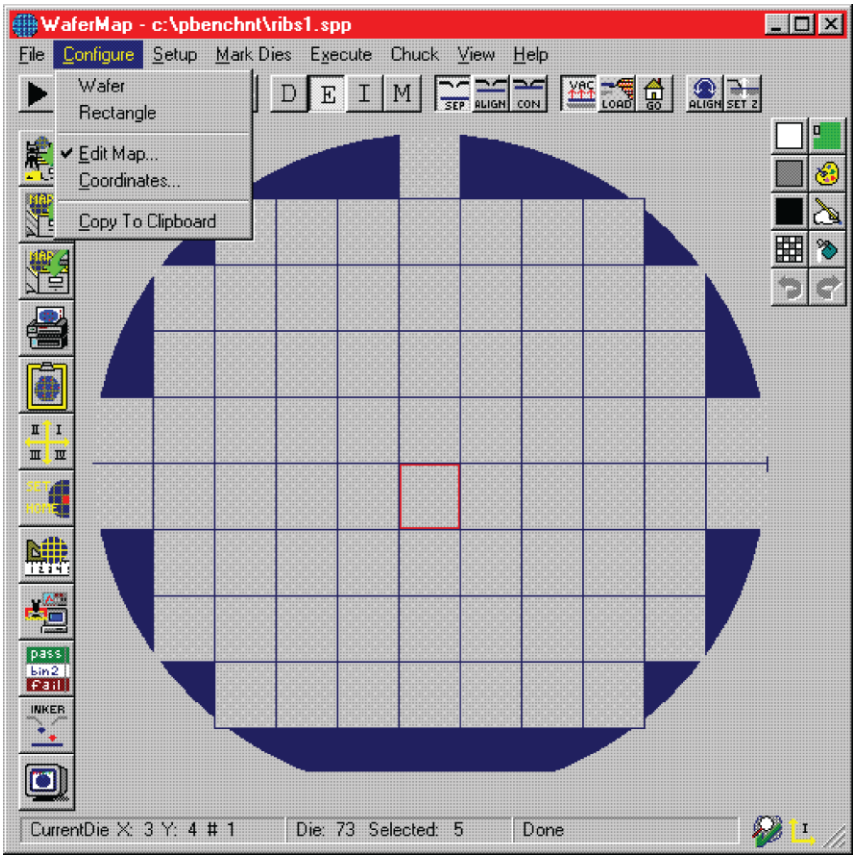
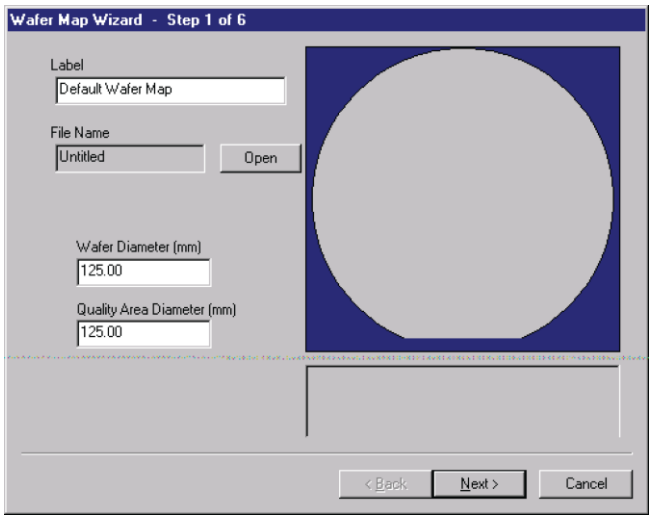


Figure 846: Wafer Map window



4. From the **File** menu of the Wafer Map window, select **Wizard** to start the Wafer Map wizard.

Figure 847: Step 1: Wafer Map Wizard

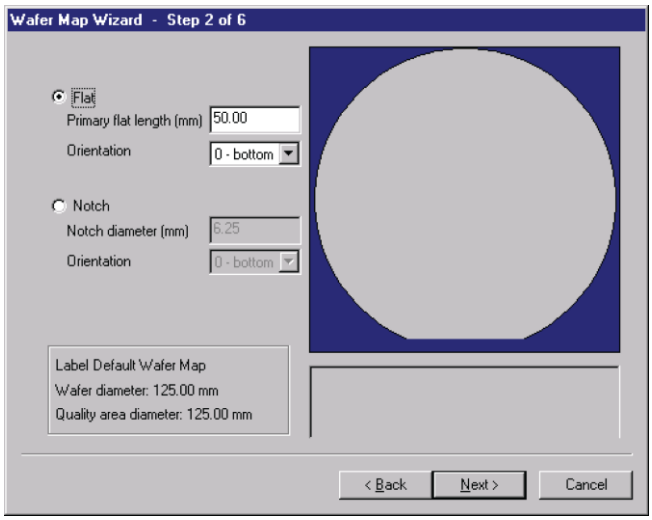


5. Enter the label and wafer diameter in the Wafer Map Wizard window.
6. Click **Next**.
7. Select **Flat** or **Notch** based on the actual wafer.
8. Enter either the primary flat length or the notch diameter in millimeters.
9. Select the orientation of the flat or notch as applicable.

NOTE

Bottom is toward the front of the prober.

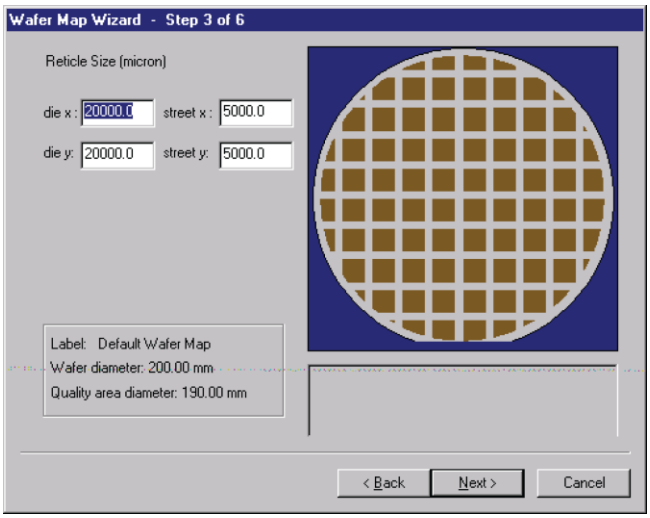
Figure 848: Step 2: Wafer Map Wizard



10. Click **Next**.

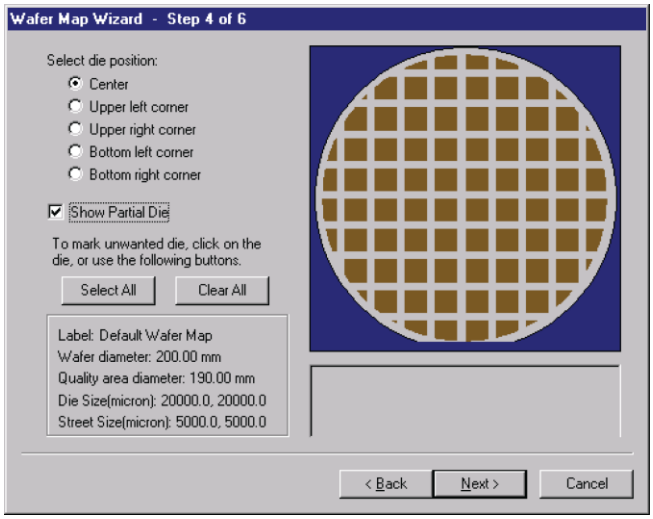
11. Enter the correct die and street sizes.

Figure 849: Step 3: Wafer Map Wizard



12. Click **Next**.
13. Select the die position. Optionally, select **Show Partial Die**.

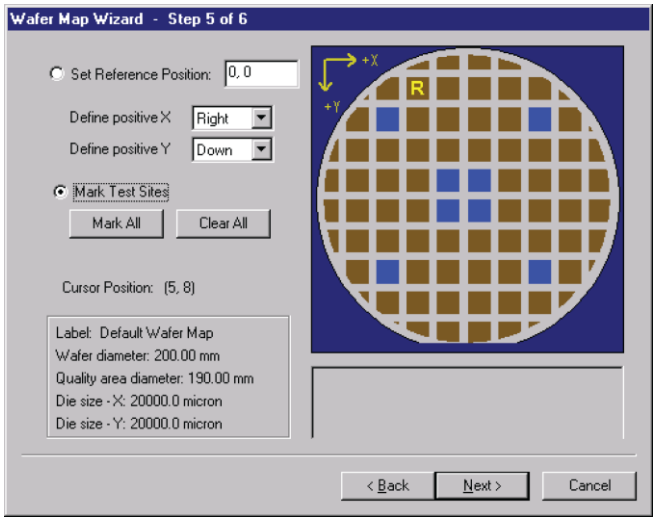
Figure 850: Step 4: Wafer Map Wizard



14. Click **Next**.

15. Set the reference position.
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. Select Mark Test Sites. You can drag to select multiple sites.

Figure 851: Step 5: Wafer Map Wizard



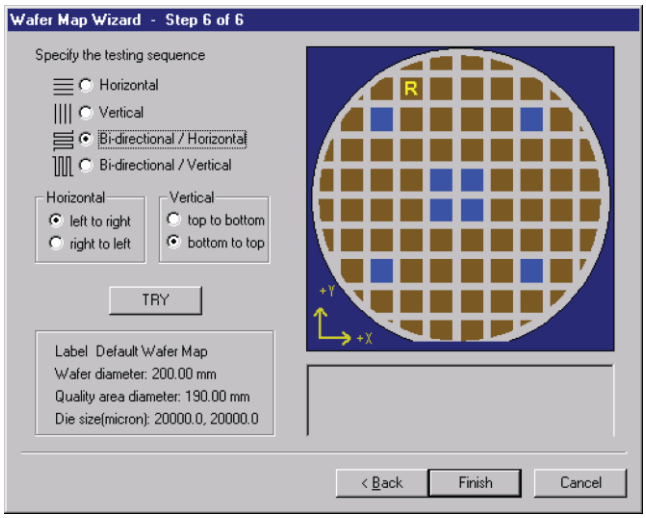
NOTE

Refer to the `probesites` and `probesubsites` Clarius project examples for specifics on selecting sites to probe.

18. Click **Next**.

19. Specify the test sequence.

Figure 852: Step 6: Wafer Map Wizard



20. Click **Finish**.
21. Save the Wafer Map settings.

Create a site definition and define a probe list

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 site (die) is to be probed, refer to Probesites Clarius Project example (on page J-23) in this appendix.

Creating a site definition for multiple subsites per die involves using the software to create a selection of dies to probe, but also includes creating a selection of the subsites for each site (die) that will be probed. If multiple subsites per site will be probed, refer to Probesubsites Clarius Project example (on page J-27) in this appendix.

For probers using the Velox prober control software:

On the probe station computer, refer to the Velox user manual and help content for information on how to add and edit sites (dies) and subsites (subdies).

Additionally, a test sequence will need to be specified. The <ModelName> operates by sending commands that will cause the prober to move to the next site or subsite as determined by the test sequence.

NOTE

The 4200A is not aware of the specific sites or subsites present in the wafer map.

For probers using the Nucleus UI prober-control software:

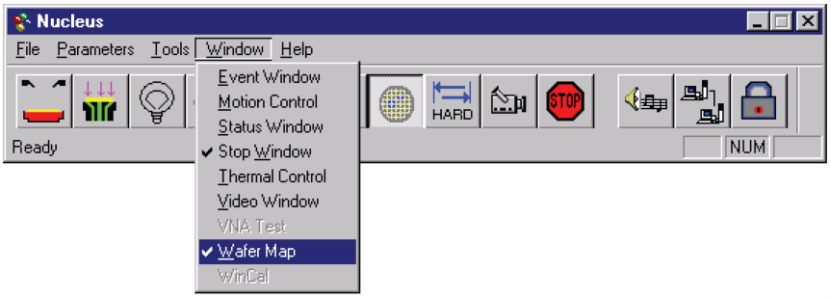
Perform the following on the probe-station computer to open a previously defined site definition and probe list.

1. If the Nucleus toolbar is not already open, double-click the Nucleus UI icon on the Windows desktop.

Figure 853: Nucleus icon

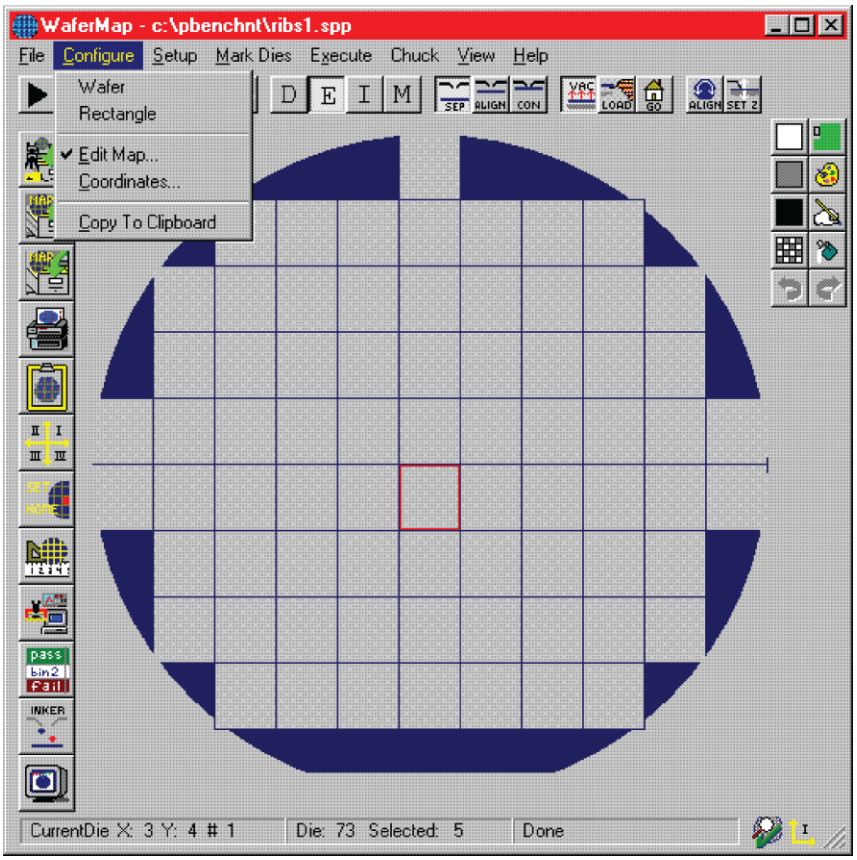
2. Log in.
3. From the **Nucleus** toolbar, select Tools > WaferMap. The Wafer Map window will be displayed. See the following two figures.

Figure 854: Nucleus toolbar



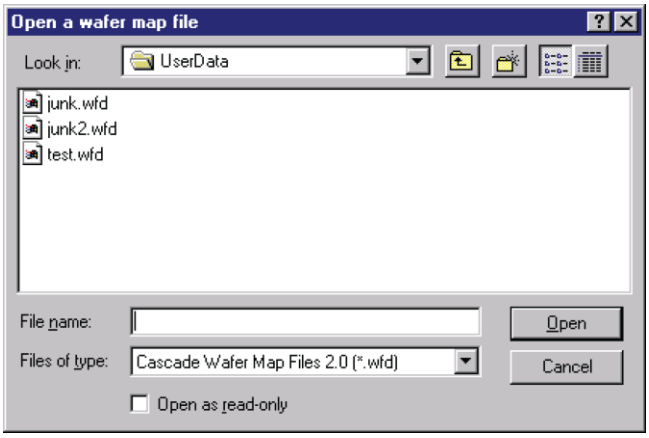
4. From the wafer Map Window, select **File > Open**.

Figure 855: Wafer Map window



5. Open the wafer map file.

Figure 856: Open a wafer map file window



Load, align, and contact the wafer

For probers using the Velox prober-control software:

On the probe station computer, refer to the Velox user manual and help content for information on how to load, unload, set chuck heights, align, contact, and set the home (also called reference) position of the wafer.

For probers using the Nucleus UI prober-control software:

1. On the probe-station computer, from the Nucleus toolbar, select **Window > Motion Control**. The Motion Control window opens. See the following two figures.

Figure 857: Nucleus toolbar

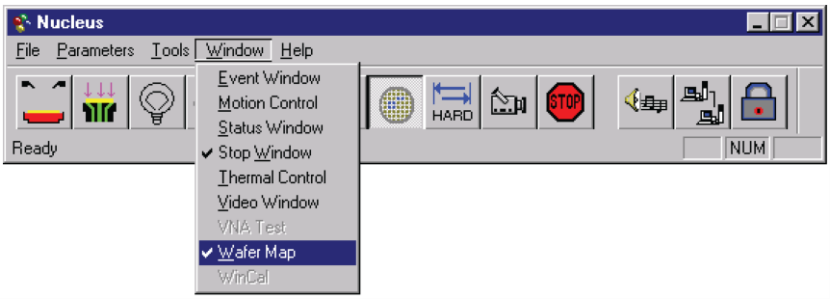
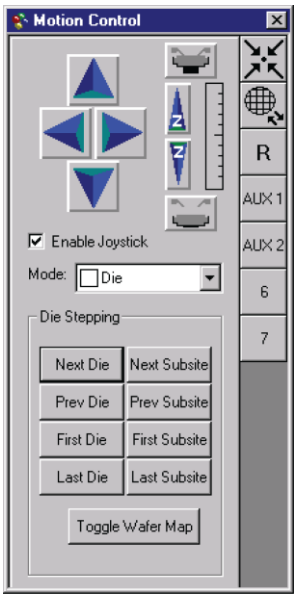
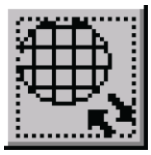


Figure 858: Motion Control window



2. From the Motion Control window, click the **Chuck to front** button.

Figure 859: Chuck to front button



3. From the Nucleus toolbar, click the **Enable Joystick** button.

Figure 860: Enable Joystick button

4. Place a wafer on the chuck.
5. From the Nucleus UI toolbar, toggle the vacuum from **OFF** to **ON**.

Figure 861: Vacuum control

6. From the Nucleus UI toolbar, turn on the camera screen by clicking the **Video** button.

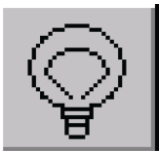
Figure 862: Video button

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 863: Light button



8. From the Wafer Map window, click the **Reference Die** button. The Align dialog box opens. See the following two figures.

Figure 864: Reference Die button

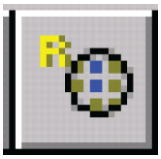
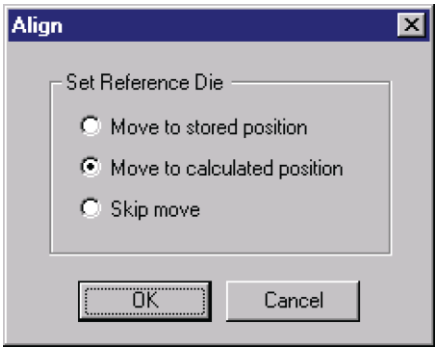


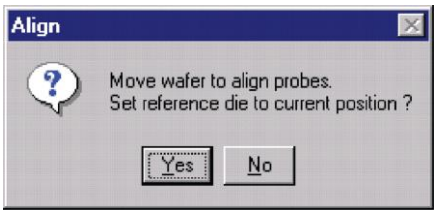
Figure 865: Move the Reference Die



9. From the Align dialog box, select **Move to calculated position**.

10. Click **OK**.
11. Manually move the wafer to the reference die.
12. Click **Yes** to set the reference die to the present position. When choosing the reference die:
    - The wafer should be on the chuck and physically in the correct reference position.
    - Click the die on the wafer map UI that will be the reference die.
    - An R appears when a die has been selected as the home die.

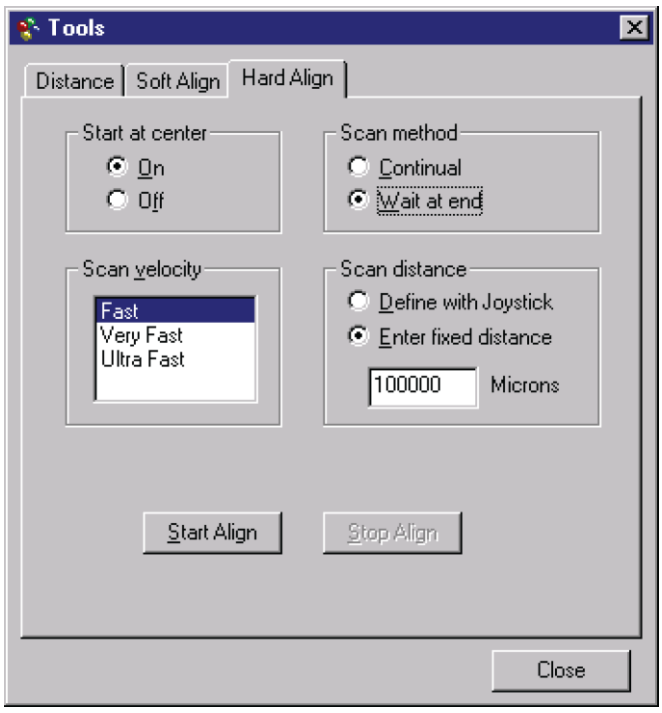
Figure 866: Align dialog



13. From the Nucleus UI toolbar, click the **Hard Align** button to display the Hard Align dialog box.

Figure 867: Hard Align button

Figure 868: Hard Align tab



14. For **Start at center**, select **On**.
15. For **Scan method**, select **Wait at end**.
16. Set the **Scan velocity**.
17. Set the **Scan distance**. You can enter a fixed distance or define it with the joystick.

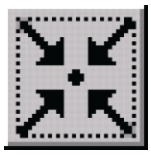
Align the wafer:

1. Move to wafer center by clicking the **Center** button on the Motion Control window.
2. Click **Start Align** on the Hard Align dialog box.

NOTE

Raise the platen arm if prompted (a prompt will only appear if the platen arm is down when you start the alignment).

Figure 869: Center button



3. Watch on the monitor while the stage moves down the street to position the needles near the left edge of the wafer.
4. Adjust the theta knob on the stage while moving across the wafer.
5. Click **Yes** at the prompt that appears on the screen.
6. 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.
7. Make a small adjustment in theta when motion stops.
8. Click **No** when the alignment is correct.
9. 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 on the left 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.

10. Either using the **Z Up/Z Down** buttons on the Motion Control window, or the joystick if set for Scan Z Axis (see **CAUTION**), make contact with the wafer.
11. When probe tips are making good contact with the wafer, right-click the **Contact** button.
12. Click the **Set to Current Position** button.

Figure 870: Set to Current Position button

CAUTION

When the Joystick mode is set to "Scan Z Axis," the joystick will control Z movement. 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).

Probesites Clarius Project example

The following is a step-by-step procedure to configure the Summit 120000 so the `probesites` Clarius project executes successfully.

Nucleus UI or Velox software

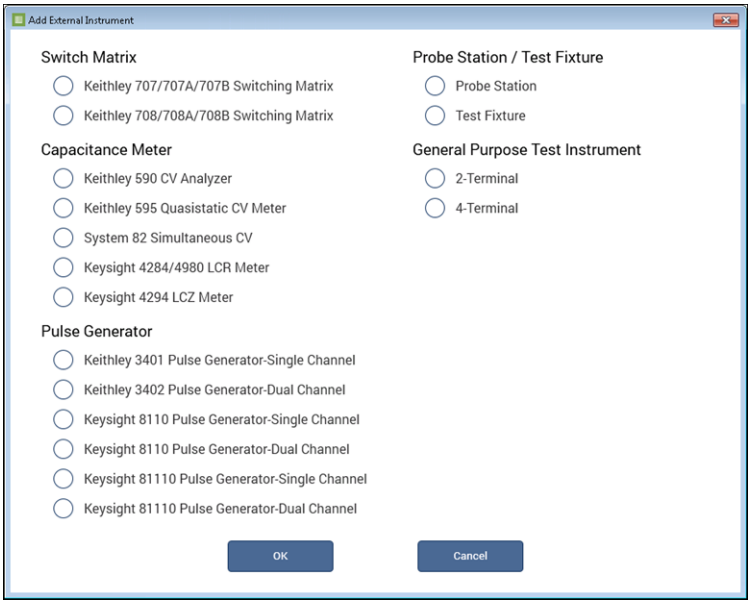
Using either the Velox software or the Nucleus UI software on the probe station computer, edit and open a wafer map file as described in Set up wafer geometry (on page J-9) and Create a site definition and define a probe list (on page J-14).

Use KCon to add a prober

On the 4200A-SCS, use KCon to add the prober to the configuration:

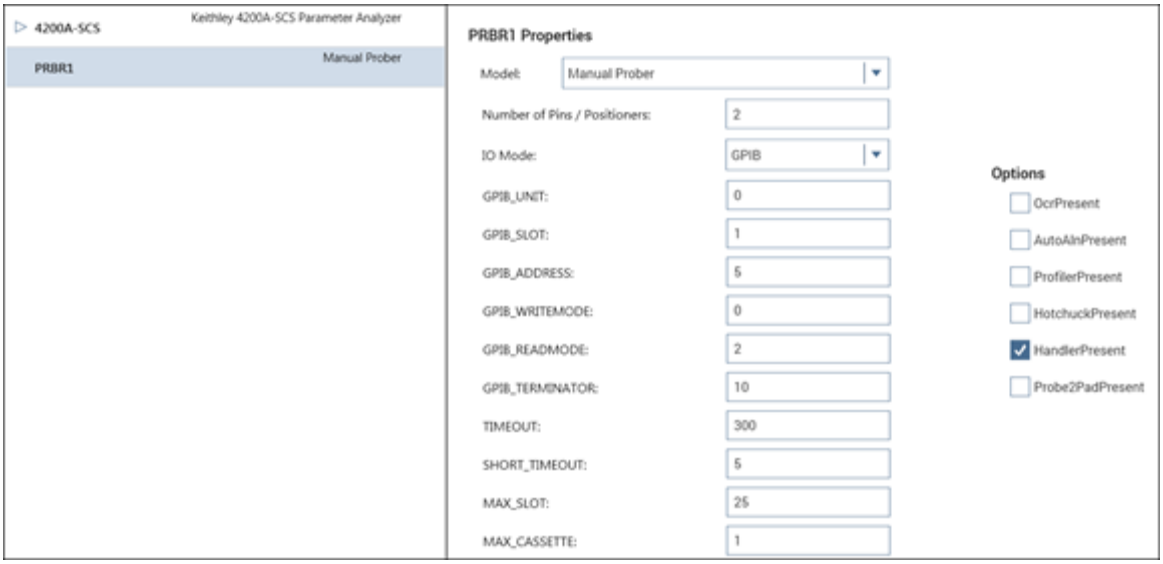
1. Open KCon.
2. At the bottom of the Configuration Navigator, select **Add External Instrument**. The Add External Instrument dialog box is displayed.

Figure 871: Add a prober in KCon



3. Select **Probe Station**.
4. Select **OK**. The KCon navigation displays PRBR1 in the Configuration Navigator and the properties.

Figure 872: Use KCon to select a prober



5. Select the Cascade 1200 prober as the model.
6. Make sure the **Number of Pins / Positioners** is correct. The number of pins defined here determines the pins that are available to assign to a switch matrix card column.
7. Verify the IO Mode is set to **GPIB.**
8. Verify the **GPIB_ADDRESS** is set to the address of the prober. This address was set in the section Set up communications (on page K-3). The default address is 28.
9. Select **Save**.
10. Exit KCon.

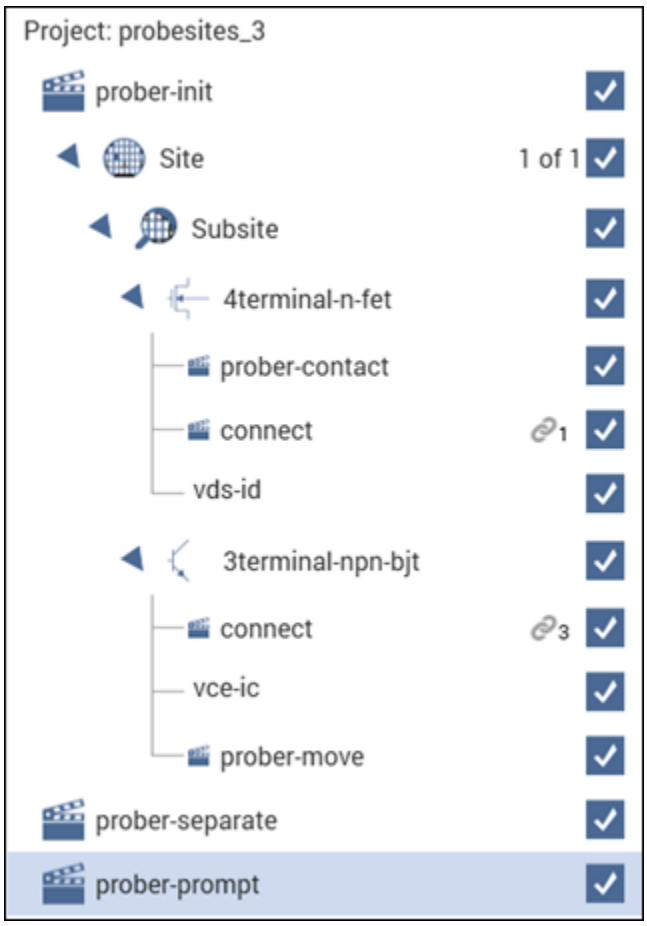
Clarius

Use Clarius to load and run the `probesites` project using the new KCon configuration file, which allows you to execute the project for this prober.

On the 4200A-SCS:

1. Open Clarius.
2. Choose **Select**.
3. Select **Projects**.
4. Search for **probe**.
5. Drag the **probesites** project to the project tree.

Figure 873: probesites project tree



6. Click **Run**.

Probesubsites Clarius Project example

The following is a step-by-step procedure to configure the Summit-12000 so the `probesubsites` project executes successfully.

For probers using the Velox prober control software:

On the probe station computer, refer to the Velox user manual and help content for information on setting up and loading a wafer map that contains sites (dies) and subsites (subdies). A test sequence also needs to be specified.

For propers using the Nucleus UI prober control software:

1. On the probe-station computer, if the Nucleus toolbar is not already open, double-click the **Nucleus** icon on the Windows desktop.

Figure 874: Nucleus icon

2. Log in.
3. From the Window menu of the Nucleus toolbar, select **WaferMap** to display the wafer map window. See the following two figures.

Figure 875: Nucleus toolbar

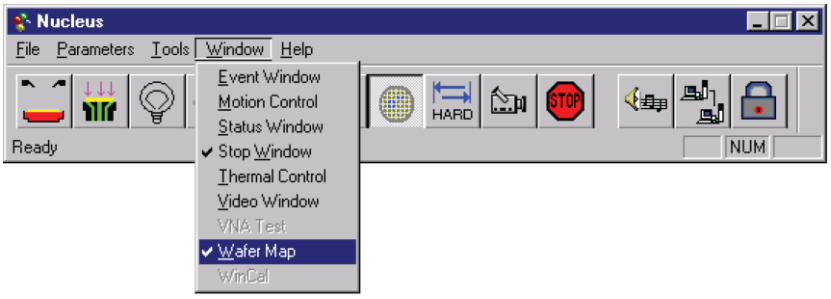
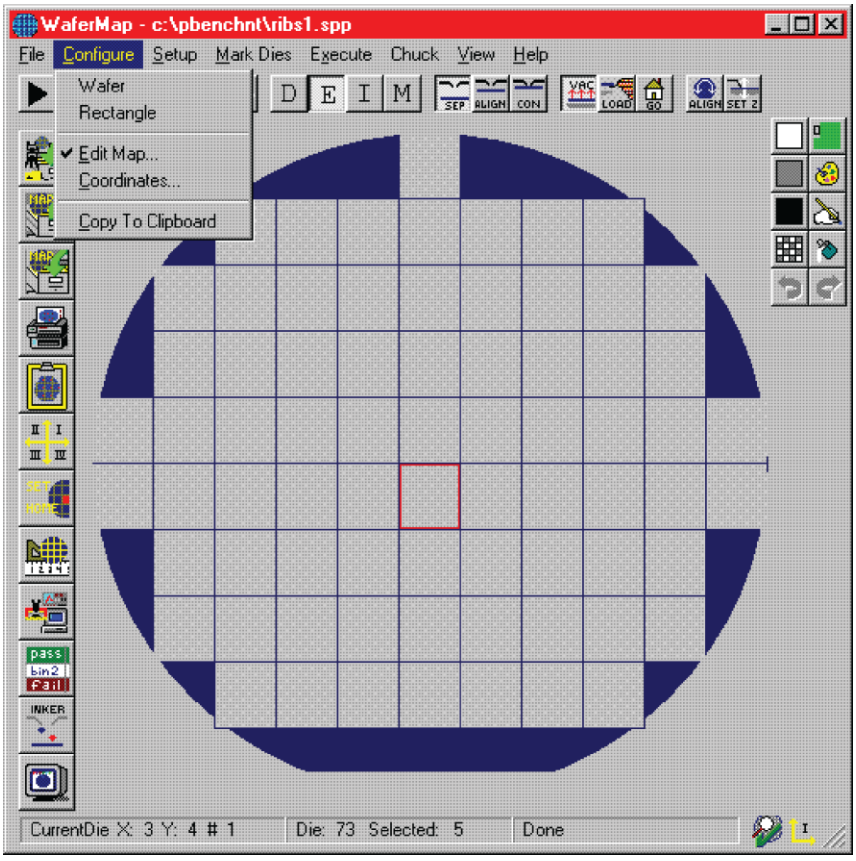
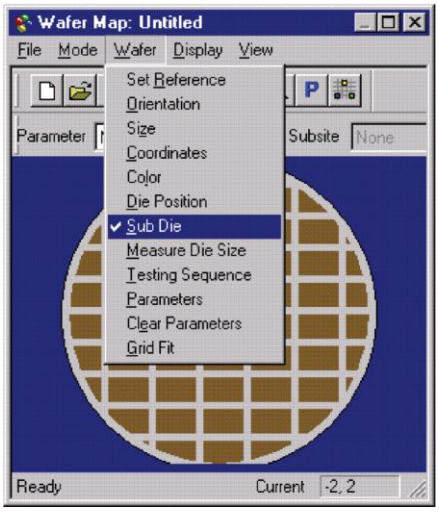


Figure 876: Wafer Map window



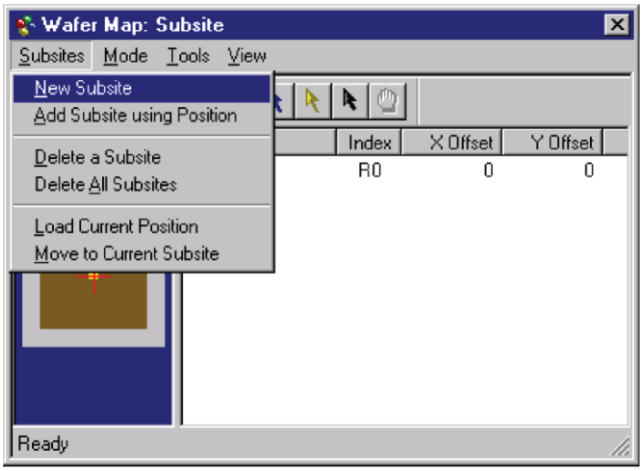
4. From the Wafer Map window, select **File > Open** to open a wafer map file.
5. Click **Wafer > Sub Die** from the Wafer Map menu. A subsite dialog box opens. See the following figure.

Figure 877: Open Sub Die dialog



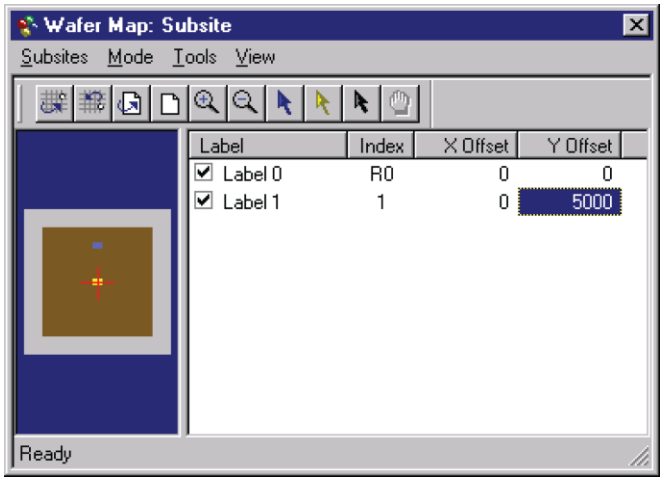
6. Click **Subsites >New Subsite** to create a new subsite Label 1.

Figure 878: Select New Subsite on the Subsites menu



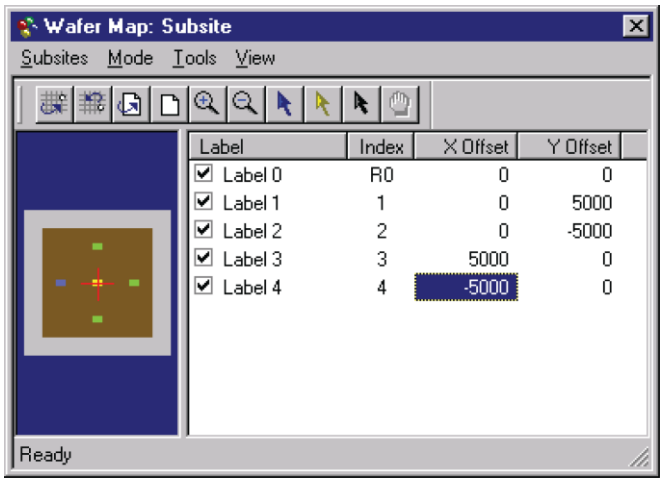
7. Enter the corresponding X and Y offset of the new subsite.

Figure 879: Enter x and y offset



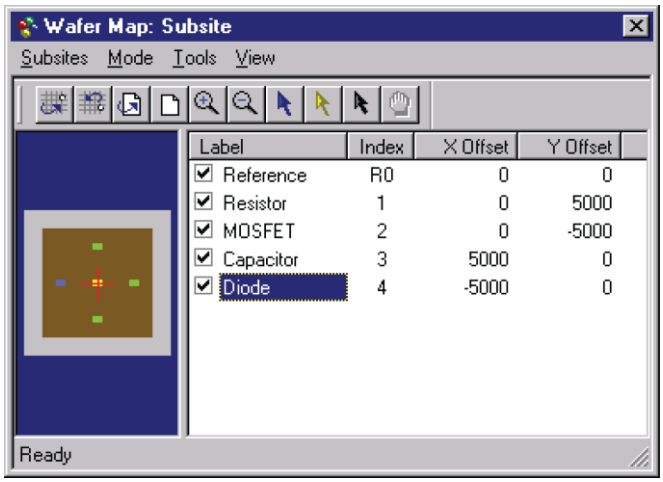
8. Continue to add new subsites until finished.

Figure 880: Make four new subsites



9. Click on the label name and type in a new description to relabel each subsite.

Figure 881: Relabel the subsites



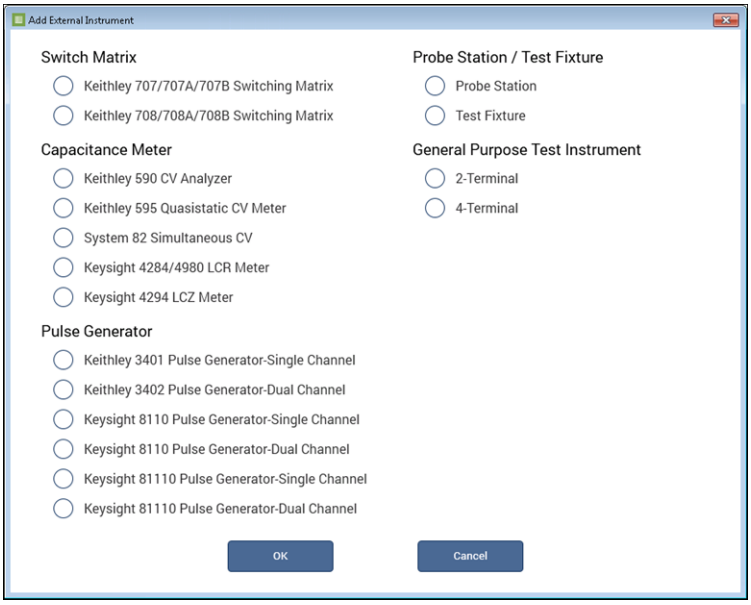
10. To choose a subsite for testing, select the box at the front of each label. To skip testing the subsite, clear the box at the front of each label.
11. Click **File > Save** on the Wafer Map dialog box to save the wafer map.

Use KCon to add a prober

On the 4200A-SCS, use KCon to add the prober to the configuration:

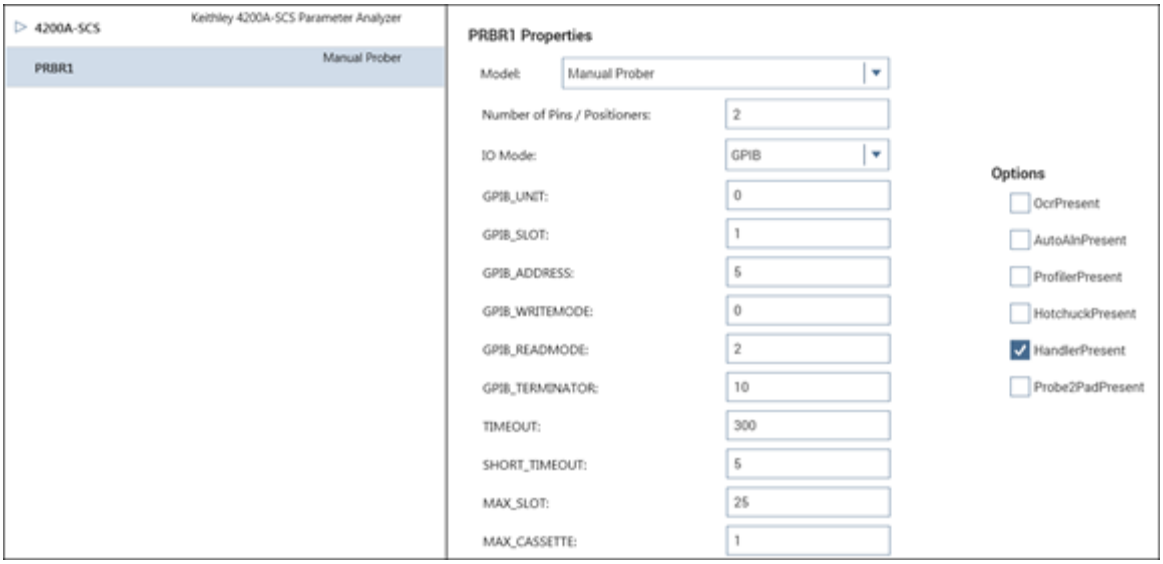
1. Open KCon.
2. At the bottom of the Configuration Navigator, select **Add External Instrument**. The Add External Instrument dialog box is displayed.

Figure 882: Add a prober in KCon



3. Select **Probe Station**.
4. Select **OK**. The KCon navigation displays PRBR1 in the Configuration Navigator and the properties.

Figure 883: Use KCon to select a prober



5. Select the Cascade 1200 prober as the model.
6. Make sure the **Number of Pins / Positioners** is correct. The number of pins defined here determines the pins that are available to assign to a switch matrix card column.
7. Verify the IO Mode is set to **GPIB.**
8. Verify the **GPIB_ADDRESS** is set to the address of the prober. This address was set in the section Set up communications (on page K-3). The default address is 28.
9. Select **Save**.
10. Exit KCon.

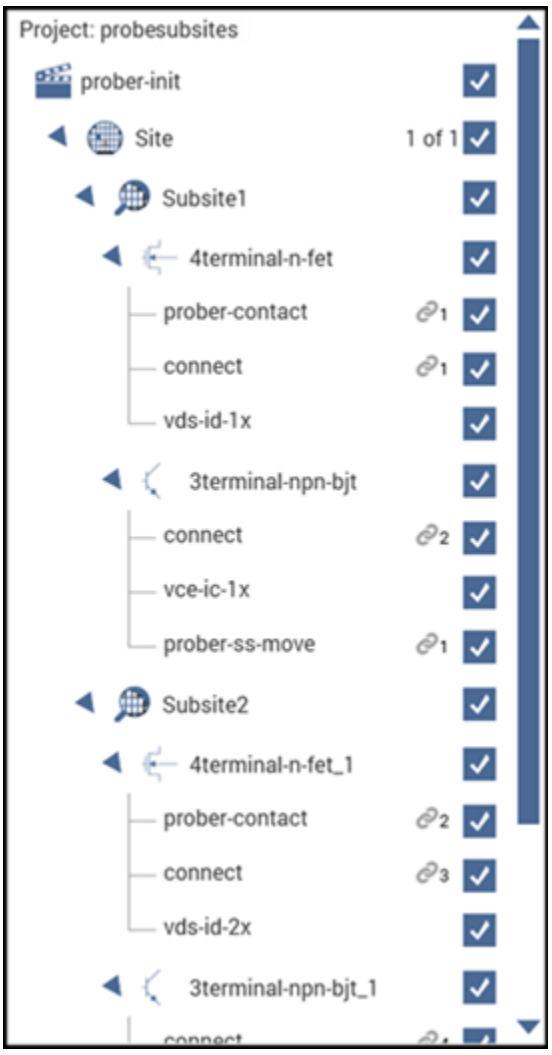
Clarius

Use Clarius to load and run the `probesites` or `probesubsites` project using the new KCon configuration file, which allows you to execute the project for this prober.

On the 4200A-SCS:

1. Open Clarius.
2. Choose **Select**.
3. Select **Projects**.
4. Search for **probesubsites**.
5. Drag the **probesubsites** project to the project tree.

Figure 884: probesubsites project tree



6. Click **Run**.

Commands and error symbols

The following table contains error and status symbols listed by command.

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
PR_WAFERCOMPLETE			X	X

Information and error code return values and descriptions

Value	Constant	Explanation
1	PR_OK	Success (OK)
4	PR_WAFERCOMPLETE	Next wafer loaded (confirmed)
-1008	INVAL_MODE	Invalid mode
-1011	BAD_MODE	Operation invalid in mode
-1013	UNINTEL_RESP	Unintelligible response
-1017	BAD_CHUCK	Bad chuck position
-1027	INVAL_PARAM	Invalid parameter

Appendix K

Using a Signatone CM500 Prober

In this appendix:

Signatone CM500 prober software

This section describes set up for the Signatone CM500 prober. Note that the CM500 driver provided with the Keithley Instruments 4200A-SCS also works with other Signatone probers with Interlink controllers, such as the WL250 and S460SE. The name CM500 used in the configuration and setup in this documentation applies to all Signatone semi-auto prober systems with Interlink controllers.

Software versions

The following software versions on the CM500 prober was used to verify the configuration of the prober with the 4200A-SCS:

- CM500.exe version 2.5
- For the S460-SE prober: S460SE.exe version 2.5

Probe station configuration

CAUTION

Refer to the Signatone CM500 or S460 Prober supporting documentation before attempting setup, configuration, or operation.

The general steps required to set up and configure the CM500 or S460 prober for use with the 4200A-SCS are:

Set up communications

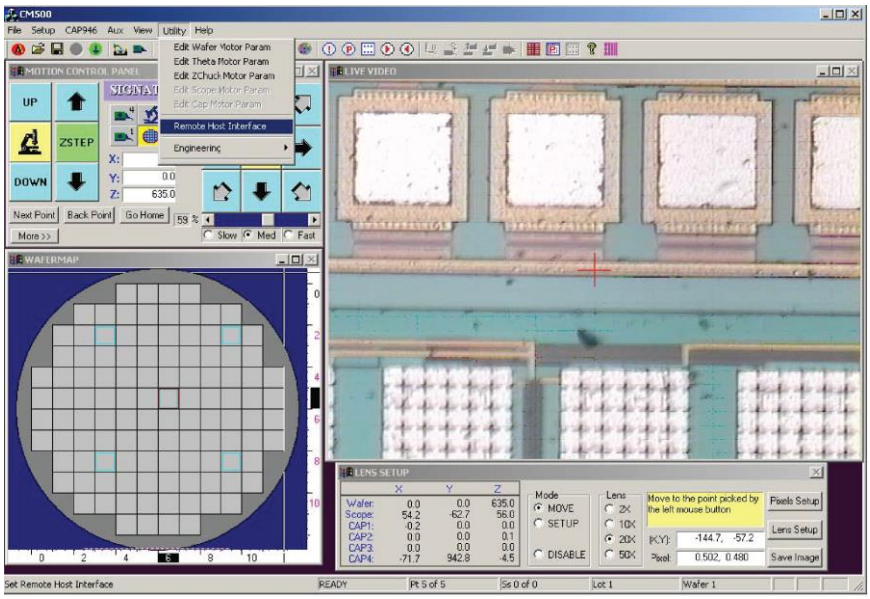
The Signatone CM500 prober is configured for GPIB communications only. Make sure the prober configuration is set up properly for the GPIB communications interface.

To set up communications:

1. Double-click the **CM500** icon on the Windows desktop. The prober initializes the wafer XY stage, theta, and Z chuck.

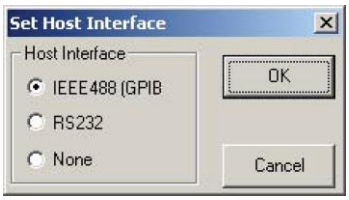
Figure 885: CM500 icon

Figure 886: CM500 Setup GPIB screen



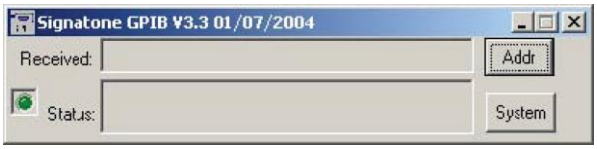
2. Select the **Utility** menu and run the **Remote Host Interface** command.

Figure 887: Select Host Interface



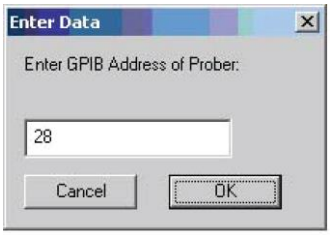
3. Select the **IEEE488** Host Interface. The Signatone GPIB driver window opens.

Figure 888: Signatone GPIB driver window



4. Click **Addr** and verify that the GPIB address matches the `GPIB_Address` setting in the 4200A-SCS prober configuration file `prbcnfg_CM500.dat` at `C:\s4200\sys\dat`. Note that the default GPIB address is set to 28.
5. If the address does not match, enter the new GPIB address, then click **OK**.

Figure 889: Set GPIB Address



Modify the prober configuration file

The default prober configuration file is shown below. As shown, the file is configured for use with a GPIB communications setup. Use a text editor such as Microsoft® Notepad to work with this file if needed.

On the 4200A-SCS, the configuration file is at `C:\s4200\sys\dat\prbcnfg_CM500.dat`.

```
# prbcnfg_CM500.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:
# 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
#
#
```

Set up wafer geometry

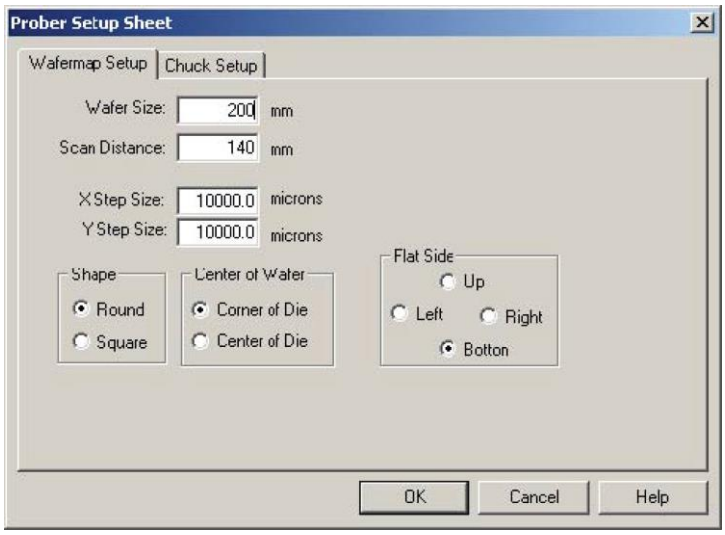
To set up wafer geometry:

1. Click the **Prober Setup** icon on the toolbar, shown below.

Figure 890: CM500 Prober Setup icon

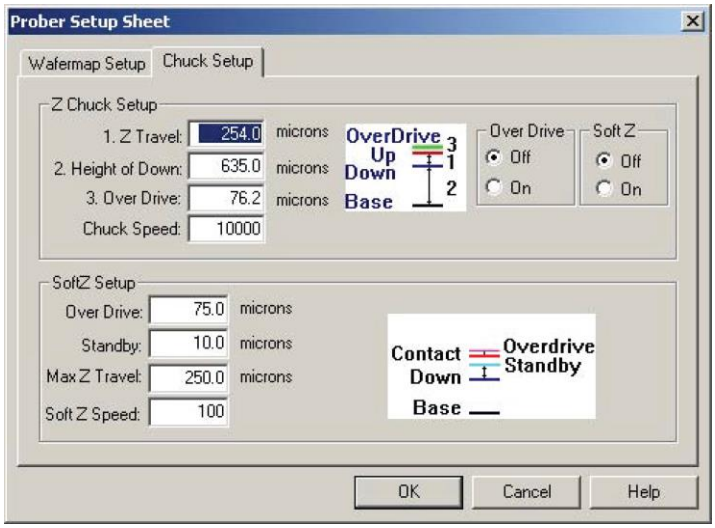
2. Select **Wafermap Setup** tab to set up wafer information, such as wafer size, scan distance, X step size, and Y step size.

Figure 891: CM500 Prober Setup Sheet



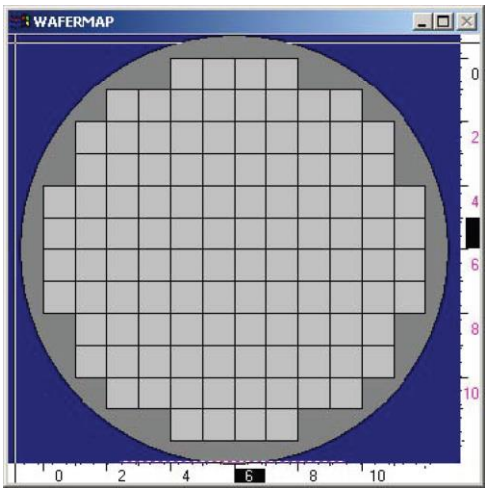
3. Select the **Chuck Setup** tab to enter Z chuck information, such as Z travel and overdrive distance.

Figure 892: CM500 Chuck Setup Sheet



4. After selecting **OK**, a new wafermap is displayed.

Figure 893: CM500 Prober wafermap



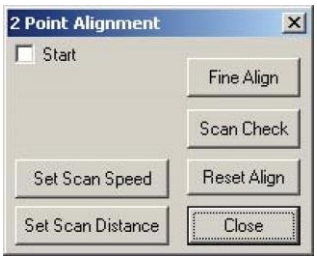
Load, align, and contact the wafer

1. Click the **Load wafer** icon on toolbar.

Figure 894: CM500 Prober load wafer icon

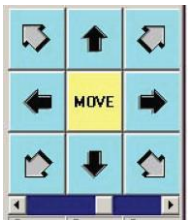
2. Select **Start** to move the wafer to Home and begin the sequences of 2-point alignment.

Figure 895: CM500 Prober 2 Point Alignment 1



3. Click the **Arrow** buttons on the window to move the wafer stage to reference point 1.

Figure 896: CM500 Prober manual MOVE buttons



4. Select **Set Point1**.

Figure 897: CM500 Prober 2 Point Alignment 3

5. The Wafer stage moves to the other side as set by the scan distance.
6. Click the **Arrow** buttons on the window to move wafer stage to reference point 2.
7. Select **Set Point2**.
8. The Prober software rotates the theta motor for the proper alignment.
9. Click **Scan Check** to verify that the wafer is aligned correctly.
10. Click **Fine Align** to make a minor alignment.
11. After the wafer is aligned, set the HOME die of the wafer and wafermap.

Set the Home die of the wafer:

1. Move the wafer stage to the actual location that needs to be set as HOME.
2. When completed moving the wafer stage, click the **Set Home** icon on the toolbar.

Figure 898: CM500 Prober Set Home icon

Set the Home die of the wafermap:

1. To set **HOME** on the wafermap, click the **Edit wafermap** icon on the toolbar.

Figure 899: CM500 Prober Editmap function icon

2. Select the **Set Home of Map** function.

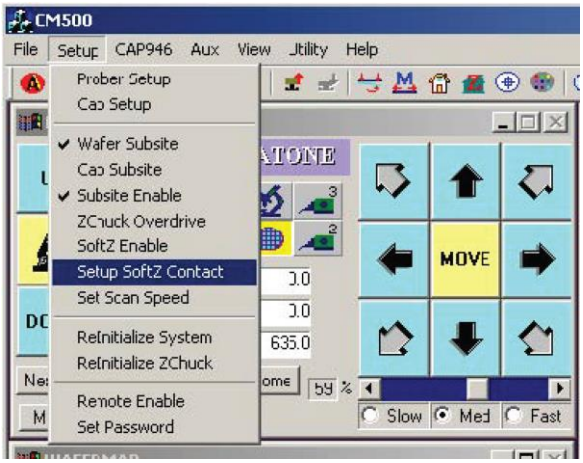
Figure 900: CM500 Prober Edit Map Function window

3. Click the Home die on the wafermap that needs to be set as Home.
4. **Close** the Edit Map Function window.

Adjust the Z chuck:

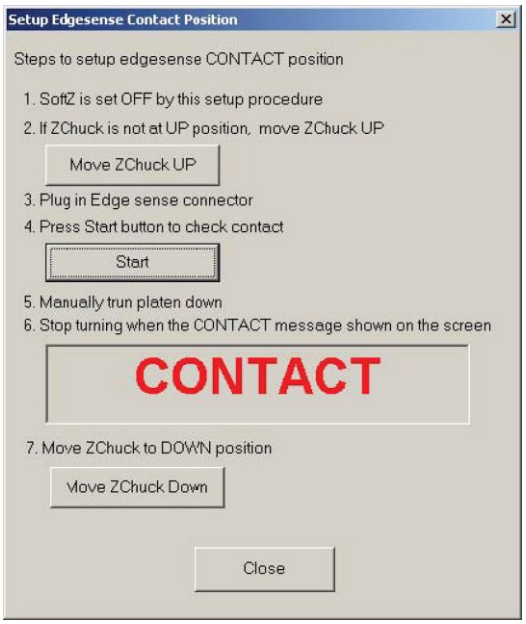
1. If an edge sense card is being used as the contact input for Z Chuck, you must select the **Setup SoftZ Contact** option from the **Setup** menu.

Figure 901: CM500 Prober Setup softz contact command



2. Follow the instructions on the window to adjust the height of platen and to determine the contact position of the Z Chuck.

Figure 902: CM500 Prober Setup Edgesense Contact Position window

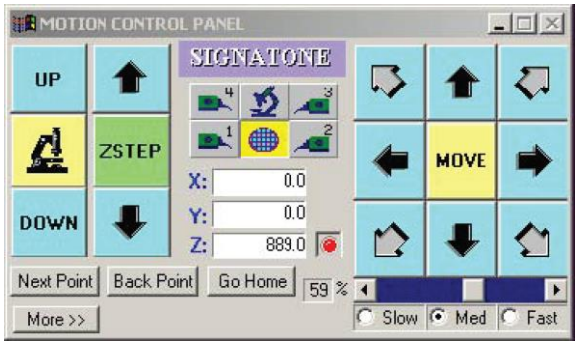


3. Move the Z Chuck up to confirm contact condition using the **Contact** icon on the toolbar.

Figure 903: CM500 Prober Z Chuck Up (CONTACT) icon

4. If the edge sense is plugged in for contact input, turn **ON** SoftZ. A red LED will appear in the motion control panel.

Figure 904: CM500 Prober Motion Control Panel



5. Move the Z Chuck down using the **Separate** icon on the toolbar.

Figure 905: CM500 Prober Z Chuck Down (SEPARATE) icon

Set up programmed sites without a subsite

1. Click the **Program Site** icon on the toolbar.

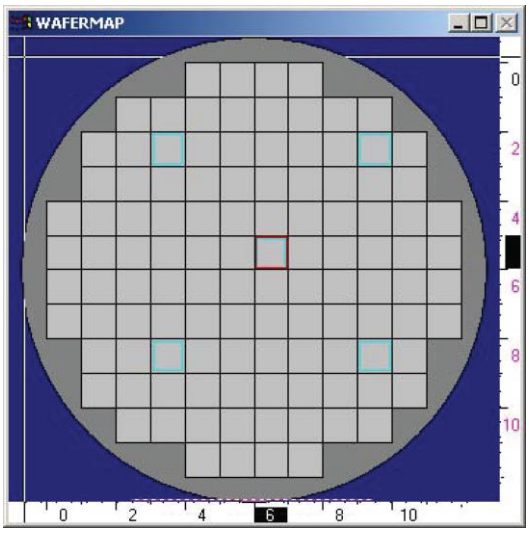
Figure 906: CM500 Prober Edit Program Sites icon

2. Select the **Enter Site Map** function.

Figure 907: CM500 Prober Edit Program Site window

3. Move the mouse onto the WAFERMAP window and then do one of the following actions:
   - Select the dies to be tested on the wafermap and click **Enter**.
   - Click **Enter All** to test all dies.

Figure 908: CM500 Prober wafermap includes program sites



4. To step through all the programmed sites, select the **Run Program Site** icon on the toolbar.

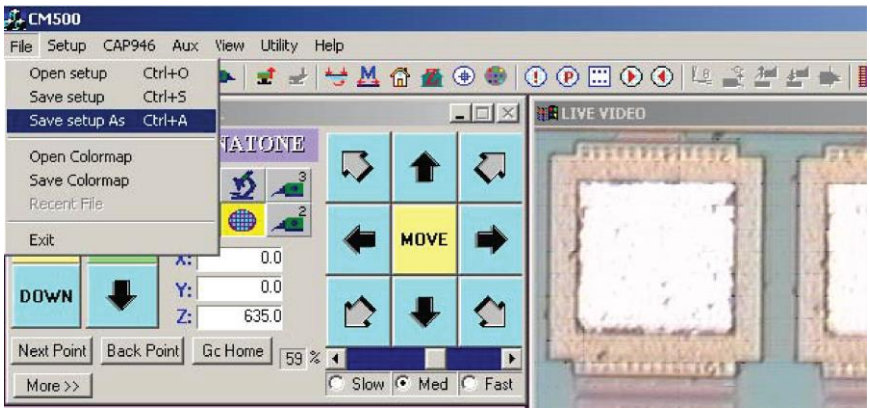
Figure 909: CM500 Prober Run Program Sites icon

5. Click the **To First Site** button to move the prober to the first programmed site for testing. Make sure the Subsite (template) is disabled here.

Figure 910: CM500 Prober Run Program Site window

6. In the **File** menu bar, select the **Save setup As** command to save the file to a hard disk. You can load this setup later to restore this setup if needed.

Figure 911: CM500 Prober save setup



The Prober is now ready to accept a remote command from the 4200A-SCS.

Set up programmed sites with a subsite

1. Click the **Edit Subsite** icon on the toolbar.

Figure 912: CM500 Prober Edit Subsite icon

2. Select **Wafer** as the subsite device.

Figure 913: CM500 Prober Edit Subsite window

3. Move the wafer stage to the HOME position.

NOTE

All data recorded for the subsite is relative to the corner of the home die. You 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 clicking **Enter**.

4. To step through all the programmed sites and subsites, click the **Run Program Site** icon on the toolbar.
5. Make sure the Subsite (template) is **Enabled** if subsites are to be used.
6. Click the **To First Site** button to move wafer stage to first site of probing lists.

Figure 914: CM500 Prober Enable Subsite

7. In the **File** menu bar, select the **Save setup As** command to save the file to a hard disk. You can load this setup next time without going through all of the procedures again.
8. The Prober is now ready to accept a remote command from the 4200A-SCS.

Clarius project example for probe sites

The following is a step-by-step procedure to configure the Clarius project to execute testing and automatic wafer stepping to all programmed sites successfully. When the CM500 prober is connected to the 4200A-SCS by GPIB interface, the 4200A-SCS is the GPIB master controller and the CM500 is always in listening mode. The 4200A-SCS will send control commands to the CM500 to move the prober to next site during the automatic testing. The interface commands are `PrInit`, `PrChuck`, `PrMovNxt`, and `PrSSMovNxt`. You will need to add these commands into the Clarius project.

CM500

On the probe station computer, complete the procedures in the Probe station configuration (on page K-2) section.

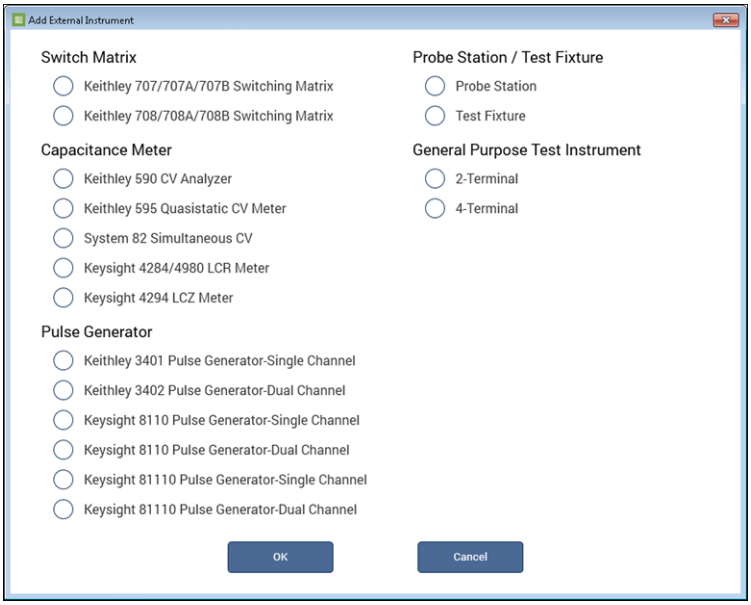
Use KCon to add a prober

You use KCon on the 4200A-SCS to add the prober to the configuration.

On the 4200A-SCS, use KCon to add the prober to the configuration:

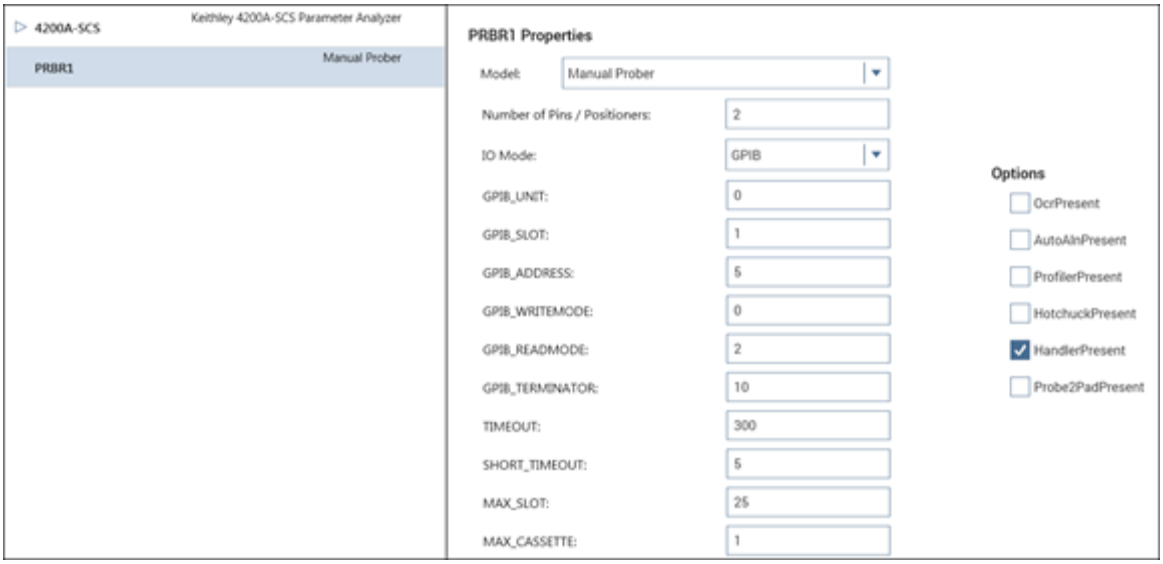
1. Open KCon.
2. At the bottom of the Configuration Navigator, select **Add External Instrument**. The Add External Instrument dialog box is displayed.

Figure 915: Add a prober in KCon



3. Select **Probe Station**.
4. Select **OK**. The KCon navigation displays PRBR1 in the Configuration Navigator and the properties.

Figure 916: Use KCon to select a prober



5. For the Model, select the **Signatone CM500 (WL250) Prober**.
6. Make sure the **Number of Pins / Positioners** is correct. The number of pins defined here determines the pins that are available to assign to a switch matrix card column.
7. Save the configuration.
8. Exit KCon.

Clarius project example

To set up a new prober project:

1. Start **Clarius**.
2. Choose **Select**.
3. Select the **Projects** tab.
4. Drag **New Project** into the project tree.
5. Choose **Yes** to create a new project.
6. Rename the project **PRB_CM500**.
7. Select the **Wafer Plan** tab.
8. Drag **Site** to the project tree.
9. Select the **Actions** tab.
10. In the Search box, enter **prober**.
11. Drag the **Prober Initialization (prober-init)** action to the project tree. Make sure it is under the subsite.

Configure the prober project:

1. Select **Configure**. Make sure `prober-init` is selected in the project tree.

Figure 917: Set prober-init parameters

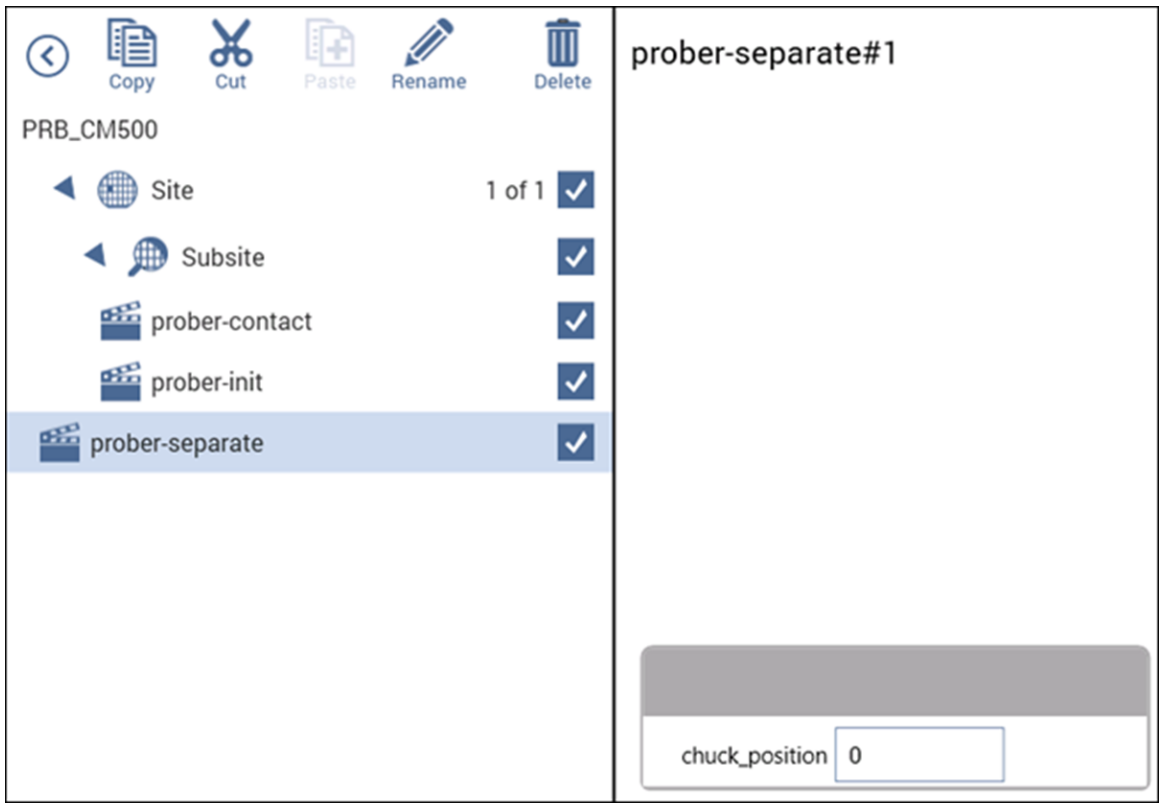
2. Set the **mode** to 6.
3. Set the **xdie_size** and **ydie_size** for your wafer.
4. Set units to either **0** for English or **1** for metric.
5. Check the **subprobtype**. If the CM500 prober is presently not at its first site, set **subprobtype** to **1**; otherwise, set it to **0**.

Set up actions:

1. In the project tree, select **Subsite**.
2. Choose **Select**.
3. Select the **Actions** library.
4. Add the **Prober Chuck Position (prober-contact)** action twice.
5. Select the **prober-contact_1** action.
6. Rename the action **prober-separate**.
7. Select **Configure**.
8. Set **chuckposition** to **0**. This moves the Z chuck to the down (separate) position.

9. Select the **prober-contact** action.
10. Set **chuckposition** to **1**. This moves the Z chuck to the contact position.
11. Place the **prober-separate** action at the bottom of the project tree.
12. Right-click **prober-separate** and select Promote Action twice so that `prober-separate` is at the site level.

Figure 918: New prober-separate UTM



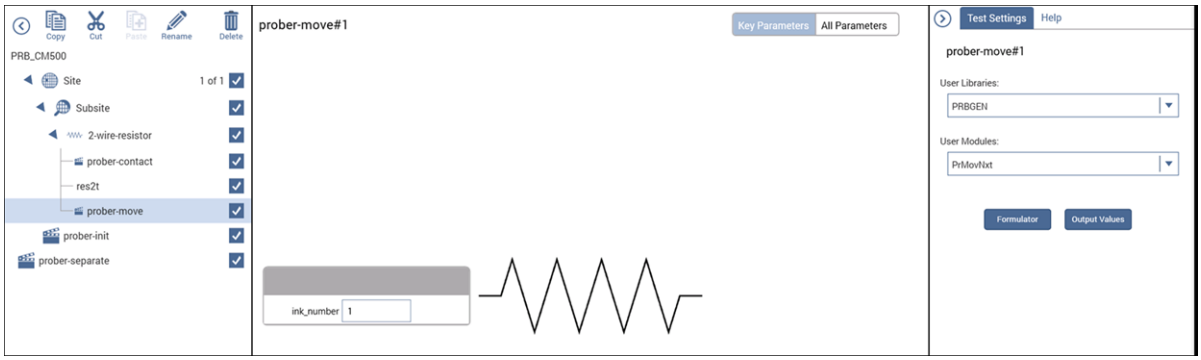
NOTE

The position of the action in the project tree determines when the action is run during a test. For example, in a device with multiple tests, you can run the device level directly. The tests under that device are executed sequentially. If an action is under the device level, the action runs in sequence with the tests. Similarly, actions under the subsite, site, or project levels execute automatically when the subsite, site, or project is run.

Create a test in the subsite level:

1. Choose **Select**.
2. Choose the **Tests** library.
3. Select a test for the device on your wafer.
4. Add the test to the subsite. When you add a test, an appropriate device is automatically added. You can also add a device from the Devices library and then add a test to the device.
5. Choose the **Actions** library.
6. Add the **Prober Move to Next Site (prober-move)** action.
7. Drag **prober-contact** so that it is immediately before your test.
8. Drag **prober-move** so that it is immediately after your test.
9. Select **Configure**.
10. For `prober-move`, set the **inknumber** to 1 if you need to trigger inker 1; otherwise, set it to 0.

Figure 919: prober-next in the project tree



Probesites Clarius project example

On the 4200A-SCS, use Clarius to open and run the `probesites` project using the new configuration file, which allows you to execute the project for this prober. This project uses a Series 700 Switching System and the `connect` action to change the instruments connected to each pin without changing the physical configuration.

To set up `probesites`:

1. In Clarius, choose **Select**.
2. Select the **Projects** library.
3. In the Search box, enter **probesites**.
4. Create the `probesites` project.
5. Select **Configure**.
6. Set the `prober-init` mode to **6**.
7. Set the **subprobtype**. If the CM500 prober is presently not at its first site, set the **subprobtype** to **1**; otherwise, set it to **0**.

Figure 920: Set prober-init mode parameters

8. In the project tree, select **probesites**.
9. Choose **Run** to execute the entire project.

Probesubsites Clarius project example

The following procedure configures a Clarius project to execute testing and automatic wafer stepping to all programmed subsites.

Use the 4200A-SCS to do this example. Use Clarius to open and run the `probesubsites` project using the new configuration file, which will allow you to execute the project for this prober.

This project uses a Series 700 Switching System and the `connect` actions to change the instruments connected to each pin without changing the physical configuration.

From Clarius:

1. Choose **Select**.
2. Select the **Projects** library.
3. Create the `probesubsites` project.
4. Select **Configure**.
5. In the project tree, select the **prober-init** action.
6. Set the **mode** to **6**.

Figure 921: Set prober-init mode parameters

7. Set the **subprobtype**. If the CM500 prober is not at its first site, set the `subprobtype` to **1**; otherwise, set it to **0**.
8. In the project tree, select **probesubsites_1**.
9. Select **Run** to execute the project.

Commands and error symbols

The following list contains error and status symbols listed by command.

Available commands and responses

	PrChuck	**PrInit**	**PrMovNxt**	**PrSSMovNxt**
PR_OK	X	X	X	X
BAD_CHUCK	X			
INVAL_MODE	X		X	X
UNINTEL_RESP	X	X	X	X
INVAL_PARAM		X		
BAD_MODE			X	X
PR_WAFERCOMPLETE			X	X

Information and error code return values and descriptions

Value	Constant	Explanation
1	PR_OK	Success (OK)
4	PR_WAFERCOMPLETE	Next wafer loaded (confirmed)
-1008	INVAL_MODE	Invalid mode
-1011	BAD_MODE	Operation invalid in mode
-1013	UNINTEL_RESP	Unintelligible response
-1017	BAD_CHUCK	Bad chuck position
-1027	INVAL_PARAM	Invalid parameter

Appendix L

Wafer-level reliability testing

In this appendix:

Introduction

This section provides information on wafer-level reliability (WLR) testing. Included are tests for:

- Hot-carrier injection (HCI)
- Negative-bias temperature instability (NBTI)
- Electromigration
- Charge-to-breakdown measurement (QBD)

AC, or pulsed, stress is a useful addition to the typical stress-measure tests for investigating both semiconductor charge trapping and degradation behaviors. NBTI and time-dependent dielectric breakdown (TDDB) tests consist of stress / measure cycles.

The applied stress voltage is a DC signal, which is used because it maps more easily to device models. Incorporating pulsed stress testing provides additional data that permits a better understanding of device performance in frequency-dependent circuits.

The test pulse stresses the device for HCI, NBTI, and TDDB test instead of DC bias by outputting a train of pulses for a period of time (stress time). Pulse characteristics are not changed during the stress-measure test. The test then uses SMUs to measure device characteristics such as V_{th} and G_m.

This section includes background information on HCI degradation and summaries for using 4200A-SCS projects to measure HCI degradation and other WLR tests.

NOTE

The projects for HCI and QBD testing comply with the standard procedures established by JEDEC. In 4200A-SCS documentation, all references to the JEDEC standards and duplicated JEDEC documentation are 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

This document is intended for use in the MOS Integrated Circuit manufacturing industry fabrication processing and test and 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), the current-ramp (J-Ramp), and the constant current (Bounded J-Ramp) test. Each test is designed for simplicity, speed, and ease of use.

NOTE

The JEDEC standard procedures are available on the JEDEC website (jedec.org).

When you visit the JEDEC website, you must register before you can access the standards. Registration is free.

HCI and WLR projects

The 4200A-SCS projects for HCI and WLR testing include:

- `hci-1-dut`
- `hci-4-dut`
- `nbti-1-dut`
- `em-const-i`
- `qbd`

All of these projects except `qbd` use subsite cycling in the stress/measure mode. For details, see Subsite cycling (on page 6-231).

You can use each of these projects as configured or modify them for your testing requirements.

Hot Carrier Injection projects

The Hot Carrier Injection (HCI) projects determine HCI on MOSFETs. The `hci-1-dut` project determines HCI degradation on a single 4-terminal n-MOSFET. The `hci-4-dut` project determines HCI degradation on two 4-terminal n-MOFETs and two 4-terminal p-MOSFETs.

The `hci-1-dut` project is shown in the following figure.

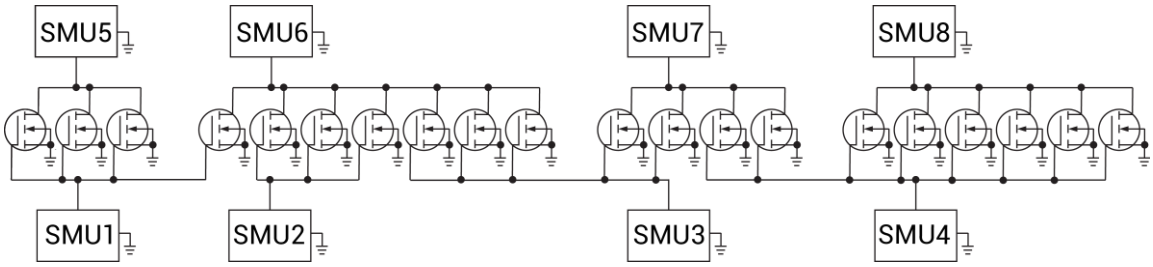
Figure 922: Project tree showing hci-1-dut project

For the `hci-1-dut` project, the `hci` subsite is set up 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 time before repeating the tests.

The `hci-4-dut` project is similar to the `hci-1-dut` project except that 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. The figure below shows an example of 20 parallel-connected devices being stressed by eight gate and drain voltages.

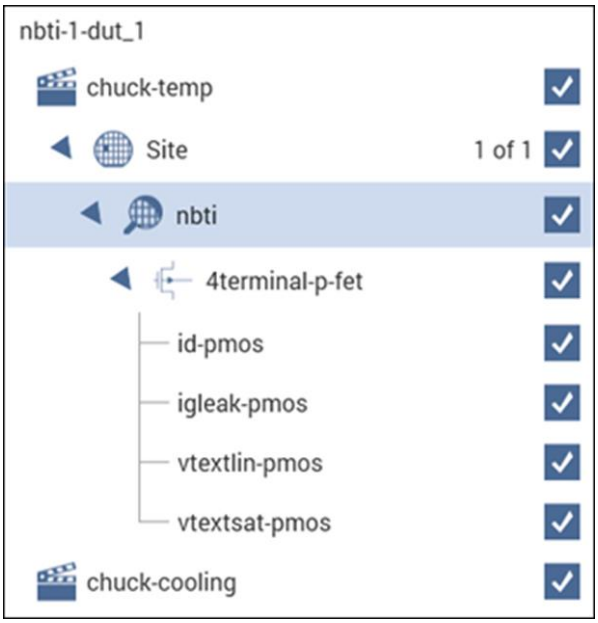
Figure 923: HCI and NBTI tests: 20 parallel-connected devices stressed by voltage



Negative Bias Temperature Instability project

The Negative Bias Temperature Instability (`nbti-1-dut`) project performs NBTI testing on a p-MOSFET with temperature and DC stress. The following figure shows the project tree when the `nbti-1-dut` project is selected.

Figure 924: Project tree for nbti-1-dut

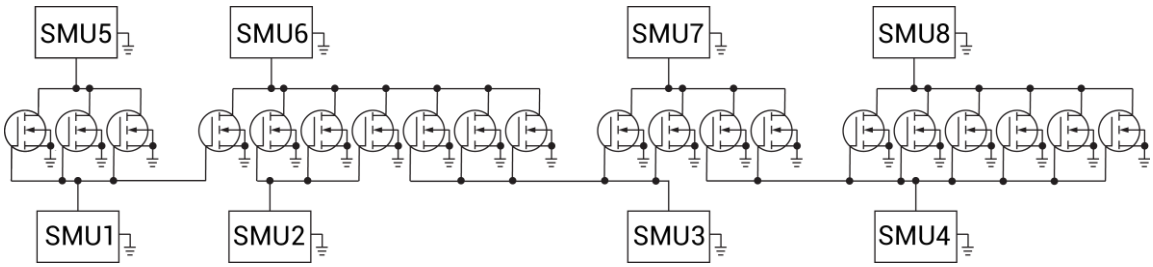


The `nbti` subsite is configured for subsite cycling using voltage stressing for a p-channel MOSFET (PMOS) device.

This project includes actions that control the temperature of the chuck. The subsite test will not start until the chuck reaches the specified temperature. After the first pre-stress cycle to characterize the device, subsequent cycles voltage stress the device for a specified time before repeating the tests. After the subsite cycling is complete, the `chuck-cooling` action cools the chuck.

In a parallel connection scheme, up to 20 devices can be stressed by voltage. The figure below shows an example of 20 parallel-connected devices being stressed by eight gate and drain voltages.

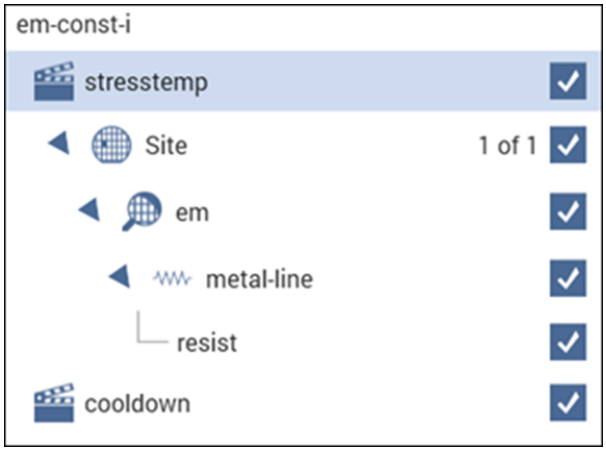
Figure 925: HCI and NBTI tests: 20 parallel-connected devices stressed by voltage



Electromigration project

The Electromigration project (`em-const-i`) is shown in the figure below.

Figure 926: em-const-i project tree



The subsite (`em`) is configured for subsite cycling using current stressing on a single device (`metal-line`).

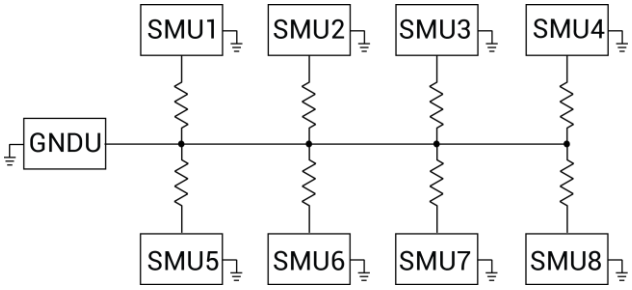
This project includes actions to control the temperature of the chuck. The subsite will not start cycling until the chuck reaches the specified temperature. After the first pre-stress cycle to perform a characterization test on the device, subsequent cycles current stress the device for a specified time before repeating the test. After the subsite completes, the `cooldown` action cools the chuck.

You can modify the `em-const-i` project 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, you can stress up to eight devices can be stressed, as shown in the following figure.

NOTE

Current stressing: When setting the current stress level for each device in the subsite, 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 927: EM test: Eight devices being current stressed by eight SMUs

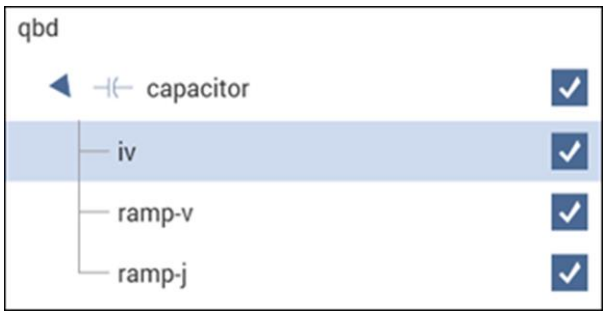


Charge-to-Breakdown Test of Dielectrics project

The `qbd` project includes tests 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 does not use subsite cycling.

Details on these tests are described in V-ramp and J-ramp tests (on page L-10).

Figure 928: qbd project tree



HCI degradation: Background information

HCI degradation is one of the most important device issues facing the semiconductor industry. Small gate length and process variations in the 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.

The devices of today 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, 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.

Configuration sequence for subsite cycling

The following projects use subsite cycling:

- `hci-1-dut`
- `hci-4-dut`
- `nbti-1-dut`
- `em-const-i`

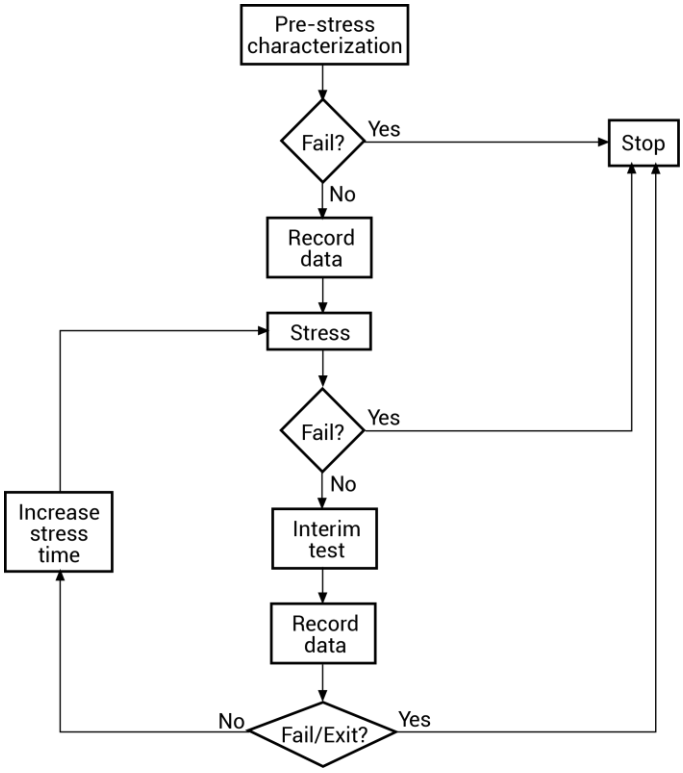
The process flow for these projects is shown in the figure below.

NOTE

You can create a new project for subsite cycling or you can use one of the existing projects as a starting point and change it as needed. For details, see Set up a basic project (on page 6-10).

To configure the subsite for subsite cycling, refer to Configure Subsite Cycling (on page 6-195).

Figure 929: Process flow HCI/NBTI/constant current EM



V-ramp and J-ramp tests

Charge-to-breakdown measurement (Q_{BD}) tests are a measure of time-dependent gate oxide breakdown. They are a standard method used to determine quality of gate oxides in MOS devices.

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.

User modules for these tests are provided in the `wlrlib` user library. The user modules in the `wlrlib` user library run linear regression and charge-to-breakdown (Q_{BD}) ramp tests for wafer-level reliability (WLR) testing. These user modules are summarized in the table below.

wlrlib user modules

User module	Description
`llsq1`	Performs simple linear regression.
`qbd_rmpv`	Performs a charge-to-breakdown test using the QBD V-ramp test.
`qbd_rmpj`	Performs a charge-to-breakdown test using the QBD J-ramp test.

V-ramp test: qbd_rmpv User Module

The V-ramp test uses the `qbd_rmpv` user module of the `wlrlib` user library.

Usage

See JEDEC standards (on page L-2), JESD35-A, "PROCEDURE FOR WAFER-LEVEL TESTING OF THIN DIELECTRICS."

NOTE

Some of the descriptions of the following variables are quoted from the JESD35-A standard. The variables quoted from the standard include this reference identification: (Ref. JESD35-A).

```
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_stress, 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);
```

Input variables

status	Returned values are placed in the Analyze sheet
hi_pin	High pin (usually the gate pin) (-1 to 72); enter -1 to not connect
lo_pin1 *lo_pin2* *lo_pin3*	Usually for source drain and substrate connection; depending on device structure, some of those pins are optional; enter -1 to not connect
HiSMUId	ID string of the SMU outputting the stress
LoSMUId1 *LoSMUId2* *LoSMUId3*	ID string of the SMU connected to ground terminal; these three IDs can be same
v_use	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	Oxide breakdown failure current when biased at *v_use*; see **Details**
hold_time	Time in ms to hold the first stress (*v_start*)
v_start	Starting voltage (V) for voltage ramp; typical value is *v_use* (Ref. JESD35-A)
v_step	Voltage (V) ramp step height; maximum of 0.1 MV/cm; refer to **Details**
t_step	Voltage ramp step time in ms, used to determine the voltage ramp rate; should be less or equal than 100 ms (typically 40 ms to 100 ms)
measure_delay	Time delay in ms for measurement after each voltage stress step; should be less than *t_step* (ms)
I_crit	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	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 1 µA (Ref. JESD35-A)
I_max	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	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 to 5, recommended value: 10 to 100)
exit_slope_mult	Change of slope failure criteria; this is the factor of change in FN slope, which, if exceeded, will result in failure (2.5 to 5, recommended value: 3)
q_max	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	Maximum stress time allowed in seconds; reaching this limit will result in test to finish (s)
v_max	The maximum voltage limit for the voltage ramp; this limit is specified at 30 MV/cm for oxides less than 20 nm thick and 15 MV/cm for thicker oxides; refer to **Details**
area	Area of oxide structure (cm^2)
exit_mode	Failure criteria mode; refer to **Details**
V_size	Size of data array; maximum 65535
I_size	Size of data array; maximum 65535
T_size	Size of data array; maximum 65535
q_size	Size of data array; maximum 65535

Output variables

V_stress	Voltage stress array
I_stress	Measured current array
T_stress	Time stamp array indicating when current is measured
q_stress	Accumulated charge array
I_use_pre	Measured oxide current at *v_use*, before starting the ramp (Ref. JESD35-A)
I_use_post	Measured oxide current at *v_use*, after the ramp finished (Ref. JESD35-A)
Q_bd	Charge-to-breakdown; cumulative charge passing through the oxide before breakdown (C) (Ref. JESD35-A)
q_bd	Charge-to-breakdown density (C/cm^2) (Ref. JESD35-A)
v_bd	Applied voltage at the step just before oxide breakdown (Ref. JESD35-A)
I_bd	Measured current at *v_bd*, just before oxide breakdown
t_bd	Time stamp when measuring *I_bd*
v_crit	Applied voltage at the step when the oxide current exceeds *I_crit* (Ref. JESD35-A)
v_box	Applied voltage at the step when the oxide current exceeds *I_box* (Ref. JESD35-A)
failure_mode	■ Initial test failure ■ Catastrophic failure (initial test pass, ramp test fail, post test fail) ■ Masked Catastrophic (initial test pass, ramp test pass, post test fail) ■ Non-Catastrophic (initial test pass, ramp test fail, post test pass) ■ Others (initial test pass, ramp test pass, post test pass)
test_status	See **Details**

Details

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," April 2011. This algorithm forces a linear voltage ramp until the oxide layer breaks down. This algorithm is capable of a maximum voltage of ±200 V. The flow diagram for the V-ramp test is shown in V-Ramp Flow Diagram (on page L-14).

Notes on input variables

hi_pin and *lo_pinX*: If there is no switching matrix in the system, enter either 0 or –1 for *hi_pin* and *lo_pinX* to bypass switch.

I_init: The typical value of *I_init* is 10 µA/cm^2 and may change depending on oxide area. For maximum sensitivity, the specified value should be well above the worst 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).

v_step: As an example, the maximum value of *v_step* can be calculated using Tox*0.1 MV/cm, where Tox is in unit of centimeters. This is 0.1 V for a 10 nm oxide (Ref. JESD35-A).

v_max: As an example, *v_max* can be estimated from T_{ox}*30 MV/cm, where T_{ox} is in centimeters. This is 35 V for a 10.0 nm Oxide (Ref. JESD35-A).

exit_mode: Select:

- 0: Specifies that oxide failure is determined by a measured current that exceeds the user specified failure current (*fail_current*)
- 1: 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 10 times the user-specified noise current. For measured currents below this value, the *fail_current* is used as the exit criteria.

Notes on output variables

test_status:

- 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

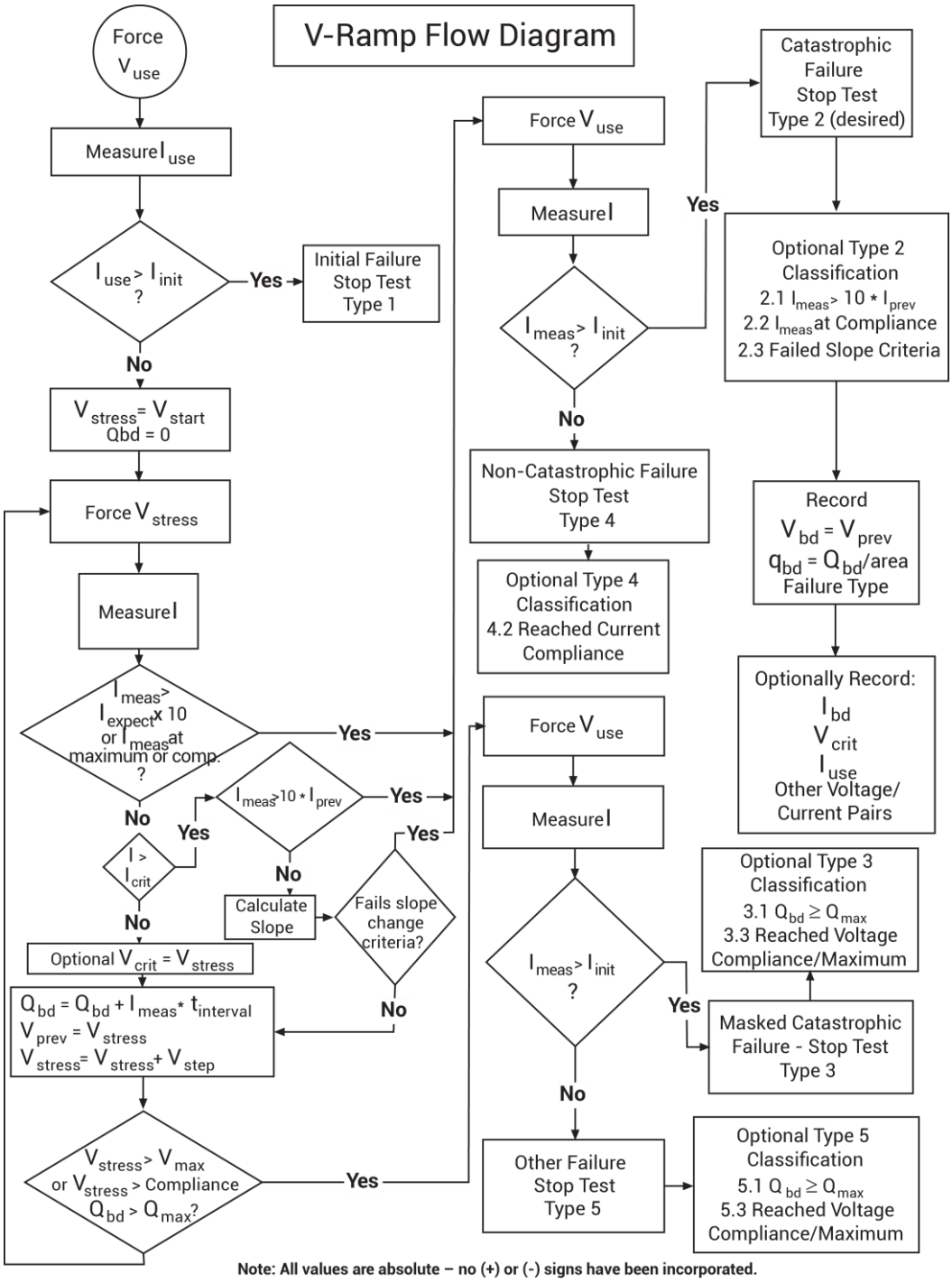
Invalid Test Result - Result = 1e21.

V-Ramp Flow Diagram

NOTE

The following diagram from JESD35-A has been reproduced with permission from JEDEC. The flowchart is JEDEC.

Figure 930: Detailed V-ramp flow diagram



J-ramp test: qbd_rmpj User Module

The J-ramp test uses the `qbd_rmpj` user module of the `wlrlib` user library.

Usage

```
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);
```

Input variables

status	Returned values are placed in the Analyze sheet
hi_pin	High pin (usually the gate pin) (-1 to 72); enter -1 to not connect
lo_pin1 *lo_pin2* *lo_pin3*	Usually for source drain and substrate connection; depending on device structure, some of those pins are optional; enter -1 to not connect
HiSMUId	ID string of the SMU outputting the stress
LoSMUId1 *LoSMUId2* *LoSMUId3*	ID string of the SMU connected to ground terminal; these three IDs can be same
v_use	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	Oxide breakdown failure current when biased at *v_use*; typical value is 10 μA/cm^2 and may change depending on oxide area; see **Details**
I_start	Starting current (A) for current ramp; typical value is *I_init* (Ref. JESD35-A)
F	Current multiplier between two successive current steps (Ref. JESD35-A)
t_step	Current ramp step time (s) (Ref. JESD35-A)
exit_volt_mult	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	Maximum ramp current (A) (Ref. JESD35-A)
q_max	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	Area of oxide structure (cm^2)
V_size	Size of data array; maximum 65535
I_size	Size of data array; maximum 65535
T_size	Size of data array; maximum 65535
q_size	Size of data array; maximum 65535

Output variables

V_stress	Voltage stress array
I_stress	Measured current array
T_stress	Time stamp array indicating when current is measured
q_stress	Accumulated charge array
Q_bd	Charge-to-breakdown; cumulative charge (C) passing through the oxide before breakdown (Ref. JESD35-A)
q_bd	Charge-to-breakdown density (C/cm^2) (Ref. JESD35-A)
v_bd	Applied voltage at the step just before oxide breakdown (Ref. JESD35-A)
I_bd	Measured current at *v_bd*, just before oxide breakdown
t_bd	Time stamp when measuring *I_bd*
failure_mode	■ Initial test failure ■ Catastrophic failure (initial test pass, ramp test fail, post test fail) ■ Masked Catastrophic (initial test pass, ramp test pass, post test fail) ■ Non-Catastrophic (initial test pass, ramp test fail, post test pass) ■ Others (initial test pass, ramp test pass, post test pass)
test_status	See **Details**

Details

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," April 2011. This algorithm forces a logarithmic current ramp until the oxide layer breaks down. This algorithm is capable of a maximum current of ±1 A if a high power SMU is used. The flow diagram for the V-ramp test is shown in J-ramp flow diagram (on page L-19).

See JEDEC standard JESD35-A "Procedure for Wafer-Level Testing of Thin Dielectrics," April 2011, referenced in Signatone CM500 Prober (on page K-1).

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).

Notes on input variables

NOTE

If there is no switching matrix in the system, input either `0` or `-1` for *hi_pin* and *lo_pins* to bypass switch.

I_init: For maximum sensitivity, the specified value should be well above the worst-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).

Notes on output variables

test_status:

- 0: No test 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

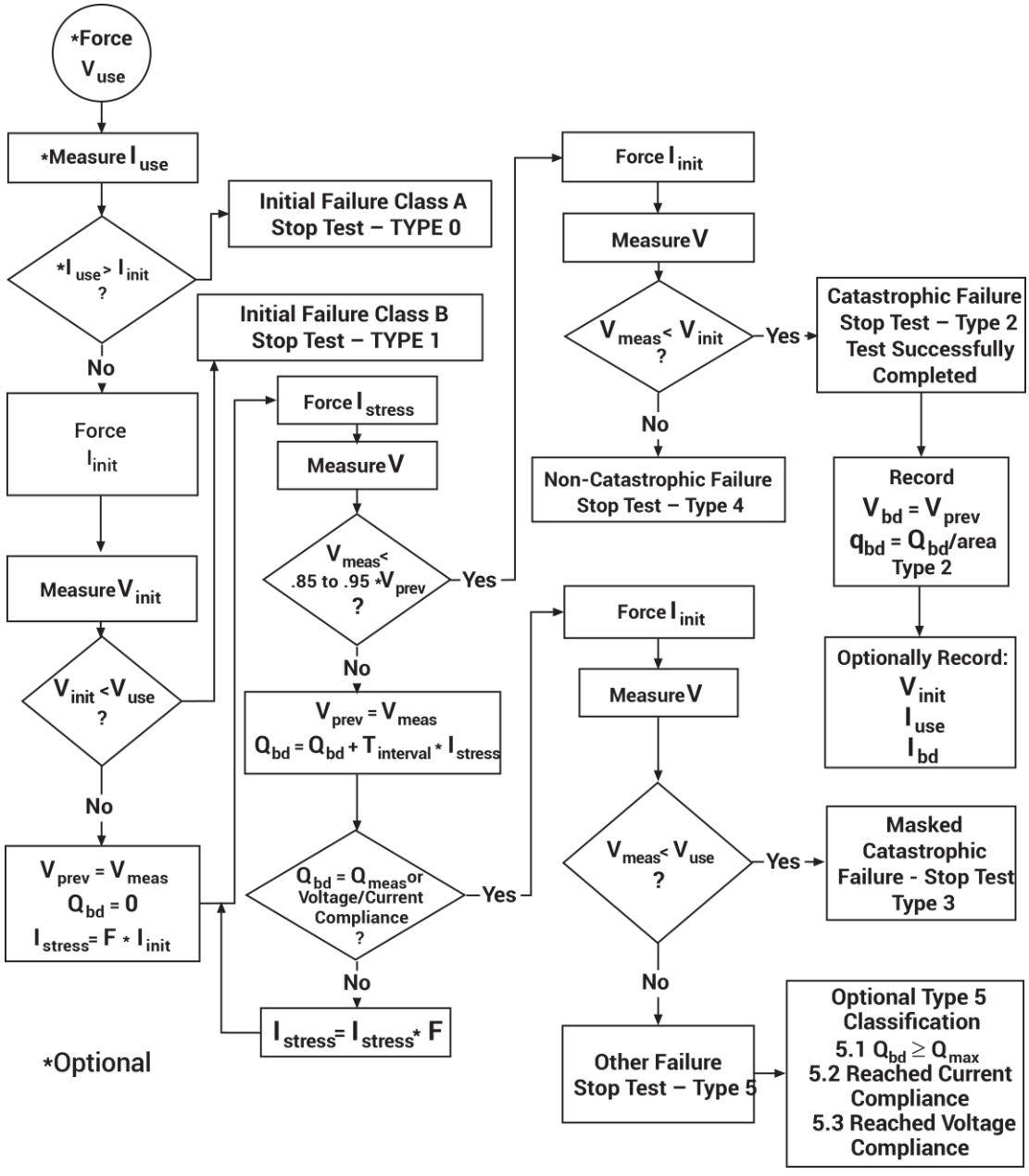
Invalid Test Result - Result = 1e21.

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.

Figure 931: J-ramp flow diagram



NOTE

All values are absolute – no (+) or (-) signs have been incorporated.

Appendix M

Using an MPI Probe Station

In this appendix:

MPI prober software

MPI supported probers include the TS2000, TS2000-DP, TS2000-HP, TS2000-SE, TS3000, and TS3000-SE.

To configure and operate one of the supported MPI probers with the Keithley Instruments 4200A-SCS, you need the MPI Sentio Software Suite application. This application provides access to configuration and help programs.

The MPI prober computer has MPI Sentio Software Suite installed. This is the main control software for the MPI prober. It provides the configuration and setup needed so that the prober can be controlled remotely using the 4200A-SCS.

Software version

The following software version was used to verify the configuration of the MPI probe station with the 4200A-SCS:

- MPI Sentio Software Suite version 2.9

Probe station configuration

CAUTION

Make sure that you are familiar with the MPI prober and its supporting documentation before attempting setup, configuration, or operation.

The general steps required to set up and configure the MPI prober for use with the 4200A-SCS include:

- Set up communications (on page M-2)
- Load, align, and contact the wafer (on page M-4)
- Set up wafer geometry (on page M-4)
- Create a site definition and define a probe list (on page M-5)

Set up communications

The MPI prober supports either GPIB or RS-232 communications to the 4200A-SCS. The following sections describe the steps to configure the prober and 4200A-SCS communications for either GPIB or RS-232.

Set up communications on the prober

The following steps describe how to set up the MPI prober for GPIB or RS-232 communications with a 4200A-SCS.

Set up the GPIB connection

To set up the GPIB connection:

1. Connect the MPI probers GPIB port to the 4200A-SCS GPIB port using a shielded GPIB cable (such as Keithley Instruments 7007-1 or 7007-2 GPIB cable).
2. Open the MPI Sentio configuration file, which is located on the MPI prober computer at: `C:\ProgramData\MPI Corporation\Sentio\config\config.xml`
3. Locate the communication configuration, which is in the node `Configuration / Main / RemoteServer`.
4. In the `RemoteServer` node, set the Type attribute to `GPIB`.
5. Set the Config attribute to *`BoardName`*`:`*`BoardAddress`*`:`*`VenderCode`*, where:
   - *`BoardName`* is the name of GPIB interface of the prober, such as `GPIB0`. Refer to the GPIB documentation to determine the name of the GPIB interface.
   - *`BoardAddress`* is the GPIB address of the MPI prober. This is an integer from 1 to 31. This address must be unique. You cannot use duplicate addresses on the same GPIB communication channel.
   - *`VenderCode`* is a string that identifies the vendor GPIB driver that will be used on the MPI prober. This must be either `NI` or `ADLINK`.

An example configuration for a GPIB card identified as `GPIB0`, with GPIB address `11`, and that is a National Instruments GPIB card is:

```
<RemoteServer Type="GPIB" Config="GPIB0:11:NI" />
```

Set up RS-232 communications

To set up the RS-232 connection:

1. Connect the COM port of the MPI probe station computer to the 4200A-SCS COM1 port using a DB9 female to DB9 female cable (shielded null modem cable).
2. Open the MPI Sentio configuration file, which is located on the MPI prober computer at: `C:\ProgramData\MPI Corporation\Sentio\config\config.xml`
3. Locate the communication configuration, which is in the node `Configuration / Main / RemoteServer`.
4. In the `RemoteServer` node, set the Type attribute to `RS232`.
5. Set the Config attribute to *`ComPort`*`:`*`BaudRate`*`:`*`Parity`*`:`*`Handshake`*, where:
   - *`ComPort`* is the name of the RS-232 COM port, such as `COM1`, that is being used on the MPI prober.
   - *`BaudRate`* is the baud rate of the selected COM port. Set to `9600`.
   - *`Parity`* is the parity checking to be used. Set to `NONE`.
   - *`Handshake`* is the handshaking to be used. Set to `OFF`.

An example configuration of an R-S232 connection on COM1 with a baud rate of 9600, parity checking of none, and handshaking turned off is:

```
<RemoteServer Type="RS232" Config="COM1:9600:NONE:OFF" />
```

Set up communications on the 4200A-SCS

On the 4200A-SCS, KCon is used to add the MPI prober to the present system configuration and to edit the prober communication settings. Refer to Use KCon to add a prober (on page M-6) for detailed information.

For more information on adding equipment to the 4200A-SCS, refer to Keithley Configuration Utility (on page 7-1).

Load, align, and contact the wafer

Refer to the *MPI Sentio User Manual* for information on how to load, unload, set chuck heights, align, contact, and set the home position of the wafer.

Set up wafer geometry

Refer to the *MPI Sentio User Manual* for information on how to set up the wafer map.

Create a site definition and define a probe list

Refer to the *MPI Sentio User Manual* for information on how to add and edit subsites and sites. To create a site definition for a single subsite for each die, you need to use the MPI software to create a selection of dies to probe.

To create a site definition for multiple subsites for each die, you need to use the MPI software to create a selection of dies to probe and create a selection of subsites on each die to be probed.

Clarius probesites and probesubsites project example

The following is a step-by-step procedure to configure an MPI prober so the `probesites` or `probesubsites` Clarius projects execute successfully.

MPI Sentio setup

Using MPI Sentio on the prober, edit and open a wafer map file. Refer to the MPI Sentio documentation to:

- Load, align, and contact the wafer
- Set up wafer geometry
- Create a site definition
- Define a probe list

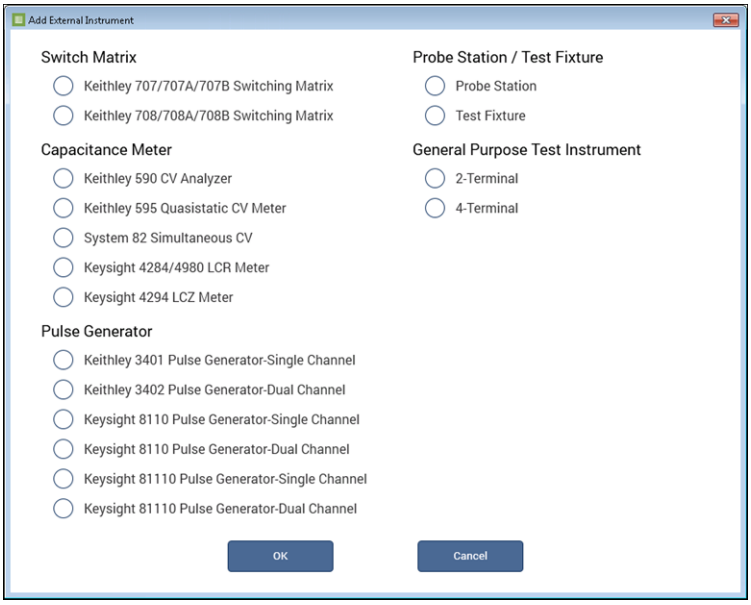
The wafer map file allows Clarius to send commands to instruct MPI Sentio to move the prober to the next site or the next subsite.

Use KCon to add a prober

On the 4200A-SCS, use KCon to add the prober to the configuration:

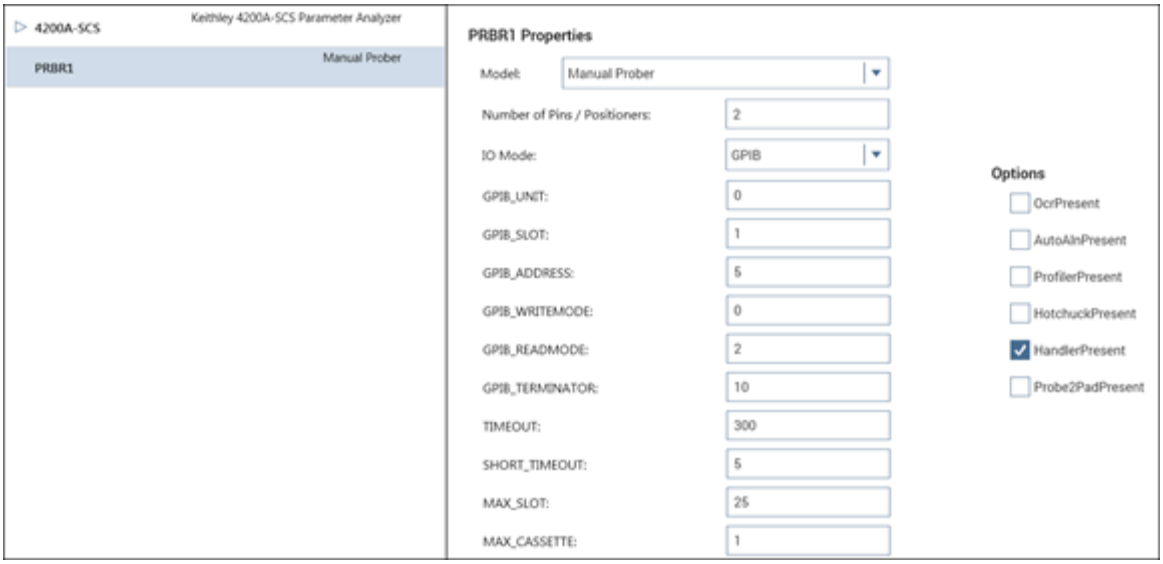
1. Open KCon.
2. At the bottom of the Configuration Navigator, select **Add External Instrument**. The Add External Instrument dialog box is displayed.

Figure 932: Add a prober in KCon



3. Select **Probe Station**.
4. Select **OK**. The KCon navigation displays PRBR1 in the Configuration Navigator and the properties.

Figure 933: Use KCon to select a prober



5. Select **MPI Prober** as the model.
6. If using a switch matrix, make sure the **Number of Pins / Positioners** is correct. The number of pins defined here determines the pins that are available to assign to a switch matrix card column.
7. Set the **IO Mode** parameter to the type of communication that is being used with the prober, either GPIB or RS-232 (Serial).
8. If using GPIB, be sure to set the **GPIB_ADDRESS** parameter to the address of the MPI prober.
9. If using RS-232, make sure the **BAUDRATE** parameter is set to the same speed as the MPI prober COM port, typically 9600.
10. Select **Save**.
11. Exit KCon.

Clarius

Use Clarius to load and run either the `probesites` or `probesubsites` project using the new MPI prober configuration.

On the 4200A-SCS:

1. Open Clarius.
2. Choose **Select**.
3. Select **Projects**.
4. Search for **probe**.
5. Drag the **probesites** project (if only one subsite is used) or **probesubsites** project (if more than one subsite is used) to the project tree.

Figure 934: probesites project tree

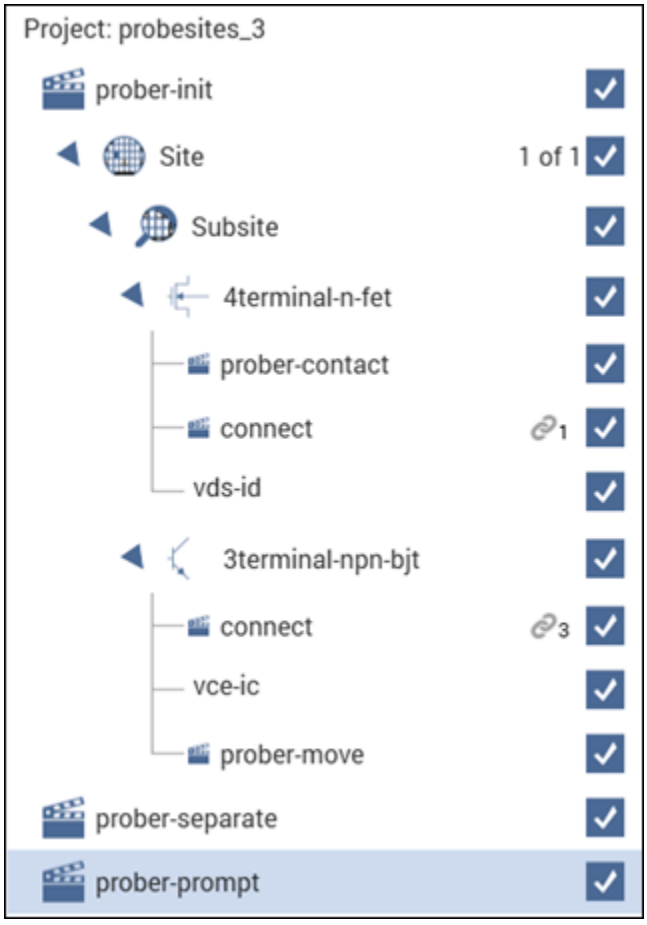
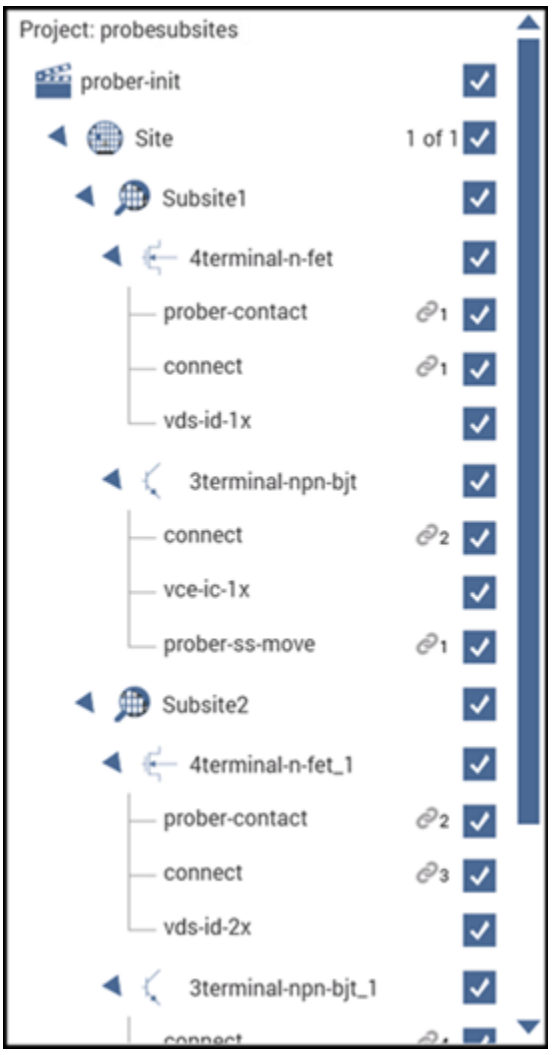


Figure 935: probesubsites project tree



6. Edit the project so that the number of sites or subsites matches the wafer map.
7. Delete or replace tests with relevant device tests for the wafer being tested. Refer to Configure sites (on page 6-193) for information on editing and adding sites in Clarius. Refer to Subsites (on page 6-186) for information on adding new subsites in Clarius.
8. Select project name in the project tree in Clarius.
9. Select **Run**.

Clarius initializes the MPI prober and runs through all sites and subsites in the project, sending the MPI prober commands as needed to either contact the chuck, step to the next site, or step to the next subsite. Refer to Run a complex test (on page 6-224) for information on running a test from the project level.

Commands and error symbols

The following table contains error and status symbols listed by command when using the MPI prober through the PRBGEN user library.

Available commands and responses

	PrChuck	PrInit	PrMovNxt	PrSSMovNxt
PR_OK	X	X	X	X
BAD_CHUCK	X			
UNINTEL_RESP	X	X	X	X
INVAL_PARAM		X	X	X
BAD_MODE			X	X
UNEXPE_ERROR	X	X	X	X
MOVE_FAIL			X	X

Information and error code return values and descriptions

Value	Constant	Explanation
1	PR_OK	Command executed properly
-1011	BAD_MODE	Operation invalid in mode
-1013	UNINTEL_RESP	Unintelligible response
-1014	MOVE_FAIL	Movement failure
-1015	UNEXPE_ERROR	Unexpected error number
-1017	BAD_CHUCK	Bad chuck position
-1027	INVAL_PARAM	Invalid parameter

Keithley Instruments
Corporate Headquarters • 28775 Aurora Road • Cleveland, Ohio 44139 • 440-248-0400 • Fax: 440-248-6168 • 1-800-935-5595 • www.tek.com/keithley

KEITHLEY
A Tektronix Company