

Model 4200A-SCS Clarius

User's Manual

4200A-914-01 Rev. B October 2021



4200A-914-01B

Model 4200A-SCS

Clarius

User's Manual

© 2021, Keithley Instruments

Cleveland, Ohio, U.S.A.

All rights reserved.

Any unauthorized reproduction, photocopy, or use of the information herein, in whole or in part, without the prior written approval of Keithley Instruments is strictly prohibited.

All Keithley Instruments product names are trademarks or registered trademarks of Keithley Instruments, LLC. Other brand names are trademarks or registered trademarks of their respective holders.

Actuate®

Copyright © 1993-2003 Actuate Corporation.

All Rights Reserved.

Microsoft, Visual C++, Excel, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Document number: 4200A-914-01 Rev. B October 2021

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

Introduction	1-1
Get started with Clarius	1-1
Clarius interface	1-2
Touchscreen basics	1-3
Choose the project phase	1-3
Run tests and set up your workspace	1-4
Organize items in the project tree	1-5
Select items from the libraries	1-6
Configure the project	1-7
Analyze data	1-8
Messages	1-9
Help pane	1-9
Additional Clarius+ applications	1-10
Embedded computer policy	1-11
Projects and tests	2-1
Introduction	2-1
Set up a simple project	2-1
Select project components	2-2
Add a device and test to the project	2-3
Rearrange items in the project tree	2-4
Delete objects in the project tree	2-4
Configure a simple test	2-5
Set the key parameters	2-7
Run a simple test	2-8
Working with the Projects dialog box	2-9
Open a project	2-10
Edit project information	2-11
Create a new project from the Projects dialog box	2-12
Export a project	2-12
Import a project	2-12
Copy or cut a project	2-14
Show Directories	2-14
Delete a project	2-15
View Projects	2-15
Migrate projects from 4200-SCS systems	2-16
Manage projects for multiple users	2-17
Set up a complex project	2-17
Customize tests	2-18
Link tests or actions	2-61
Add actions	2-62
Example: Creating a project	2-63
Equipment required	2-63
Device connections	2-64
Set up the measurements in Clarius	2-65
Configure a complex test	2-72
Test and terminal settings	2-73
Step or sweep multiple device terminals in the same test	2-74

Configure actions	2-77
Run a complex test	2-78
Run devices and tests	2-78
Monitor a test	2-82
Enable the Monitor option	2-83
Running a test using Monitor	2-83
Demo Project overview	2-84
4-terminal n-MOSFET tests	2-86
3-terminal NPN BJT tests	2-86
Resistor tests	2-87
Diode tests	2-87
Capacitor tests	2-87
Analyze data	3-1
Introduction	3-1
Spreadsheet	3-3
Run sheet	3-4
Formulas List of the Run worksheet	3-6
Terminal Settings pane (Analyze)	3-6
Module Settings pane (Analyze)	3-7
Measurement status	3-7
Analyze test data at the project level	3-10
Select data for the project-level Analyze pane	3-11
Remove data from the project-level Analyze pane	3-12
Run History	3-12
Changing the name of a test run	3-14
Changing the display of Run Histories	3-15
Searching for a Run History	3-15
Copy the settings of a Run History to the Configure screen	3-15
Setting the number of run histories	3-16
Subsite cycling Analyze sheets	3-16
Stress/measure mode Analyze sheet	3-17
Stress/measure mode Analyze graph	3-18
Subsite Settings sheet	3-20
Calc sheet	3-22
Settings worksheet	3-24
Save test results and graphs	3-25
Graph	3-26
Open a graph	3-27
Define data to be graphed	3-27
View plot coordinates and data series properties	3-30
Change the graph settings	3-30
Cycle mode graphs	3-59
Calc worksheet function definitions	3-60
ABS Calc worksheet function	3-60
ACOS Calc worksheet function	3-61
ACOSH Calc worksheet function	3-61
ASIN Calc worksheet function	3-62
ASINH Calc worksheet function	3-62
ATAN Calc worksheet function	3-63
ATAN2 Calc worksheet function	3-64
ATANH Calc worksheet function	3-64
AVERAGE Calc worksheet function	3-65

COS Calc worksheet function	3-65
COSH Calc worksheet function	3-66
DAY Calc worksheet function	3-66
EXP Calc worksheet function	3-67
FIXED Calc worksheet function	3-67
HOUR Calc worksheet function	3-68
IF Calc worksheet function	3-68
LN Calc worksheet function	3-69
LOG Calc worksheet function	3-69
LOG10 Calc worksheet function	3-70
LOOKUP Calc worksheet function	3-71
MATCH Calc worksheet function	3-72
MAX Calc worksheet function	3-73
MIN Calc worksheet function	3-74
MINUTE Calc worksheet function	3-75
MONTH Calc worksheet function	3-76
NOW Calc worksheet function	3-77
PI Calc worksheet function	3-77
PRODUCT Calc worksheet function	3-78
ROUND Calc worksheet function	3-78
SECOND Calc worksheet function	3-79
SIGN Calc worksheet function	3-79
SIN Calc worksheet function	3-80
SINH Calc worksheet function	3-80
SQRT Calc worksheet function	3-81
STDEVP Calc worksheet function	3-81
SUM Calc worksheet function	3-82
SUMSQ Calc worksheet function	3-82
TAN Calc worksheet function	3-83
TANH Calc worksheet function	3-83
VARP Calc worksheet function	3-84
YEAR Calc worksheet function	3-84

Customizing Clarius 4-1

Customize Clarius	4-1
Add objects to the library	4-1
Add a test to the library	4-2
Add a device to the Device Library	4-3
Add an action to the library	4-3
Add a project to the library	4-3
Edit a library object you added	4-4
Edit an object in the library	4-5
Project tree display options	4-7
Messages display option	4-7
My Settings	4-7
Specify environment settings	4-7
Specify run settings	4-10
Graph defaults	4-12
Custom GPIB Abort Options	4-13
Logging	4-13
Tools	4-14
Instrument Tools	4-14
Data Export tool	4-35

Formulator	5-1
Introduction	5-1
Open the Formulator	5-2
Configure Formulator calculations	5-3
Formulator dialog box	5-3
Formula area.....	5-3
Data Series	5-3
Number pad	5-4
Functions.....	5-4
Constants.....	5-4
Apply Set to.....	5-5
Using the Formulator options.....	5-5
Real-time functions, operators, and formulas	5-6
Post-test-only functions and formulas.....	5-7
Editing Formulator formulas and constants	5-8
Deleting Formulator formulas and constants	5-8
Identify data analysis requirements	5-8
Determining the type of calculation: an example.....	5-9
Determining range data for a calculation: an example	5-9
Creating an analysis formula.....	5-11
Adding an analysis formula to the test	5-11
Executing an analysis formula.....	5-12
Viewing analysis results in the Analyze sheet.....	5-12
Viewing analysis results in the Analyze graph.....	5-13
Formulator function reference.....	5-13
ABS Formulator function	5-14
SQRT Formulator function	5-14
EXP Formulator function	5-15
LOG Formulator function.....	5-15
LN Formulator function.....	5-16
DELTA Formulator function.....	5-16
Statistics	5-17
AVG Formulator function.....	5-17
MAVG Formulator function.....	5-17
MAX Formulator function	5-18
MEDIAN Formulator function	5-18
MIN Formulator function.....	5-18
STDEV Formulator function	5-19
Trigonometry	5-19
ACOS Formulator function	5-19
ASIN Formulator function.....	5-20
ATAN Formulator function.....	5-20
COS Formulator function	5-21
DEG Formulator function	5-21
RAD Formulator function.....	5-22
SIN Formulator function	5-22
TAN Formulator function	5-23
Array.....	5-23
AT Formulator function.....	5-23
DIFF Formulator function	5-24

FINDD Formulator function	5-25
FINDLIN Formulator function	5-26
FINDU Formulator function	5-27
FIRSTPOS Formulator function	5-27
INTEG Formulator function	5-28
INDEX Formulator function	5-29
LASTPOS Formulator function.....	5-30
MINPOS Formulator function	5-30
MAXPOS Formulator function.....	5-30
SUBARRAY Formulator function.....	5-31
SUMMV Formulator function.....	5-31
Line Fits.....	5-32
EXPFIT Formulator function.....	5-33
EXPFITA Formulator function	5-34
EXPFITB Formulator function	5-35
LINFIT Formulator function	5-36
LINFITSLP Formulator function.....	5-37
LINFITXINT Formulator function	5-38
LINFITYINT Formulator function	5-39
LOGFIT Formulator function	5-40
LOGFITA Formulator function.....	5-41
LOGFITB Formulator function.....	5-42
TANFIT Formulator function.....	5-43
TANFITSLP Formulator function.....	5-44
TANFITXINT Formulator function.....	5-45
TANFITYINT Formulator function.....	5-46
POLY2FIT Formulator function	5-47
POLY2COEFF Formulator function.....	5-48
POLYNFIT Formulator function.....	5-49
REGFIT Formulator function	5-50
REGFITSLP Formulator function	5-51
REGFITXINT Formulator function	5-52
REGFITYINT Formulator function	5-53
FFT.....	5-53
FFT_R Formulator function	5-54
FFT_I Formulator function.....	5-55
FFT_FREQ Formulator function.....	5-56
FFT_FREQ_P Formulator function	5-57
IFFT_R Formulator function	5-58
IFFT_I Formulator function.....	5-59
SMOOTH Formulator function.....	5-60
Misc.....	5-60
COND Formulator function.....	5-61

Setting up site and subsite operation..... 6-1

Introduction	6-1
Sites	6-1
Subsites	6-2
Configure sites	6-3
Configure subsite cycling	6-5
Connect devices for stress/measure cycling	6-6
Connections for matrix card	6-7
Connections for pulse card to device under test	6-8
Connections for system hardware.....	6-9
Set up the Subsite Operation	6-10

Export output values to Analyze sheet	6-32
Run an individual subsite	6-32
Run a single site.....	6-33
Cycle a subsite	6-34
Multi-site execution	6-35
User library descriptions	7-1
Introduction	7-2
AVMControl user library	7-2
BeepLib user library	7-2
chargepumping user library	7-3
cvivulib user library.....	7-3
cvucompulib user library	7-4
cvuulib user library	7-4
DLCP user library.....	7-5
dmm-6500-7510-temp-ulib user library	7-5
flashulib user library	7-6
GateCharge user library.....	7-6
generic_gpib_ulib user library	7-7
generic_visa_ulib user library	7-8
hivcvulib user library.....	7-9
Hotchuck_Temptronics3010B user library	7-9
Hotchuck_Triotek user library	7-9
HP4284ulib user library	7-10
HP4294ulib user library	7-10
HP8110ulib user library.....	7-11
ki340xulib user library	7-11
KI42xxulib user library.....	7-11
KI590ulib user library	7-12
KI595ulib user library	7-12
ki622x_2182ulib user library	7-13
ki82ulib user library	7-14
LS336ulib user library	7-15
Matrixulib user library	7-15
MultiSegmentSweep_ulib user library.....	7-15
nvm user library.....	7-16
OVPControl user library	7-17

parlib user library.....	7-18
pmuCompulib	7-18
pmuulib user library.....	7-19
PMU_examples_ulib user library	7-19
PMU_freq_time_ulib user library.....	7-20
PMU_PCRAM_ulib.....	7-21
PRBGEN user library	7-21
QSCVulib user library.....	7-22
RPM_ILimit_Control user library	7-22
utilities_ulib.....	7-22
van der Pauw user library	7-23
VLowFreqCV user library	7-24
wrlib user library	7-25
Winulib user library.....	7-25
AbortRetryIgnoreDialog user module	7-26
InputOkCancelDialog user module.....	7-28
OkCancelDialog user module.....	7-30
OkDialog user module.....	7-32
RetryCancelDialog user module.....	7-34
YesNoCancelDialog user module	7-36
YesNoDialog user module.....	7-38
Wafer-level reliability testing.....	8-1
JEDEC standards.....	8-1
Introduction	8-2
HCI and WLR projects	8-3
Hot Carrier Injection projects.....	8-3
Negative Bias Temperature Instability project.....	8-4
Electromigration project	8-5
Charge-to-Breakdown Test of Dielectrics project.....	8-6
HCI degradation: Background information.....	8-7
Configuration sequence for subsite cycling	8-7
V-ramp and J-ramp tests.....	8-8
V-ramp test: qbd_rmpv User Module	8-9
J-ramp test: qbd_rmpj User Module	8-14

In this section:

Get started with Clarius	1-1
Clarius interface	1-2
Additional Clarius+ applications	1-10
Embedded computer policy	1-11

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.

The Clarius Software user interface provides touch-and-swipe or point-and-click control for advanced test definition, parameter analysis, graphing, and automation capabilities for modern semiconductor, materials, and process characterization.

Key features:

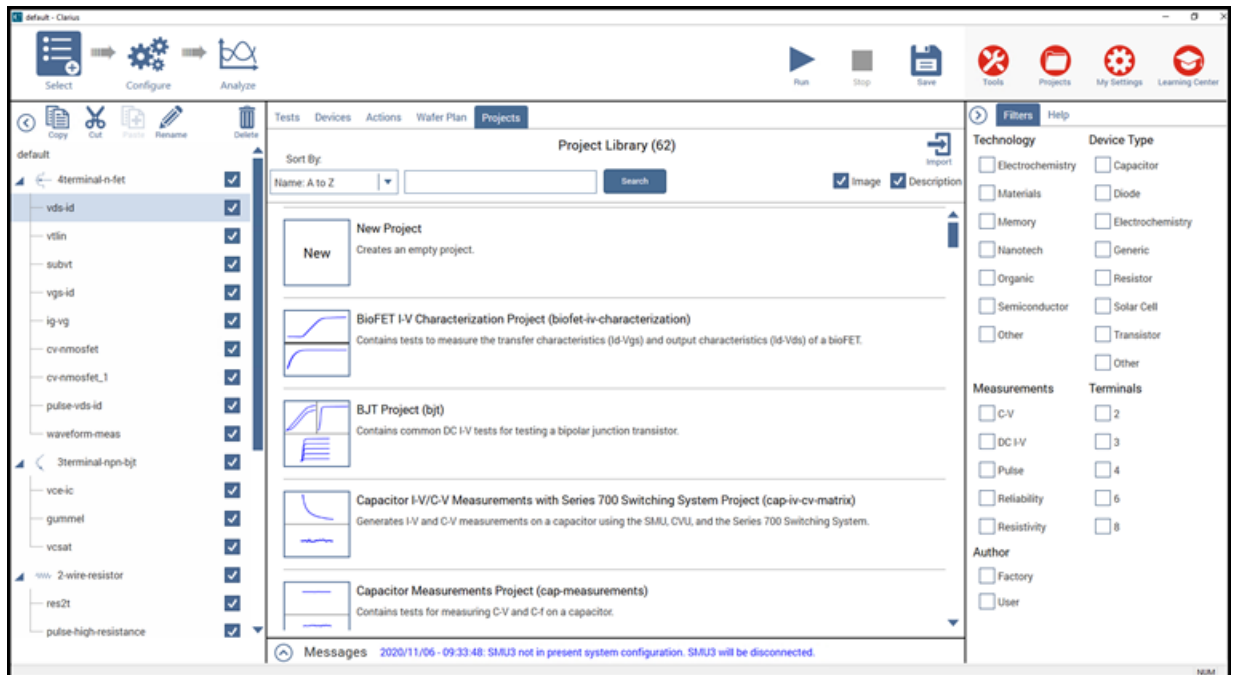
- Ready-to-use, modifiable application tests, projects, and devices that reduce test development time
- Built-in measurement videos from world-wide Application Engineers in four languages
- Pin-to-pad contact check ensures reliable measurements
- Multiple measurement functions
- Data display, analysis, 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 1: 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.

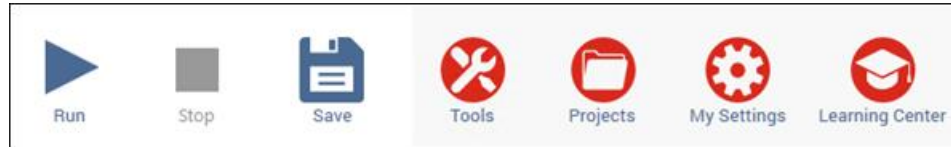
Analyze displays the results of the test in a spreadsheet and graph. You can also access analysis tools to explore and export your data.

Figure 2: Select, Configure, and Analyze



Run tests and set up your workspace

Figure 3: 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 and data export options. 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. The Data Export options allow you to export Run History data files to Microsoft Excel.

Use **Projects** to manage your projects. It includes options to create, import, export, copy, cut, paste, edit, and delete projects. Projects are automatically stored in 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, GPIB abort settings, and error and warning logging. It also includes the About Clarius option, which lists the Clarius version and copyright information.

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.

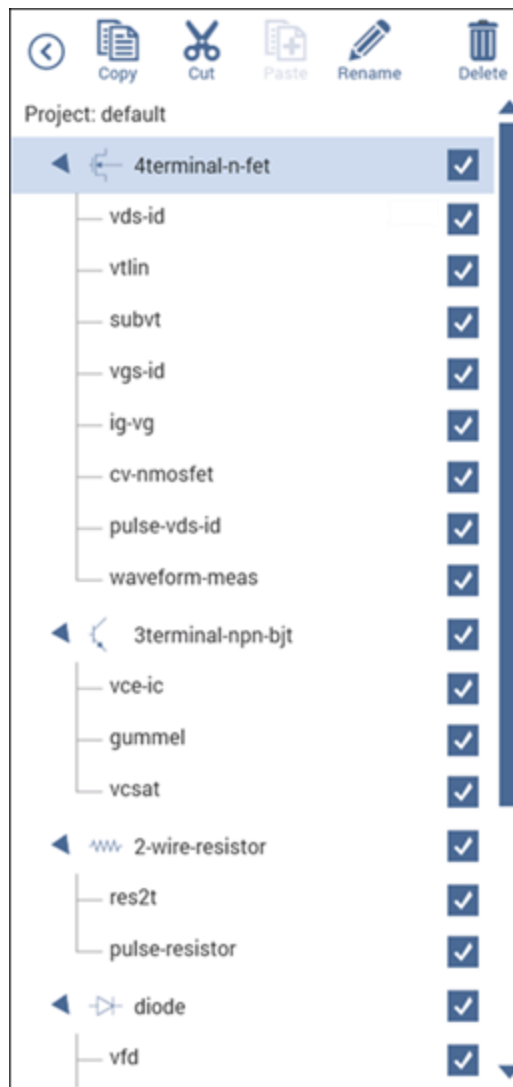
Organize items in the project tree

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 4: 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 project 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 4-3).

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.

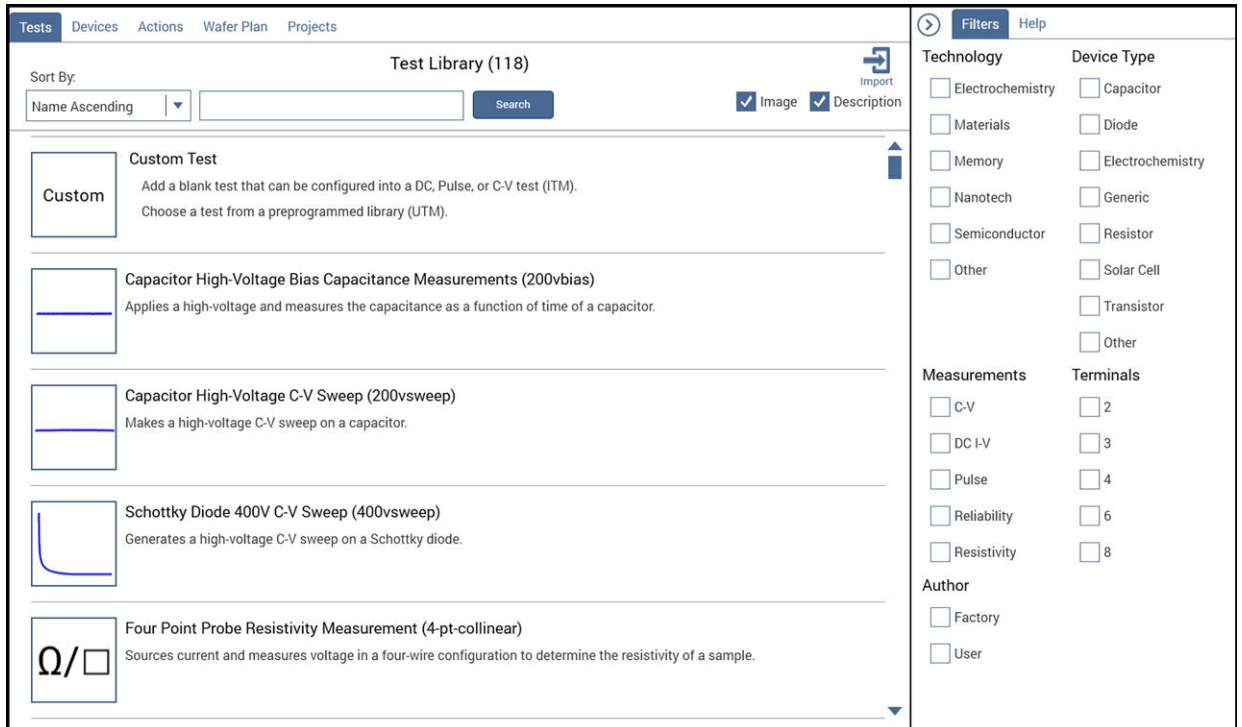
For most of the libraries, the right pane displays filters that you can use to reduce the list of library items to the items you need. You can also use the Search option at the top of the library to type a search term to reduce the number of items.

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 allows more items to be visible on the screen.

Import allows you to import items into the 4200A-SCS.

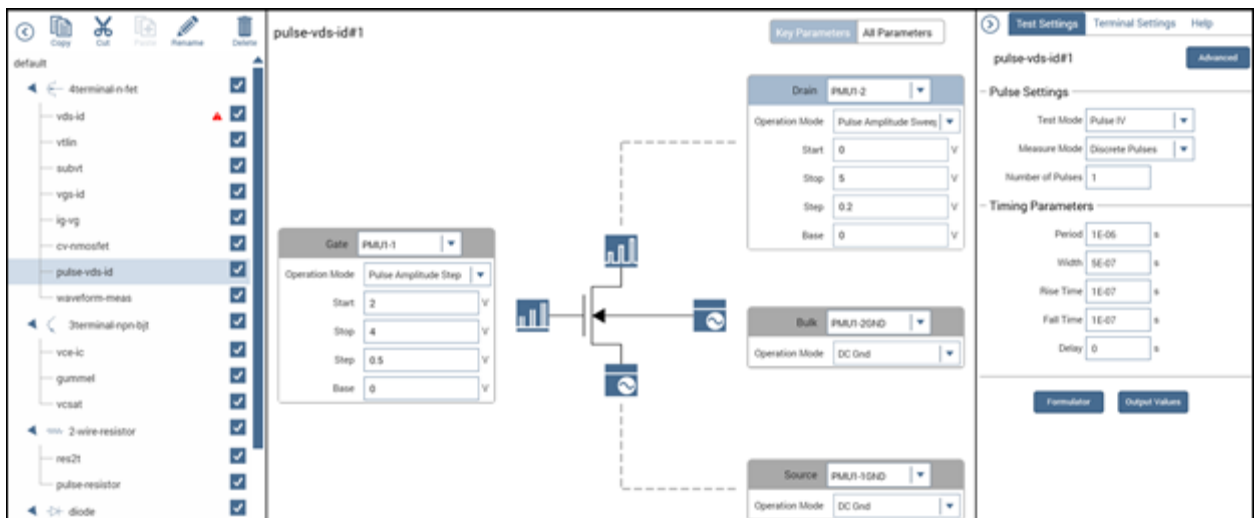
Figure 5: Test library



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 6: 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 7: Analyze showing 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) 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) manages the configuration and interconnections between the test system components that are controlled by Clarius.

To control the 4200A-SCS remotely using an external GPIB controller, you can use another Clarius+ software tool, the Keithley External Control Interface (KXCI). You cannot run KXCI and Clarius simultaneously.

To configure and control the optional pulse cards, you can use the Keithley Pulse tool (KPulse). A pulse card is a dual-channel pulse card that is integrated inside the 4200A-SCS mainframe. Although KPulse can be launched at the same time as Clarius, KPulse and Clarius cannot communicate with hardware simultaneously.

For information about these applications, refer to:

- *Model 4200A-SCS KULT and KULT Extension Programming*
- “Keithley Configuration Utility (KCon)” in the *Model 4200A-SCS Setup and Maintenance User's Manual*
- *Model 4200A-SCS KXCI Remote Control Programming*
- “KPulse (for Keithley Pulse Cards)” in the *Model 4200A-SCS Pulse Card (PGU and PMU) User's Manual*

Embedded computer policy

CAUTION

If you install software that is not part of the standard application software for the 4200A-SCS, the nonstandard 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” in *Model 4200A-SCS Setup and Maintenance*.

Projects and tests

In this section:

Introduction	2-1
Set up a simple project.....	2-1
Configure a simple test	2-5
Run a simple test	2-8
Working with the Projects dialog box	2-9
Set up a complex project.....	2-17
Example: Creating a project.....	2-63
Configure a complex test	2-72
Run a complex test	2-78
Monitor a test	2-82
Demo Project overview	2-84

Introduction

This chapter describes how to set up projects and tests in Clarius.

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 and 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.

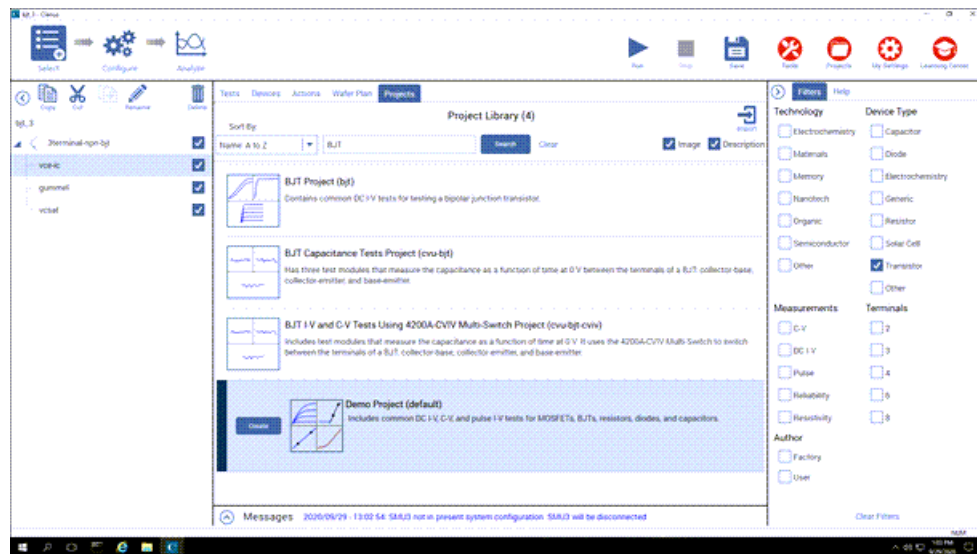
To clear filters, select **Clear Filters** at the bottom of the Filters pane. To clear the search, select **Clear** next to the Search button.

The following example shows you how to select tests for bipolar junction transistors (BJTs).

To set up a test of BJTs:

1. Select **Save** to save your existing project.
2. Choose **Select**.
3. Select the **Projects** tab.
4. In the Filters pane, select **Transistor**.
5. In the Search box, type **BJT** and select **Search**. The Project 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 8: Filter and search for the bjt project



Add a device and test to the project

You can add additional items to a project. When you add a project from the library to the project tree, it is copied from the project in the library. Any changes you make do not affect the original project. The new project in the project tree is automatically stored in Projects.

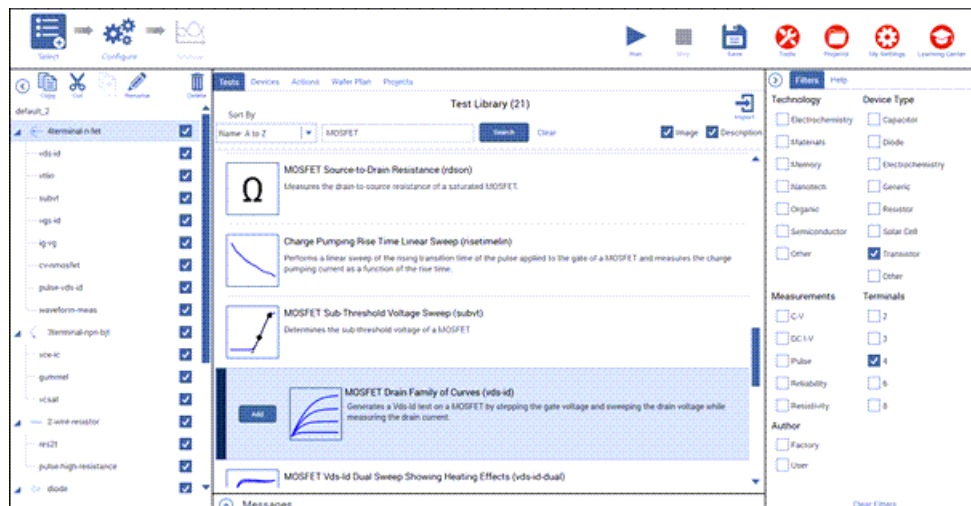
This example shows you how to add a predefined test to the project. Predefined tests are configured with commonly used parameter settings and a set of typical data. Once they are in a project, you can change the parameters as needed. They can be an efficient way for you to add a test to your project.

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. In the center pane, select **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 are 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 9: Add a MOSFET test and device to the project



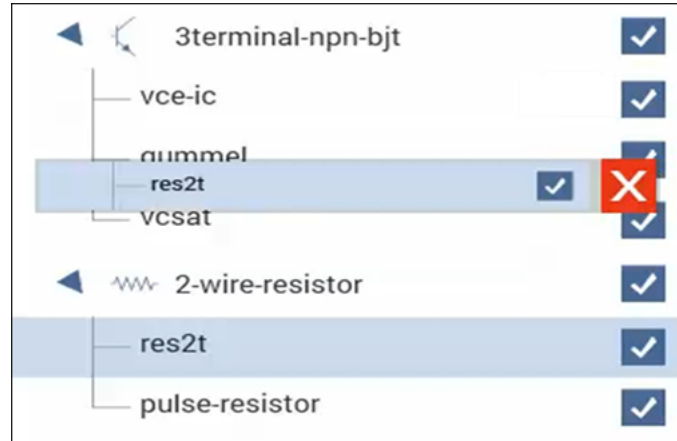
NOTE

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 10: 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.

NOTE

The following discusses the Test Settings pane for interactive test modules (ITMs). For tests that are based on user test 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 2-20) for information on settings available for UTMs.

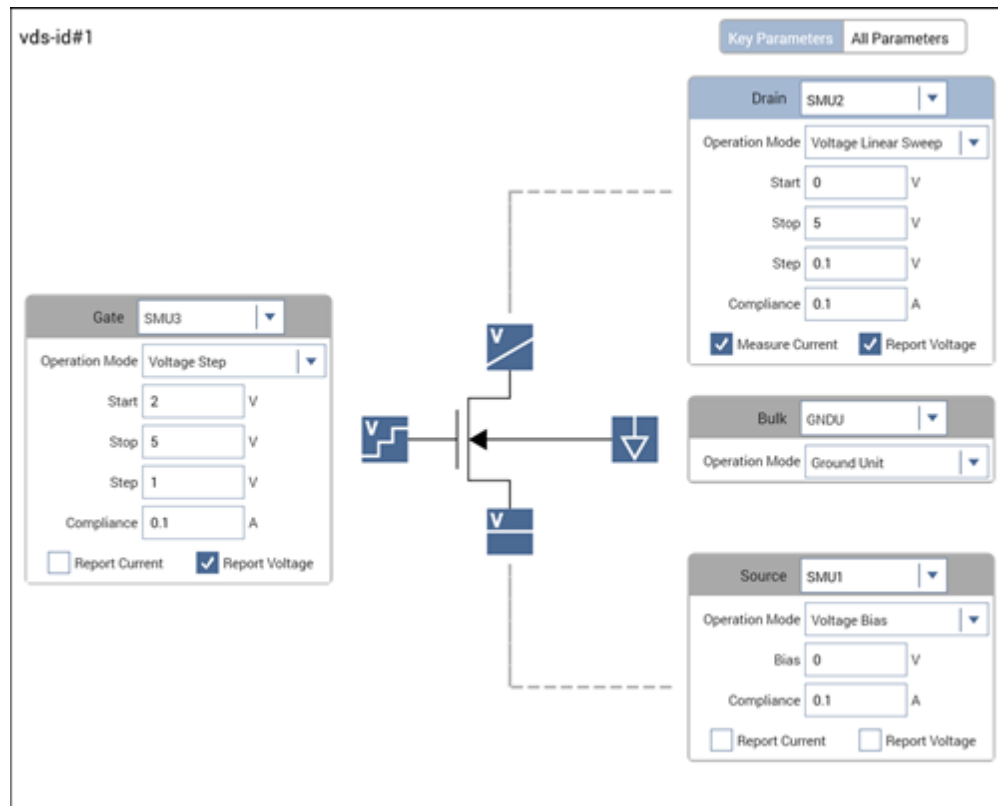
CAUTION

The connections selected in the Clarius software 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 11: 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.

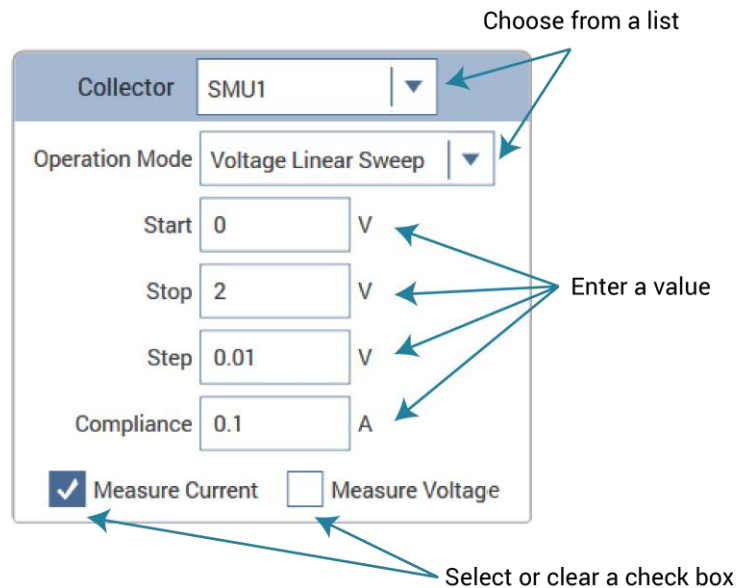
The parameters that are available depend on the instrument that is selected. For descriptions of parameters, refer to:

- “SMU - all parameters” in Model 4200A-SCS Source-Measure Unit (SMU) User's Manual
- “CVU - all parameters” in Model 4200A-SCS Capacitance-Voltage Unit (CVU) User's Manual
- “PMU - all parameters” in Model 4200A-SCS Pulse Card (PGU and PMU) User's Manual

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 12: Clarius selection options



3. Select **Save**.

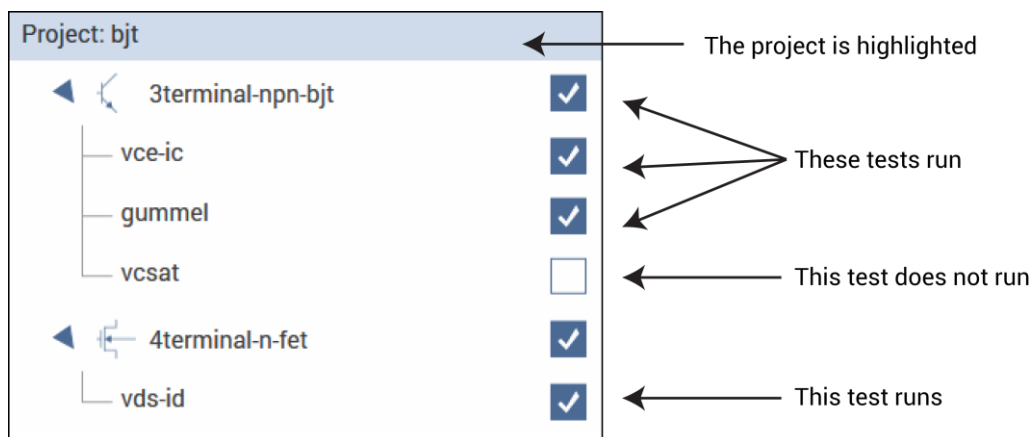
Run a simple test

When you select Run, selected tests and actions at a lower level than the highlighted item in the project tree are executed, 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:

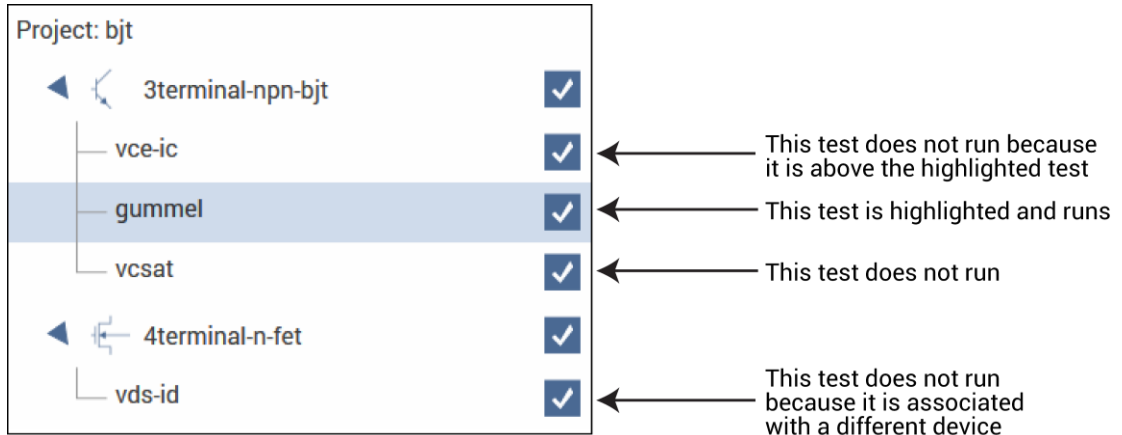
- The `vce-ic` test runs.
- The `gummel` test runs.
- The `vcsat` test is skipped.
- The `vds-id` test runs.

Figure 13: 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 14: 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 the Projects dialog box

The Projects option in the Clarius ribbon allows you to work with the projects you have created. Projects from all user accounts and any projects added to the project directory, which is defined in My Settings, are available.

You can use the Projects dialog box to create new projects, import and export projects, and to copy, cut, paste, edit, delete, search for, and open projects.

NOTE

To change the project directory, see [My Projects Directory](#) (on page 4-9).

Open a project

Your projects are automatically added to the Projects dialog box when they are added to the project tree. This procedure describes how to retrieve a project.

To open a project:

1. Choose **Projects** from the ribbon.
2. Type the project name in the Search box.
3. Select **Search**.
4. Select the project.
5. Select **Open Project**. If the project in the project tree has unsaved changes, you are prompted to save the changes.

Edit project information

You can edit information that is displayed in the Project Library for a specific project. You can also edit the filters and keywords that are used.

To edit project information:

1. In Clarius, select **Projects** from the ribbon.
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 following table 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 Project Information Editor.
Help	Cannot be changed. 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.
Library Image	The image that is displayed in the library. Click the image to select a different image. Images should be 400x400 pixels in <code>png</code> 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 <code>bmp</code> image to <code>png</code> 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 the Projects dialog box

You can create a new project from the Projects dialog box. This is the same as creating a new empty project using the Project Library "New Project" option.

To create a new project:

1. Open **Projects** from the ribbon.
2. Select **New**. A confirmation message is displayed.
3. Select **Yes**. The new project opens in Clarius and the existing project is closed. The Projects dialog box closes.
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.

The exported file has the extension `.kzp`.

To export a project:

1. In Clarius, select **Projects** from the ribbon.
2. Select the project to be exported.
3. Select **Export**. The Export Project dialog box opens.
4. Select the location for the exported file. You can right-click 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 2-12) 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.

NOTE

If you are importing a project from a 4200-SCS, see [Migrate projects from 4200-SCS systems](#) (on page 2-16).

To import a project:

1. In Clarius, select **Projects**.
2. Select **Import**. The Import Project dialog box opens.
3. Select the exported project (*kzp*) or project directory (*kpr*).
4. Select **Import**. The Project Information Editor opens.
5. In the Basic tab, complete the information as needed. Refer to the following table 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 Project** 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 Project 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.
Library Image	The image that is displayed in the library. Click the image to select a different image. Images should be 400x400 pixels in <i>png</i> 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 <i>bmp</i> image to <i>png</i> format. You can use a tool such as Microsoft® Paint to convert the image. To leave the image area blank, select Clear .

Copy or cut a project

You can copy or cut a project. The new project does not maintain any links to the old project. Changes to the new project do not affect the old project and changes to the old project do not affect the new project.

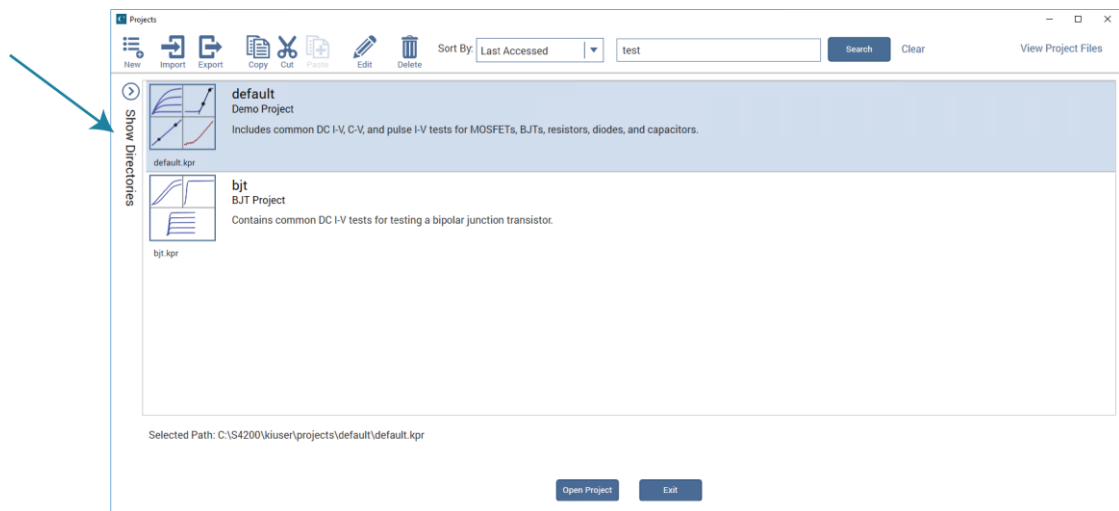
To copy a project:

1. Select **Projects** from the ribbon.
2. Select the project to be copied.
3. Select **Copy** or **Cut**.
4. Select **Paste**.
5. If you are copying a file, to:
 - Copy the project and load it as the active project: Select **Copy and Load**.
 - Copy the project only in Projects: Select **Copy Only**.

Show Directories

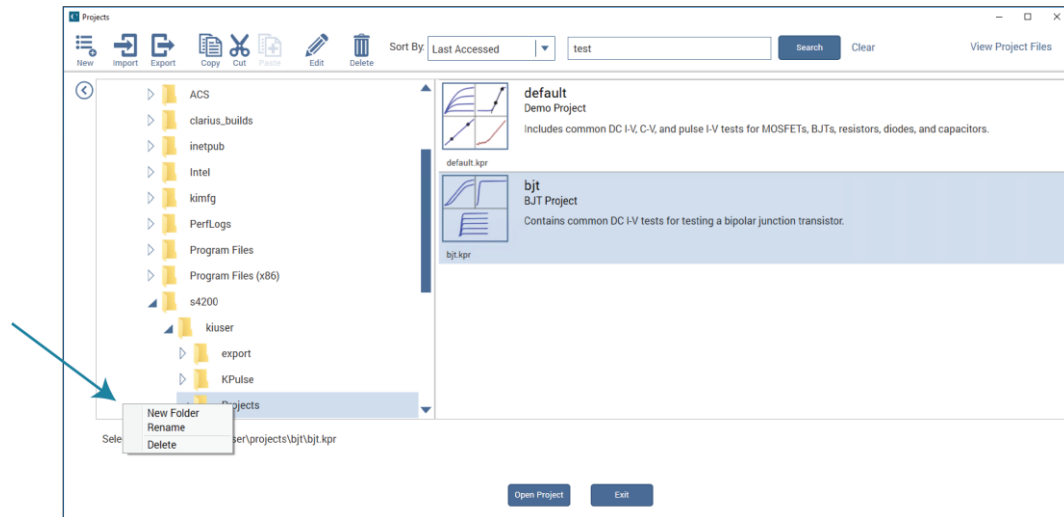
To display projects in a Microsoft File Explorer view, you can select **Show Directories** in the Projects dialog box.

Figure 15: Show Directories pane



When Show Directories is selected, you can right-click in the directory to add new folders, rename folders, or delete folders.

Figure 16: Show Directories open



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 project directory that is set in My Settings are also deleted. If the deleted project is open in the project tree, the project tree is cleared.

To delete a project:

1. Select **Projects**.
2. Select the project.
3. Select **Delete**. A confirmation message is displayed.
4. Select **Yes**.

View Projects

You can change the view of the 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.

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 Model 4200A-SCS Parameter Analyzer KULT and KULT Extension Programming, “Copy user libraries and files” 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.
- Data that was appended in the 4200-SCS is stored in Run History in the 4200A-SCS.

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 the **Projects** tab.
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.

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 Project dialog box and 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 the Projects dialog box, refer to [Edit project information](#) (on page 2-11). For information on adding keywords to projects that are added to the library, refer to [Edit information for a library object](#) (on page 4-4).

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 the Projects dialog box 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 Projects, export the project. Refer to [Export a project](#) (on page 2-12).
3. Delete the project from Projects. Refer to [Delete a project](#) (on page 2-15).
4. If you need to use the project again, import the project into the 4200A-SCS. Refer to [Import a project](#) (on page 2-12).

Set up a complex project

[Set up a simple project](#) (on page 2-1) 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.
- Using switch matrices to cycle electrical connections from the 4200A-SCS between the devices of a subsite. Refer to “Using Switch Matrices” in *Model 4200A-SCS Prober and External Instrument Control* for detail.

This section describes how to add custom tests and actions to Clarius and to the project tree.

NOTE

For information on adding, duplicating, and importing projects, refer to [Working with the Projects dialog box](#) (on page 2-9).

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 7-1) 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 *Model 4200A-SCS KULT and KULT Extension Programming* 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 2-3).

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. Configure the options as needed.
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.

The options that are available depend on the instrument that is selected. For descriptions of parameters, refer to:

- “SMU Test Settings” in the *Model 4200A-SCS Source-Measure Unit (SMU) User's Manual*
- “CVU Test Settings” in the *Model 4200A-SCS Capacitance-Voltage Unit (CVU) User's Manual*
- “PMU Test Settings” in the *Model 4200A-SCS Pulse Card (PGU and PMU) User's Manual*

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 *Model 4200A-SCS KULT and KULT Extension Programming*.

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 5-1) 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 2-21) 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:

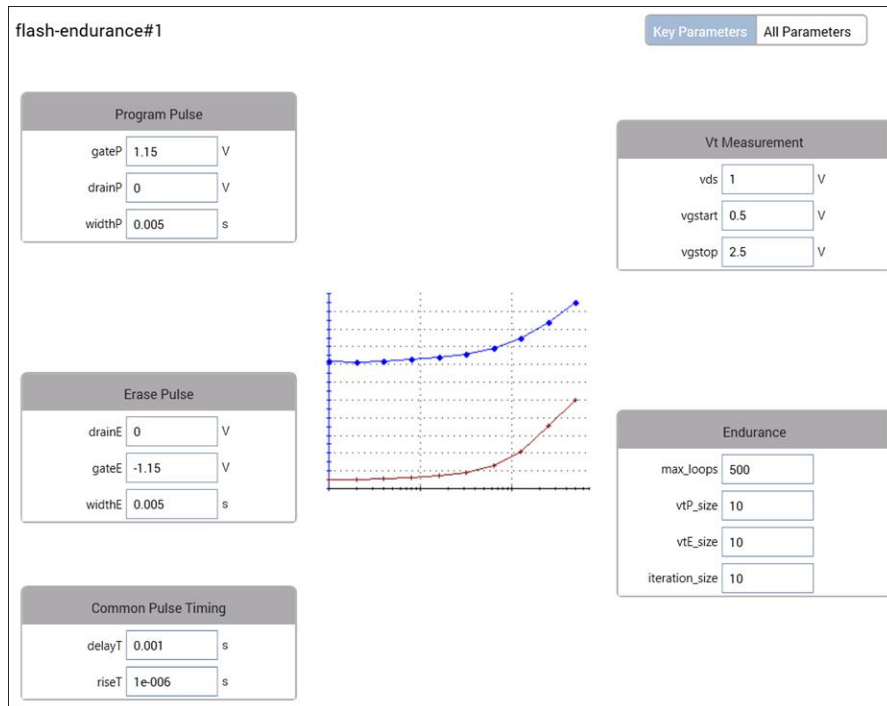
- Group parameters logically
- Add an image of the test to illustrate the overall test capability
- Add tooltips for each parameter

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 17: 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.

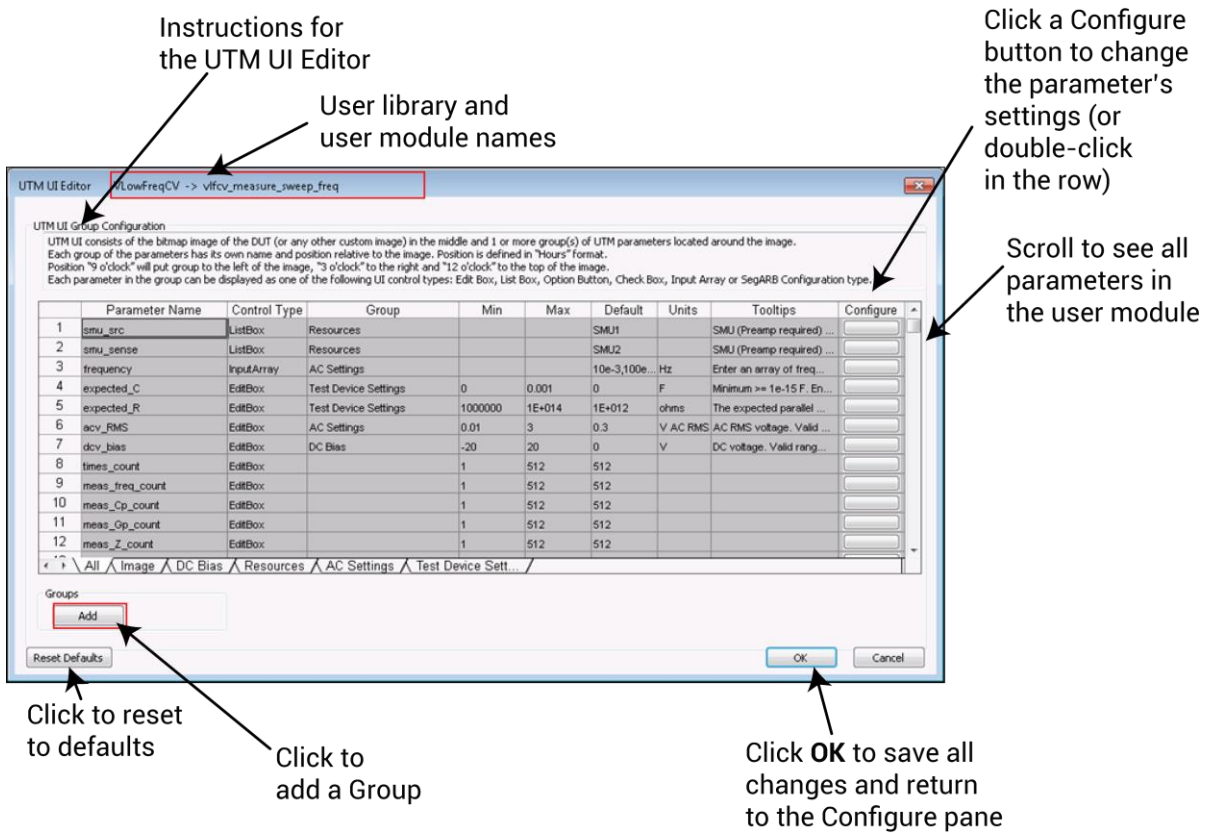
Open the UTM UI editor

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 18: 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 2-21), parameters are organized into the groups Program Pulse, Erase Pulse, Common Pulse Timing, Vt Measurement, and Endurance.

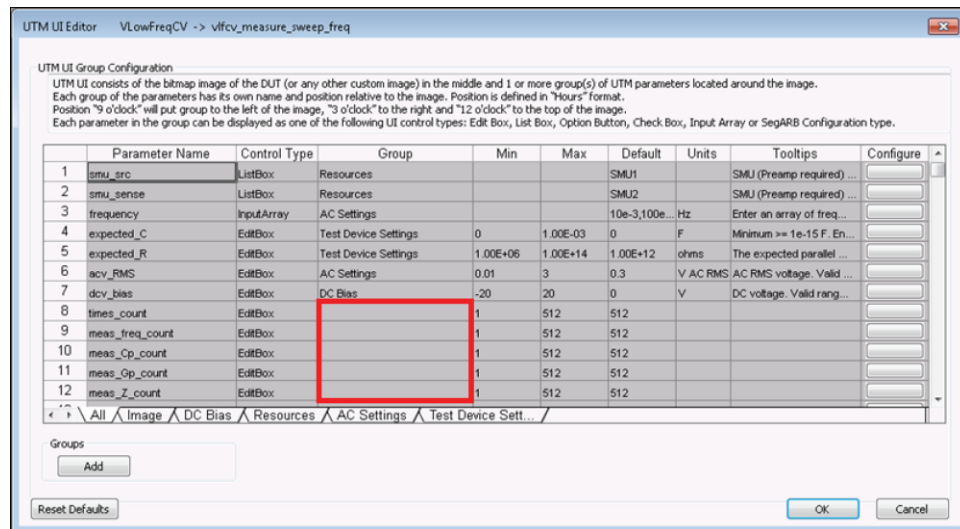
The maximum number of parameters in a group is ten.

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 19: Parameters not assigned to a group

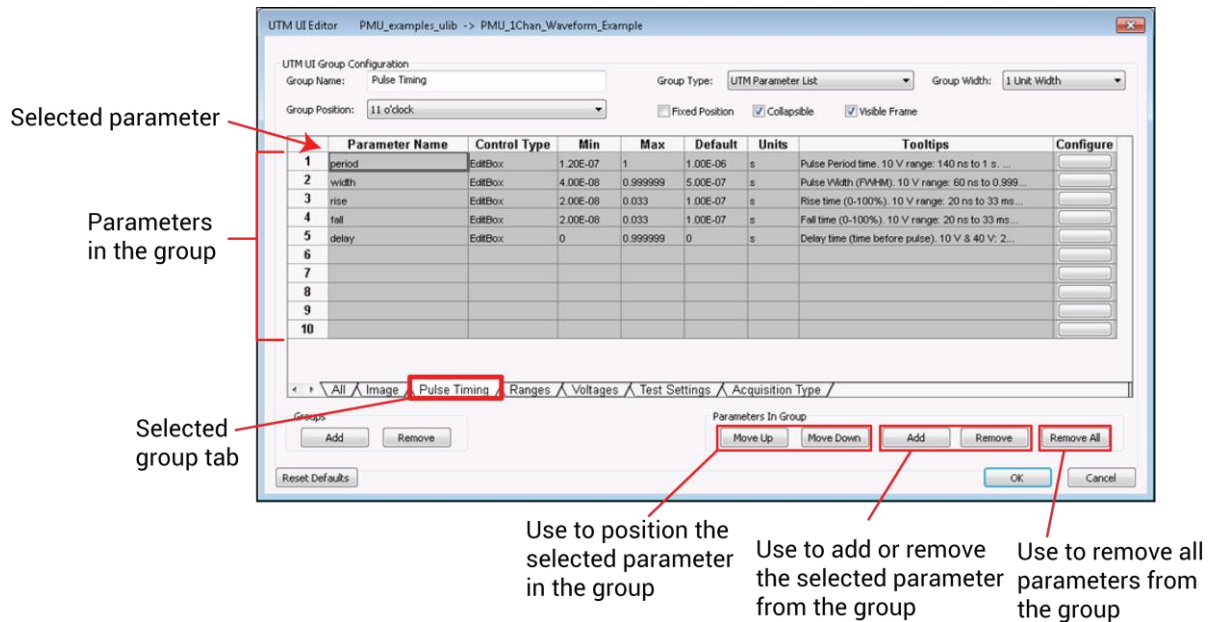


Add a group

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 2-28) for detail.
7. Click **OK**.

Figure 20: Example group view for pulse timing

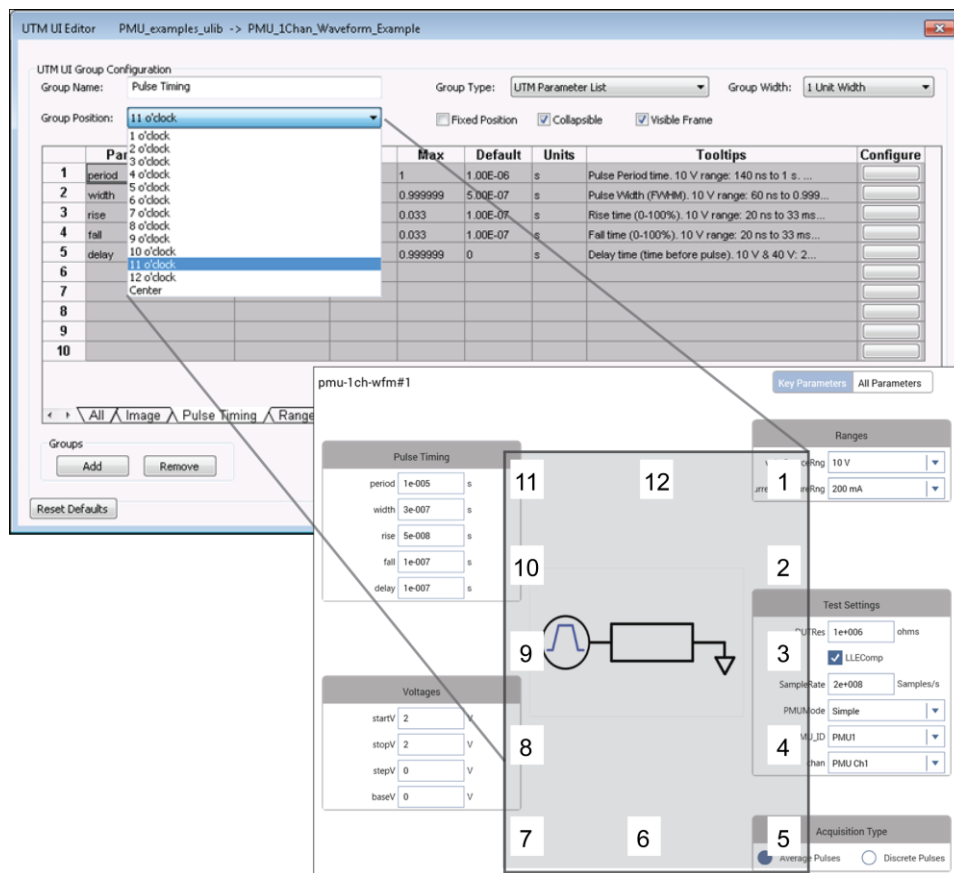


Modify a group

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 2-25)):
 - **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 21: 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)
- 120 × 100 pixels to 480 × 400 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 2-31) 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. Do not use nontext 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 22: GUI Configuration for the voltsSourceRng parameter (ListBox)

UTM UI Parameter Configuration

Parameter Name: voltsSourceRng

Control Type: ListBox

Displayed Group: Ranges

Minimum Value: 5

Maximum Value: 40

Default Value: 10

Displayed Units:

Displayed Tooltips: PMU Voltage Source Range: +/- 10V or +/-40V.

List Items Configuration

	Displayed Name	UTM Value	Use Case Condition
1	40 V	40	
2	10 V	10	
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			

Predefined List Items: Custom List

UI presentation of the parameter via ListBox control will include:

- Parameter name
- List box with items correspondent to valid parameter values
- Parameter units
- Tooltips

List Items Configuration table allows to configure items in the list. Each item must have Displayed Name (as it will be shown in list) and correspondent to it UTM value. List item also may have Use Case Condition that defines when such item will be listed. There are several predefined conditions:

PresentInSystem – Can be used with list of SMUs/CVU/PMUs. Configured list may contain for example 9 SMUs but UI list will show only SMUs installed in particular system.

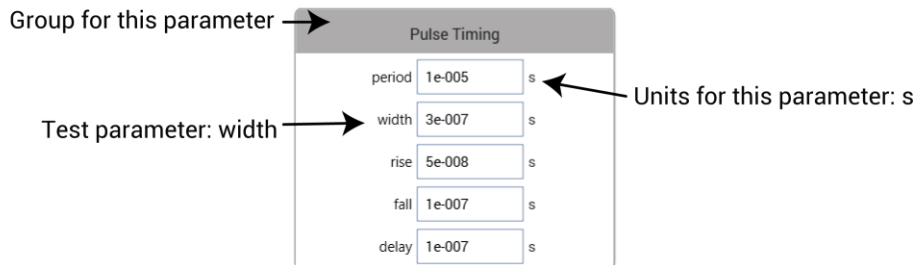
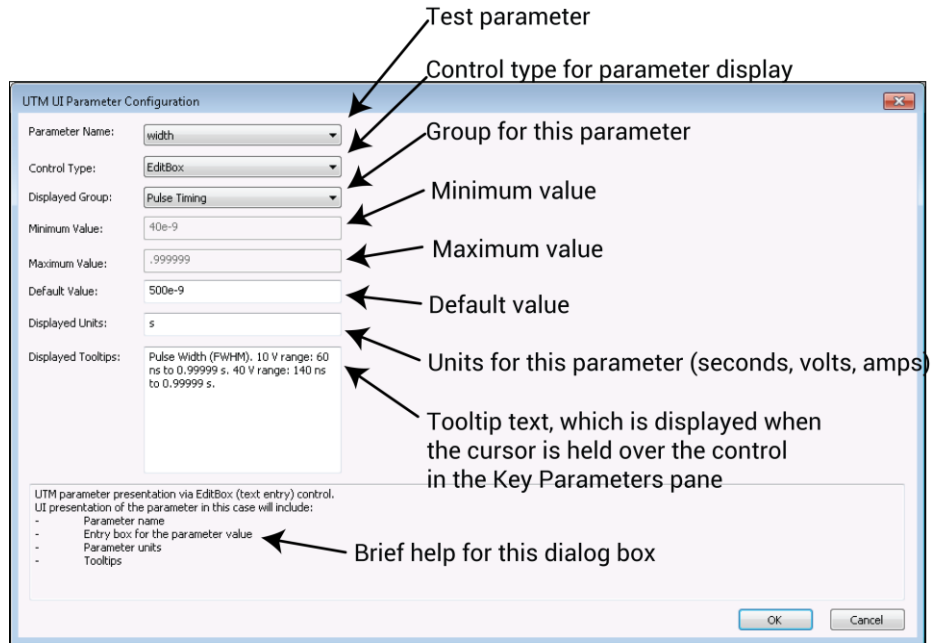
Group for this parameter

Control type: ListBox

Test parameter

Displayed names

Figure 23: 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 nonarray inputs. It is also the default control type for all nonarray inputs in dynamically generated UTM UI views.

The EditBox Control type is shown in [Edit the attributes for a test parameter](#) (on page 2-28).

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 nontext 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.

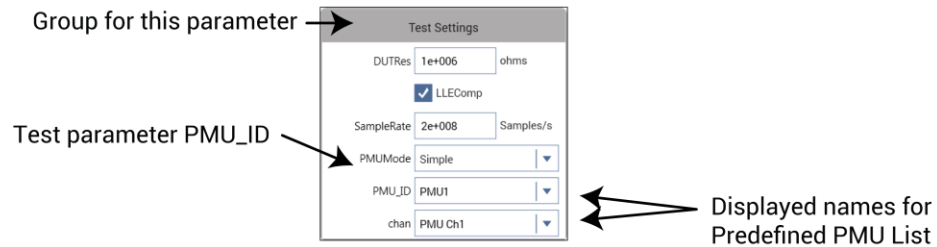
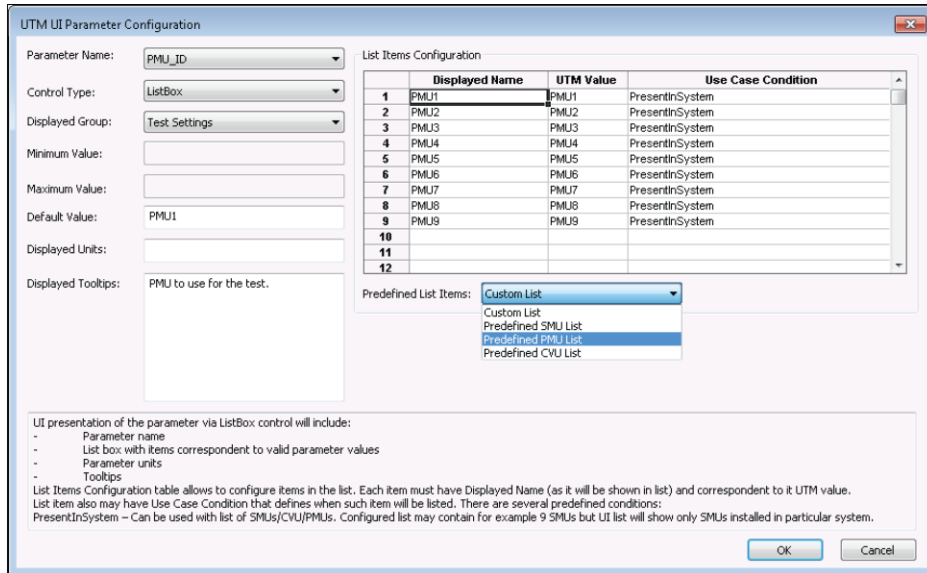
When finished, click **OK**.

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 24: 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 2-36) 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 `currentMeasureRng` 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 2-36) lists keywords and operators available for use in the Use Case Condition field.

Figure 25: ListBox UTM UI Parameter Configuration for currentMeasureRng

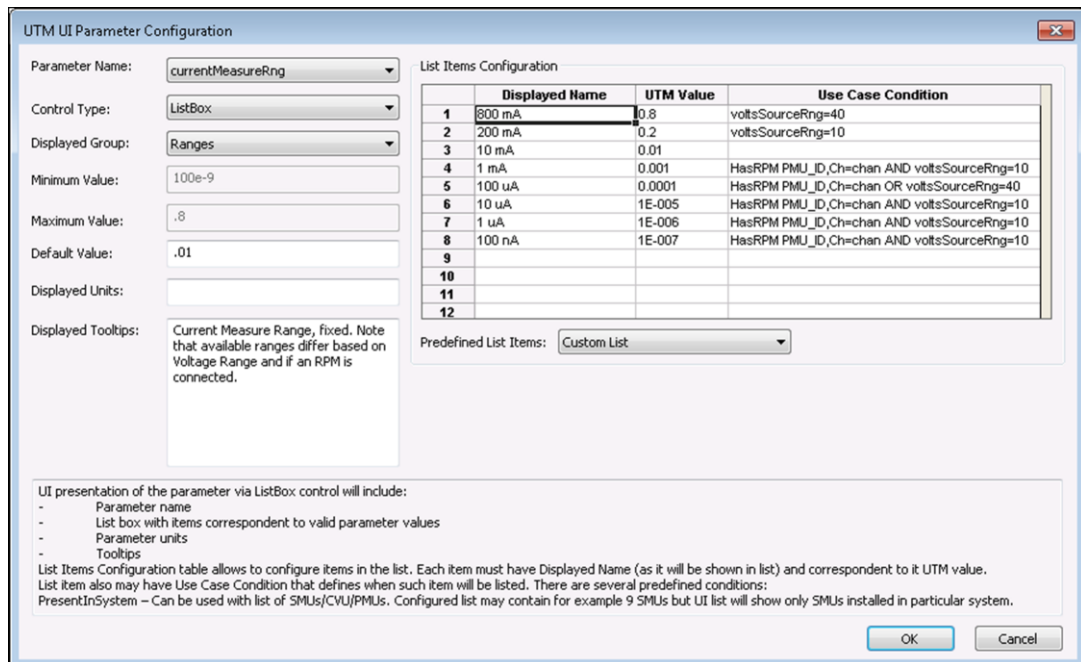
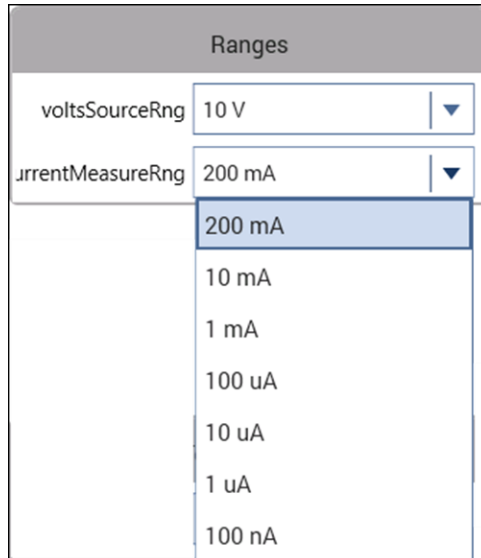


Figure 26: currentMeasureRng list box with a connected RPM



	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 <i>smuid</i>	True if SMU preamplifier is connected to <i>smuid</i>	Note that <i>smuid</i> can be a constant, “SMU1”, or a parameter name.
HasRPM <i>pmuid</i> , Ch= <i>chanid</i>	True if RPM is connected to <i>chanid</i> of <i>pmuid</i> .	Both <i>pmuid</i> and <i>chanid</i> 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

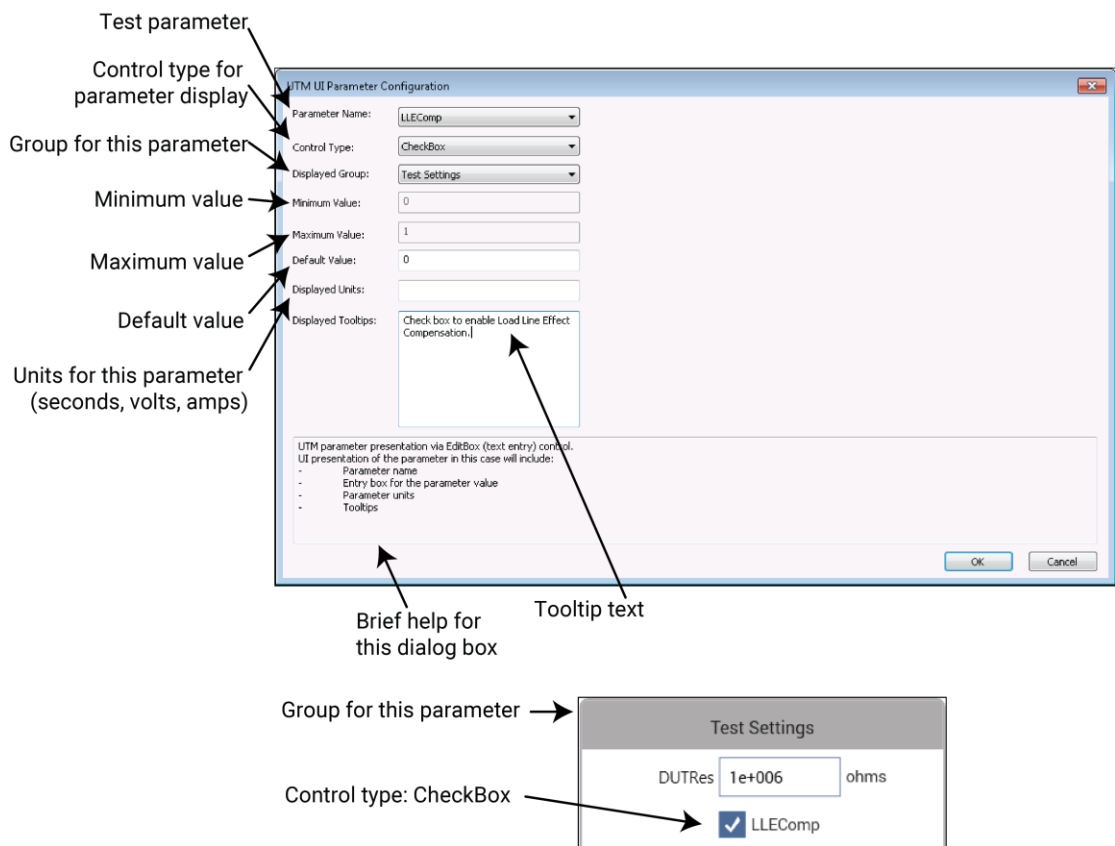
Use a check box control (CheckBox) for parameters that have two values or states. This control returns a zero (0) to indicate the box is not selected or a one (1) to indicate that it is selected.

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 27: 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 2-58) 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`. The 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). For more information on the measurement modes, refer to “Waveform measurements” in the *Model 4200A-SCS Pulse Card (PGU and PMU) User's Manual*.

Figure 28: OptionBtn UTM UI Parameter Configuration dialog box

Test parameter

Control type for parameter display

Group for this parameter

Minimum value

Maximum value

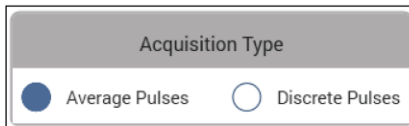
Default value

Units for this parameter, such as seconds or volts

Brief help for this dialog box

Tooltip text, which is displayed when mouse pointer is hovered over the control (in the UTM UI Key Parameters pane)

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

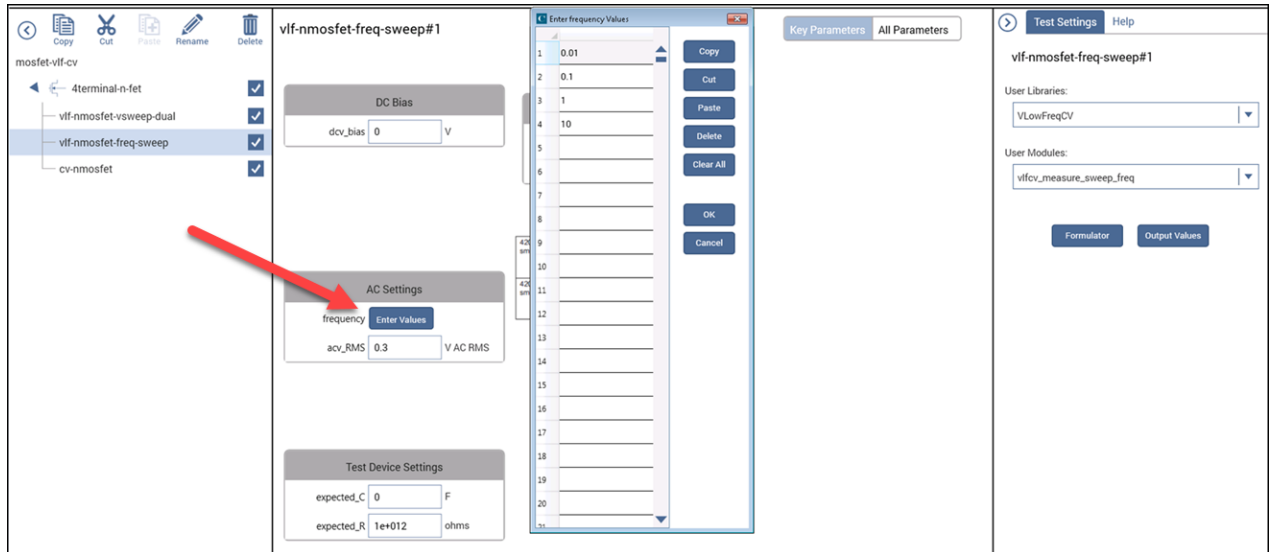
Use the input array control (InputArray) for array input types in user modules. Configure the parameter variable types using the Parameters tab in KULT.

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 29: 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 30: InputArray UTM parameter UI configuration

Test parameter

Control type for parameter display

Group for this parameter

Minimum value

Maximum value

Default value (for multiple entries use commas, no spaces)

Units for this parameter: such as seconds or volts

Brief help for this dialog box

Tooltip text, which is displayed when cursor is held over the control in the Key Parameters pane

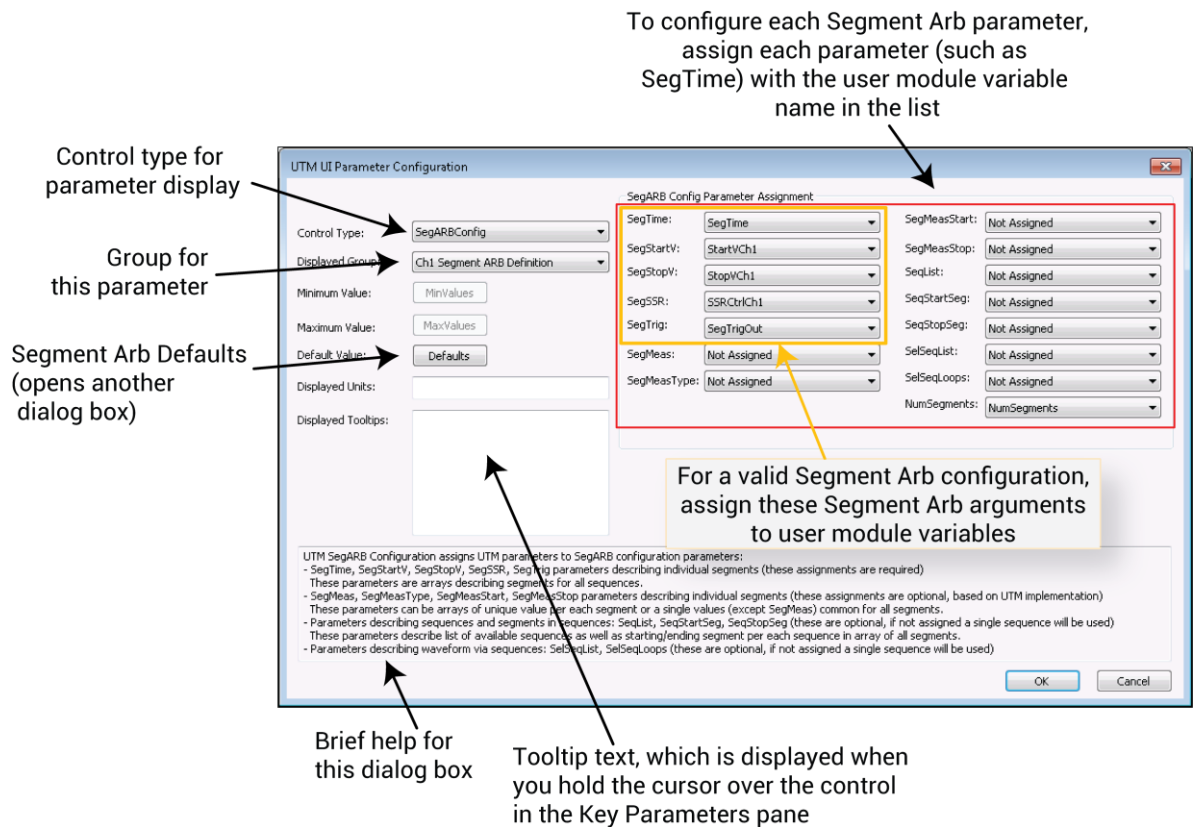
Group for this parameter

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` and `seg_arb_waveform`. 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 2-52).

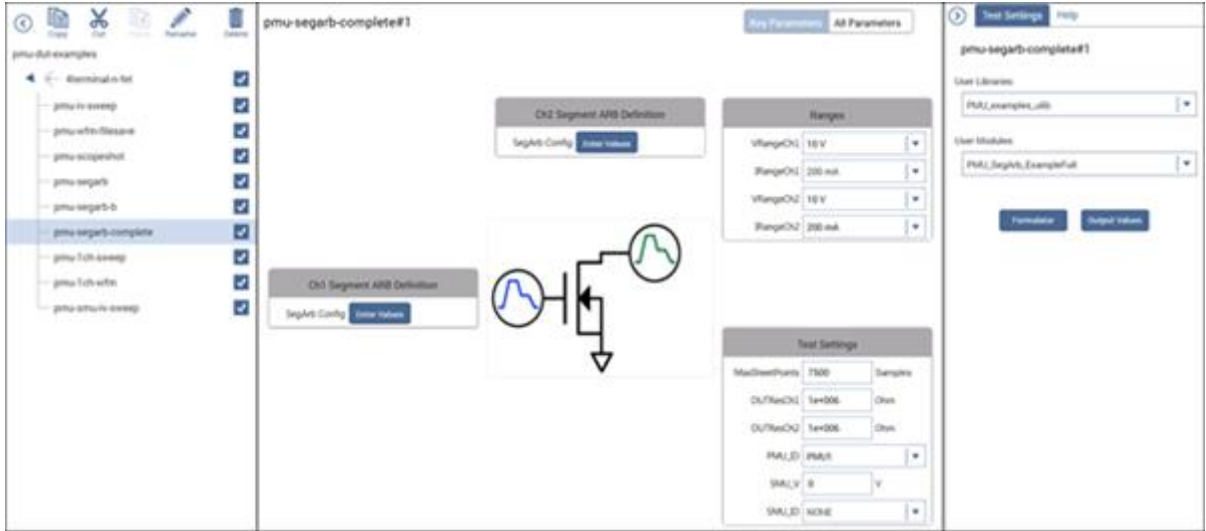
Figure 31: SegARBConfig UTM UI Parameter Configuration dialog box



PMU_SegArb_ExampleFull user module

For the example of this control type, the user module `PMU_SegArb_ExampleFull` is used. This module is included in the `pmu-dut-examples` project as the UTM named `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.

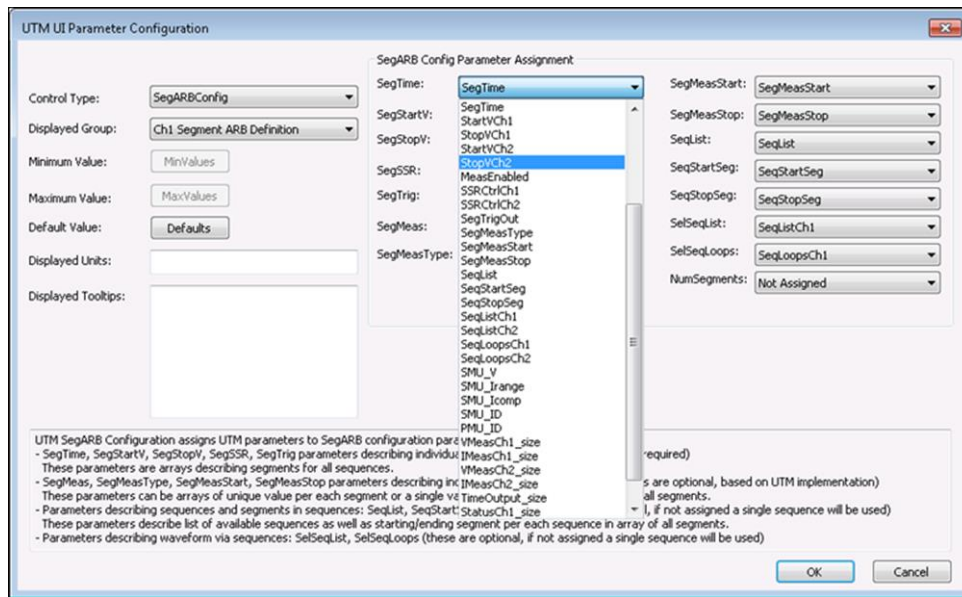
Figure 32: UTM Configure pane for `PMU_SegArb_ExampleFull`



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 2-42) and the figure below). Also, each parameter assignment involves many parameters (see [PMU_SegArb_ExampleFull user module](#) (on page 2-43)). This is in contrast to other controls. For example, the `InputArray` control has a single parameter in its parameter configuration dialog box.

Figure 33: 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. 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.

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.

SegARBConfig parameter names mapped to LPT function `seg_arb_sequence`

Parameter name	Variable name in function	Type	Required in SegARBConfig?	Description
<code>SegTime</code>	<code>Time</code>	Double array	Yes	Time duration for each segment
<code>SegStartV</code>	<code>StartV</code>	Double array	Yes	Segment start voltage
<code>SegStopV</code>	<code>StopV</code>	Double array	Yes	Segment stop voltage
<code>SegSSR</code>	<code>SSR</code>	Integer array	Yes	Solid State Relay control, per segment
<code>SegTrig</code>	<code>Trig</code>	Integer array	Yes	Trigger output state for each segment
<code>SegMeasType</code>	<code>MeasType</code>	Integer array or Integer*	No	Measurement Type: None, spot mean, or waveform
<code>SegMeasStart</code>	<code>MeasStart</code>	Double array or Double ¹	No	Start point for measurement window, for each segment; from 0 to 1 (100%)
<code>SegMeasStop</code>	<code>MeasStop</code>	Double array or Double ¹	No	Stop point for measurement window, per segment; from 0 to 1 (100%)
<code>NumSegments</code>	<code>NumSegments</code>	Integer	No	Number of segments in a sequence

* If using a single value instead of an array, the `SegARBConfig` control automatically assigns the value to each segment.

Parameter targets mapped to LPT function `seg_arb_waveform`

Parameter name	Variable name in function	Type	Required in SegARBConfig?	Description
<code>SelSeqList</code>	<code>Seq</code>	Integer array	No	List of sequences that define the Seg-Arb waveform
<code>SelSeqLoops</code>	<code>SeqLoopCount</code>	Integer array	No	Array of loops values for each sequence in the Seg-Arb waveform

SegARBConfig parameter names unique to function `SegARBConfig`

Parameter name	Variable name in function	Type	Required in SegARBConfig?	Description
<code>SegMeas</code>	<code>SegMeas</code>	Integer array	No	Measurement Enabled ¹
<code>SeqList</code>	<code>SeqList</code>	Integer array	No	List of defined sequences in SegARBConfig
<code>SeqStartSeg</code>	<code>SeqStartSeg</code>	Integer array	No	Start segment array value in SegARBConfig
<code>SeqStopSeg</code>	<code>SeqStopSeg</code>	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).

Multi-sequence tests

Three parameters, although not required for the `SegARBConfig` control, are required for multi-sequence tests. These 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` function and the multi-sequence waveform using the `seg_arb_waveform` 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 the figure named "SegARBConfig uses SeqList, SeqStartSeg, and SeqStopSeg 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 SeqStopSeg 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 34: Segment Arb UI UTM configuration for channel 1

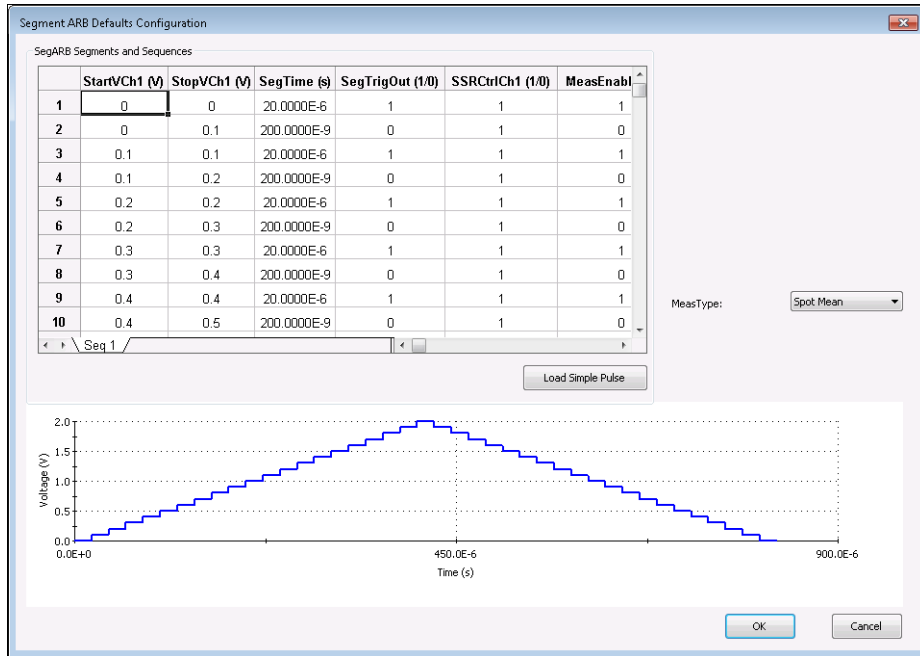


Figure 35: Two-sequence Segment Arb waveform

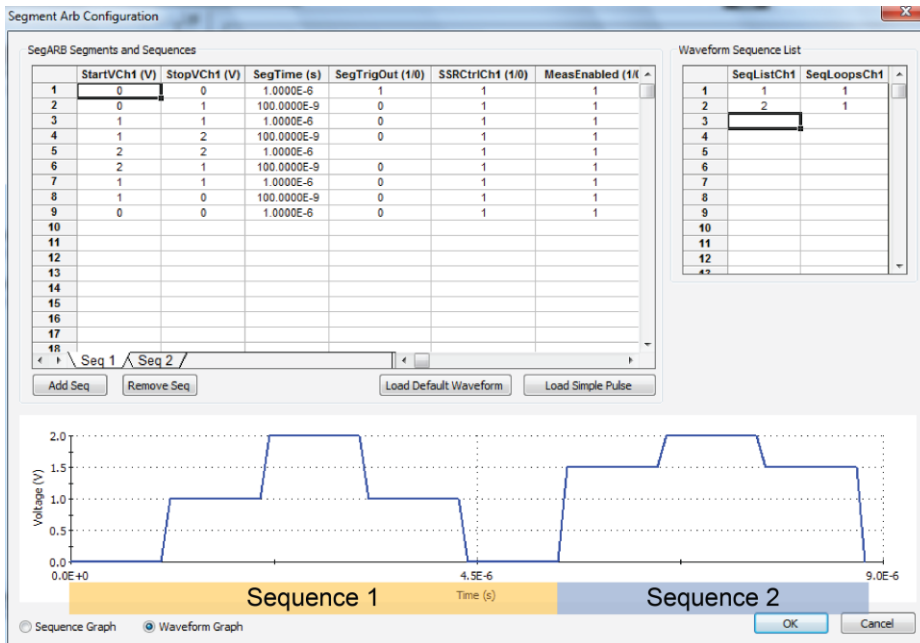
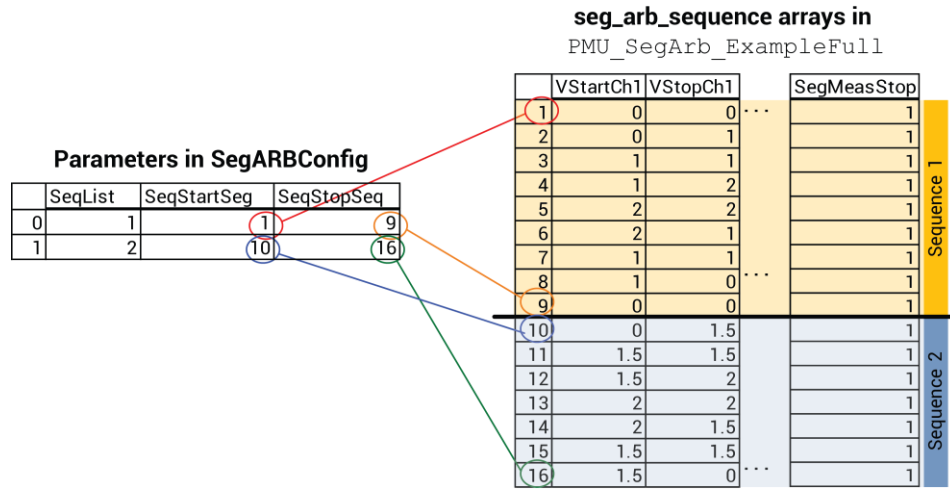


Figure 36: SegARBConfig uses SeqList, SeqStartSeg, and SeqStopSeg to store sequence



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" in the Model 4200A-SCS LPT Library Programming Manual).

Figure 37: Channel 1 SegARBConfig parameter configuration

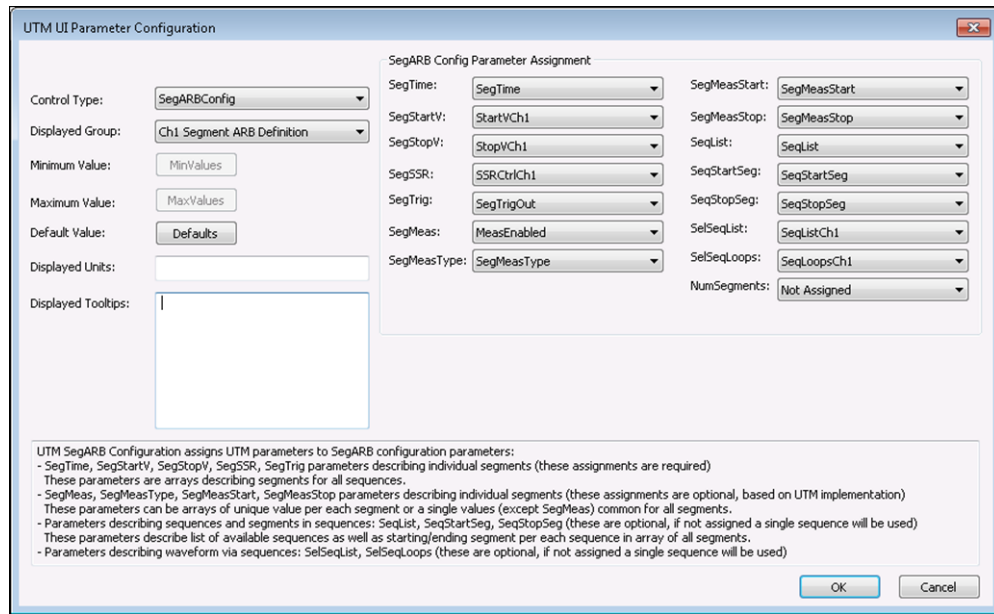
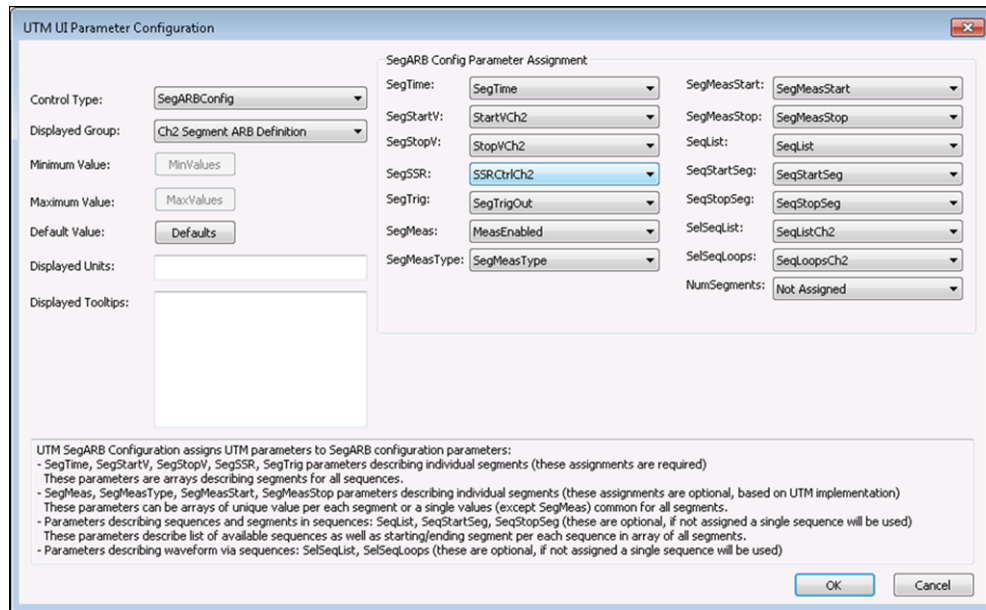


Figure 38: Channel 2 SegARBConfig parameter configuration



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 2-50).

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 39: Segment Arb Defaults Configuration

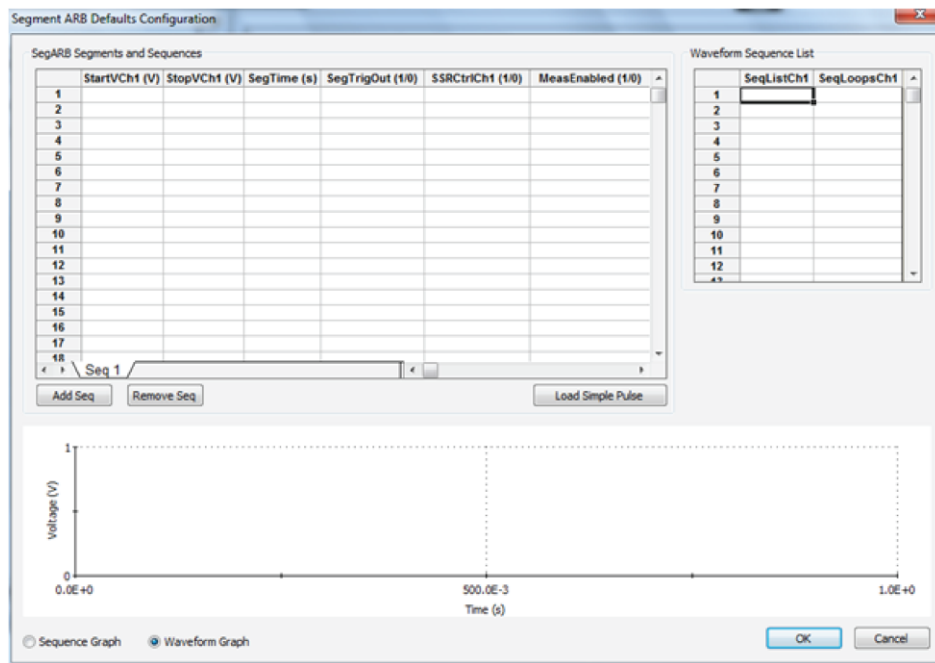
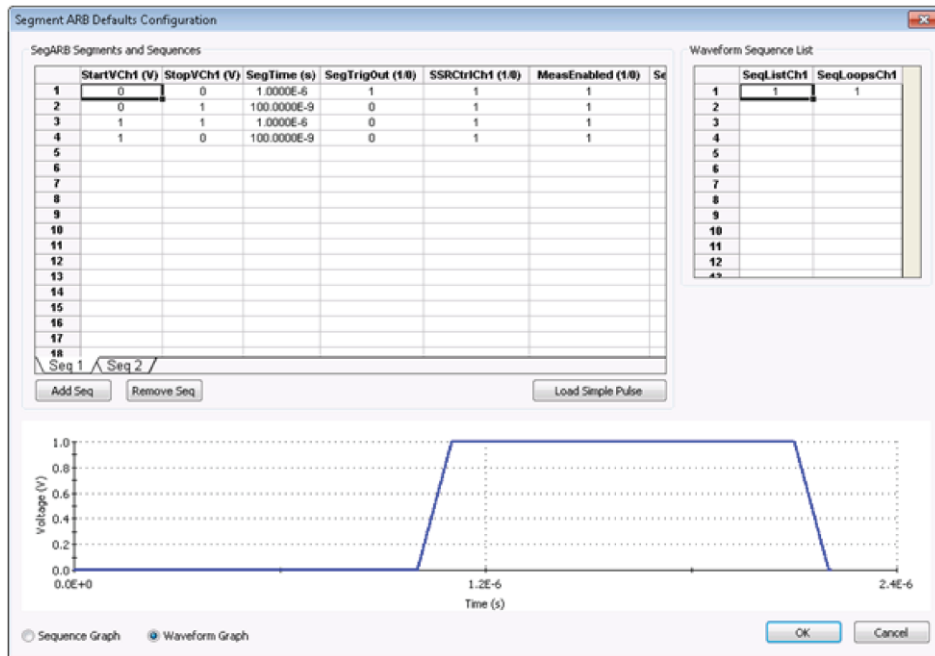


Figure 40: UTM UI Editor Segment Arb Defaults Configuration (after pressing Load Simple Pulse)



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 41: UTM UI Editor Segment Arb Defaults Configuration for channel 1

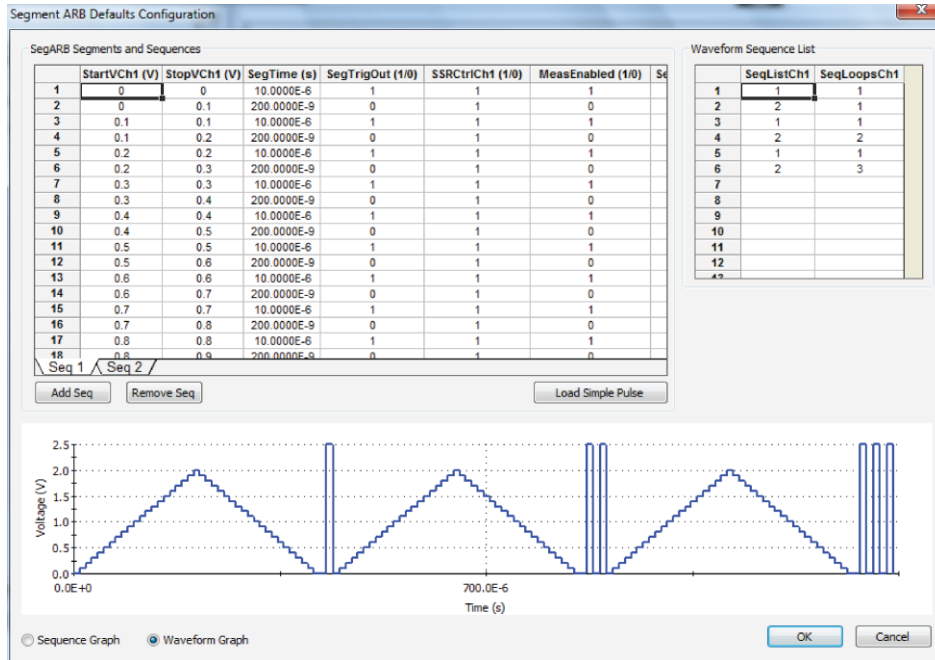
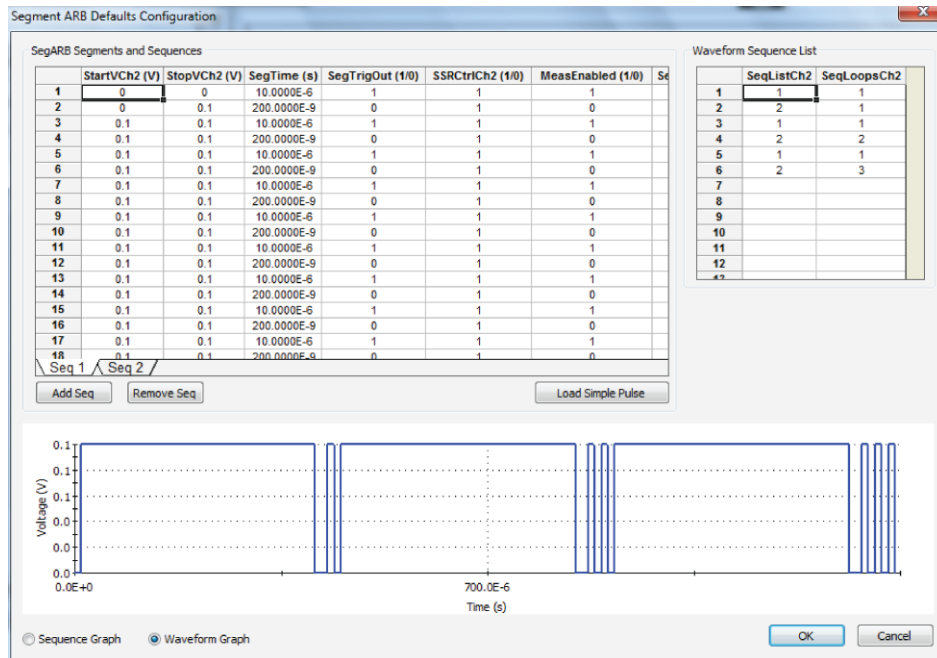


Figure 42: UTM UI Editor Segment Arb Defaults Configuration for channel 2

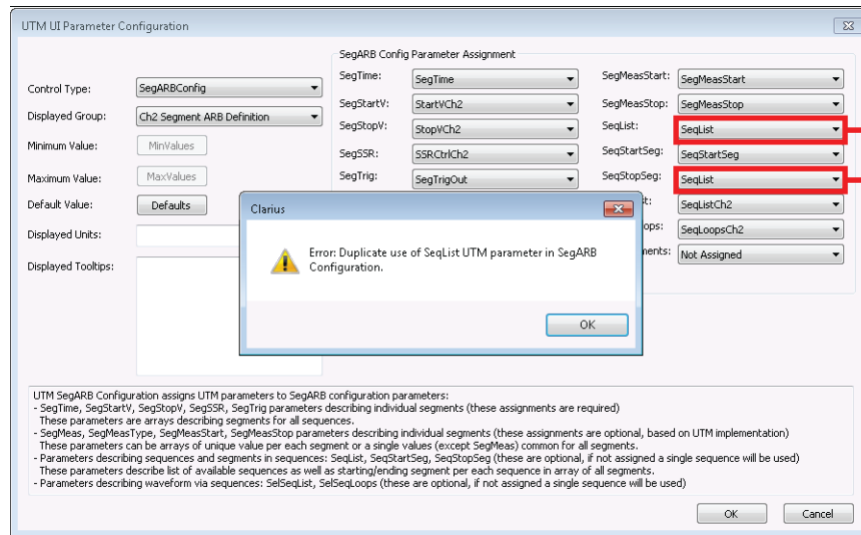


Error checking

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 2-50). 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 43: 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 44: UTM UI editor

Instructions for the UTM UI Editor

User library and user module names

Click a Configure button to change the parameter's settings (or double-click in the row)

Scroll to see all parameters in the user module

Click to reset to defaults

Click to add a Group

Click OK to save all changes and return to the Configure pane

Parameter Name	Control Type	Group	Min	Max	Default	Units	Tooltips	Configure
1 smu_src	ListBox	Resources			SMU1		SMU (Preamp required) ...	
2 smu_sense	ListBox	Resources			SMU2		SMU (Preamp required) ...	
3 frequency	InputArray	AC Settings			10e-3,100e... Hz		Enter an array of freq...	
4 expected_C	EditBox	Test Device Settings	0	0.001	0	F	Minimum >= 1e-15 F. En...	
5 expected_R	EditBox	Test Device Settings	1000000	1E+014	1E+012	ohms	The expected parallel ...	
6 acv_RMS	EditBox	AC Settings	0.01	3	0.3	V AC RMS	AC RMS voltage. Valid ...	
7 dcv_bias	EditBox	DC Bias	-20	20	0	V	DC voltage. Valid rang...	
8 times_count	EditBox		1	512	512			
9 meas_freq_count	EditBox		1	512	512			
10 meas_Cp_count	EditBox		1	512	512			
11 meas_Op_count	EditBox		1	512	512			
12 meas_Z_count	EditBox		1	512	512			

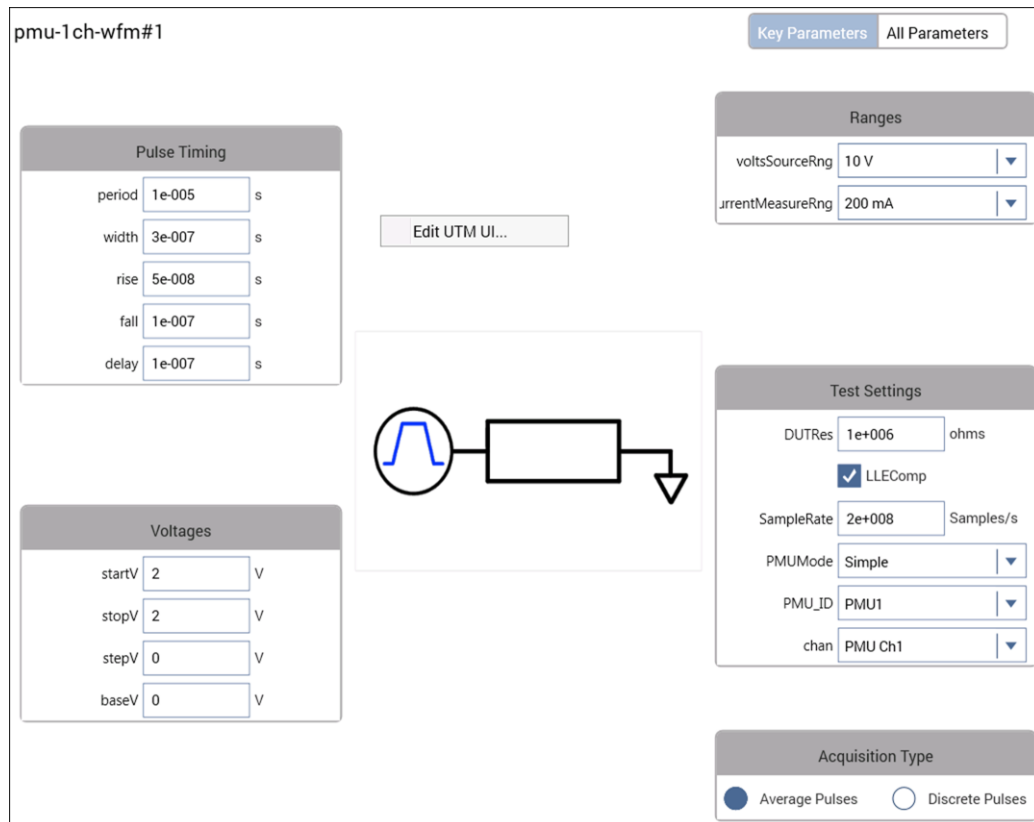
Example of using the editor

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 2-22).
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.

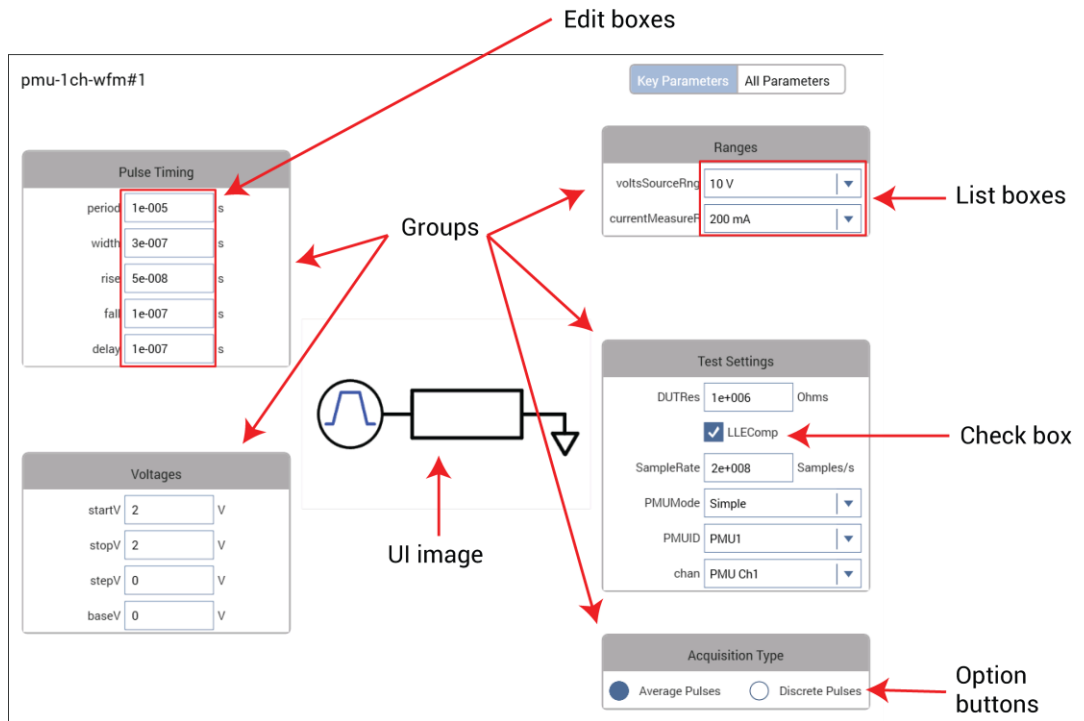
Figure 45: Start the UTM UI editor (right-click the Configure pane)



6. Select **Edit UTM UI**. The UTM UI Editor dialog box opens, as shown in [UTM UI Editor](#) (on page 2-57).

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 46: 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” in *Model 4200A-SCS KULT and KULT Extension Programming*.

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 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 creates a user file. Do not modify the XML files outside of the UTM UI Editor, as errors or nonfunctional 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 2-57)). 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

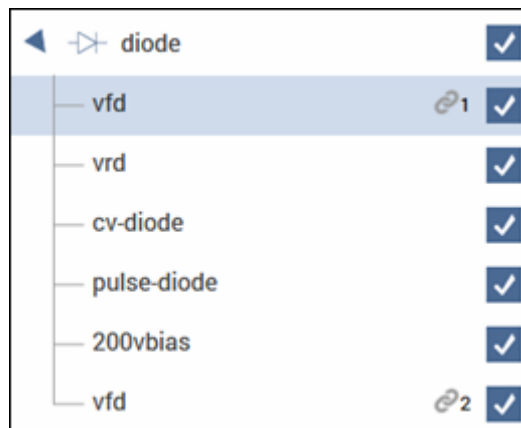
The UTM UI definition consists of one file for each user library. Use the `kultcopy` command to move a user library to another 4200A-SCS. This command copies the user modules (c code), UTM UI definition (XML files), and any bitmaps for the UTM UI (bitmapped files with the `.bmp` extension). For more information, see `kultcopy` in “Copying user libraries using `kultcopy`” in *Model 4200A-SCS KULT and KULT Extension Programming*.

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 47: 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 one 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 2-3) or [Create a custom test](#) (on page 2-20)).
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.

Example: Creating a project

This section provides an example of how to create a new blank project and configure a new blank test. You will create a test to be performed on a MOSFET, but the procedure is general and can be applied to different devices and applications.

NOTE

The default settings used for the devices, tests, and projects in Clarius are generally sufficient to produce usable data when executing a test. However, you may have additional settings you want to apply when you configure your measurements.

Equipment required

- One 4200A-SCS, with the following instruments:
 - Two medium power (4200-SMU or 4201-SMU) or high power (4210-SMU or 4211-SMU) SMUs
 - Two 4200-PAs
- Three 4200-TRX-2 or 4200-MTRX-2 triaxial cables (supplied with SMU)
- One shielded, three-terminal test fixture with triaxial inputs (such as the 8101-PIV)

Device connections

Using the supplied cables, connect the output terminals of the instruments directly to the MOSFET terminals in the shielded test fixture. The triaxial terminals on the shielded test fixture allow you to connect to the device and maintain a completely shielded and guarded test setup.

⚠ WARNING

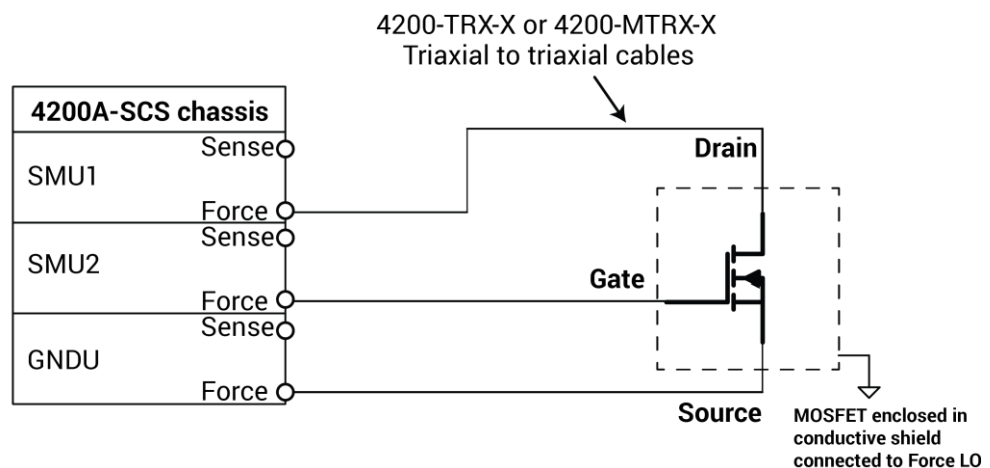
Hazardous voltages may be present on all output and guard terminals. To prevent electrical shock that could cause injury or death, never connect or disconnect from the 4200A-SCS while the output is on.

To prevent electric shock, test connections must be configured such that the user cannot come in contact with test leads, conductors, or any device under test (DUT) that is in contact with the conductors. It is good practice to disconnect DUTs from the instrument before powering up the instrument. Safe installation requires proper shields, barriers, and grounding to prevent contact with test leads and conductors.

Connection schematic

The hardware connections from the output of the instruments in the 4200A-SCS chassis to the test fixture that contains the MOSFET are shown in the following figure. All of the connections are 2-wire and only the Force terminal of each SMU is used. The SMUs and GNDU are each connected to a different terminal of the 3-terminal MOSFET.

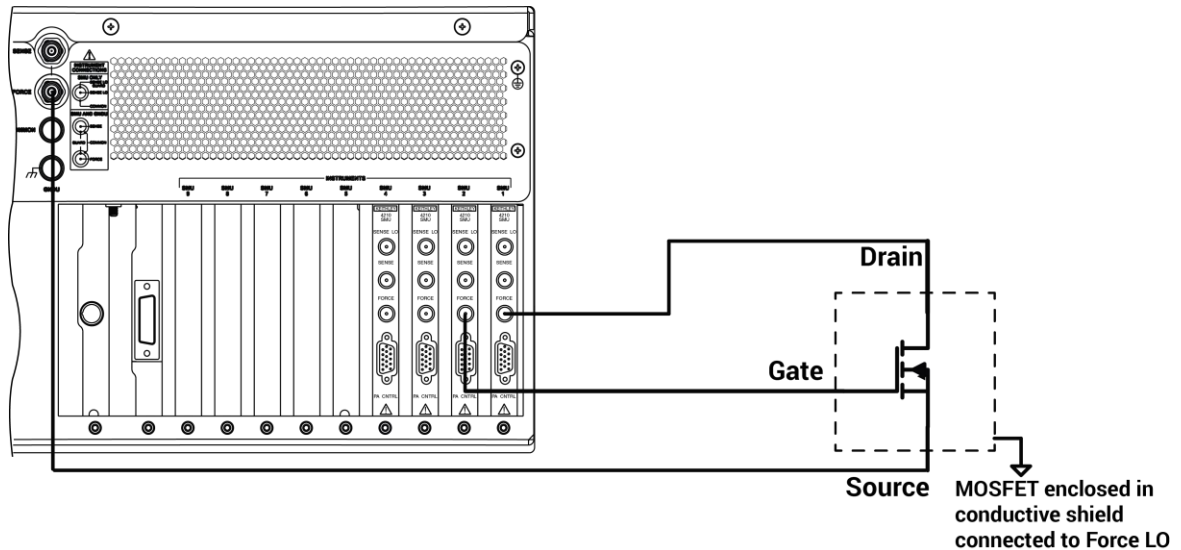
Figure 48: Connections from the 4200A-SCS to a MOSFET



Connect the 4200A-SCS to the DUT

The hardware connections from the output of the instruments in the 4200A-SCS chassis to the test fixture that contains the MOSFET are shown in the following figure.

Figure 49: Rear-panel connections from the 4200A-SCS to a MOSFET



Set up the measurements in Clarius

This section describes how to set up the 4200A-SCS to generate a V_{ds} - I_d family of curves for a 3-terminal n-type MOSFET. This general procedure can also be used to create tests for other devices and other applications.

For this example, you will use the Clarius application to:

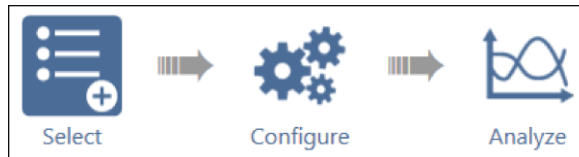
- Select and rename a new project
- Add a device
- Select a custom test
- Configure the test
- Execute the test
- View and analyze the test results

Select and rename a new project

To select and rename a new project:

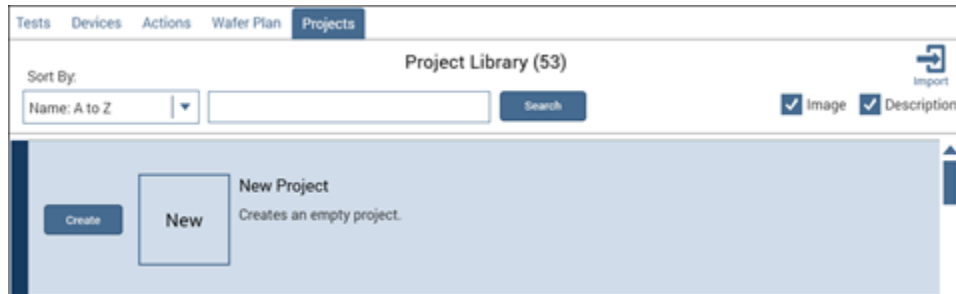
1. Choose **Select**.

Figure 50: Select highlighted



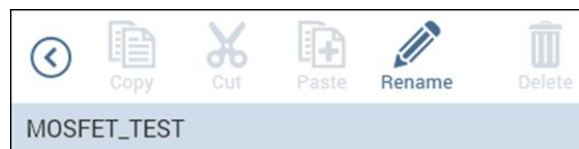
2. In the Library, select **Projects**.
3. Select **New Project**.
4. Select **Create**.

Figure 51: Select a New Project from the Project Library



5. Select **Yes** when prompted to replace the existing project.
6. Assign a title to the project by selecting **Rename** above the project tree.
7. Enter a project name into the text box, then select **Enter**. `MOSFET_TEST` has been chosen for this example.

Figure 52: Toolbar with Rename function



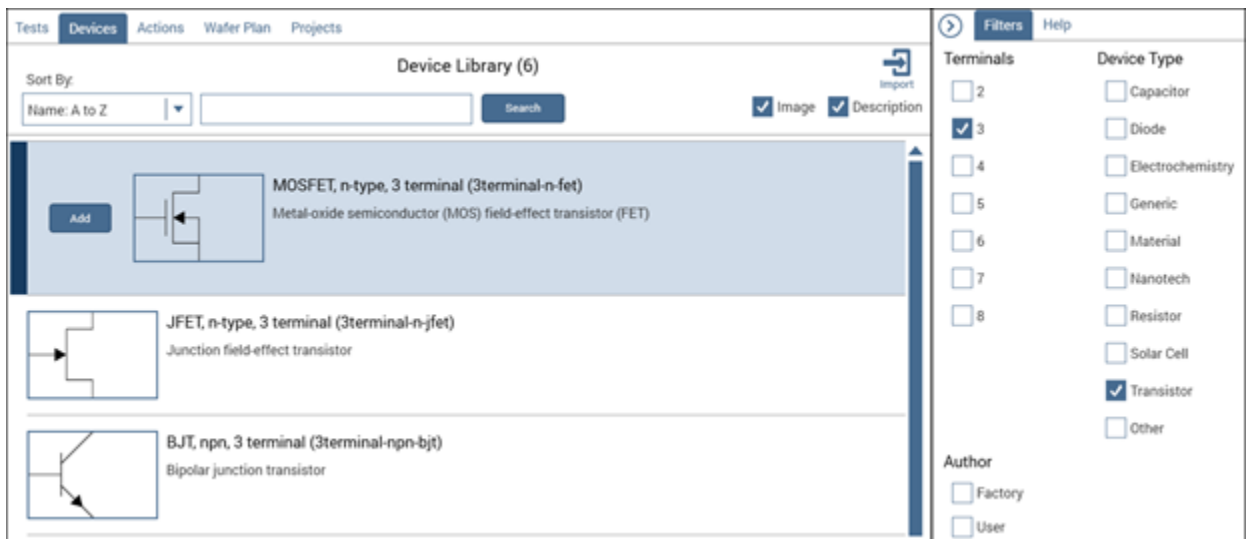
Add a device

Tests must be placed in the project under a device.

To add a device:

1. Select **Devices**.
2. From the Filters pane, select the **3** under the Terminals heading and **Transistor** under the Device Type option.

Figure 53: Searching for a device using Filters



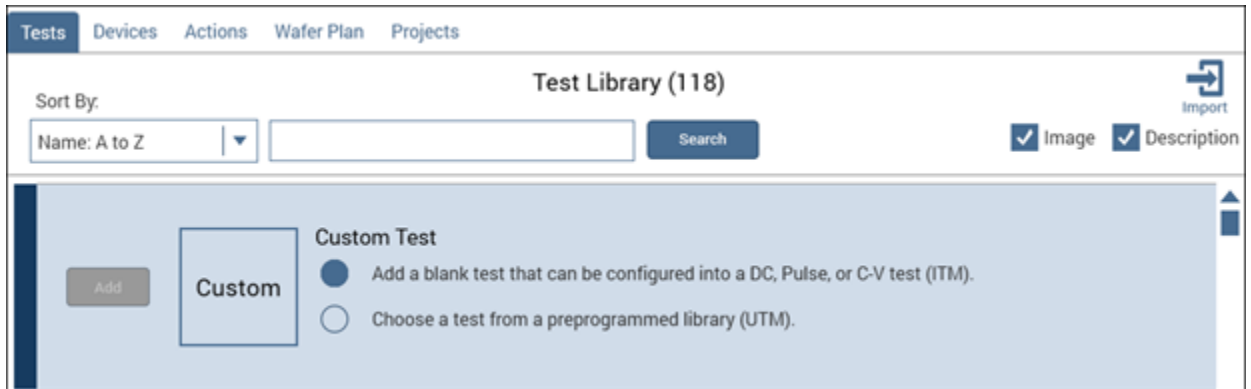
3. Select the MOSFET, n-type, 3 terminal (3terminal-n-fet) device.
4. Select **Add** to copy it to the project tree.

Select a custom test

To select a custom test:

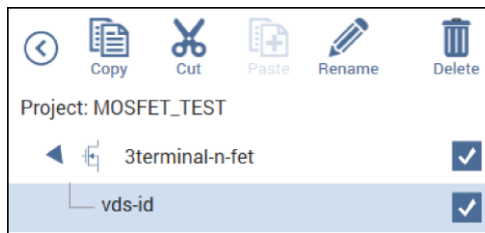
1. Select **Tests**.
2. Select **Custom Test**, then select **Add** to create a new 3-terminal, n-type MOSFET test in the project tree.

Figure 54: Custom Test option



3. Select **Rename** from the toolbar. Enter a test name in the text box, then select **Enter**. `vds-id` was chosen for this example.

Figure 55: MOSFET_TEST project tree with one device and one test

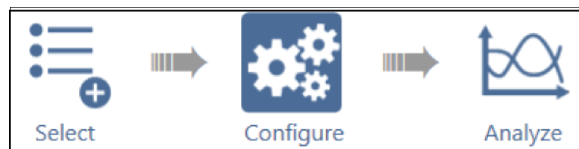


Configure the test

To configure the test:

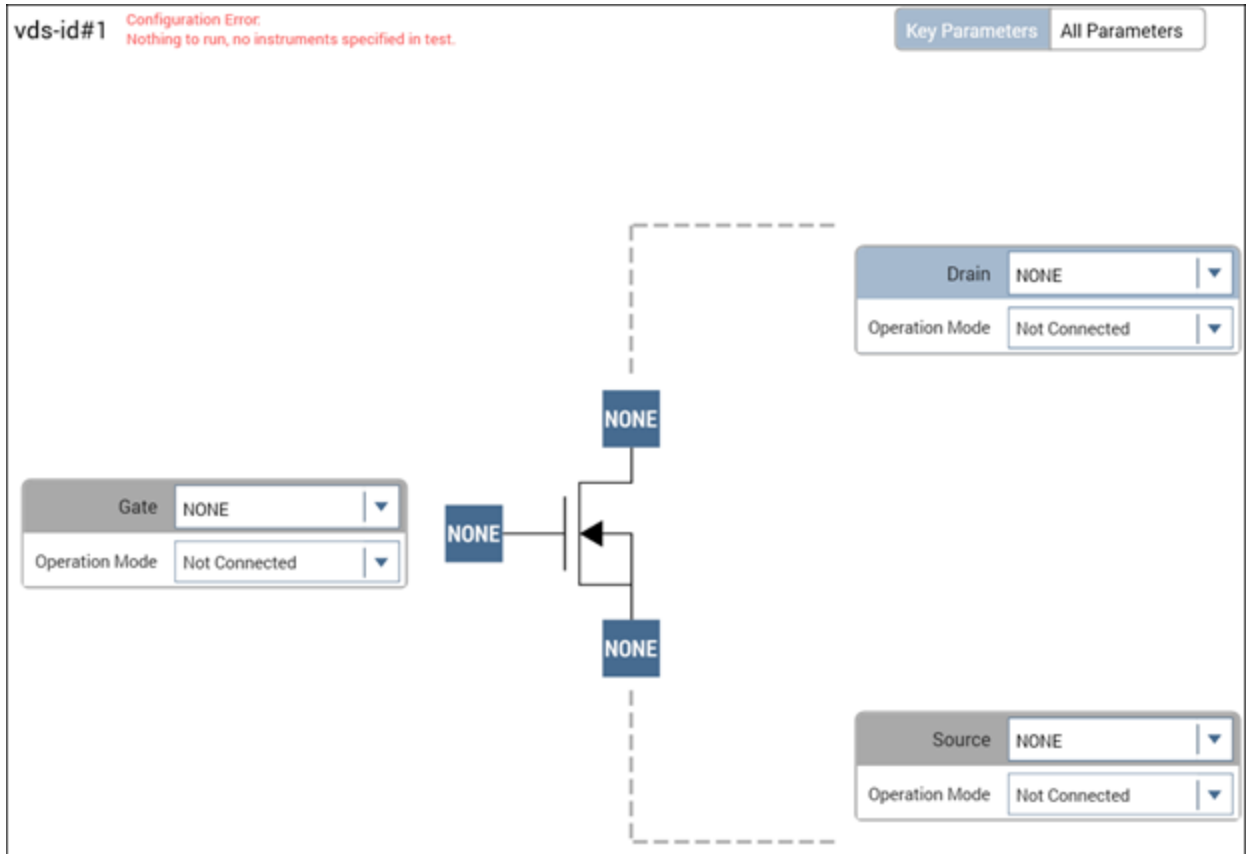
1. Select **Configure**.

Figure 56: Configure highlighted



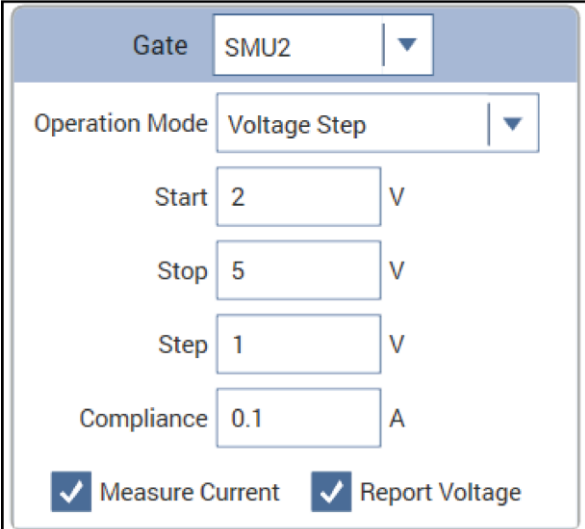
- In the project tree, select `vds-id`. Because this test is custom, you must assign functions to all terminals connected to the MOSFET before you can run the test.

Figure 57: All MOSFET terminals unassigned in a custom test



3. Set the Gate terminal connection to **SMU2**.
4. Set the Operation Mode to **Voltage Step**.
5. Change the Start, Stop, Step, and Compliance settings to match the following figure or to the gate settings appropriate for your device.

Figure 58: SMU2 steps from 2 V to 5 V, connected to MOSFET Gate terminal

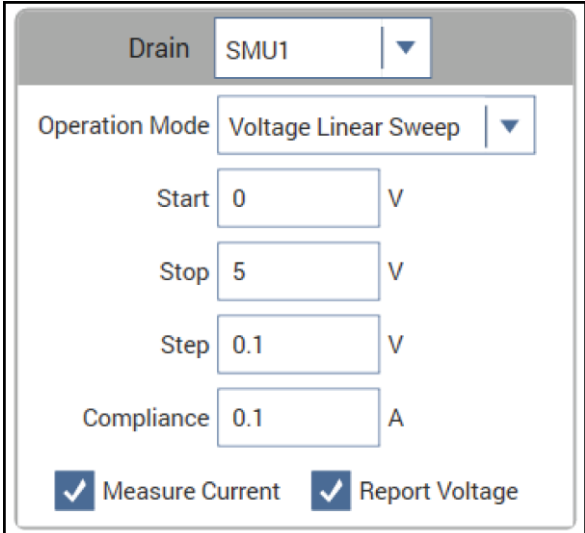


The screenshot shows the configuration window for SMU2. The 'Gate' dropdown is set to 'SMU2'. The 'Operation Mode' dropdown is set to 'Voltage Step'. The 'Start' field is 2 V, the 'Stop' field is 5 V, the 'Step' field is 1 V, and the 'Compliance' field is 0.1 A. Both 'Measure Current' and 'Report Voltage' checkboxes are checked.

Parameter	Value	Unit
Gate	SMU2	
Operation Mode	Voltage Step	
Start	2	V
Stop	5	V
Step	1	V
Compliance	0.1	A
Measure Current	<input checked="" type="checkbox"/>	
Report Voltage	<input checked="" type="checkbox"/>	

6. Set the Drain terminal connection to **SMU1**.
7. Set the Operation Mode to **Voltage Linear Sweep**.
8. Change the Start, Stop, Step, and Compliance settings to match the following figure.

Figure 59: SMU1 sweeps from 0 V to 5 V, connected to MOSFET Drain terminal



The screenshot shows the configuration window for SMU1. The 'Drain' dropdown is set to 'SMU1'. The 'Operation Mode' dropdown is set to 'Voltage Linear Sweep'. The 'Start' field is 0 V, the 'Stop' field is 5 V, the 'Step' field is 0.1 V, and the 'Compliance' field is 0.1 A. Both 'Measure Current' and 'Report Voltage' checkboxes are checked.

Parameter	Value	Unit
Drain	SMU1	
Operation Mode	Voltage Linear Sweep	
Start	0	V
Stop	5	V
Step	0.1	V
Compliance	0.1	A
Measure Current	<input checked="" type="checkbox"/>	
Report Voltage	<input checked="" type="checkbox"/>	

9. Set the Operation Mode of the Source terminal to **GNDU**.

Execute the test

Select **Run** to execute the test.

Figure 60: Run



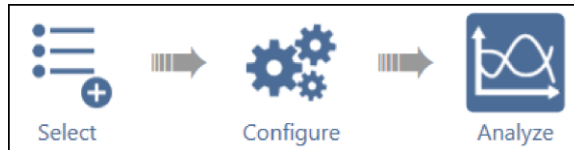
View and analyze the test results

While the test is running, you can view the data in the spreadsheet of the Analyze pane. Because you created a new test, the data must be assigned to the axes of the graph before you can view graphical results.

To view and analyze the test results:

1. Select **Analyze**. The Analyze screen displays data as it is gathered in the spreadsheet and a blank graph with unassigned axes.

Figure 61: Analyze highlighted



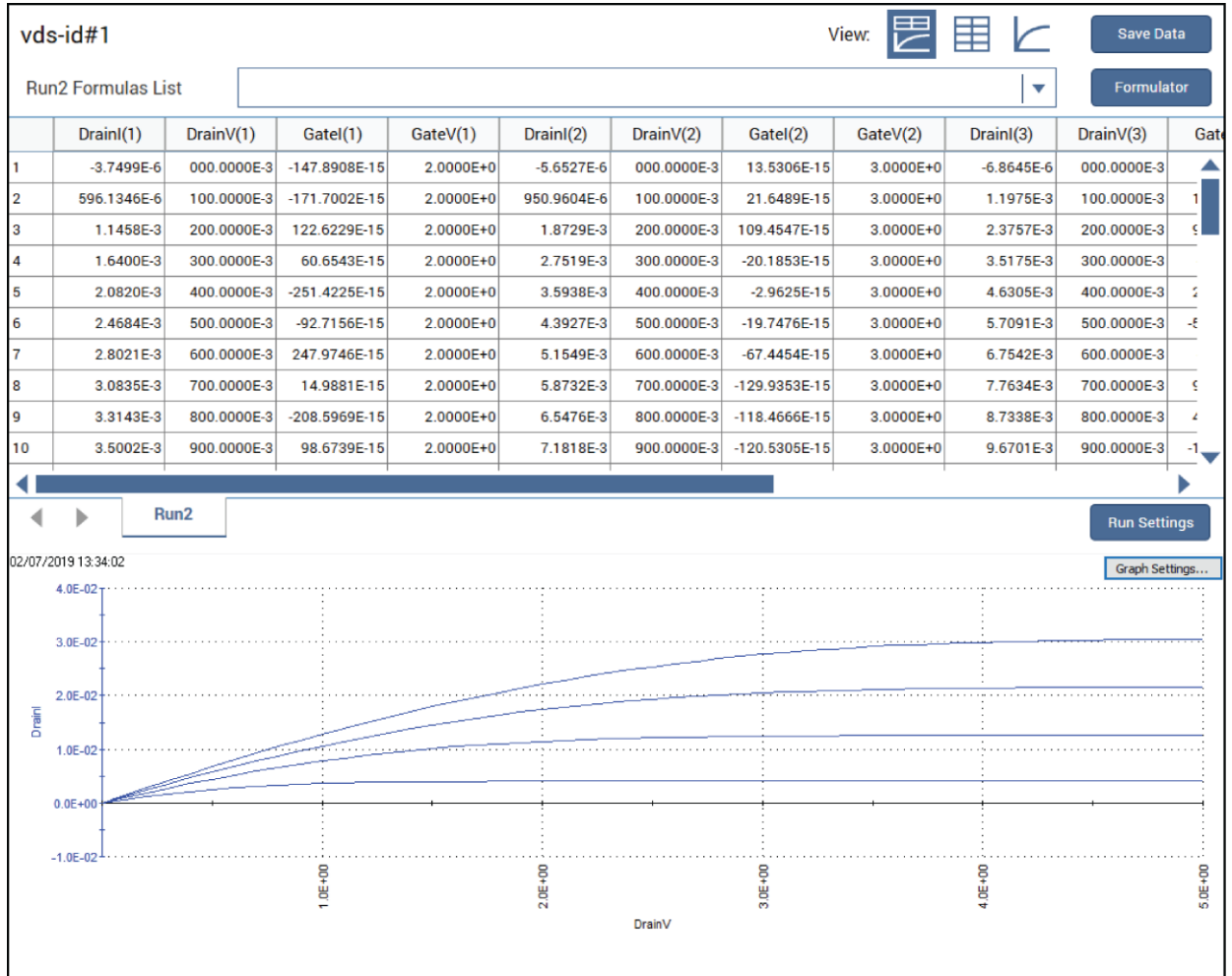
2. Select **Graph Settings**.
3. Select **Define Graph**.
4. In the Graph Definition screen, assign X to **DrainV** and Y1 to **DrainI**.

Figure 62: Define the graph

Data Series	Sheet	Column	X	Y1	Y2
DrainI*	Data	1		*	
DrainV*	Data	2	*		
GateI*	Data	3			
GateV*	Data	4			

5. Select **OK**.
6. The graph displays the `vds-Id` family of curves.

Figure 63: Analyze showing test results



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.

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.

The options that are available depend on the instrument that is selected. For descriptions of parameters, refer to:

- “SMU Test Settings” in the *Model 4200A-SCS Source-Measure Unit (SMU) User's Manual*
- “CVU Test Settings” in the *Model 4200A-SCS Capacitance-Voltage Unit (CVU) User's Manual*
- “PMU Test Settings” in the *Model 4200A-SCS Pulse Card (PGU and PMU) User's Manual*

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 64: Master step versus subordinate step

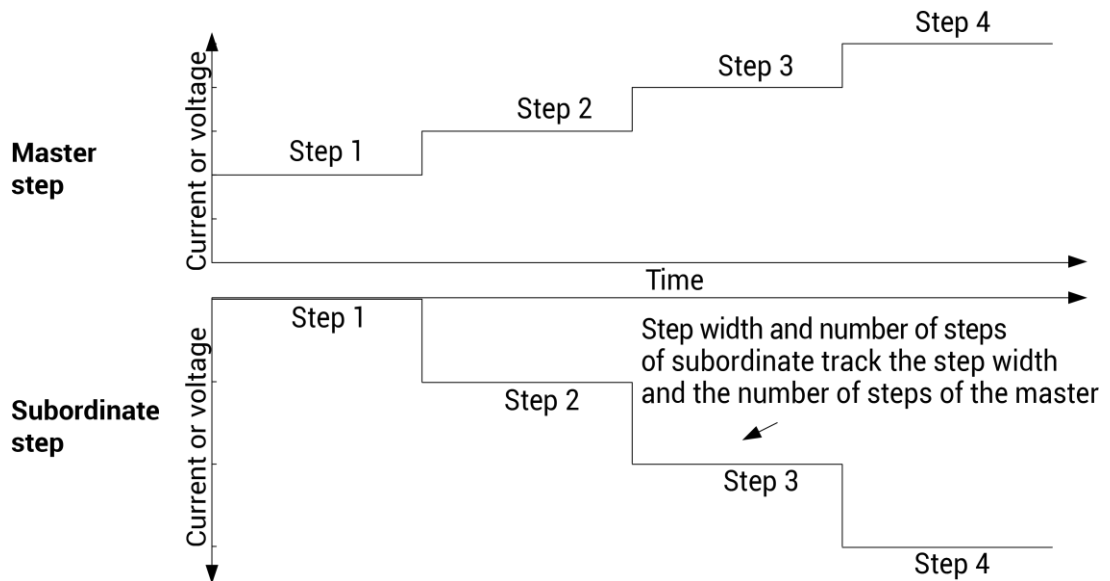


Figure 65: Master and subordinate sweeps

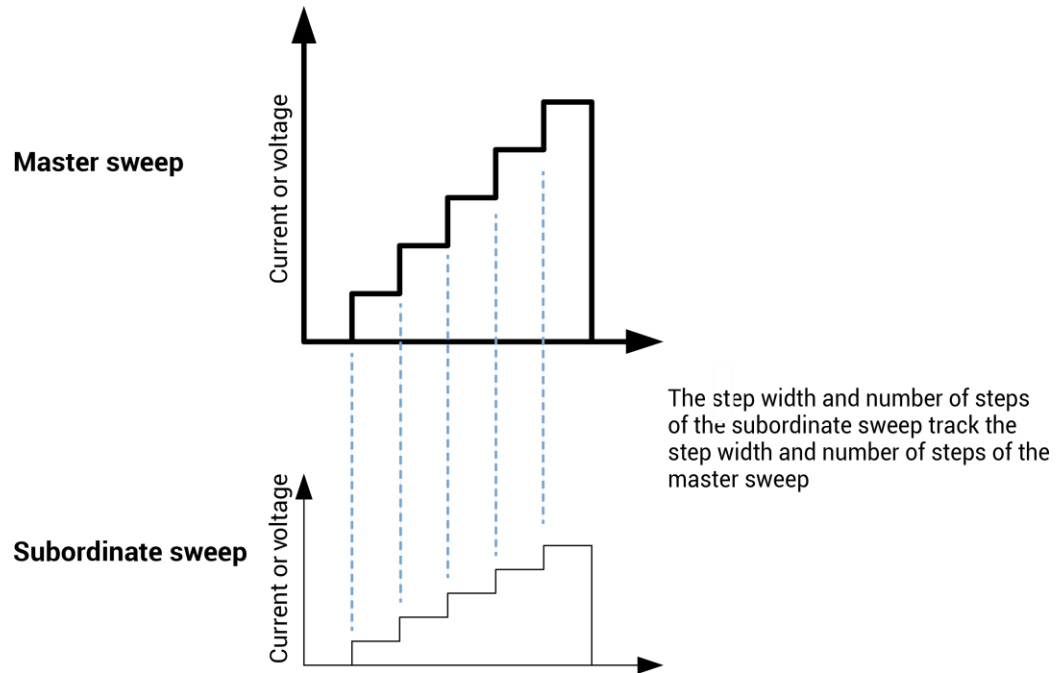
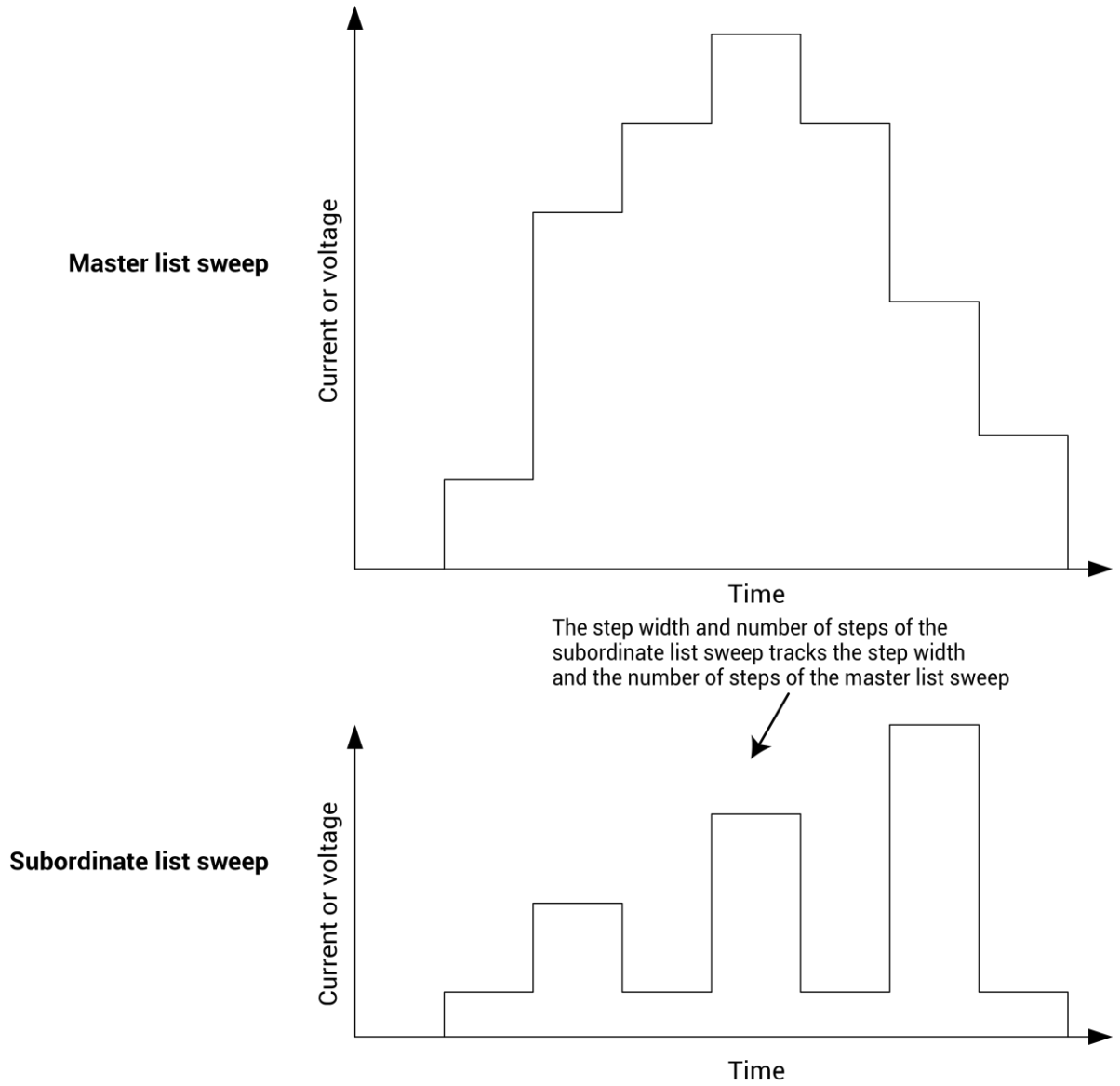


Figure 66: Master list sweeps versus subordinate list sweeps



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.

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 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 3-1).

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:

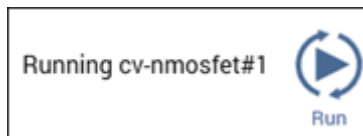
1. Open the **Demo Project**. Refer to [Open a project](#) (on page 2-10).
2. Make sure the check boxes are selected for all items in the project tree.
3. Highlight the project name, as shown in the figure below.

Figure 67: 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 68: 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

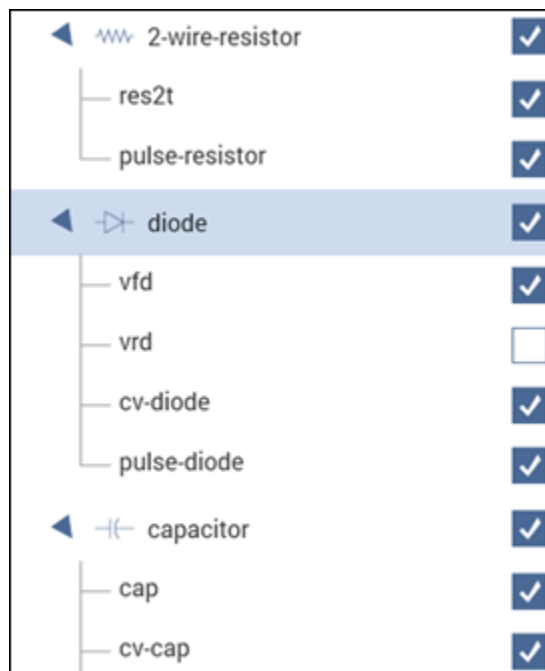
To abort a test, select **Stop**. All test and action execution stops immediately.

The following example uses the Demo Project to demonstrate how to run tests for a device.

To run tests for a device:

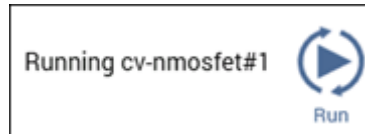
1. Open the **Demo Project**. Refer to [Open a project](#) (on page 2-10).
2. Make sure the check boxes are selected for all items under the `diode` device except for `vrd`, as shown in the figure below.
3. Highlight the device name, **diode**, as shown in the figure below.

Figure 69: 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 70: 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

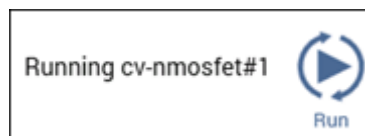
NOTE

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.

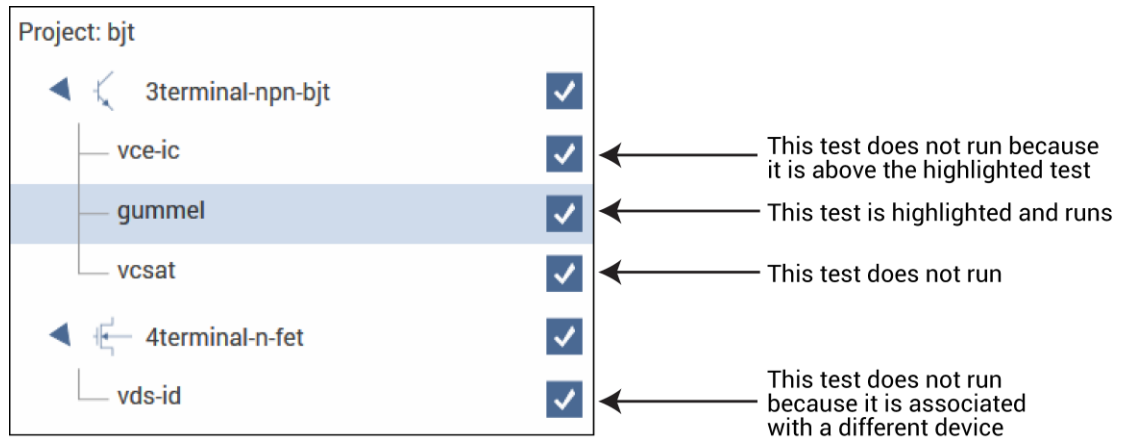
Figure 71: Run icon while a test is running



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 72: Run specific tests



NOTE

You can also use Monitor to run an individual test. For more information, see [Monitor a test](#) (on page 2-82).

Monitor a test

You can use the Monitor option to set a test to run continuously until stopped. This option is available for ITMs and UTMs when an individual test is selected and when the Monitor option is enabled.

Enable the Monitor option

The Monitor option is enabled in My Settings.

To enable the Monitor option:

1. Select **My Settings**.
2. Select **Run Settings**.
3. Select **Enable Monitor button**.
4. Select **OK**. Monitor is now available to the left of Run, as shown in the following figure.

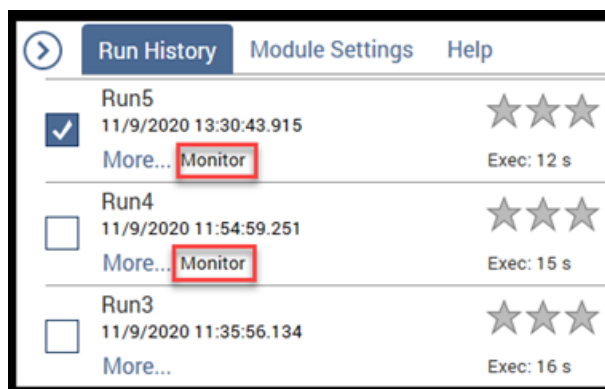
Figure 73: Monitor option



Running a test using Monitor

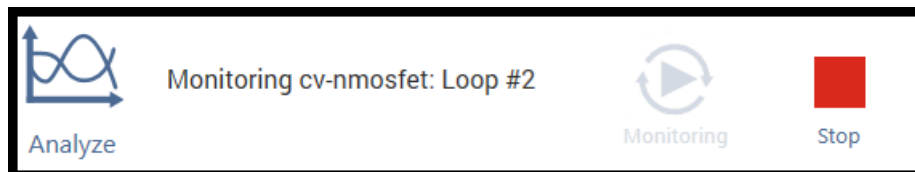
When a test is run using Monitor, the test loops until you stop it. Only one run is stored in Run History. The stored run contains the data that was acquired during the last complete test run. The time that is shown for the run is the test of the last completed test run, not the entire time that Monitor was running. Runs that were generated with Monitor include "Monitor" in the description in the Run History list, as shown in the following figure.

Figure 74: Run History when test is run using Monitor



To use Monitor to run a test:

1. In the project tree, make sure the check box for the test is selected.
2. Highlight the test.
3. Select **Monitor**. The Monitor icon changes as shown below. The active test is listed to the left of Monitor, followed by the number of the loop that is running. The Stop icon changes to red.

Figure 75: Monitor when a test is running

4. To stop the test, select **Stop**.

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 76: 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). You can also use four SMUs, one for each device-under-test (DUT) terminal.

Descriptions of the 4-terminal n-MOSFET tests

Test	Description
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

Test	Description
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

Test	Description
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.
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 four pulses. The Formulator multiplies the current of CH2 by -1.

Diode tests

By default, the following tests use up to two source-measure units (SMUs), a CVU, and a PMU.

Descriptions of the diode tests

Test	Description
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.
vrđ	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

By default, the following tests use up to two source-measure units (SMUs), one CVU, and one PMU.

Descriptions of the capacitor tests

Test	Description
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.

Analyze data

In this section:

Introduction	3-1
Spreadsheet	3-3
Analyze test data at the project level.....	3-10
Run History	3-12
Subsite cycling Analyze sheets	3-16
Save test results and graphs.....	3-25
Graph	3-26
Calc worksheet function definitions	3-60

Introduction

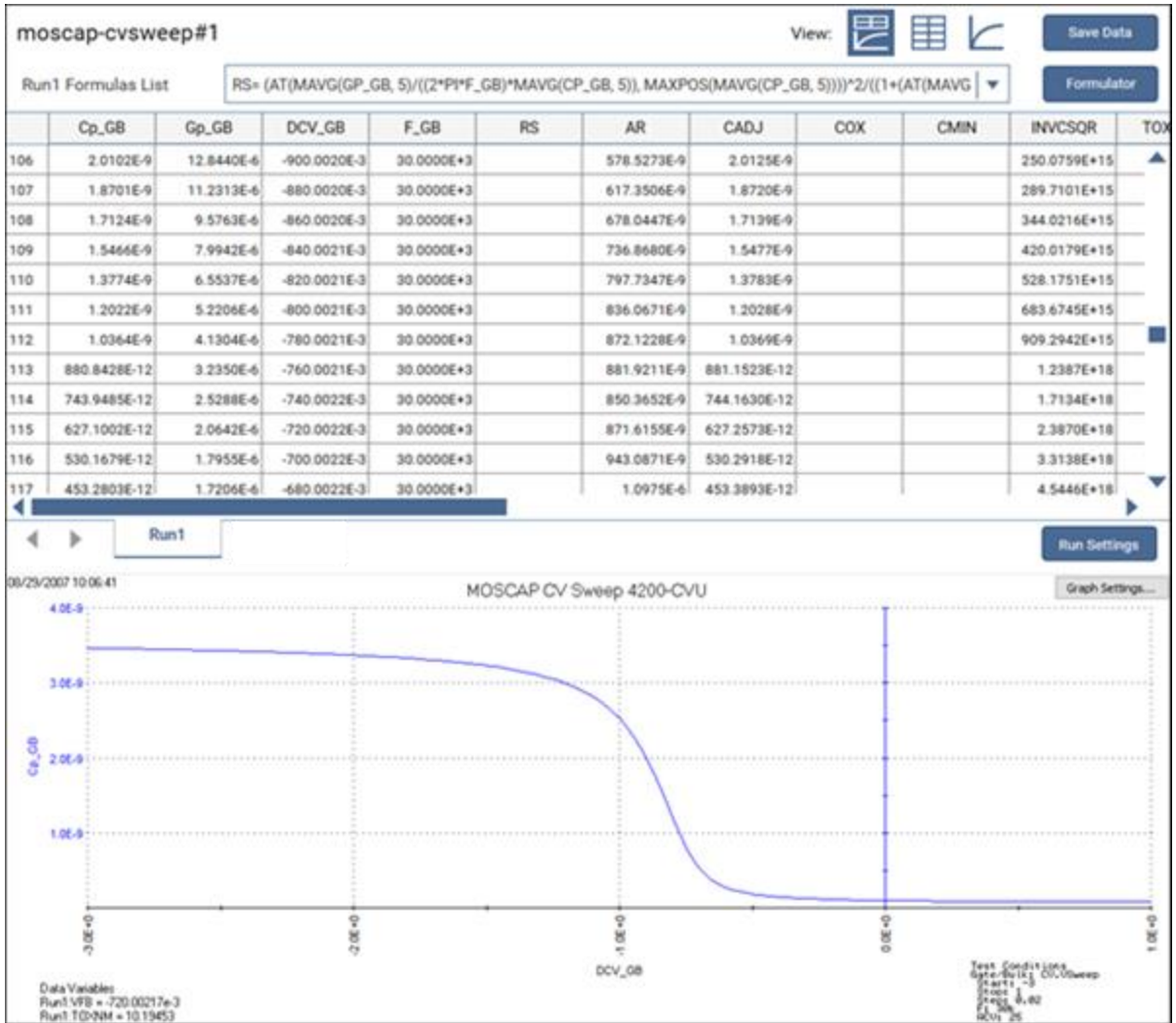
When you run tests for projects or individual subsites, devices, and tests, Clarius records the data in the Analyze pane. You can display the data in a spreadsheet and as a graph. You can also select data for specific tests to display at the project level. Select **Analyze** to view the spreadsheet and 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 5-1) for more information.

Figure 77: Analyze



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 78: Analyze Run sheet

moscap-cvsweep#1

View:

Save Data

Run1 Formulas List | | 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 | 271.2896E-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 | |

Run1

Run Settings

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.

The maximum number of columns in a sheet is 256. An error is generated if a test results in more than 256 columns.

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 3-25).

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 79: Run Settings example

The screenshot shows a 'Test Settings' dialog box with a table of parameters. The table has 6 columns and 23 rows. The parameters listed are:

	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				

At the bottom of the dialog box, there is a 'Run1' button and a 'Close' button.

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.

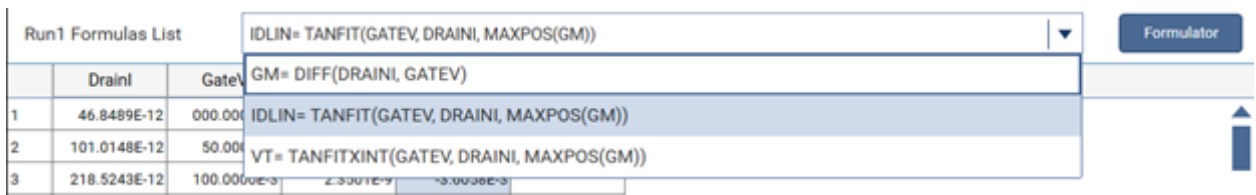
The #REF notation in a cell indicates that the Formulator could not calculate a valid value. This can occur if a Formulator function needs multiple rows as arguments, if a calculated value is out of range, or if a divide by zero is attempted.

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 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 **RUN Formulas List**. 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 80: 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.

Module Settings pane (Analyze)

When Analyze is selected for a test that is based on a user module, the Module Settings pane is available. The options in this pane are the same as the options in Configure for the module. Changes made in the Module Settings pane are also changed in Configure.

Descriptions of the options are available in the Help pane for most tests.

Measurement status

Many tests provide status information for the measurements in the Analyze pane of Clarius. For the 4210-CVU or 4215-CVU, the data column for the 32-bit status codes is labeled CVU1S. CVU status code indicates the current 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:

Color	Meaning
Red	Measurement timeout
Magenta	Measurement overflow
Orange	Auto Balance Bridge (ABB) not locked

Figure 81: 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 CVU measurement status are listed in following table. Each code is represented as a 32-bit hexadecimal value (0x).

CVU measurement status codes (CVU1S)

#	Code	Description
0	000000yy	No faults
1	8xxxxxyy	Measurement timeout occurred
2	01xxxxyy	CVH1 current measurement overflow
3	02xxxxyy	CVH1 voltage measurement overflow
4	03xxxxyy	CVH1 current and voltage measurement overflow
5	08xxxxyy	CVH1 ABB not locked
6	09xxxxyy	CVH1 ABB not locked, current measurement overflow
7	0Axxxxyy	CVH1 ABB not locked, voltage measurement overflow
8	0Bxxxxyy	CVH1 ABB not locked, current and voltage measurement overflow
9	xx01xyy	CVL1 current measurement overflow
10	xx02xyy	CVL1 voltage measurement overflow
11	xx03xyy	CVL1 current and voltage measurement overflow
12	xx08xyy	CVL1 ABB not locked
13	xx09xyy	CVL1 ABB not locked, current measurement overflow
14	xx0Axyy	CVL1 ABB not locked, voltage measurement overflow
15	xx0Bxyy	CVL1 ABB not locked, current and voltage measurement overflow

The yy value indicates the range used, as shown in the following table.

yy value	00	Lowest range (1 μ A) used for the impedance measurement
	01	Middle range (30 μ A) 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 [CVU Confidence Check](#) (on page 4-28) 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 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 to be reset.
4. Click **Reset**.

Current measurement overflow:

- Try a higher current measure range (or Auto).
- Try a lower ac drive voltage.

Voltage 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.

ABB unbalance errors

The CVU uses the autobalancing bridge (ABB) technique to achieve accurate impedance measurements. 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.

The most common reasons that 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 CVU Confidence Check to help troubleshoot ABB errors. Refer to [Run an open check and short check](#) (on page 4-29) for instructions on performing a confidence check.

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 5-1) for information on creating formulas.

You can also graph data. Refer to [Graph](#) (on page 3-26) 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

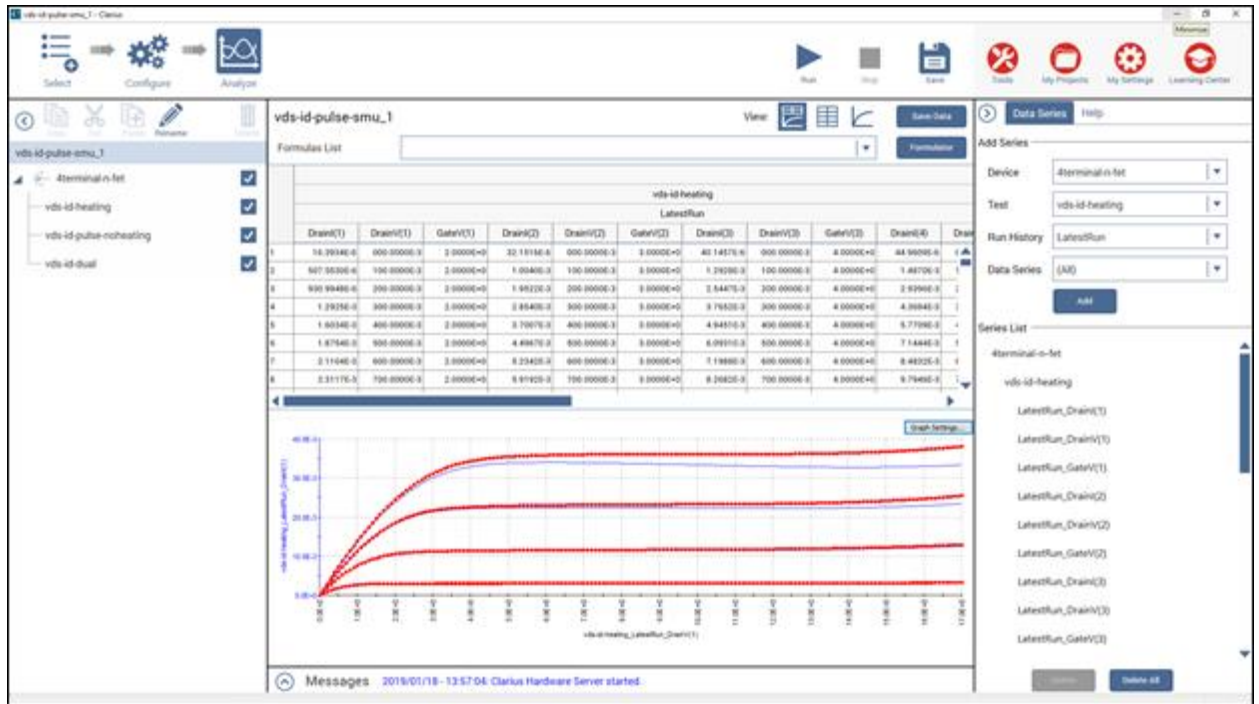
You can select up to 255 columns of data to display in the project-level Analyze pane.

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 82: Analyze 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, select Delete All in the right pane.

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**.

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, Run History contains sample data and a sample graph is displayed 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 are displayed in the Analyze pane.

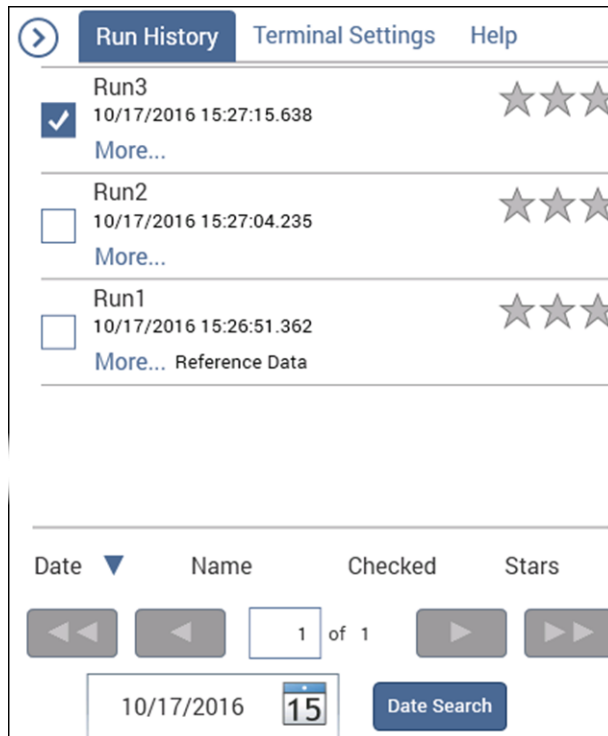
You can have up to 10,000 runs for each test.

When you run stress tests, only data from the latest loop is saved to the Run History data file in order to optimize disk space usage.

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.
- The indicator "Monitor" if the data was generated using the Monitor option.

Figure 83: 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 sheet.
- 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 are displayed in the Analyze sheet and graph. Select a different Run History to view that information in the Analyze sheet and graph.

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 3-30) 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.

Setting the number of run histories

You can set the number of runs that are stored and displayed in the Run History pane.

When you run a test, the number of runs stored in the Run History is limited to the number that is set. To maintain the number of runs, Clarius deletes the oldest unselected run. If there is existing data that exceeds the limit, only the oldest unselected run is deleted. If you have all runs selected, the most recent run is deleted when a new run is created.

For example, if you have 200 existing runs and change the maximum run history size to 5, the number of runs in the Run History remains at 200. The next test run triggers the deletion of the oldest unselected run.

If you are running a project with subsite cycling or stressing, all existing runs are automatically unselected at the start of execution. Only data collected during the last run of the project is selected and displayed on the graphs. To save data between project runs, increase the run history size to be more than the number of cycles or stresses in your project.

If the run history size setting is less than the number of cycles or stresses in your project, a run is still generated for each cycle or stress.

To set the number of runs stored in Run History:

1. In Clarius, select **My Settings**.
2. Select **Run Settings**.
3. In the **Run History Size** field, enter the number of runs (1 to 10,000).
4. Select **OK**.

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:

$$\% \text{ Change} = \text{ABS}[(\text{Post-Stress Rdg} - \text{Pre-Stress Rdg}) / \text{Pre-Stress Rdg} \times 100]$$

For the example in the following figure, percent change I_{DOFF} for the second cycle is calculated as:

$$\begin{aligned} \% \text{ Change I}_{\text{DOFF}} &= \text{ABS}[(82.2013\text{e-}15 - 291.1666\text{e-}15) / 291.1666\text{e-}15 \times 100] \\ &= \text{ABS}[-208.9653\text{e-}15 / 291.1666\text{e-}15 \times 100] \\ &= \text{ABS}[-0.7176 \times 100] \\ &= 71.8 \end{aligned}$$

- 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.

Figure 84: Subsite Analyze sheet in Stress/Measure mode

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	vte:
	Cycle Index	Stress Time	id#1 IDOFF	% Change IDOFF	Target Value	id#1 IDLIN	% Change IDLIN	Target Value	id#1 IDSAT	% Change IDSAT	Target Value	igleak#1 IGLEAK	% Change IGLEAK	Target Value	VTE
1	1	0.00	308.1823E-15		0.000	269.9511E-6		0.000	4.6286E-3		0.000	-6.8007E-12		0.000	926
2	2	10.00	74.6224E-15	75.786		276.2232E-6	2.323		4.6160E-3	0.271		168.0886E-15	102.472		921
3	3	21.54	102.9612E-15	66.591		279.2826E-6	3.457		4.6100E-3	0.402		169.3108E-15	102.490		919
4	4	46.42	177.5246E-15	42.396		284.6862E-6	5.458		4.5991E-3	0.636		202.7564E-15	102.981		915
5	5	100.00	133.8737E-15	56.560		292.0504E-6	8.186		4.5843E-3	0.956		213.4304E-15	103.138		910
6	6	215.44	171.1995E-15	44.449		297.4943E-6	10.203		4.5738E-3	1.183		238.4493E-15	103.506		906
7	7	464.16	188.5646E-15	38.814		299.3781E-6	10.901		4.5711E-3	1.241		235.6590E-15	103.465		905
8	8	1000.00	75.7835E-15	75.409		299.9014E-6	11.095		4.5715E-3	1.233		231.4591E-15	103.403		904
9	9	2154.43	131.7894E-15	57.237		299.8906E-6	11.091		4.5733E-3	1.195		202.8820E-15	102.983		904
10	10	4641.59	96.0319E-15	68.839		301.8061E-6	11.800		4.5714E-3	1.235		162.6893E-15	102.392		903
11	11	10000.00	45.7663E-15	85.150		302.9184E-6	12.212		4.5712E-3	1.239		104.9323E-15	101.543		902
12	12	20000.00	96.5853E-15	68.660		302.4817E-6	12.051		4.5738E-3	1.184		59.8365E-15	100.880		902

If Measure Current or Measure Voltage is selected in the Terminal Settings pane, the 4200A-SCS takes one measurement per terminal before each stress cycle begins and returns that one measurement to the subsite sheet. It takes one current measurement if the terminal is set to dc voltage stressing or one voltage measurement if the terminal is set to current stressing. The column name that contains the measurement includes the name of the device, the name of the terminal, and a measurement ID (I or V).

Stress/measure mode Analyze graph

The graphs for the stress/measure mode plot degradation (in %) versus the stress times. Each point in the graph represents the device degradation (% Change) for tests after each stress cycle (stress time).

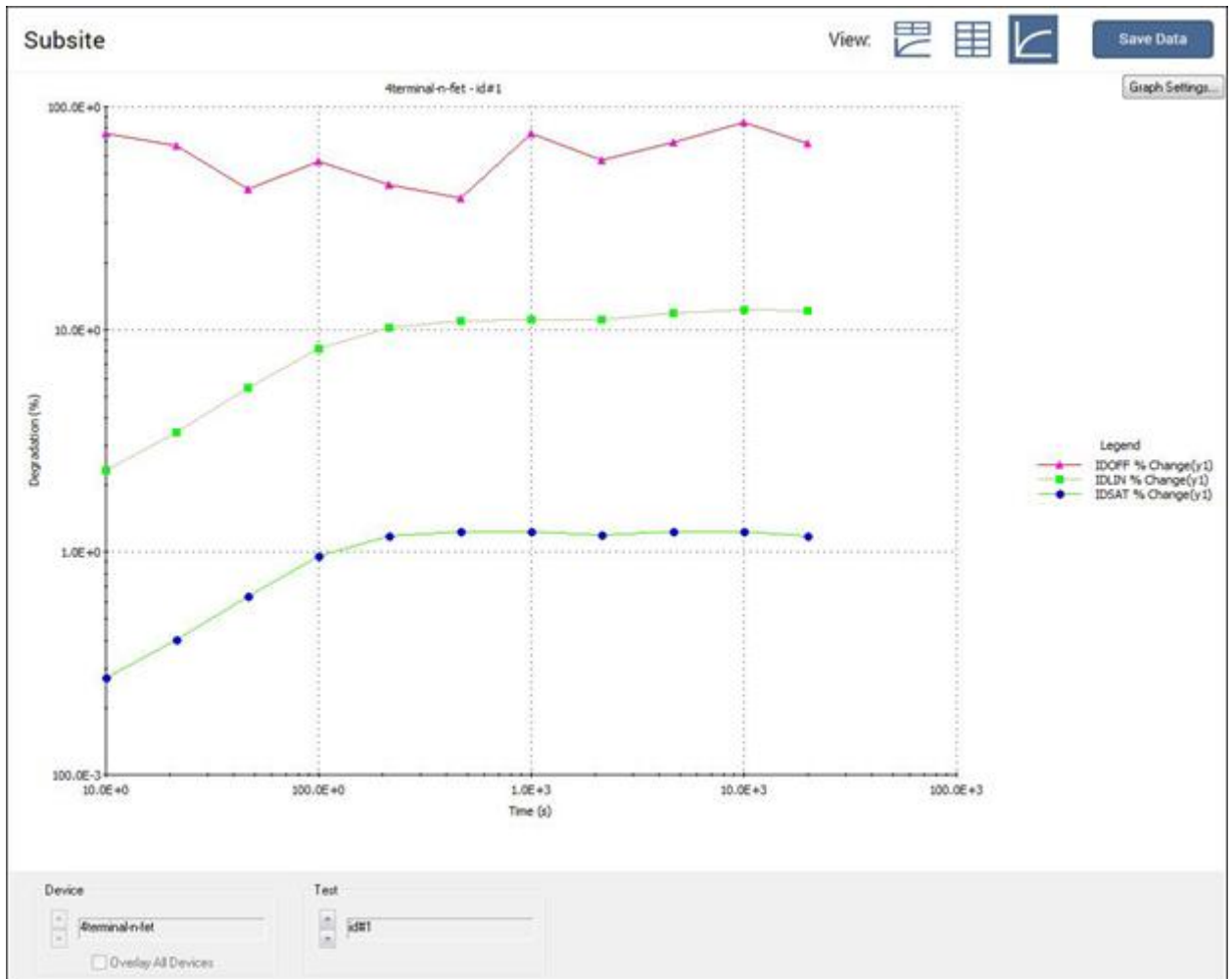
The graph below traces for test `id#1` for the `4terminal-n-fet` device. The three traces are for Output Values `IDOFF`, `IDLIN`, and `IDSAT`.

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 85: 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 86: Analyze Subsite Settings sheet for Cycle mode

	A	B	C	D
1	Subsite Name	HCI		
2	Site Number	1		
3	Cycle Mode	Cycle Mode		
4				
5	Total Cycles	13		
6	Last Executed	08/18/2016 15:49:33		
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

4terminal-n-fet 1 4terminal-n-fet Calc Settings

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 87: 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									

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 3-27) 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 3-60).

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 `=TabNameX!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**.

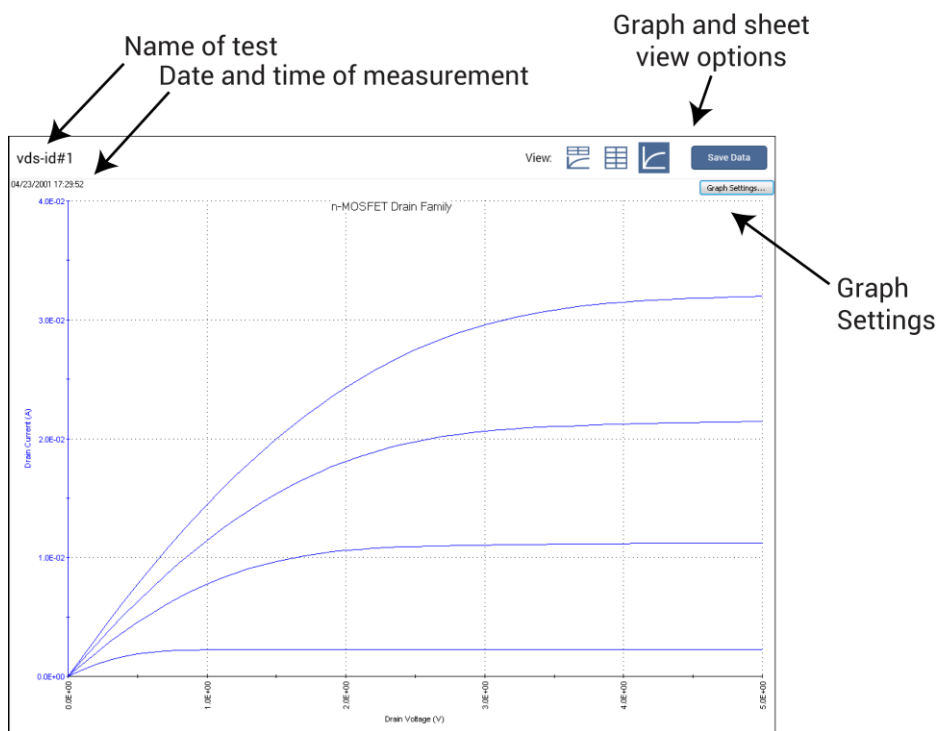
Graph

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 88: Example of an Analyze graph

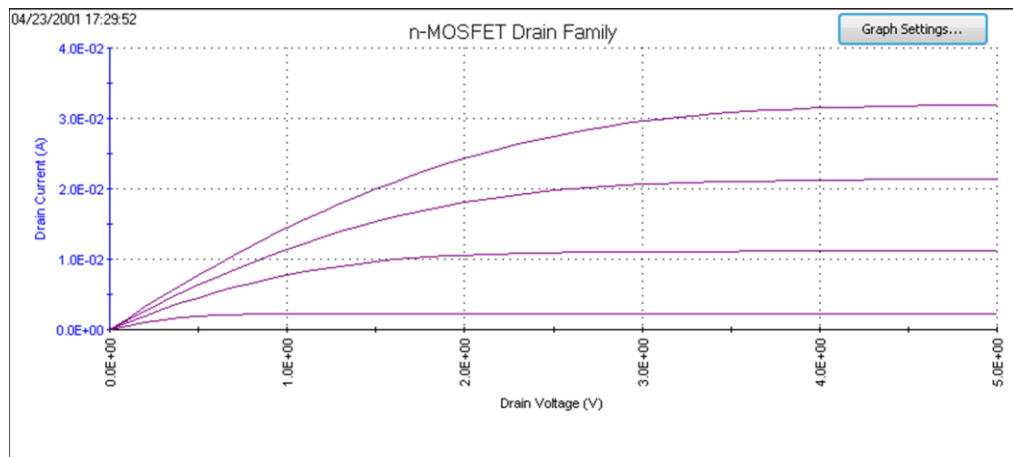


Open a graph

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 89: 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.

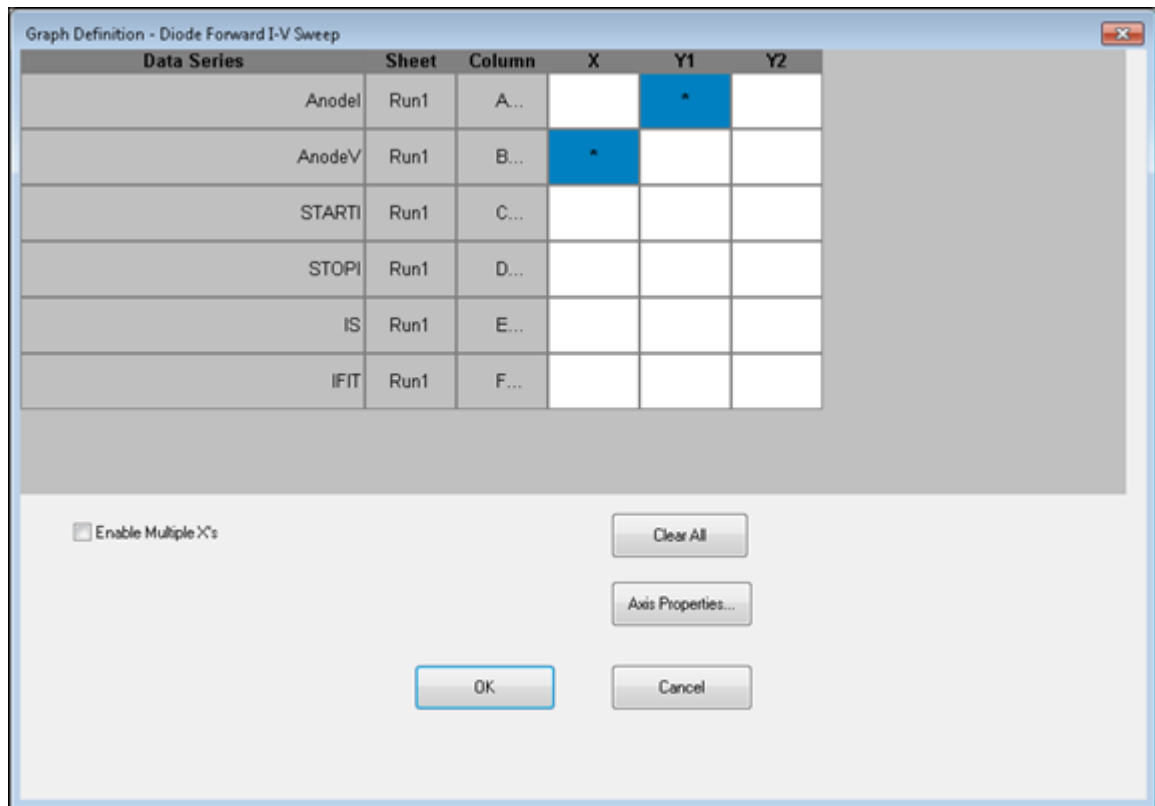
The Sheet column shows you whether the data came from the Run tab or the Calc tab.

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 3-31) for information.

To define a graph:

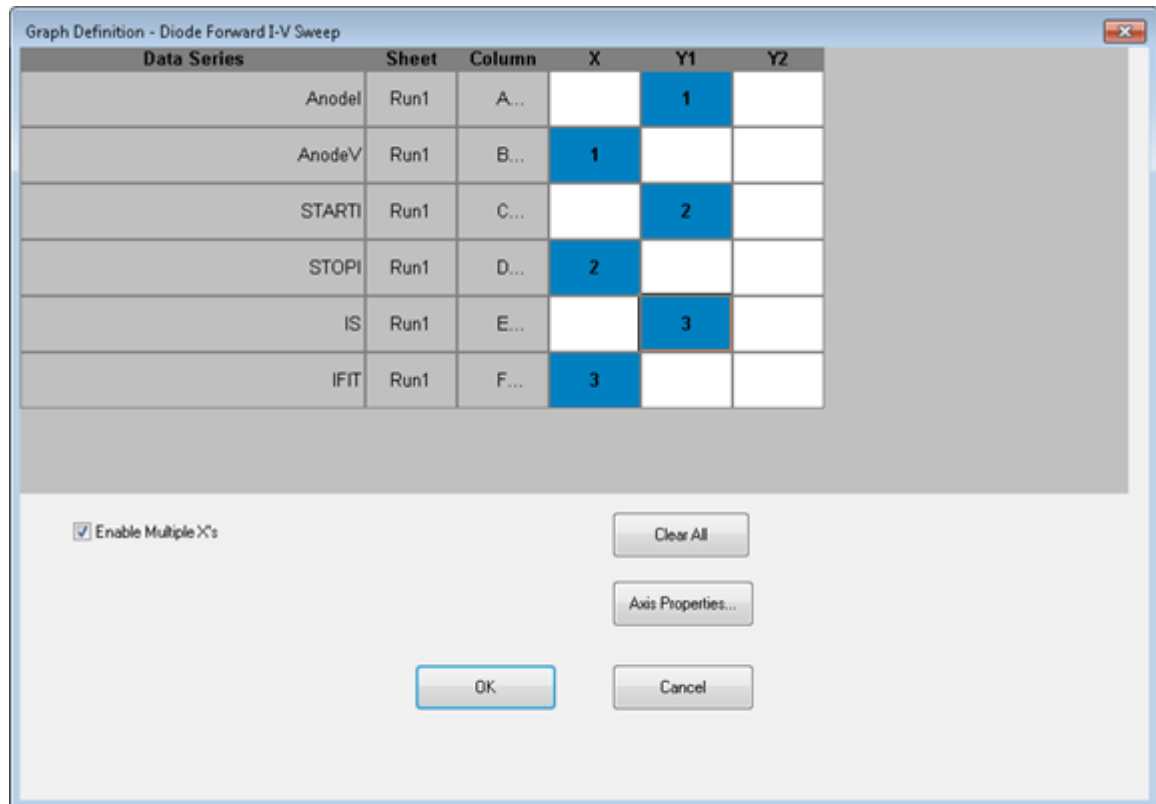
1. Select **Graph Settings**.
2. Select **Define Graph**. An example of the Graph Definition dialog box is shown in the following figure.

Figure 90: 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 91: 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 point on any graph using the mouse or other pointing device, Clarius displays the following information about the point:

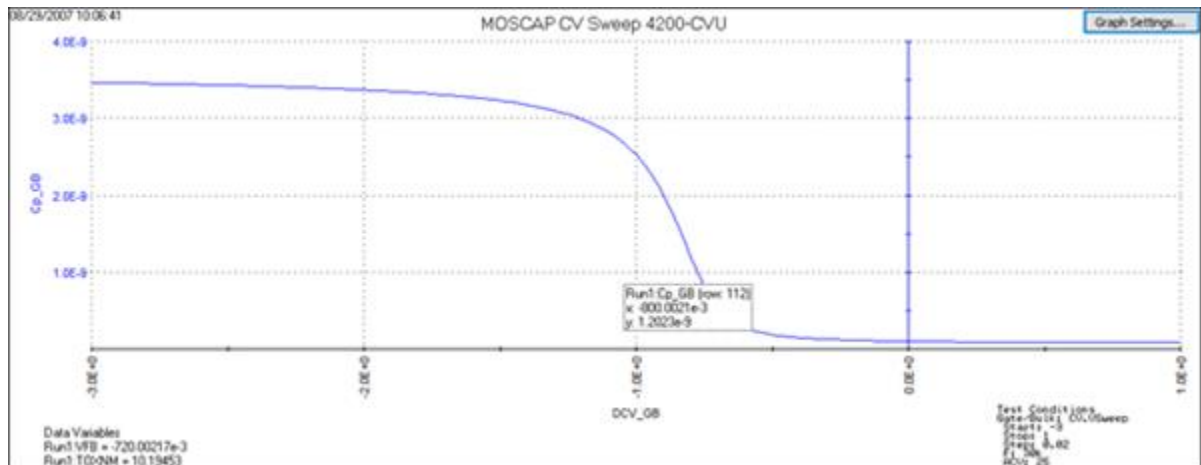
- Data series
- Run sheet row number
- Coordinates to four decimal places

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 point.
2. Move the cursor along the plot line until it is over the point. The cursor changes to the pointer cursor and the coordinates are displayed, as shown in the following figure.

Figure 92: 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 3-55) 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.

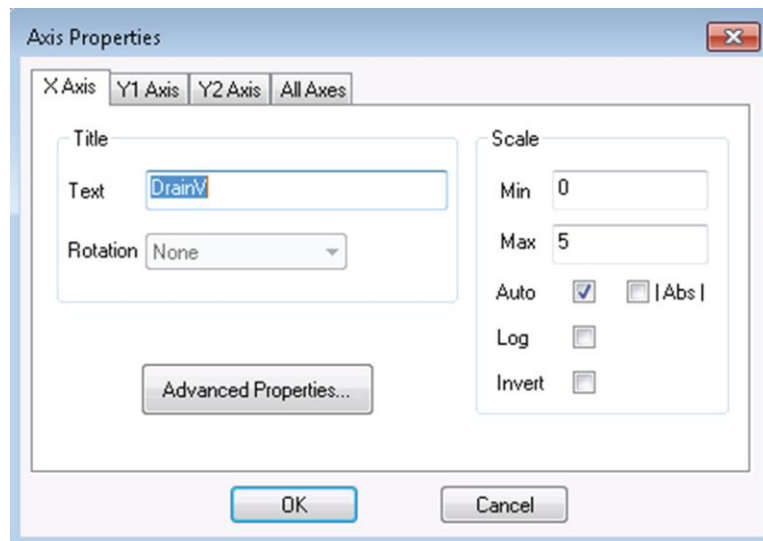
To change how axes are scaled, refer to [Define the axis properties](#) (on page 3-31).

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 93: 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: <ul style="list-style-type: none"> ▪ 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: <ul style="list-style-type: none"> ▪ 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 placed 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.

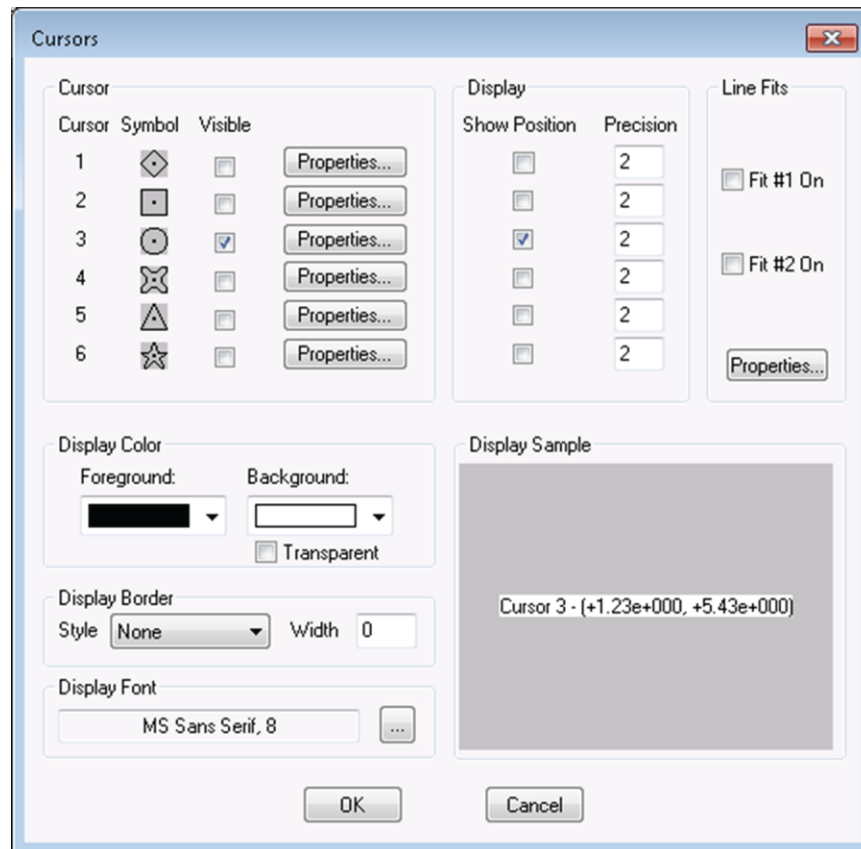
Option	Description
Run Autoscale	This setting affects any axes that are set to autoscale. Select: <ul style="list-style-type: none">▪ Real-Time: Autoscaling occurs as data is acquired. This option applies to all raw-data parameters and some calculated parameters.▪ End of Test: Autoscaling occurs at completion 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.

Configure cursors

To display cursors:

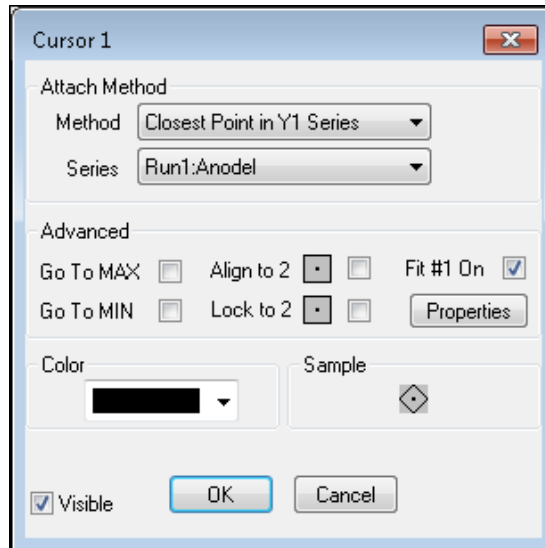
1. Select **Analyze**.
2. On the graph, right-click and select **Cursors**. The dialog box shown below opens.

Figure 94: 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 95: 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 point of the plot to which the cursor is attached.
- **Go To MIN:** Places the cursor at the minimum-Y 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 3-38).

Numerically displaying plot coordinates using cursors

You can display the precise numerical coordinates of 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.

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.

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 point at which a tangent line is fitted to the plot or the starting and ending points (data range).
- The data-point coordinates. Tangent or starting and ending 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 following table.

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 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 96: Linear fit example

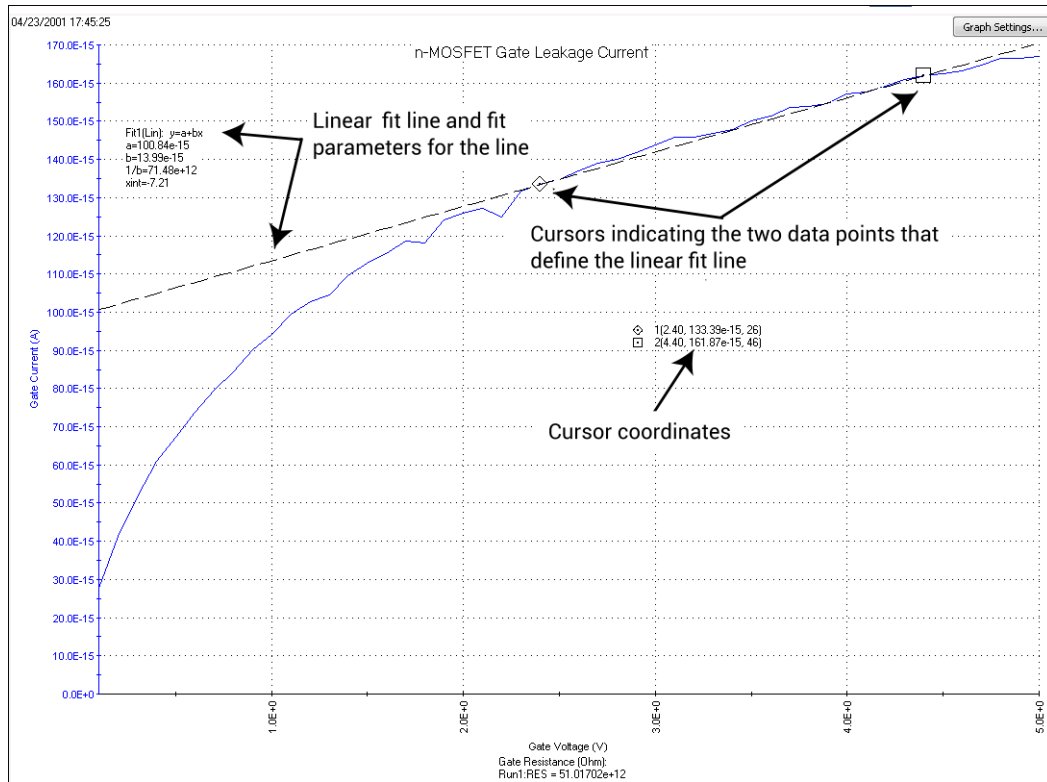
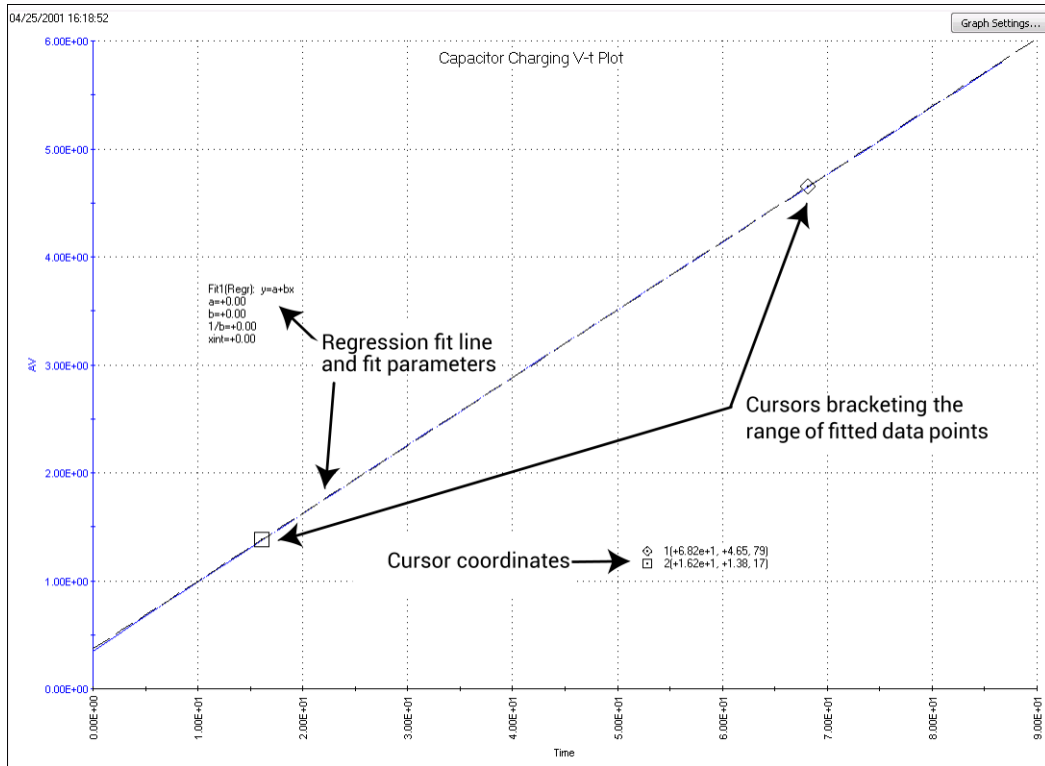


Figure 97: Regression fit example



Perform line fits

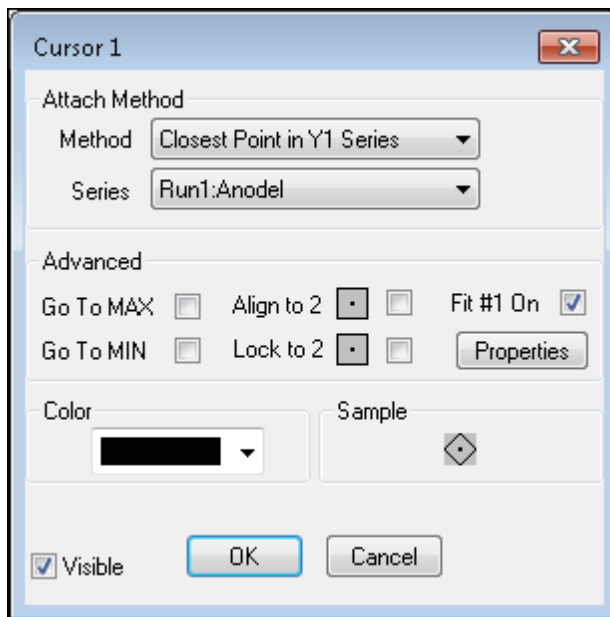
Plots of the fit lines appear as dashed lines, and fit parameter and cursor coordinate displays indicate appropriate numerical values.

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.

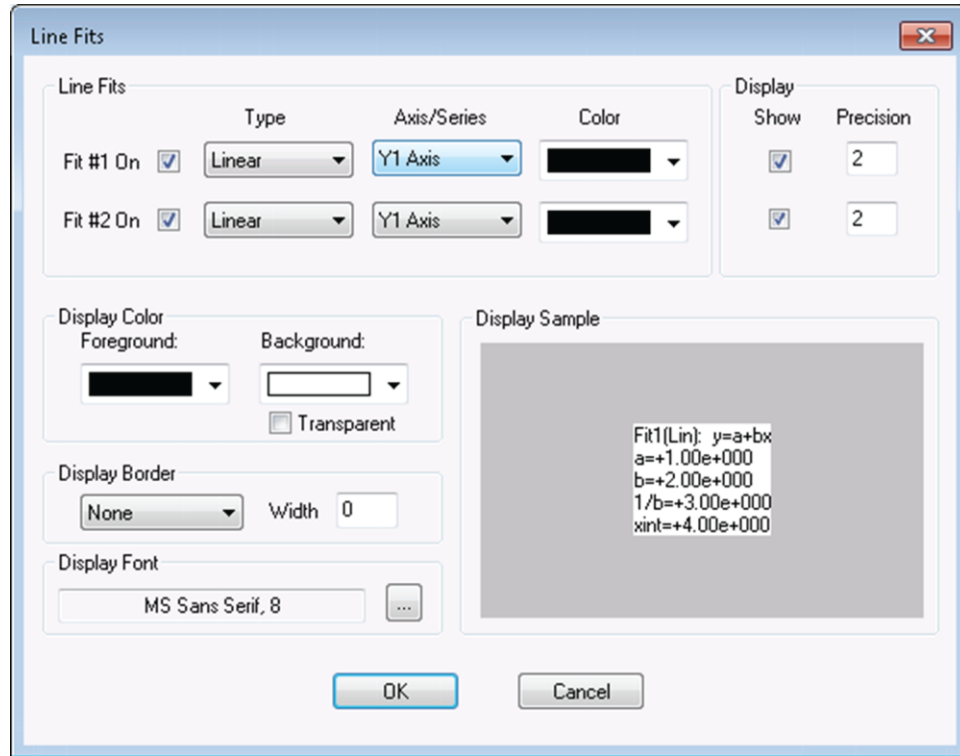
To initiate a line fit:

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 98: 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 99: 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 3-37) and [Advanced cursor options](#). (on page 3-37)

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.

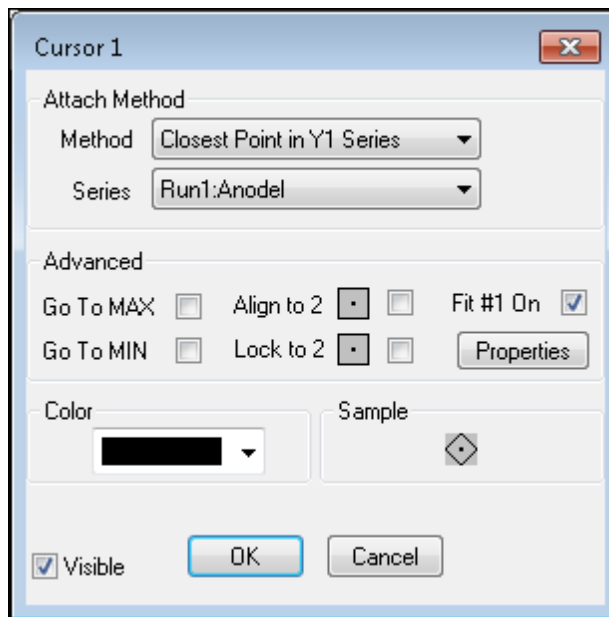
Option	Description
Fit # On	Enable or disable Fit #1 or Fit #2.
Type	The type of line fit to apply: <ul style="list-style-type: none"> ▪ Linear: Chord line of the form $y = a + bx$, drawn between two graphically defined points. ▪ Regression: Regression line of the form $y = a + bx$ for a graphically defined range of points. ▪ Exponential: Regression line of the form $y = a \cdot e^{bx}$ for a graphically defined range of points. ▪ Log: Regression line of the form $y = a + b \cdot \log_{10}(x)$ for a graphically defined range of points. ▪ Tangent: Tangent to the plot at a graphically defined 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 are 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 100: Cursor # dialog box



2. Select **Fit # On**.
3. Select **Properties** under Fit # On.
4. Refer to [Perform line fits](#) (on page 3-40) for information on the options.

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 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.

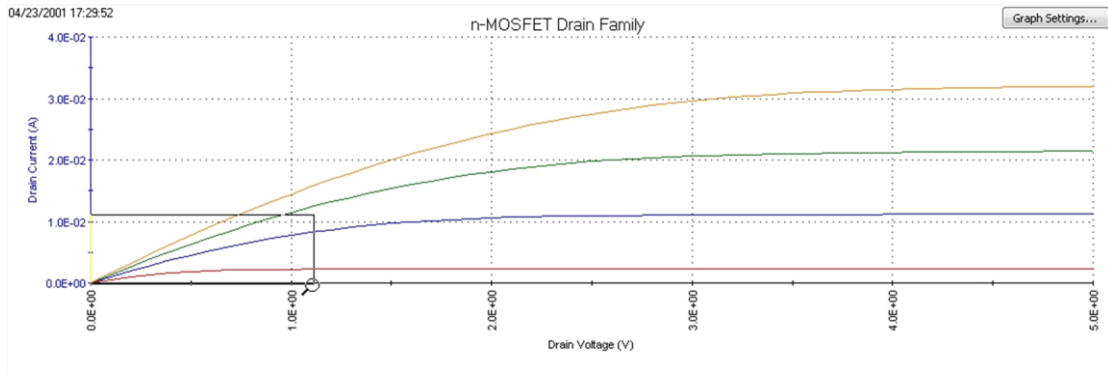
If you hold the Ctrl or Alt and Arrow key for more than a second, the cursor moves more quickly. The cursor moves one pixel at a time in normal mode and five pixels at a time in fast mode.

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.

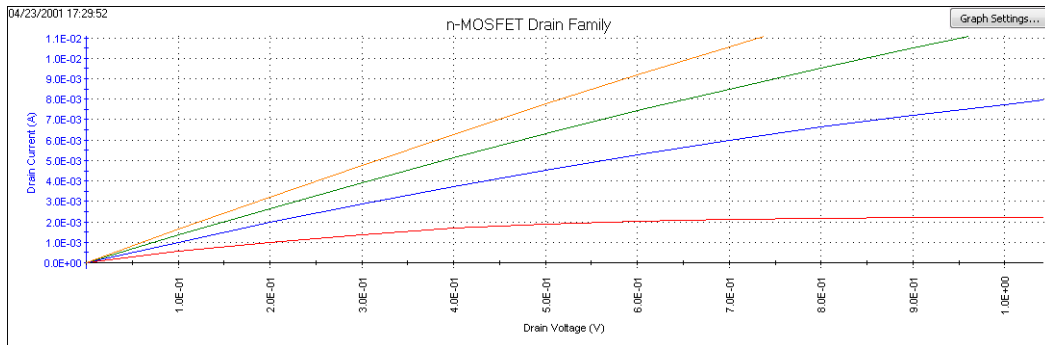
Figure 101: Zoom in on an area of the graph



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 102: 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.

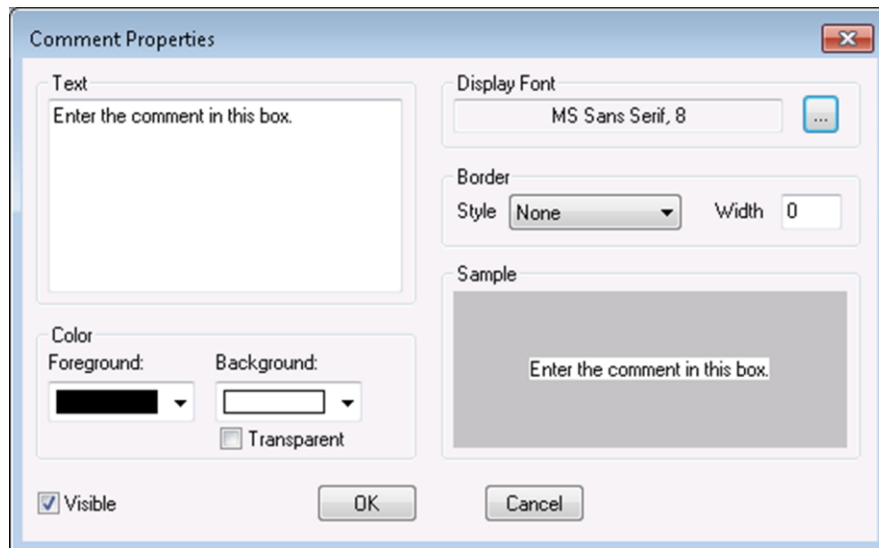
Add a comment

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 103: Graph Comment Properties dialog box



3. In the Text box, enter the comment.
4. Change the appearance of the comment as needed. See the following table 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 following table 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.

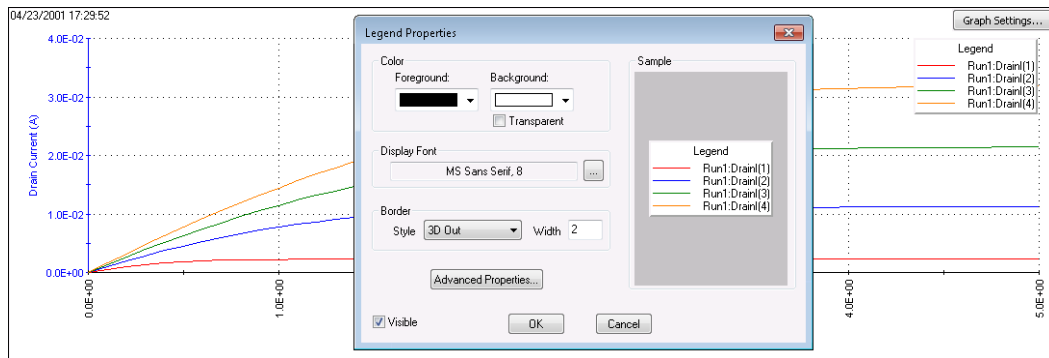
Add a legend

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 104: Graph legend and Legend Properties dialog box



3. Change the appearance of the legend as needed. See the following table 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.

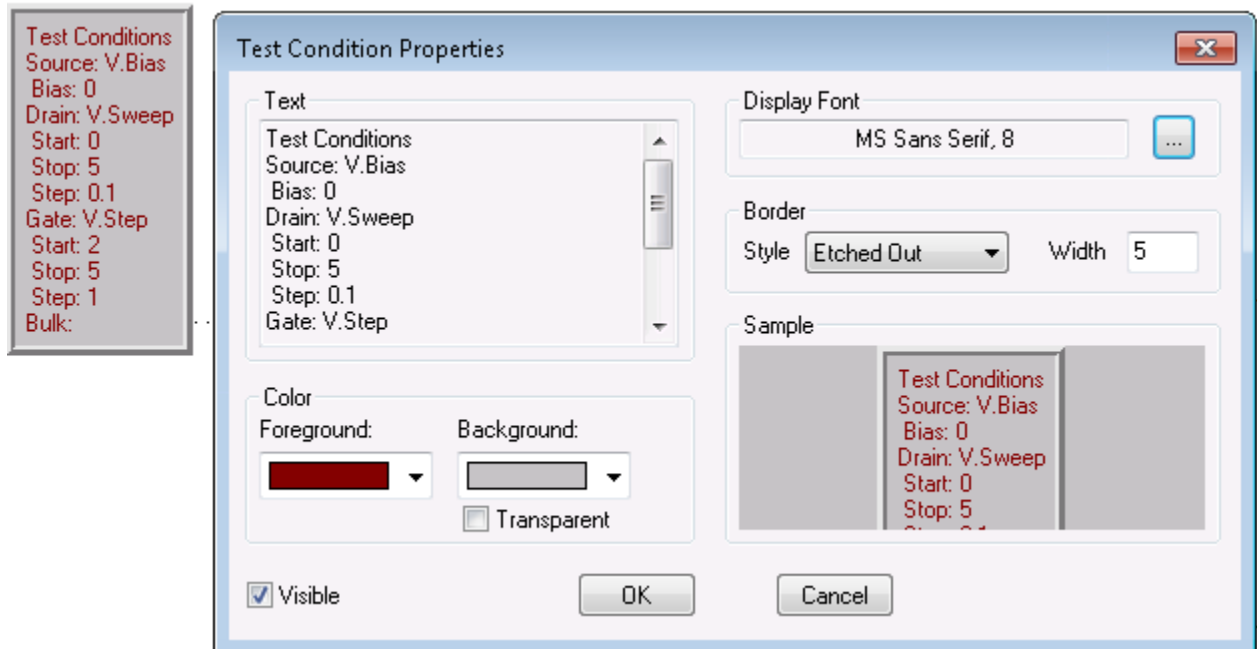
The following table lists the test conditions that are displayed. For each terminal of the DUT, the Test Conditions item displays the name, the applied operation mode, and the corresponding test conditions.

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 105: Example of test conditions and test condition properties

3. Change the appearance of the test conditions as needed. See the following table 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.

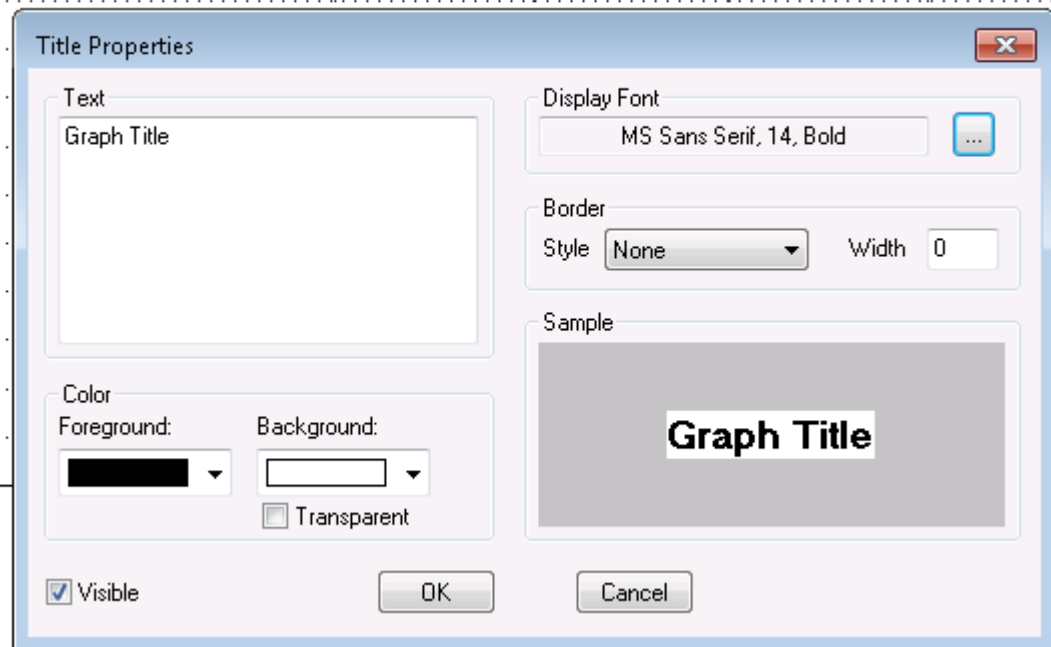
Add or update a title

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 106: Example of graph title and properties

Graph Title



2. Change the appearance of the title as needed. See the following table 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 107: Data Series Properties dialog box



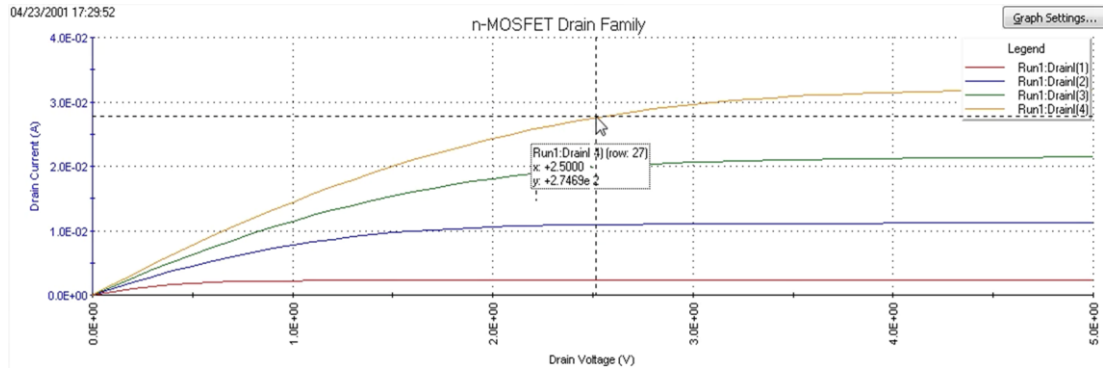
The options for the series are described in the following table.

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 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 point.

Figure 108: 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.

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:

On the graph, right-click and select **Reset**.

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.

CVU Data Type

Only available for measurements that were made with the 4210-CVU or 4215-CVU. Options include:

- **Z-THETA:** Impedance and phase angle
- **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

Cycle mode graphs

The graphs for the Cycle Mode plot output values versus the cycle index. Each 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.

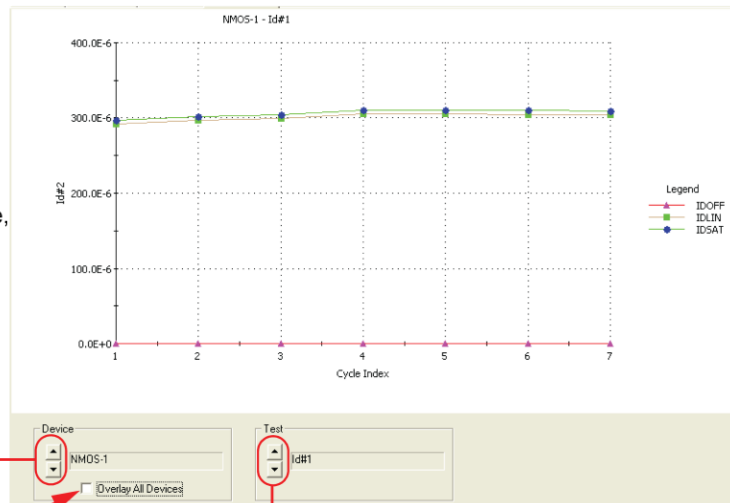
Figure 109: Subsite Analyze graph: Cycle mode

NOTE:

For a single-device subsite, the Device select buttons and the checkbox to Overlay All Devices are disabled.

For a single-test subsite, the test select buttons are disabled.

1) Use to select device



Click to display all the graph traces for all devices that were measured by the selected test.

2) Use to select the test

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)`

<i>Value</i>	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 3-79)

ACOS Calc worksheet function

This command returns the arc cosine of a value.

Usage

ACOS (*Value*)

<i>Value</i>	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 $180/\text{PI}()$.

Example

=ACOS (0.5)	Returns 1.0471E+0.
=ACOS (-0.2)	Returns 1.7721E+0.

Also see

[COS](#) (on page 3-65)

ACOSH Calc worksheet function

This command returns the inverse hyperbolic cosine of a value.

Usage

ACOSH (*Value*)

<i>Value</i>	Any number equal to or greater than 1
--------------	---------------------------------------

Example

=ACOSH (1.2)	Returns 622.3625E-3.
=ACOSH (3)	Returns 1.7627E+0.

Also see

[ASINH](#) (on page 3-62)

[ATAN](#) (on page 3-63)

[COSH](#) (on page 3-66)

ASIN Calc worksheet function

This command returns the arcsine of a value.

Usage

`ASIN (Value)`

<i>Value</i>	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 $180/\text{PI}()$.

Example

<code>=ASIN (1)</code>	Returns 1.5707E+0.
<code>=ASIN (0.4)</code>	Returns 411.5168E-3

Also see

[ASINH](#) (on page 3-62)

[PI](#) (on page 3-77)

[SIN](#) (on page 3-80)

ASINH Calc worksheet function

This command returns the inverse hyperbolic sine of a value.

Usage

`ASINH (Value)`

<i>Value</i>	Any number
--------------	------------

Example

<code>=ASINH (5.3)</code>	Returns 2.3696E+0.
<code>=ASINH (-4)</code>	Returns -2.0947E+0.

Also see

[ACOSH](#) (on page 3-61)

[ASIN](#) (on page 3-62)

[ATANH](#) (on page 3-64)

[SINH](#) (on page 3-80)

ATAN Calc worksheet function

This command returns the arctangent of a number.

Usage

ATAN (*Value*)

<i>Value</i>	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 $180/\text{PI}()$.

Example

<code>=ATAN (3.5)</code>	Returns 1.2924E+0.
<code>=ATAN (4)</code>	Returns -1.3258E+0.

Also see

[ATAN2](#) (on page 3-64)

[ATANH](#) (on page 3-64)

[PI](#) (on page 3-77)

[TAN](#) (on page 3-83)

ATAN2 Calc worksheet function

This command returns the arctangent of specified coordinates.

Usage

ATAN2 (*x*, *y*)

<i>x</i>	The x coordinate
<i>y</i>	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 3-63)
[ATANH](#) (on page 3-64)
[PI](#) (on page 3-77)
[TAN](#) (on page 3-83)

ATANH Calc worksheet function

This command returns the inverse hyperbolic tangent of a number.

Usage

ATANH (*Value*)

<i>Value</i>	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 3-61)
[ASINH](#) (on page 3-62)
[TANH](#) (on page 3-83)

AVERAGE Calc worksheet function

This command returns the average of the supplied numbers.

Usage

`AVERAGE (Value_list)`

<code>Value_list</code>	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

<code>=AVERAGE (5, 6, 8, 14)</code> <code>=AVERAGE (C15:C17)</code>	Returns 8.2500E+0. <code>AVERAGE (C15:C17)</code> returns the average of the values in cells C15 through C17 of the Calc tab.
--	--

Also see

[MAX](#) (on page 3-73)

[MIN](#) (on page 3-74)

COS Calc worksheet function

This command returns the cosine of an angle.

Usage

`COS (Value)`

<code>Value</code>	The angle in radians
--------------------	----------------------

Details

If the angle is in degrees, convert the angle to radians by multiplying it by $\text{PI}() / 180$.

Example

<code>=COS (1.4444)</code> <code>=COS (5)</code>	Returns 126.0600E-3. Returns 283.6622E-3.
---	--

Also see

[ACOS](#) (on page 3-61)

[ASINH](#) (on page 3-62)

[ATANH](#) (on page 3-64)

[COSH](#) (on page 3-66)

[PI](#) (on page 3-77)

COSH Calc worksheet function

This command returns the hyperbolic cosine of an angle.

Usage

`COSH (Value)`

Value

Any value

Example

`=COSH (2.10)`

Returns 4.1443E+0.

`=COSH (0.24)`

Returns 1.0289E+0.

Also see

[ASINH](#) (on page 3-62)

[ATANH](#) (on page 3-64)

[COS](#) (on page 3-65)

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)`

Returns 13.0000E+0.

`=DAY ("7-21-2016")`

Returns 21.0000E+0.

`=DAY (NOW ())`

Returns the present day of the month.

Also see

[HOUR](#) (on page 3-68)

[MINUTE](#) (on page 3-75)

[MONTH](#) (on page 3-76)

[NOW](#) (on page 3-77)

[SECOND](#) (on page 3-79)

[YEAR](#) (on page 3-84)

EXP Calc worksheet function

This command returns the constant e raised to the specified power.

Usage

EXP(*Value*)

<i>Value</i>	Any number as the exponent
--------------	----------------------------

Details

The constant e is 2.71828182845904 (the base of the natural logarithm).

Example

=EXP(2.5)	Returns 12.1824E+0.
=EXP(3)	Returns 20.0855E+0.

Also see

[LN](#) (on page 3-69)

[LOG](#) (on page 3-69)

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*)

<i>Value</i>	Any number
<i>Precision</i>	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
<i>No_commas</i>	<i>No_commas</i> determines if commas separate thousands in the result; send 1 to exclude commas in the result; send 0 or do not define <i>No_commas</i> 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)	Returns 2,000.500.
=FIXED(2009.5, -1, 1)	Returns 2010.
=FIXED(2009.5, -1, 0)	Returns 2,010.

Also see

[ROUND](#) (on page 3-78)

HR 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)`

<i>Serial_number</i>	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

<code>=HOUR (34259.4)</code>	Returns 9.000E+0.
<code>=HOUR (34619.976)</code>	Returns 23.000E+0.
<code>=HOUR (NOW ())</code>	Returns the present hour of the present day.

Also see

[DAY](#) (on page 3-66)

[MINUTE](#) (on page 3-75)

[MONTH](#) (on page 3-76)

[NOW](#) (on page 3-77)

[SECOND](#) (on page 3-79)

[YEAR](#) (on page 3-84)

IF Calc worksheet function

This command tests the condition and returns the specified value.

Usage

`IF (Condition, True_number, False_number)`

<i>Condition</i>	Any logical expression
<i>True_number</i>	The value to be returned if <i>Condition</i> evaluates to True
<i>False_number</i>	The value to be returned if <i>Condition</i> evaluates to False

Example

<code>=IF (A1>10, "Greater", "Less")</code>	Returns <code>Greater</code> if the content of A1 is greater than 10 and <code>Less</code> 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*)

<i>Value</i>	Any positive real number
--------------	--------------------------

Example

=LN (12.18)	Returns 2.4997E+0.
=LN (20.09)	Returns 3.0002E+0.

Also see

[EXP](#) (on page 3-67)

[LOG](#) (on page 3-69)

[LOG10](#) (on page 3-70)

LOG Calc worksheet function

This command returns the logarithm of a value to the specified base.

Usage

LOG (*Value*)

LOG (*Value*, *base*)

<i>Value</i>	Any positive real number
<i>base</i>	The base of the logarithm; if <i>base</i> is omitted, base 10 is assumed

Example

=LOG (1)	Returns 000.0000E-3.
=LOG (10)	Returns 1.0000E+0.
=LOG (8, 2)	Returns 3.0000E+0.

Also see

[EXP](#) (on page 3-67)

[LN](#) (on page 3-69)

[LOG10](#) (on page 3-70)

LOG10 Calc worksheet function

This command returns the base-10 logarithm of a value.

Usage

LOG10 (*Value*)

<i>Value</i>	Any positive real number
--------------	--------------------------

Example

=LOG10 (260)	Returns 2.4149E+0.
=LOG10 (100)	Returns 2.0000E+0.

Also see

[EXP](#) (on page 3-67)

[LN](#) (on page 3-69)

[LOG](#) (on page 3-69)

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*)

<i>Lookup_value</i>	The value for which to search in the first range
<i>Lookup_range</i>	The first range to search and contains only one row or one column; the range can contain numbers, text or logical values; to search <i>Lookup_range</i> , 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
<i>Result_range</i>	Range of one row or one column that is the same size as the <i>Lookup_range</i>

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 3-23)).

Figure 110: Example Calc worksheet cells

	A	B
1	DrainV(1)	Sourcel(1)
2	0	1.32744E-010
3	0.1000000015	-0.0008447049
4	0.2000000003	-0.0016400181
5	0.3000000119	-0.0023773371
6	0.4000000006	-0.0030588347
7	0.5	-0.003683852
8	0.6000000238	-0.0042509343
9	0.6000000001	0.0047640077

=LOOKUP (0.5, A2:A8, B2:B8)
 =LOOKUP (0.4, A2:A8, B2:B8)
 =LOOKUP (-0.1, A2:A8, B2:B8)

Returns -0.003683852.
 Returns -0.0023773371 (see **Details**).
 Returns #N/A (see **Details**).

Also see

[MATCH](#) (on page 3-72)

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)`

<i>Lookup_value</i>	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
<i>Lookup_range</i>	The range to search; contains only one row or one column; the range can contain numbers, text, or logical values
<i>Comparison</i>	A value representing type of comparison to be made between <i>Lookup_value</i> and the values in <i>Lookup_range</i> ; if you omit <i>Comparison</i> , 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 3-23)).

Figure 111: 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	0.6000000081	0.0047640077

<code>=MATCH(0.5, A2:A8, 1)</code>	Returns 6 (the sixth cell relative to cell 2, for example, cell 7).
<code>=MATCH(0.4, A2:A8, 1)</code>	Returns 4 (the fourth cell relative to cell 2, for example, cell 5).
<code>=MATCH(0.5, A2:A8, 0)</code>	Returns 6 (because an exact match is found).
<code>=MATCH(0.4, A2:A8, 0)</code>	Returns #N/A (because an exact match is not found).

Also see

[LOOKUP](#) (on page 3-71)

MAX Calc worksheet function

This command returns the largest value in the specified list of numbers.

Usage

`MAX(Value_list)`

<code>Value_list</code>	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

<code>=MAX(50, 100, 150, 500, 200)</code>	Returns 500.0000E+0.
<code>=MAX(A1:F12)</code>	Returns the largest value in this range.

Also see

[AVERAGE](#) (on page 3-65)

[MIN](#) (on page 3-74)

MIN Calc worksheet function

This command returns the smallest value in the specified list of numbers.

Usage

`MIN(Value_list)`

<code>Value_list</code>	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

<code>=MIN(50, 100, 150, 500, 200)</code> <code>=MIN(A1:F12)</code>	Returns 50.0000E+0. Returns the smallest value in this range.
--	--

Also see

[AVERAGE](#) (on page 3-65)

[MAX](#) (on page 3-73)

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)`

Returns 36.0000E+0.

`=MINUTE (34399.825)`

Returns 48.0000E+0.

`=MINUTE (NOW ())`

Returns the present minute of the present hour.

Also see

[DAY](#) (on page 3-66)

[HOUR](#) (on page 3-68)

[MONTH](#) (on page 3-76)

[NOW](#) (on page 3-77)

[SECOND](#) (on page 3-79)

[YEAR](#) (on page 3-84)

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)`

<code>Serial_number</code>	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

<code>=MONTH ("07-21-16")</code>	Returns 7.0000E+0.
<code>=MONTH (34626)</code>	Returns 10.0000E+0.
<code>=MONTH (NOW ())</code>	Returns the present month of the present year.

Also see

[DAY](#) (on page 3-66)
[HOUR](#) (on page 3-68)
[MINUTE](#) (on page 3-75)
[NOW](#) (on page 3-77)
[SECOND](#) (on page 3-79)
[YEAR](#) (on page 3-84)

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 3-66)

[HOUR](#) (on page 3-68)

[MINUTE](#) (on page 3-75)

[MONTH](#) (on page 3-76)

[SECOND](#) (on page 3-79)

[YEAR](#) (on page 3-84)

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 3-65)

[SIN](#) (on page 3-80)

[TAN](#) (on page 3-83)

PRODUCT Calc worksheet function

This command multiplies a list of numbers and returns the result.

Usage

`PRODUCT(Value_list)`

<i>Value_list</i>	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

<code>=PRODUCT(1, 2, 3, 4)</code>	Returns 24.0000E+0.
-----------------------------------	---------------------

Also see

[SUM](#) (on page 3-82)

ROUND Calc worksheet function

This command rounds the given number to the supplied number of decimal places.

Usage

`ROUND(Value, Precision)`

<i>Value</i>	Any number
<i>Precision</i>	The number of decimal places to which <i>Value</i> 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

<code>=ROUND(879.278, 2)</code>	Returns 879.2800E+0.
<code>=ROUND(9899.435, -2)</code>	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)`

<i>Serial_number</i>	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

<code>=SECOND (0.259)</code>	Returns 58.0000E+0.
<code>=SECOND (34657.904)</code>	Returns 46.0000E+0.
<code>=SECOND (NOW ())</code>	Returns the present second of the present minute.

Also see

[DAY](#) (on page 3-66)
[HOUR](#) (on page 3-68)
[MINUTE](#) (on page 3-75)
[MONTH](#) (on page 3-76)
[NOW](#) (on page 3-77)
[YEAR](#) (on page 3-84)

SIGN Calc worksheet function

This command determines the sign of a specified number.

Usage

`SIGN (Value)`

<i>Value</i>	Any number
--------------	------------

Details

`SIGN` returns 1 if the specified number is positive. It returns -1 if the specified number is negative.

Example

<code>=SIGN (-456)</code>	Returns -1.0000E+0.
<code>=SIGN (456)</code>	Returns 1.0000E+0.

Also see

[ABS](#) (on page 3-60)

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 $\text{PI}() / 180$.

Example

<code>=SIN (1.5)</code>	Returns 997.4950E-3.
<code>=SIN (4.8)</code>	Returns -996.1646E-3.

Also see

[ASIN](#) (on page 3-62)

[PI](#) (on page 3-77)

SINH Calc worksheet function

This command returns the hyperbolic sine of the specified number.

Usage

`SINH (Value)`

Value	Any number
-------	------------

Example

<code>=SINH (1)</code>	Returns 1.1752E+0.
<code>=SINH (3)</code>	Returns 10.0178E+0.

Also see

[ASINH](#) (on page 3-62)

[PI](#) (on page 3-77)

SQRT Calc worksheet function

This command returns the square root of the specified number.

Usage

`SQRT (Value)`

<i>Value</i>	Any positive number
--------------	---------------------

Details

If you specify a negative number, the error #NUM! is returned.

Example

<code>=SQRT (25)</code>	Returns 5.0000E+0.
<code>=SQRT (160)</code>	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)`

<i>Value_list</i>	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

<code>=STDEVP (4.0, 3.0, 3.0, 3.5, 2.5n 4.0, 3.5)</code>	Returns 5.0788E-3.
--	--------------------

Also see

[VARP](#) (on page 3-84)

SUM Calc worksheet function

This command returns the sum of the supplied numbers.

Usage

`SUM(Value_list)`

<code>Value_list</code>	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

<code>=SUM(1000, 3500, 500)</code> <code>=SUM(A10:D10)</code>	Returns 5.0000E+3. Returns 6.0000E+3 if each cell in the range contains 1500.
--	--

Also see

[AVERAGE](#) (on page 3-65)

[PRODUCT](#) (on page 3-78)

[SUMSQ](#) (on page 3-82)

SUMSQ Calc worksheet function

This command squares each of the supplied numbers and returns the sum of the squares.

Usage

`SUMSQ(Value_list)`

<code>Value_list</code>	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

<code>=SUMSQ(5, 9, 11)</code>	Returns 227.0000E+0.
-------------------------------	----------------------

Also see

[SUM](#) (on page 3-82)

TAN Calc worksheet function

This command returns the tangent of the specified angle.

Usage

TAN (*Value*)

<i>Value</i>	The angle in radians
--------------	----------------------

Details

If the angle is in degrees, convert the angle to radians by multiplying the angle by $\text{PI}() / 180$.

Example

<code>=TAN (1.5)</code>	Returns 14.1014E+0.
<code>=TAN (45*PI () /180)</code>	Returns 1.0000E+0.

Also see

[ATAN](#) (on page 3-63)
[PI](#) (on page 3-77)
[TANH](#) (on page 3-83)

TANH Calc worksheet function

This command returns the hyperbolic tangent of a value.

Usage

TANH (*Value*)

<i>Value</i>	Any number
--------------	------------

Example

<code>=TANH (1.5)</code>	Returns 905.1483E-3.
<code>=TANH (1.1)</code>	Returns 800.4990E-3.

Also see

[ATANH](#) (on page 3-63)
[COSH](#) (on page 3-66)
[SINH](#) (on page 3-80)
[TAN](#) (on page 3-83)

VARP Calc worksheet function

This command returns the variance of a population based on an entire population of values.

Usage

`VARP(Value_list)`

<code>Value_list</code>	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

<code>=VARP(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)</code>	Returns 265.3061E-3.
---	----------------------

Also see

[STDEVP](#) (on page 3-81)

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)`

<code>Serial_number</code>	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

<code>=YEAR(39328)</code>	Returns 2.0070E+3.
<code>=YEAR("06/21/16")</code>	Returns 2.0160E+3.
<code>=YEAR(NOW())</code>	Returns the present year.

Also see

[DAY](#) (on page 3-66)
[HOUR](#) (on page 3-68)
[MINUTE](#) (on page 3-75)
[MONTH](#) (on page 3-76)
[NOW](#) (on page 3-77)
[SECOND](#) (on page 3-79)

Customizing Clarius

In this section:

Customize Clarius	4-1
Add objects to the library	4-1
Project tree display options	4-7
Messages display option	4-7
My Settings	4-7
Tools	4-14

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

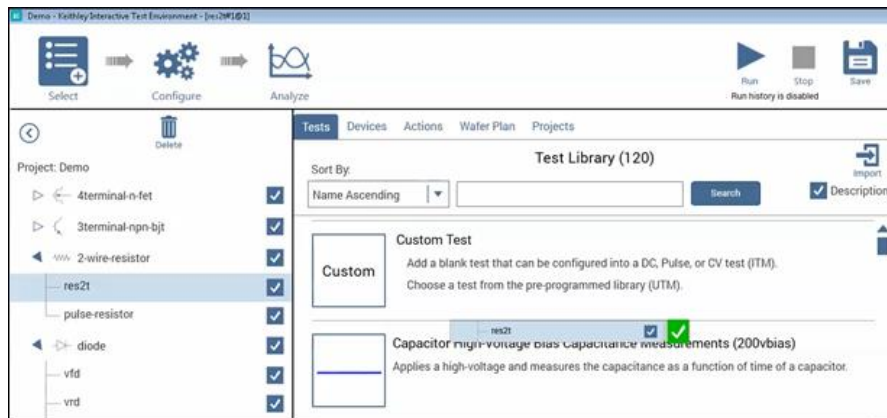
The following steps provide specifics on how to add a test to the library. You can follow the same basic procedure to add devices, actions, and projects. The primary difference is the type of object and which library the object is added to.

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. Select **Analyze**.
3. Select the run histories that you want to include in the new test.
4. Open **Select**.
5. Drag the test from the project tree to the library. You 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 112: 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 4-4) 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 4-4) 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 4-4).

Add a project to the library

The following example provides specifics on how to add a project to the library.

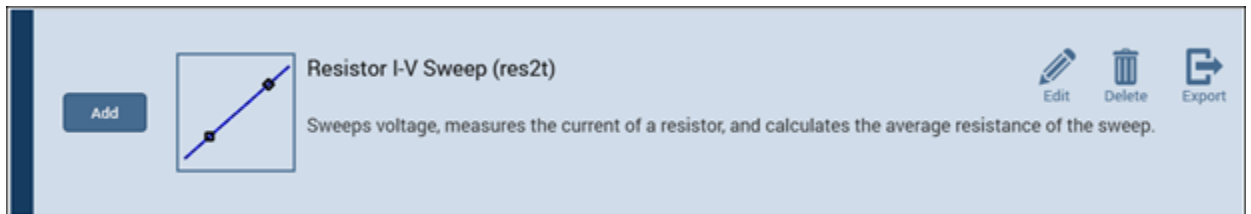
To add a project to the library:

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 4-4).

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 113: 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 4-5) for the options.

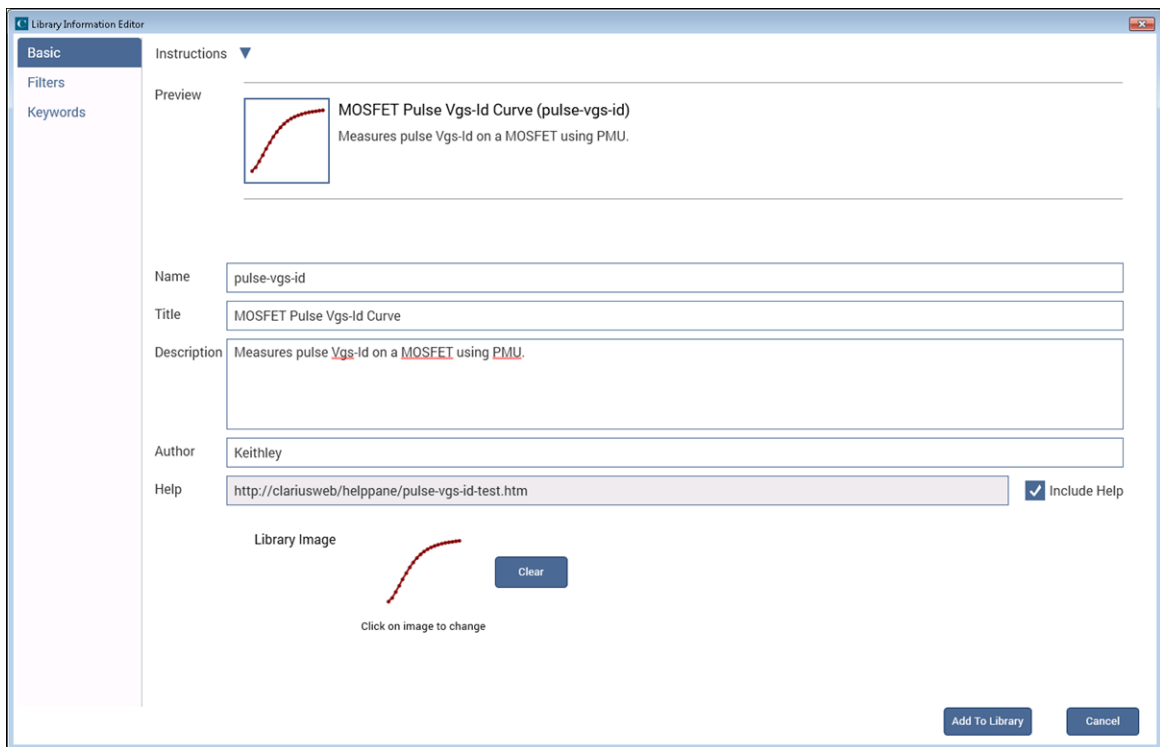
Edit an object in the library

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 114: Library Information Editor



To change information for a library object:

1. In the Basic tab, complete the information as needed. Refer to the following table 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 Project 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 <code>png</code> 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 <code>bmp</code> image to <code>png</code> 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 <code>png</code> 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 <code>bmp</code> image to <code>png</code> format. You can use a tool such as Microsoft® Paint to convert the image. To leave the image area blank, select Clear .

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 ^.

My Settings

The options in My Settings allow you to set environment settings, run settings, graph defaults, GPIB abort, and logging 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, where your projects are stored, if the save as functionality is available, and which project loads at startup.

Reset GPIB devices when Clarius starts

Select this option to reset GPIB devices in the system 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.

For additional information, see “PMU minimum settling times versus current measure range” in the *Model 4200A-SCS Pulse Card (PGU and PMU) User's Manual*.

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 2-21) 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.

Enable the Save As button

You can enable Save As functionality, which allows you to save a presently open project to a different name into the same folder or a different folder. The folder location can be on the 4200A hard drive, a USB drive, or a mapped network drive.

To enable the Save As functionality:

1. Select **My Settings**.
2. Select **Environment Settings**.
3. Select **Enable Save As button**.

The Save As button is now available to the right of the Save button in the Clarius header.

When you select the Save As button, Clarius displays a file tree dialog box. The name and path of the presently selected project is selected when the dialog box is opened.

To save the project that is open in Clarius to a new name, enter the new file name in the Project name box. If the name already exists in the selected folder, you can cancel or overwrite the existing project. The newly created project opens after the Save As operation completes.

NOTE

Clarius automatically saves any unsaved changes to the open project before completing the Save As operation. The original and new projects will have identical configurations and data after completion of Save As. If you cancel the Save As operation, changes are not saved to the presently open project.

My Projects Directory

Use this option to change the directory where your projects are stored. 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`.

In the My Projects Directory dialog box, you can right-click to add a new folder, rename an existing folder, or delete a folder.

You can move projects to the new location using Windows Explorer. When moving projects, move the project folder and all its contents.

Project to load at startup

You can choose which project opens when you start Clarius.

If you select **Load last open project at startup**, the project that was open when you last closed Clarius is opened at startup.

If you select **Load default project at startup**, you can select a project to be the default project that opens at startup. Use Browse to select the default project.

Specify run settings

The options in the Run Settings dialog box allow you to customize Clarius behavior when runs occur. The behaviors you can adjust are:

- 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.
- If hardware is reinitialized after the run.
- If a test can be monitored while repeating continuously.
- The size of run history.

Continue to run after an error

This option determines if Clarius continues running a project sequence when it encounters errors in a test. 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.

Some examples of the types of errors that are ignored are:

- Tests for which no SMUs have been specified.
- Tests that are not configured or are improperly configured.

Ignore interlock state

The "Ignore interlock state" option allows you to choose if tests should continue if the interlock circuit is open.

When "Ignore interlock state" is selected and the 4200A-SCS interlock circuit is 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 open, Clarius displays a warning message and disables the execution of all tests.

Autoscroll the Analyze sheet during test

The autoscroll option determines whether or not the Analyze sheet scrolls during test execution.

When autoscroll is selected, the sheet scrolls so that new data is always displayed during test execution.

When autoscroll is cleared, the sheet does not display data until the test is complete.

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.

When this option is cleared, all instruments in the system remain at their last programmed settings after the test completes.

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.

Enable Monitor button

The "Enable Monitor button" option allows you to set a test to run continuously until stopped. When this option is selected, the Monitor icon is displayed to the left of the Run icon. Monitor allows you to run the test continuously until Stop is selected.

For more detail on using the monitor feature, refer to [Monitor a test](#) (on page 2-82).

Run History Size

This setting allows you to control the number of runs that are stored and displayed in the Run History pane.

When you run a test, the number of runs stored in the Run History is limited to the number that is set. To maintain the number of runs, Clarius deletes the oldest unselected run. If there is existing data that exceeds the limit, only the oldest unselected run is deleted. If you have all runs selected, the most recent run is deleted when a new run is created.

For example, if you have 200 existing runs and change the maximum run history size to 5, the number of runs in the Run History remains at 200. The next test run triggers the deletion of the oldest unselected run.

If you are running a project with subsite cycling or stressing, all existing runs are automatically unselected at the start of execution. Only data collected during the last run of the project is selected and displayed on the graphs. To save data between project runs, increase the run history size to be more than the number of cycles or stresses in your project.

If the run history size setting is less than the number of cycles or stresses in your project, a run is still generated for each cycle or stress.

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 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 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 user-defined GPIB command string to the instrument.

NOTE

While most instruments will respond to the `DCL` command, if an instrument is not SCPI compliant, it will not respond. 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" in *Model 4200A-SCS Parameter Analyzer Setup and Maintenance* for instruction.

To add a custom string, select the box to the left of the Custom String box for the instrument.

Logging

The Logging dialog box allows you to select which messages are logged. It also allows you to opt in or opt out of sharing data with Keithley Instruments.

The Logging Level options selects the which error messages are logged based on severity. The logged error messages include messages at the selected logging level and messages at higher severity levels. The levels are listed from highest to lowest severe. For example, if you select **Informational**, the 4200A-SCS will also log error and warning messages, but not debug messages. Select **Debug** to log all messages.

The location of the logging directory is displayed in this dialog box. It cannot be changed. If you contact Keithley Instruments for help troubleshooting a problem, you can copy the information in this directory and send it to Keithley to help diagnose the problem.

The "Share anonymous usage data with Keithley Instruments" allows you to opt in or out of sharing usage analytic data with Keithley.

NOTE

These options are also available in the Keithley Logging Client Control option in the Microsoft Windows notification area.

Tools

The Tools menu includes tools specific to the SMU, CVU, and PMU instruments that are installed in your 4200A-SCS. It also includes a tool that allows you to export data to a Microsoft Excel spreadsheet.

Instrument Tools

The Instrument Tools tab in the Tools dialog box 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 autocalibrate 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 4-14) 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 4-16) 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 Measurement](#) (on page 4-27) for instructions.
- **CVU Confidence Check:** CVU 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 [CVU Confidence Check](#) (on page 4-28) 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 4-31) for instructions.

Calibrate the system

To maintain SMU performance specifications, you must autocalibrate the 4200A-SCS every 24 hours or any time after the ambient temperature has changed more than ± 1 °C.

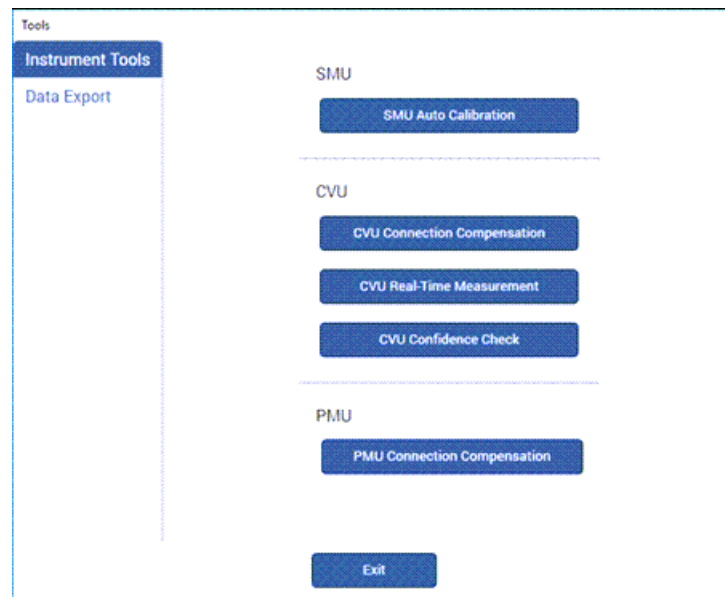
The autocalibration routine recalibrates the current and voltage offsets for all source and measurement functions of all SMUs in the system.

NOTE

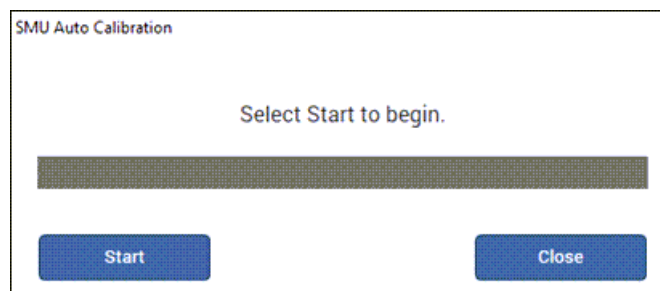
Before initiating a calibration, allow the system to warm up for at least 30 minutes. Clarius prevents autocalibration if the system is not sufficiently warmed up.

To autocalibrate:

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**.
5. Select **Instrument Tools**.

Figure 115: Clarius Tools dialog box

6. Select **SMU Auto Calibration**. A warning dialog box is displayed.
7. Select **OK**. The SMU Auto Calibration dialog box opens, as shown in the following figure.

Figure 116: SMU Auto Calibration dialog box

8. Select **Start**. A progress bar is displayed. When autocalibration is complete, the message "Auto calibration successfully completed" is displayed.
9. Select **Close**. Autocalibration is complete.

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:

- 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 CVU is connected to a 4200A-CVIV Multi-Switch, run the `cvu-cviv-comp-collect` action. Refer to the Model 4200A-CVIV 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 ($>1\text{ M}\Omega$, large impedance).
- **Short** correction: Offset correction for large capacitances ($<10\ \Omega$, small impedance).
- **Load** correction: Resistive load correction for gain error. Keithley recommends a load that is as close in impedance to the cabling system ($100\ \Omega$) as possible. 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

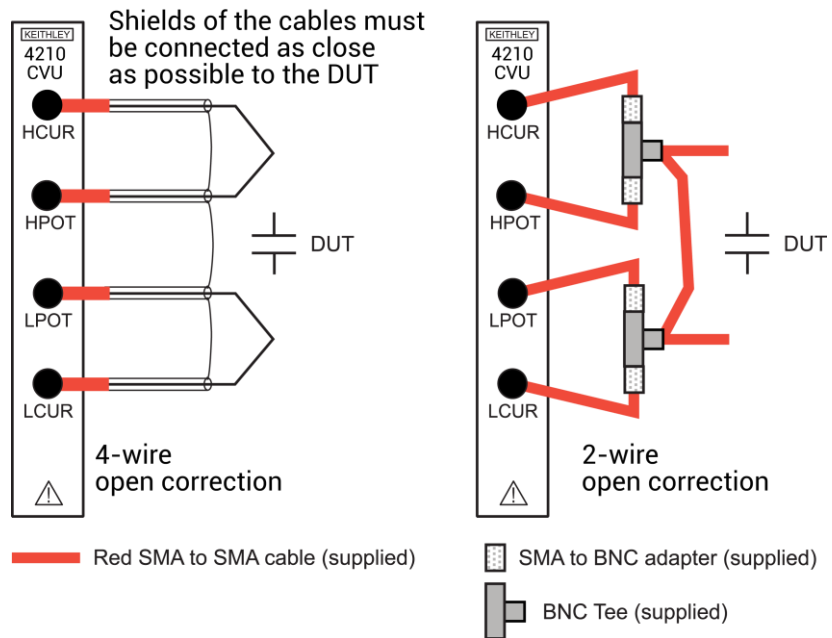
Open connection compensation is usually done to offset correction for small capacitances.

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 117: Connections for open connection compensation CVU



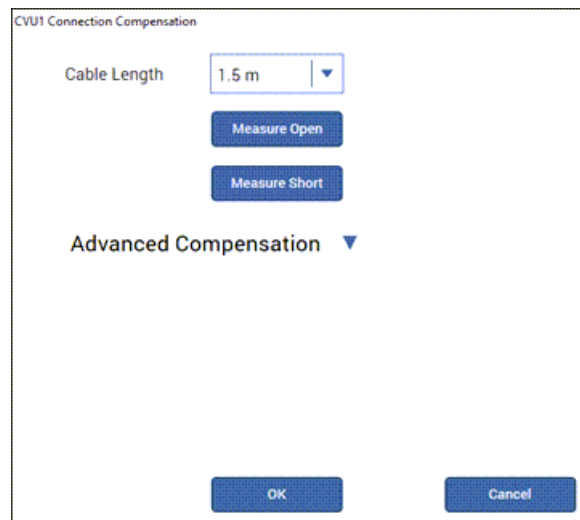
- In Clarius, select **Tools**. The Clarius Tools dialog box opens.

Figure 118: Clarius Tools dialog box



- Select **CVU Connection Compensation**.

Figure 119: 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 "Using Switch Matrices" in *Model 4200A-SCS Prober and External Instrument Control*.
7. In the Clarius CVU Connection Compensation dialog box, select **Measure Open**.
8. Follow the instructions.
9. Select **OK**.

Generate short connection compensation data

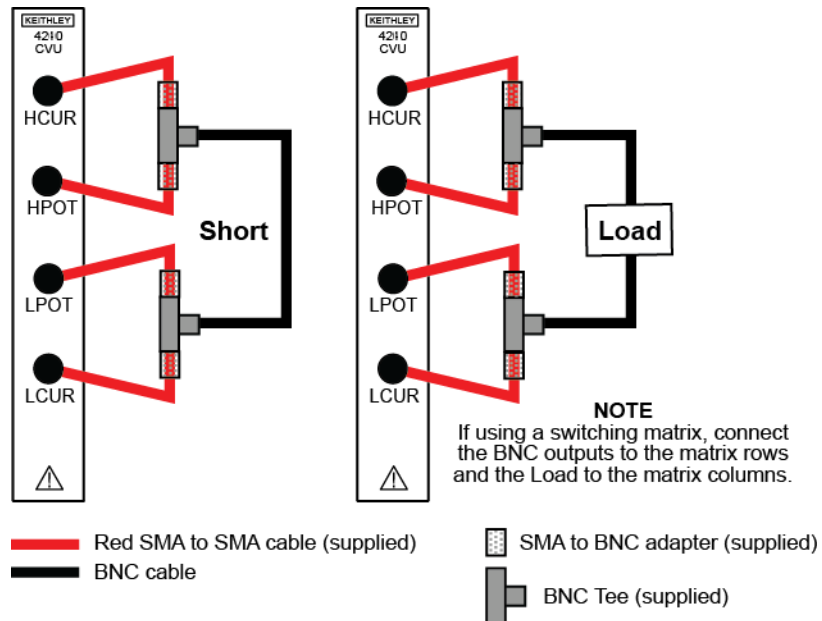
Short connection compensation is usually done to offset correction for large capacitances.

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 120: Connections for short and load connection compensation



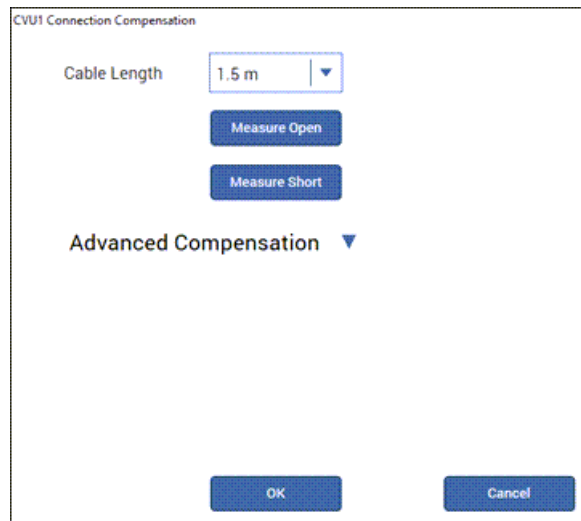
- In Clarius, select **Tools**. The Clarius Tools dialog box opens.

Figure 121: Clarius Tools dialog box



- Select **CVU Connection Compensation**.

Figure 122: 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 "Using Switch Matrices" in *Model 4200A-SCS Prober and External Instrument Control*.
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 in the test circuit.

To generate load correction data:

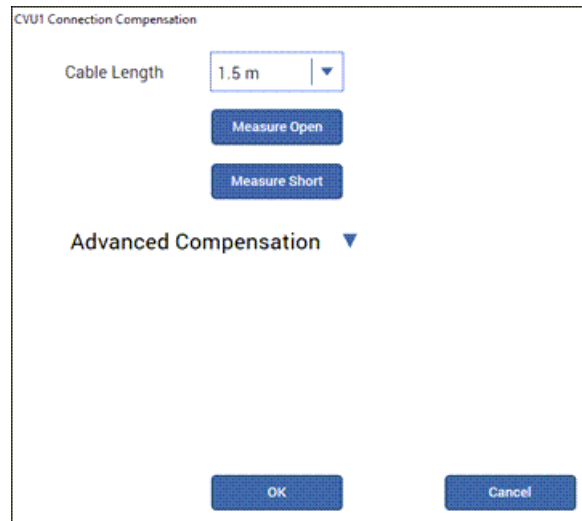
1. Make the connections to the CVU. See “Test connections for a switch matrix” in the *Model 4200A-SCS Capacitance-Voltage Unit (CVU) User's Manual*.
2. In Clarius, select **Tools**. The Clarius Tools dialog box opens.

Figure 123: Clarius Tools dialog box



3. Select **CVU Connection Compensation**.

Figure 124: 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 “Using Switch Matrices” in *Model 4200A-SCS Prober and External Instrument Control*.
7. If it is not open, select **Advanced Compensation**.
8. In 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. Next to View Compensation Data, 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 125: Open compensation values example

Frequency	1mA		30uA		1uA	
	R	jX	R	jX	R	jX
1kHz	1e-012	0	1e-012	0	1e-012	0
2kHz	0.0192332	0.0092446	0.0192332	0.0092446	0.0192332	0.0092446
3kHz	0.0240222	0.00620513	0.0240222	0.00620513	0.0240222	0.00620513
4kHz	0.0251885	0.00343866	0.0251885	0.00343866	0.0251885	0.00343866
5kHz	0.0251798	-0.00192463	0.0251798	-0.00192463	0.0251798	-0.00192463
6kHz	0.0247988	-0.00374154	0.0247988	-0.00374154	0.0247988	-0.00374154
7kHz	0.0243094	-0.00494131	0.0243094	-0.00494131	0.0243094	-0.00494131
8kHz	0.0241997	-0.00483867	0.0241997	-0.00483867	0.0241997	-0.00483867
9kHz	0.0229851	-0.00625016	0.0229851	-0.00625016	0.0229851	-0.00625016
10kHz	0.0230555	-0.00879264	0.0230555	-0.00879264	0.0230555	-0.00879264

HI LO

Note: A value of R=1e15 and jX=1e15 indicates that a measurement could not be made and default values will be used for the Open Compensation.
 Open: Mon Sep 26 14:58:31 2016 ;
 CVU Path: Direct

OK

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 setting that was used in the Tools > CVU Connection Compensation dialog box to generate connection compensation data.

Figure 126: Enable connection compensation

The screenshot shows the 'Terminal Settings' dialog box. At the top, there are tabs for 'Test Settings', 'Terminal Settings' (which is active), and 'Help'. Below the tabs is a 'Gate' section with an 'Advanced' button. The main content is divided into several sections: 'Force' (with a minus sign), 'Measure' (with a minus sign), and 'Compensation' (with a minus sign). Under 'Force', there is a 'DC Operation Mode' dropdown set to 'Voltage Linear Sweep', and input fields for 'Presoak' (0 V), 'Start' (-3 V), 'Stop' (1 V), and 'Step' (0.02 V). There is also an unchecked 'Dual Sweep' checkbox. Under 'Measure', there is a 'Parameters' dropdown set to 'Cp-Gp'. Under 'Compensation', there are three unchecked checkboxes: 'Open', 'Short', and 'Load'. At the bottom, there is a 'Cable Length' dropdown set to '1.5m'.

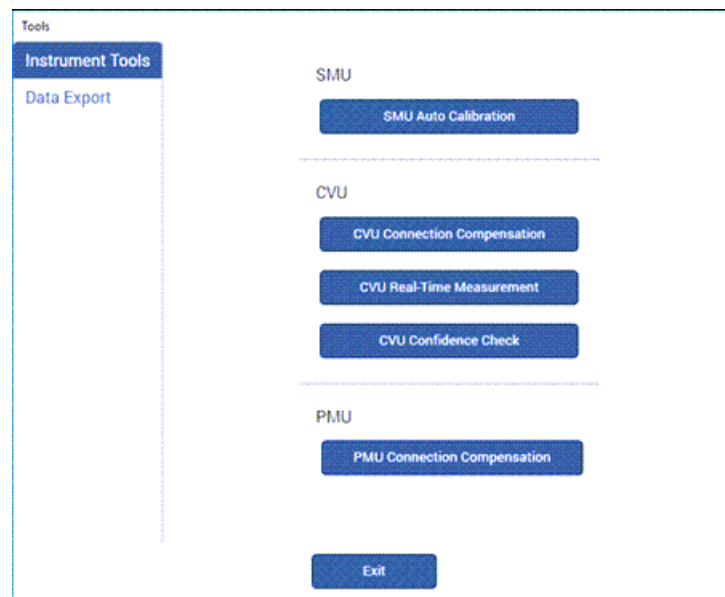
CVU Real-Time Measurement

The CVU Real-Time Measurement 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.

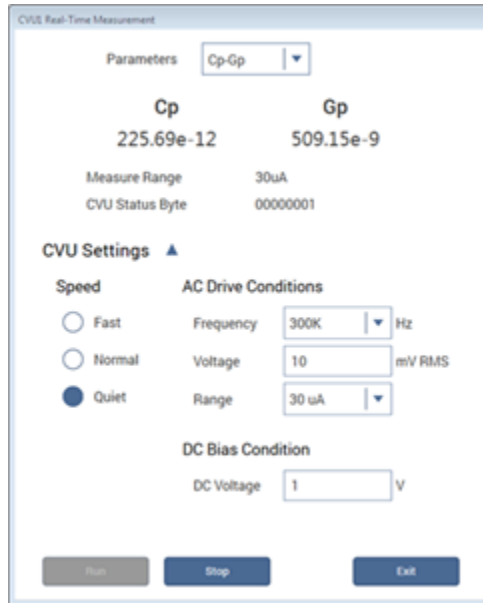
To make real-time measurements:

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

Figure 127: Clarius Tools dialog box



2. Select **Instrument Tools**.
3. Select **CVU Real-Time Measurement**.

Figure 128: Real-Time Measurement dialog box

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

CVU Confidence Check

CVU 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

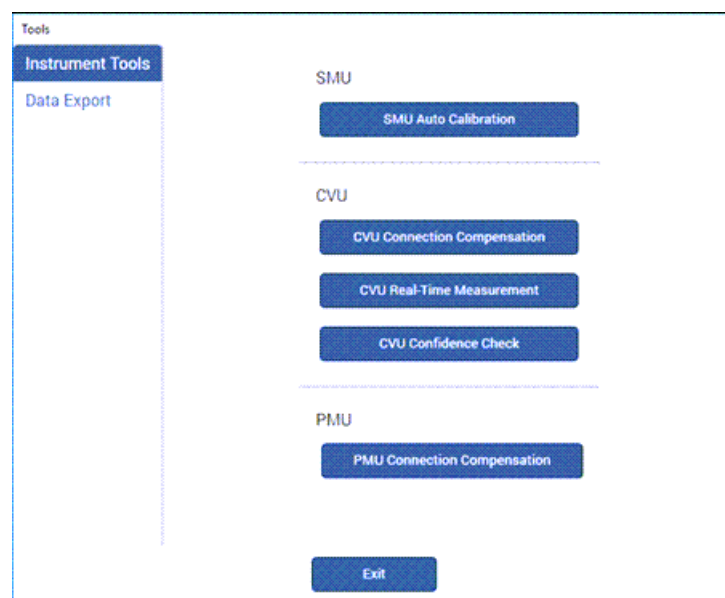
CVU Confidence Check is a diagnostic tool that allows you to check the integrity of open and short connections and connections to a device under test (DUT). When the CVU is connected to the DUT, the Confidence Check displays the measured readings in real time in the Messages area of Clarius.

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

To do a CVU confidence check:

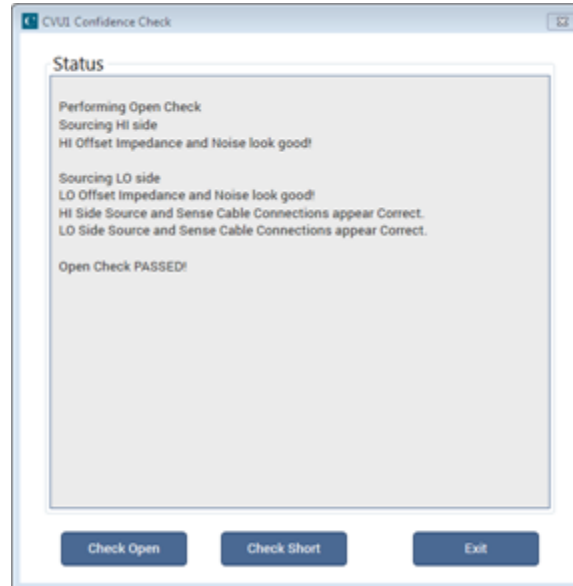
1. If you are using a switching matrix, connect the switching matrix to the CVU and DUT or the short as explained in “Test connections for a switch matrix” in the Model 4200A-SCS Capacitance-Voltage Unit (CVU) User's Manual.
2. For the short check, close the matrix switches to connect the CVU to the DUT or short. For the open check, also close the matrix switches, but lift the probes or disconnect the DUT.
3. In Clarius, select **Tools**. The Clarius Tools dialog box opens.

Figure 129: Clarius Tools dialog box



4. Select **CVU Confidence Check**.
5. Select **Check Open** or **Check Short**.

Figure 130: CVU Confidence Check dialog box



6. Follow the instructions and click **OK**.

When the check is complete, the dialog box displays the results of the test. If the test failed, the results include suggestions for troubleshooting.

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 LPT function `pulse_conncomp`.

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

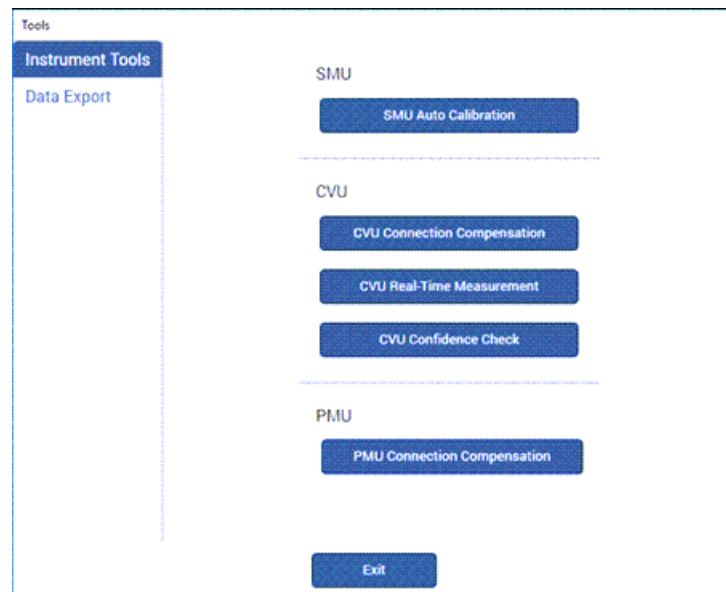
Error currents can be introduced into your pulsed measurements setup by PMUs. PMU offset compensation reduces error currents by subtracting measurements taken at 0 V from all subsequent readings.

Perform connection compensation

To compensate for connections:

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

Figure 131: 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 PMU list, select the PMU that you want to perform compensation for.

Figure 132: PMU Connection Compensation dialog box

PMU Short and Offset Current Compensation Values and Defaults

PMU1 ▼

Channel 1 Channel 2

Measure Short Short Resistance (Ω) 0.7 0.7

Offset Current Correction (Measured at 0V)

Measure Offset

Force Range	Measure Range	Channel 1 (A)	Channel 2 (A)
40V	800mA	0.00	0.00
40V	10mA	0.00	0.00
40V	100uA	0.00	0.00
10V	200mA	0.00	0.00
10V	10mA	0.00	0.00
10V	1mA	0.00	0.00
10V	100uA	0.00	0.00
10V	10uA	0.00	0.00
10V	1uA	0.00	0.00
10V	100nA	0.00	0.00

Note: N/A indicates that a measurement could not be made at RPM range because RPM is not connected.

Default

Exit

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 uses both PMU channels, you will have new custom data for both channels.

Enabling connection compensation

NOTE

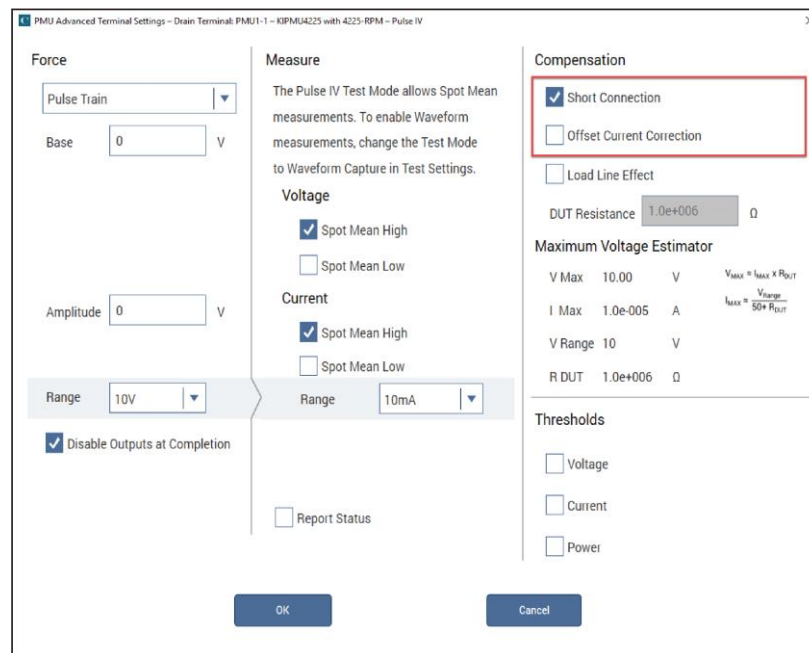
This procedure is for ITMs. For UTMs, you need to enable connection compensation data using the LPT command `pulse_conncomp` or `setmode` functions.

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**. Refer to the following figure.

Figure 133: Enabling connection compensation



7. Select **OK**.
8. To disable connection compensation, clear either **Short Connection** or **Offset Current Correction**, then select **OK**. When disabled, connection compensation values are not applied to DUT measurements.

Data Export tool

The Data Export tab in the Tools dialog box allows you to export data files to a Microsoft Excel .xls format.

NOTE

To export specific runs, select the runs in Run History before exporting data and set Run to <Selected Runs>. See [Changing the display of Run Histories](#) (on page 3-15) for information on selecting runs.

To export data:

1. Select **Tools**.
2. On the left navigation pane, select **Data Export**.
3. If you have sites and subsites, select the sites and subsites that contain data to be exported.
4. Select the devices, tests, and runs that contain data to be exported.
5. Select **Path** to set the location of the exported file.
6. In the File Name field, enter the patterns for the file name. Select **Help** for detail on the patterns that are available.
7. To overwrite existing files, select **Overwrite any existing data files**.
8. Select **Export Selected Data**. A progress bar is displayed during the export.

Formulator

In this section:

Introduction	5-1
Open the Formulator	5-2
Configure Formulator calculations.....	5-3
Formulator dialog box	5-3
Using the Formulator options	5-5
Real-time functions, operators, and formulas.....	5-6
Post-test-only functions and formulas	5-7
Editing Formulator formulas and constants	5-8
Deleting Formulator formulas and constants.....	5-8
Identify data analysis requirements.....	5-8
Formulator function reference	5-13
Statistics.....	5-16
Trigonometry	5-19
Array	5-23
Line Fits.....	5-32
FFT	5-53
Misc.....	5-60

Introduction

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 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).

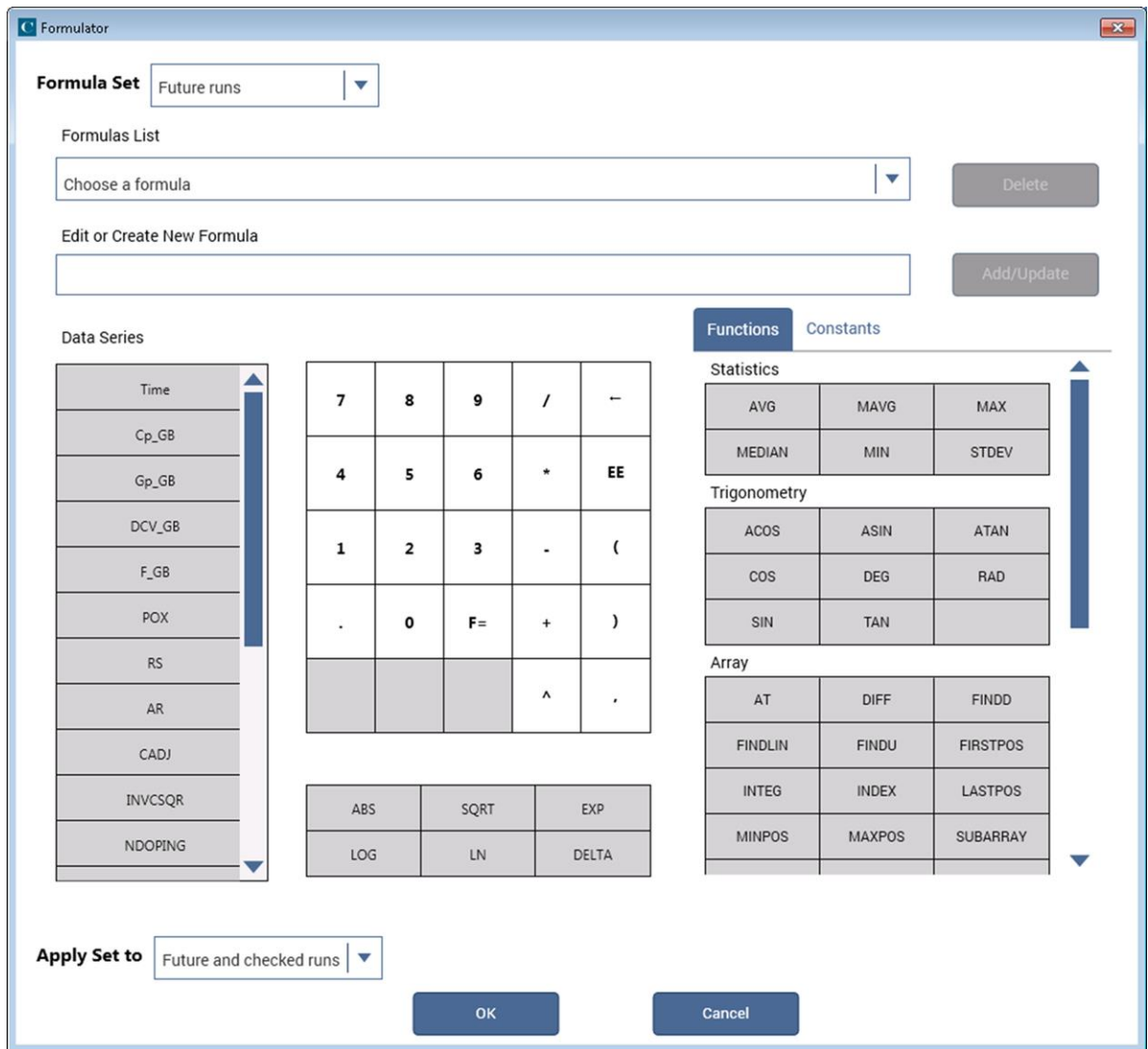
The results of some calculations may be a column of values (vector) in the Analyze Run sheet or a column that contains only a single value (scalar).

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 134: Formulator with no entries



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 5-13).

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.

From the **Formula Set** list, select whether you want to apply the formula to the runs selected in Run History 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 and any existing formulas. 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 5-5).

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 5-13).

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.
- **Currently 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 future test runs and the runs that are selected in the Run History pane.

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.

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
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 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 5-23)	IFFT_I (on page 5-59)	MIN (on page 5-18)
AVG (on page 5-17)	IFFT_R (on page 5-58)	MINPOS (on page 5-30)
COND (on page 5-61)	LASTPOS (on page 5-30)	REGFIT (on page 5-50)
EXPFIT (on page 5-33)	LINFIT (on page 5-36)	REGFITSLP (on page 5-51)
EXPFITA (on page 5-34)	LINFITSLP (on page 5-37)	REGFITXINT (on page 5-52)
EXPFITB (on page 5-35)	LINFITXINT (on page 5-38)	REGFITYINT (on page 5-53)
FFT_FREQ (on page 5-56)	LINFITYINT (on page 5-39)	SMOOTH (on page 5-60)
FFT_FREQ_P (on page 5-57)	LOGFIT (on page 5-40)	SUBARRAY (on page 5-31)
FFT_I (on page 5-55)	LOGFITA (on page 5-41)	SUMMV (on page 5-31)
FFT_R (on page 5-54)	LOGFITB (on page 5-42)	TANFIT (on page 5-43)
FINDD (on page 5-25)	MAVG (on page 5-17)	TANFITSLP (on page 5-44)
FINDLIN (on page 5-26)	MAX (on page 5-18)	TANFITXINT (on page 5-45)
FINDU (on page 5-27)	MAXPOS (on page 5-30)	TANFITYINT (on page 5-46)
FIRSTPOS (on page 5-27)		

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.

Editing Formulator formulas and constants

To edit a Formulator formula:

1. From the **Formulas List**, choose a formula. It is displayed in the Edit or Create New Formula box.
2. Edit the formula as needed.
3. Click **Add/Update**.
 - If you renamed the result variable on the left side of the formula, the Formulator adds the edited formula to the Formulas List 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 to create a new formula. Nothing happens to either of the formula boxes. Edit the name of the result variable, then click **Add/Update** again.
 - **Yes** if you edited the formula to update it. The replacement formula appears in the Formulas List.

Deleting Formulator formulas and constants

To delete a Formulator formula:

1. From the **Formulas List**, select the formula.
2. Select **Delete**.

Identify data analysis requirements

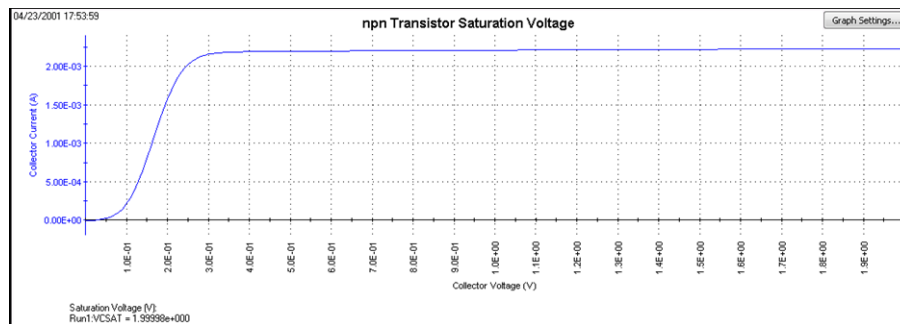
In many cases, you may already know a needed analysis formula, even before running a test. In fact, for a test, you may create a real-time formula in advance so that you can monitor its output during a test, either in the Analyze sheet or graph.

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 135: 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 demonstrate 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 5-5)). 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 101 and 201. This is the range information that you need to create the regression analysis equation using `REGFIT`.

Figure 136: 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.

To create an analysis formula, complete the following steps:

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 5-9), enter the equation in the figure below.

Figure 137: Creating the regression formula for the data



Adding an analysis formula to the test

To display a formula in the Edit or Create New Formula, box, select it.

After editing an existing formula or creating a new one, select **Add/Update**. 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 5-8)

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 select **Add/Update**. 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 Formulas List 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 #REF 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.

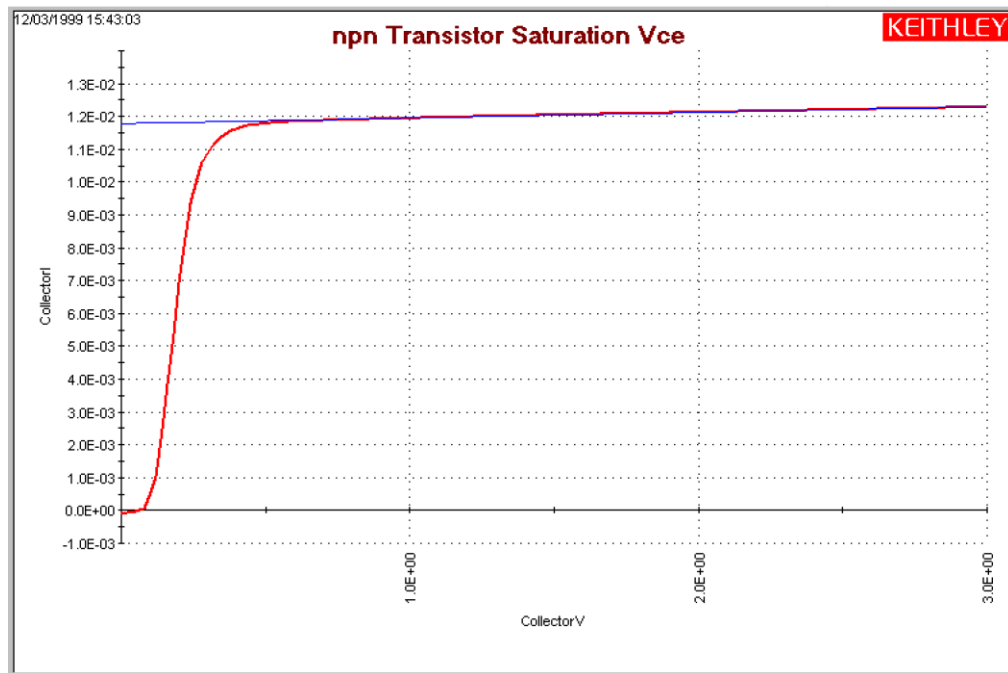
Therefore, the Formulator returns #REF in the first row.

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 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 138: Added linear regression line to the graph



NOTE

For information about using the Graph tab, refer to [Graph](#) (on page 3-26).

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

$ABS(Value)$

<i>Value</i>	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

<code>F2 = ABS(GateI)</code>	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

$SQRT(Value)$

<i>Value</i>	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

<code>TWO = SQRT(4)</code>	
----------------------------	--

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)`

<i>Value</i>	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 5-16)

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)`

<i>Value</i>	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 (DRAIN1)
```

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

$LN(Value)$

<i>Value</i>	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 5-15)

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

$DELTA(Value)$

<i>Value</i>	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

Statistics

The following Formulator functions provide statistics operations.

AVG Formulator function

Returns the average of all values in the column (vector).

Usage

`AVG (Value)`

<code>Value</code>	The name of any column (vector) in the Data Series list
--------------------	---

Example

`LEAKAGE = AVG (GATEI)`

Also see

[MAVG](#) (on page 5-17)

MAVG Formulator function

Returns a new column (vector) consisting of the moving averages of successive groups of points from another column (vector).

Usage

`MAVG (V, N)`

<code>V</code>	The name of any column (vector) in the Data Series list or any operand
<code>N</code>	The number of points to be averaged in each group

Details

You can configure the number of points in a group.

If $N = 3$ and V contains the 12 values $x_1, x_2, x_3, x_4, x_5, \dots, x_{10}, x_{11}, x_{12}$, 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)`

<code>Value</code>	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)`

<code>Value</code>	The name of any column (vector) in the Data Series list or any operand
--------------------	--

Details

Returns the value in radians.

Example

```
F1 = ACOS (DRAIN1)
```

Also see

None

ASIN Formulator function

Returns the arc sine of each value in a designated column (vector) under **Columns** or any operand.

Usage

$ASIN(Value)$

<i>Value</i>	The name of any column (vector) in the Data Series list or any operand
--------------	--

Details

Returns the value in radians.

Example

```
F1 = ASIN(DRAIN1)
```

Also see

None

ATAN Formulator function

Returns the arc tangent of each value in a designated column (vector) under **Columns** or any operand.

Usage

$ATAN(Value)$

<i>Value</i>	The name of any column (vector) in the Data Series list or any operand
--------------	--

Details

Returns the value in radians.

Example

```
F1 = ATAN(DRAIN1)
```

Also see

None

COS Formulator function

Returns the cosine of each value operand.

Usage

$\text{COS}(\text{Value})$

<i>Value</i>	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

$\text{DEG}(\text{Value})$

<i>Value</i>	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)`

<i>Value</i>	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)`

<i>Value</i>	The name of any column (vector) in the Data Series list or any operand
--------------	--

Details

Returns the value in radians.

Example

```
F1 = SIN(DRAIN1)
```

Also see

None

TAN Formulator function

Returns the tangent of each value in a designated column (vector) under **Columns** or any operand.

Usage

$TAN(Value)$

<i>Value</i>	The name of any column (vector) in the Data Series list or any operand
--------------	--

Details

Returns the value in radians.

Example

F1 = TAN(DRAIN1)

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

$AT(Value, POS)$

<i>Value</i>	The name of any column (vector) in the Data Series list or any operand
<i>POS</i>	The row number of column <i>Value</i> where the single value is located

Example

IDSAT = AT(DRAIN1, 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 (DRAIN1, 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 V , beginning at $START$. Then it returns the row number 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 of the starting value for the search

Details

If **FINDD** does not find an exact match for X , it returns the row number of the V value that is closest to X .

Example

```
IF = AT(ANODEI, FINDD(ANODEV, 0.7, FIRSTPOS(ANODEV)))
```

Also see

[FINDLIN](#) (on page 5-26)

[FINDU](#) (on page 5-27)

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 .

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 of the starting value for the search

Details

`FINDLIN` searches a column 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. The returned index number (in decimal format) indicates the position of the specified value.

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` returns Index 3.5.

Example

```
IF = AT(ANODEI, FINDLIN(ANODEV, 0.7, FIRSTPOS(ANODEV)))
```

Also see

[FINDD](#) (on page 5-25)

[FINDU](#) (on page 5-27)

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 of that value. `FINDU` searches a column V , beginning at $START$. Then it returns the row number 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 of the starting value for the search

Details

If `FINDU` does not find an exact match for X , it returns the row number of the V value that is closest to X .

Example

```
IF = AT (ANODEI, FINDU (ANODEV, 0.7, LASTPOS (ANODEV)))
```

Also see

[FINDD](#) (on page 5-25)

[FINDLIN](#) (on page 5-26)

FIRSTPOS Formulator function

Returns the row number of the first value in a column, typically the number 1.

Usage

`FIRSTPOS (V)`

V	The name of any column (vector) listed under Columns
-----	---

Example

```
STARTOFARRAY = FIRSTPOS (DRAINI)
```

Also see

[LASTPOS](#) (on page 5-30)

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 $n = 1, 2, \dots, N-1, N$.

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 A_1 equal to 0 (zero area at the start of a curve, at X_1) and A_2, A_3, A_4 , and A_5 equal to curve areas, as shown in the following graphics.

Figure 139: INTEG Formulator function

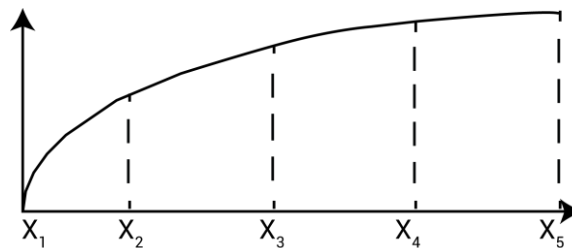


Figure 140: A5 curve area

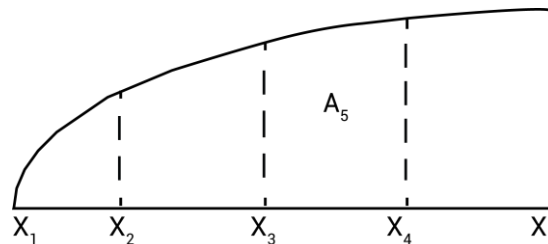


Figure 141: A2, A3, and A4 curve areas

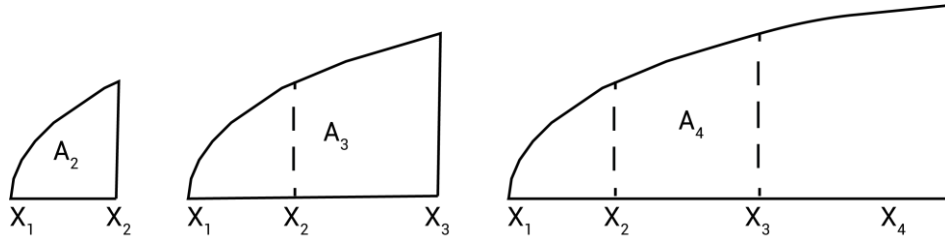
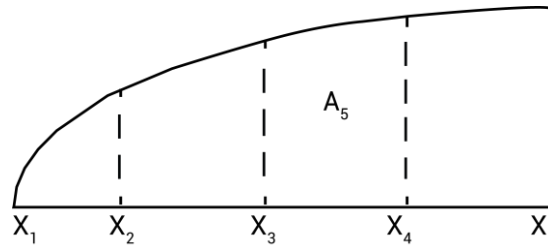


Figure 142: A5 curve area



You can use this function to do calculations in real time (while a test is executing).

Example

```
QBD = INTEG (TIME, GATEI)
```

Also see

None

INDEX Formulator function

Returns a specified number of points starting with a specified value and consecutive values incremented by one.

Usage

`INDEX (START, N)`

<i>START</i>	The starting value
<i>N</i>	The number of 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

LASTPOS Formulator function

Returns the row number of the last value in a column.

Usage

LASTPOS (*Value*)

Value

The name of any column (vector) listed under **Columns**

Example

```
NUMSWEEPPTS = LASTPOS (COLLECTORI)
```

Also see

[FIRSTPOS](#) (on page 5-27)

MINPOS Formulator function

Searches all values in a column, finds the minimum value, and returns the row number of the minimum value.

Usage

MINPOS (*V*)

V

The name of any column (vector) listed under **Columns**

Example

```
LOCATION = MINPOS (DRAIN1)
```

Also see

None

MAXPOS Formulator function

Searches all values in a column, finds the maximum value, and returns the row number 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

SUBARRAY Formulator function

Returns a new column containing a specified range of the values from an existing column.

Usage

`SUBARRAY (V, STARTPOS, ENDPOS)`

V	The name of any column (vector) listed under Columns
STARTPOS	The row number of the existing value that you choose to become the first value in the new column (vector)
ENDPOS	The row number 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 (VEXIST, 10, 20)`

Given an existing column, *V_{EXIST}*, containing values in rows 1 through 60, you could use *SUBARRAY* to return a new column, *V_{NEW}*, containing only the values from rows 10 through 20 of *V_{EXIST}*.

Also see

None

SUMMV Formulator function

Returns a column (vector) *V_Y* 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 *n*th value in *V_Y* (*Y_n*) is the sum of the *n*th and preceding values in *V*. This relationship may be expressed mathematically as follows:

$$Y_n = \sum_{i=1}^{i=n} X_i$$

Where x_i = 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 and returns a new column of Y values.

Usage

`EXPFIT(VX, VY, STARTPOS, ENDPOS)`

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>STARTPOS</i>	For the range of X and Y values to be exponentially fitted, the row number of the starting values
<i>ENDPOS</i>	For the range of X and Y values to be exponentially fitted, the row number of the ending values

Details

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:

$$Y = \text{EXPFIT}_A * e^{(\text{EXPFIT}_B * X)}$$

Where `EXPFITA` and `EXPFITB` are fit constants. Using this 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, 1, LASTPOS(ANODEV))`

Also see

[EXPFIT_A](#) (on page 5-34)

[EXPFIT_B](#) (on page 5-35)

EXPFITA Formulator function

Returns the value of the constant `EXPFITA` as part of the formula to perform an exponential fit.

Usage

`EXPFITA(VX, VY, STARTPOS, ENDPOS)`

<code>VX</code>	The name of any column (vector) listed under Columns
<code>VY</code>	The name of any column (vector) listed under Columns
<code>STARTPOS</code>	For the range of X and Y values to be exponentially fitted, the row number of the starting values
<code>ENDPOS</code>	For the range of X and Y values to be exponentially fitted, the row number of the ending values

Details

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:

$$Y = \text{EXPFITA} * e^{(\text{EXPFITB} * X)}$$

Where `EXPFITA` and `EXPFITB` are fit constants.

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

```
OFFSETI = EXPFITA(ANODEV, ANODEI, 1, LASTPOS(ANODEV))
```

Also see

[EXPFIT](#) (on page 5-33)

[EXPFITB](#) (on page 5-35)

EXPFITB Formulator function

Returns the value of the constant `EXPFITB` as part of the formula to perform an exponential fit.

Usage

`EXPFIT(VX, VY, STARTPOS, ENDPOS)`

<code>VX</code>	The name of any column (vector) listed under Columns
<code>VY</code>	The name of any column (vector) listed under Columns
<code>STARTPOS</code>	For the range of X and Y values to be exponentially fitted, the row number of the starting values
<code>ENDPOS</code>	For the range of X and Y values to be exponentially fitted, the row number of the ending values

Details

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:

$$Y = \text{EXPFITA} * e^{(\text{EXPFITB} * X)}$$

Where `EXPFITA` and `EXPFITB` are fit constants.

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, 1, LASTPOS(ANODEV))*0.0257)
```

Also see

[EXPFITA](#) (on page 5-34)

[EXPFITB](#) (on page 5-35)

LINFIT Formulator function

Finds a linear equation and returns a new column.

Usage

`LINFIT (VX, VY, STARTPOS, ENDPOS)`

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>STARTPOS</i>	The row number of the first set of X and Y values
<i>ENDPOS</i>	The row number 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 5-50)

LINFITSLP Formulator function

Finds a linear equation and returns the slope.

Usage

LINFITSLP(*VX*, *VY*, *STARTPOS*, *ENDPOS*)

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>STARTPOS</i>	The row number of the first set of X and Y values
<i>ENDPOS</i>	The row number 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, *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 5-51)

LINFITXINT Formulator function

Finds a linear equation and returns the X intercept.

Usage

`LINFITXINT(VX, VY, STARTPOS, ENDPOS)`

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>STARTPOS</i>	The row number of the first set of X and Y values
<i>ENDPOS</i>	The row number 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 5-52)

LINFITYINT Formulator function

Finds a linear equation and returns the Y intercept.

Usage

`LINFITYINT(VX, VY, STARTPOS, ENDPOS)`

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>STARTPOS</i>	The row number of the first set of X and Y values
<i>ENDPOS</i>	The row number 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, *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, use the `REGFITYINT` function.

Example

```
OFFSET = LINFITYINT(GATEV, GATEI, FIRSTPOS(GATEV), LASTPOS(GATEV))
```

Also see

[REGFITYINT](#) (on page 5-53)

LOGFIT Formulator function

Performs a base-10 log-linear fit.

Usage

`LOGFIT(VX, VY, STARTPOS, ENDPOS)`

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>STARTPOS</i>	For the range of <i>X</i> and <i>Y</i> values to be logarithmically fitted, the row number of the starting values
<i>ENDPOS</i>	For the range of <i>X</i> and <i>Y</i> values to be logarithmically fitted, the row number 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 (one column, *VX*, containing *X* values and the other column, *VY*, containing *Y* values):

$$Y = \text{LOGFITA} + \text{LOGFITB} * \log(X)$$

where *LOGFITA* and *LOGFITB* are fit constants.

- Using the above logarithmic relationship, returns a new column 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 5-41)

[LOGFITB](#) (on page 5-42)

LOGFITA Formulator function

Performs a base-10 log-linear fit.

Usage

`LOGFITA(VX, VY, STARTPOS, ENDPOS)`

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>STARTPOS</i>	For the range of <i>X</i> and <i>Y</i> values to be logarithmically fitted, the row number of the starting values
<i>ENDPOS</i>	For the range of <i>X</i> and <i>Y</i> values to be logarithmically fitted, the row number 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 (one column, *VX*, containing *X* values and the other column, *VY*, containing *Y* values):

$$Y = \text{LOGFITA} + \text{LOGFITB} * \log(X)$$

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, DRAIN1, 30, 50)
```

Also see

[LOGFIT](#) (on page 5-40)

[LOGFITB](#) (on page 5-42)

LOGFITB Formulator function

Performs a base-10 log-linear fit.

Usage

`LOGFITB(VX, VY, STARTPOS, ENDPOS)`

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>STARTPOS</i>	For the range of <i>X</i> and <i>Y</i> values to be logarithmically fitted, the row number of the starting values
<i>ENDPOS</i>	For the range of <i>X</i> and <i>Y</i> values to be logarithmically fitted, the row number 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 (one column, *VX*, containing *X* values and the other column, *VY*, containing *Y* values):

$$Y = \text{LOGFITB} + \text{LOGFITB} * \log(X)$$

where `LOGFITB` 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 5-40)

[LOGFITB](#) (on page 5-41)

TANFIT Formulator function

Finds a linear equation of the form $Y = a + bX$ from two columns, *VX* and *VY*.

Usage

`TANFIT (VX, VY, POS)`

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>POS</i>	The row number where the tangent is to be found

Details

Finds a linear equation of the form $Y = a + bX$ from two columns, *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 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, DRAIN1, MAXPOS (GM) )
```

Also see

[TANFITSLP](#) (on page 5-44)

[TANFITXINT](#) (on page 5-45)

[TANFITYINT](#) (on page 5-46)

TANFITSLP Formulator function

Finds a linear equation of the form $Y = a + bX$ from two columns, *VX* and *VY*.

Usage

TANFITSLP(*VX*, *VY*, *POS*)

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>POS</i>	The row number where the tangent is to be found

Details

Finds a linear equation of the form $Y = a + bX$ from two columns, *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 5-43)

[TANFITXINT](#) (on page 5-45)

[TANFITYINT](#) (on page 5-46)

TANFITXINT Formulator function

Finds a linear equation of the form $Y = a + bX$ from two columns, *VX* and *VY*.

Usage

`TANFITXINT(VX, VY, POS)`

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>POS</i>	The row number where the tangent is to be found

Details

Finds a linear equation of the form $Y = a + bX$ from two columns, *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 5-43)

[TANFITSPL](#) (on page 5-44)

[TANFITYINT](#) (on page 5-46)

TANFITYINT Formulator function

Finds a linear equation of the form $Y = a + bX$ from two columns, *VX* and *VY*.

Usage

`TANFITYINT(VX, VY, POS)`

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>POS</i>	The row number where the tangent is to be found

Details

Finds a linear equation of the form $Y = a + bX$ from two columns, *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 5-43)

[TANFITSLP](#) (on page 5-44)

[TANFITXINT](#) (on page 5-45)

POLY2FIT Formulator function

Enables quadratic regression line fitting.

Usage

`POLY2COEFF(VX, VY, STARTPOS, ENDPOS)`

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>STARTPOS</i>	The row number of the first set of X and Y values
<i>ENDPOS</i>	The row number 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 5-48)

[POLYNFIT](#) (on page 5-49)

POLY2COEFF Formulator function

Enables quadratic regression line fitting.

Usage

`POLY2COEFF(VX, VY, STARTPOS, ENDPOS)`

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>STARTPOS</i>	The row number of the first set of X and Y values
<i>ENDPOS</i>	The row number 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 5-47)

[POLYNFIT](#) (on page 5-49)

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*)

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>ORDER</i>	The order
<i>STARTPOS</i>	The row number of the first set of X and Y values
<i>ENDPOS</i>	The row number 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 5-48)

[POLY2FIT](#) (on page 5-47)

REGFIT Formulator function

Performs a linear regression fit.

Usage

`REGFIT (VX, VY, STARTPOS, ENDPOS)`

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>STARTPOS</i>	For the range of X and Y values to be fitted, the row number of the starting values
<i>ENDPOS</i>	For the range of X and Y values to be fitted, the row number 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 (column *VX* containing X values and column *VY* containing Y values):

$$Y = \text{REGFIT}Y\text{INT} + \text{REGFIT}SLP * X$$

where *REGFITSLP* and *REGFITINT* are slope and Y-intercept constants.

- Using the above linear relationship, returns a new column that contains 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 5-51)

[REGFITINT](#) (on page 5-53)

REGFITSLP Formulator function

Performs a linear regression fit.

Usage

REGFITSLP(*VX*, *VY*, *STARTPOS*, *ENDPOS*)

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>STARTPOS</i>	For the range of <i>X</i> and <i>Y</i> values to be fitted, the row number of the starting values
<i>ENDPOS</i>	For the range of <i>X</i> and <i>Y</i> values to be fitted, the row number 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 (column *VX* containing *X* values and column *VY* containing *Y* values):

$$Y = \text{REGFITYINT} + \text{REGFITSLP} * X$$

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 5-50)

[REGFITYINT](#) (on page 5-53)

REGFITXINT Formulator function

Performs a linear regression fit.

Usage

REGFITXINT(*VX*, *VY*, *STARTPOS*, *ENDPOS*)

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>STARTPOS</i>	For the range of X and Y values to be fitted, the row number of the starting values
<i>ENDPOS</i>	For the range of X and Y values to be fitted, the row number 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 (column *VX* containing X values and column *VY* containing Y values):

$$Y = \text{REGFITXINT} + \text{REGFITSLP} * X$$

where REGFITSLP and REGFITXINT are slope and Y-intercept constants.

- Returns the value of the X intercept for relationship above.
($-\text{REGFITXINT}/\text{REGFITSLP}$).

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 5-50)

[REGFITXINT](#) (on page 5-53)

REGFITYINT Formulator function

Performs a linear regression fit.

Usage

`REGFITYINT(VX, VY, STARTPOS, ENDPOS)`

<i>VX</i>	The name of any column (vector) listed under Columns
<i>VY</i>	The name of any column (vector) listed under Columns
<i>STARTPOS</i>	For the range of X and Y values to be fitted, the row number of the starting values
<i>ENDPOS</i>	For the range of X and Y values to be fitted, the row number 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 (column *VX* containing X values and column *VY* containing Y values):

$$Y = \text{REGFITYINT} + \text{REGFITSLP} * X$$

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 5-50)

[REGFITXINT](#) (on page 5-52)

FFT

The fast Fourier transform (FFT) formulas are used to convert a signal or sequential group of measurements from the time domain to the frequency domain. They can also be used to convert from the frequency domain to the time domain.

FFT_R Formulator function

Performs an FFT on the provided input arrays and then returns the real parts.

Usage

`FFT_R(REAL, IMAG)`

<i>REAL</i>	The real portion of a complex number input array
<i>IMAG</i>	The imaginary portion of a complex number input array

Details

If *REAL* is set to 0 or any other constant, the real portion of all values for the input array are set to 0 or the constant.

If *IMAG* is set to 0 or any other constant, the imaginary portion of all values for the input array are set to 0 or the constant.

If either *REAL* or *IMAG* is not a power of 2, the input arrays are adjusted so that they are the same power of 2.

The output of an `FFT_R` formula is the real component of the calculated FFT, with an output size of a power of 2. The output is 0 Hz to $-F_s/2$, where F_s is the sampling frequency, such as the following:

... [0, 1, ..., n/2-1, -n/2, ..., -1] * F_s/n

If the input is invalid or the input size is less than 2, #REF is returned.

Also see

- [FFT_FREQ Formulator function](#) (on page 5-56)
- [FFT_FREQ_P Formulator function](#) (on page 5-57)
- [FFT_I Formulator function](#) (on page 5-55)
- [IFFT_I Formulator function](#) (on page 5-59)
- [IFFT_R Formulator function](#) (on page 5-58)
- [SMOOTH Formulator function](#) (on page 5-60)

FFT_I Formulator function

Performs an FFT on the provided input arrays and then returns the imaginary parts.

Usage

`FFT_I (REAL, IMAG)`

<i>REAL</i>	The real portion of a complex number input array
<i>IMAG</i>	The imaginary portion of a complex number input array

Details

If *REAL* is set to 0 or any other constant, the real portion of all values for the input array are set to 0 or the constant.

If *IMAG* is set to 0 or any other constant, the imaginary portion of all values for the input array are set to 0 or the constant.

If either *REAL* or *IMAG* is not a power of 2, the input arrays are adjusted so that they are the same power of 2. If possible, a message is displayed to indicate that the input arrays were adjusted.

The output of an `FFT_I` formula is the imaginary component of the calculated FFT, with an output size of a power of 2. The output is 0 Hz to $-F_s/2$, where F_s is the sampling frequency, such as the following:

... [0, 1, ..., n/2-1, -n/2, ..., -1] * F_s/n

If the input is invalid or the input size is less than 2, #REF is returned.

Also see

- [FFT_FREQ Formulator function](#) (on page 5-56)
- [FFT_FREQ_P Formulator function](#) (on page 5-57)
- [FFT_R Formulator function](#) (on page 5-54)
- [IFFT_I Formulator function](#) (on page 5-59)
- [IFFT_R Formulator function](#) (on page 5-58)
- [SMOOTH Formulator function](#) (on page 5-60)

FFT_FREQ Formulator function

Returns an array of positive and negative frequencies that correspond to the frequencies of an FFT output.

Usage

`FFT_FREQ (TIME, TOLERANCE)`

<i>TIME</i>	Input time array
<i>TOLERANCE</i>	The tolerance that is used to determine if the array is evenly spaced

Details

If *TIME* is not a power of 2, the input array is adjusted so that it is a power of 2. This is a single value, not an array.

The sampling period, or time step, is derived by computing the total period of the signal (after being adjusted if needed) and dividing by the number of samples. The samples are a power of 2.

The difference between two consecutive points is checked. If this difference is greater than the tolerance multiplied by the sampling period, an error is returned as #REF. This check happens for all points in the input array after the array has been adjusted to be a power of 2.

The output of an `FFT_FREQ` formula is 0 Hz to $-F_s/2$, where F_s is the sampling frequency and n is the size of the input array after being scaled to a power of 2, such as the following:

$\dots [0, 1, \dots, n/2-1, -n/2, \dots, -1] * F_s/n$

The output size is a power of 2.

#REF is returned if any one of the following are true:

- An input array delta between two points is greater than the tolerance value.
- The input size is less than 2.
- The average time step is less than or equal to zero.

Also see

[FFT_FREQ_P Formulator function](#) (on page 5-57)

[FFT_I Formulator function](#) (on page 5-55)

[FFT_R Formulator function](#) (on page 5-54)

[IFFT_I Formulator function](#) (on page 5-59)

[IFFT_R Formulator function](#) (on page 5-58)

[SMOOTH Formulator function](#) (on page 5-60)

FFT_FREQ_P Formulator function

This formula returns an array of the positive frequencies that correspond to the frequencies of an FFT output.

Usage

`FFT_FREQ_P (TIME, TOLERANCE)`

<i>TIME</i>	Input time array; adjusted to be a power of 2
<i>TOLERANCE</i>	The tolerance that is used to determine if the array is evenly spaced

Details

The output of this formula is a single value, not an array.

The sampling period, or time step, is derived by computing the total period of the signal, after being adjusted if needed, and dividing by the number of samples. The number of samples will be a power of 2.

The difference between two consecutive points is checked. If this difference is greater than the tolerance multiplied by the sampling period, an error is returned as #REF. This check happens for all points in the input array after the array has been adjusted to be a power of 2.

The output of an `FFT_FREQ_P` formula is $0 \text{ Hz to } (n/2-1) * Fs/n$, where Fs is the sampling frequency and n is the size of the input array after being scaled to a power of 2.

The output size is the input array size divided by two.

#REF is returned if any one of the following occurs:

- An input array delta between two points is greater than the tolerance value.
- The input size is less than 2.
- The average time step is less than or equal to zero.

Also see

[FFT_FREQ](#) (on page 5-56)

[FFT_I Formulator function](#) (on page 5-55)

[FFT_R Formulator function](#) (on page 5-54)

[IFFT_I Formulator function](#) (on page 5-59)

[IFFT_R Formulator function](#) (on page 5-58)

[SMOOTH Formulator function](#) (on page 5-60)

IFFT_R Formulator function

Performs an inverse FFT on the provided input arrays and then returns the real parts scaled by $1/N$, where N is the number of samples.

Usage

`IFFT_R (REAL, IMAG)`

<i>REAL</i>	The real portion of a complex number input array
<i>IMAG</i>	The imaginary portion of a complex number input array

Details

If *REAL* is set to 0 or any other constant, the real portion of all values for the input array are set to 0 or the constant.

If *IMAG* is set to 0 or any other constant, the imaginary portion of all values for the input array are set to 0 or the constant.

If either *REAL* or *IMAG* is not a power of 2, the input arrays are cut so that they are the same power of 2.

The output of an `IFFT_R` formula is the real component of the calculated inverse FFT after being scaled by $1/N$. The output size is a power of 2.

The output is 0 Hz to $-F_s/2$, where F_s is the sampling frequency, such as the following:

`... [0, 1, ..., n/2-1, -n/2, ..., -1] * F_s/n`

If the input is invalid or the input size is less than 2, #REF is returned.

Also see

- [FFT_FREQ Formulator function](#) (on page 5-56)
- [FFT_FREQ_P Formulator function](#) (on page 5-57)
- [FFT_I Formulator function](#) (on page 5-55)
- [FFT_R Formulator function](#) (on page 5-54)
- [IFFT_I Formulator function](#) (on page 5-59)
- [SMOOTH Formulator function](#) (on page 5-60)

IFFT_I Formulator function

Performs an inverse FFT on the provided input arrays and then returns the imaginary parts scaled by $1/N$, where N is the number of samples.

Usage

`IFFT_I (REAL, IMAG)`

<i>REAL</i>	The real portion of a complex number input array
<i>IMAG</i>	The imaginary portion of a complex number input array

Details

If *REAL* is set to 0 or any other constant, the real portion of all values for the input array are set to 0 or the constant.

If *IMAG* is set to 0 or any other constant, the imaginary portion of all values for the input array are set to 0 or the constant.

If either *REAL* or *IMAG* is not a power of 2, the input arrays are cut so that they are the same power of 2. If possible, a message is displayed to indicate that the input arrays were adjusted.

The output of an `IFFT_I` formula is the imaginary component of the calculated FFT after being scaled by $1/N$. The output size is a power of 2. The output is 0 Hz to $-F_s/2$, where F_s is the sampling frequency, such as the following:

... [0, 1, ..., $n/2-1$, $-n/2$, ..., -1] * F_s/n

If the input is invalid or the input size is less than 2, #REF is returned.

Also see

- [FFT_FREQ Formulator function](#) (on page 5-56)
- [FFT_FREQ_P Formulator function](#) (on page 5-57)
- [FFT_I Formulator function](#) (on page 5-55)
- [FFT_R Formulator function](#) (on page 5-54)
- [IFFT_R Formulator function](#) (on page 5-58)
- [SMOOTH Formulator function](#) (on page 5-60)

SMOOTH Formulator function

Performs digital filtering on an input array by zeroing out high frequency components.

Usage

`SMOOTH (X, PERCENT)`

<i>X</i>	The input data array; does not need to be a power of 2
<i>PERCENT</i>	The percent of high frequencies to smooth (0 to 100); this is a single value (not an array)

Details

The SMOOTH formula takes the FFT of the input signal, then uses the returned FFT array to zero out the frequency bin based on the percentage value. For example, if *PERCENT* is set to 25%, the top 25% frequency bins are zeroed. Note that the highest frequency bins are at the center of the array. After these bins are zeroed, this array is fed into the inverse FFT to return the value back to the time domain. The 0 Hz bin, which is the first item in the array, is never zeroed.

If *PERCENT* is set to 0%, no frequencies are zeroed and the original signal is returned. If *PERCENT* is set to 100%, all but the 0 Hz frequency are zeroed (a straight line is produced).

Also see

- [FFT_FREQ Formulator function](#) (on page 5-56)
- [FFT_FREQ_P Formulator function](#) (on page 5-57)
- [FFT_I Formulator function](#) (on page 5-55)
- [FFT_R Formulator function](#) (on page 5-54)
- [IFFT_I Formulator function](#) (on page 5-59)
- [IFFT_R Formulator function](#) (on page 5-58)

Misc

The Misc Formulator function allows you to compare user-defined expressions.

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)`

<code><i>EXP1</i>, <i>EXP2</i>, <i>EXP3</i>, <i>EXP4</i></code>	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 $EXP1 \geq EXP2$, then *EXP4* is returned.

Example

```
CLIPCURRENT = COND(DRAIN1, 1E-6, DRAIN1, 1E-6)
```

Also see

None

Setting up site and subsite operation

In this section:

Introduction	6-1
Sites	6-1
Subsites	6-2
Configure sites	6-3
Configure subsite cycling	6-5
Run an individual subsite	6-32
Run a single site.....	6-33
Cycle a subsite.....	6-34
Multi-site execution	6-35

Introduction

This chapter describes how to configure sites and subsites. It also describes how to set up subsite cycling for stress and measure cycles.

For additional information on setting up prober movement between project sites and subsites, refer to *Model 4200A-SCS Prober and External Instrument Control* for detail.

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

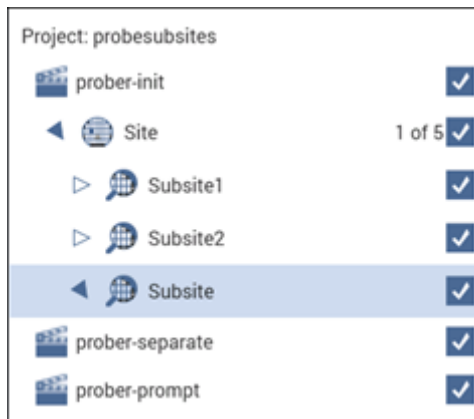
You can add subsites as show below. If you are using a prober, you can also use `probesubsites` from the Project Library to start with a site and subsite template with prober actions.

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 143: Project tree with subsites example



Configure sites

For sites, you can set the following options:

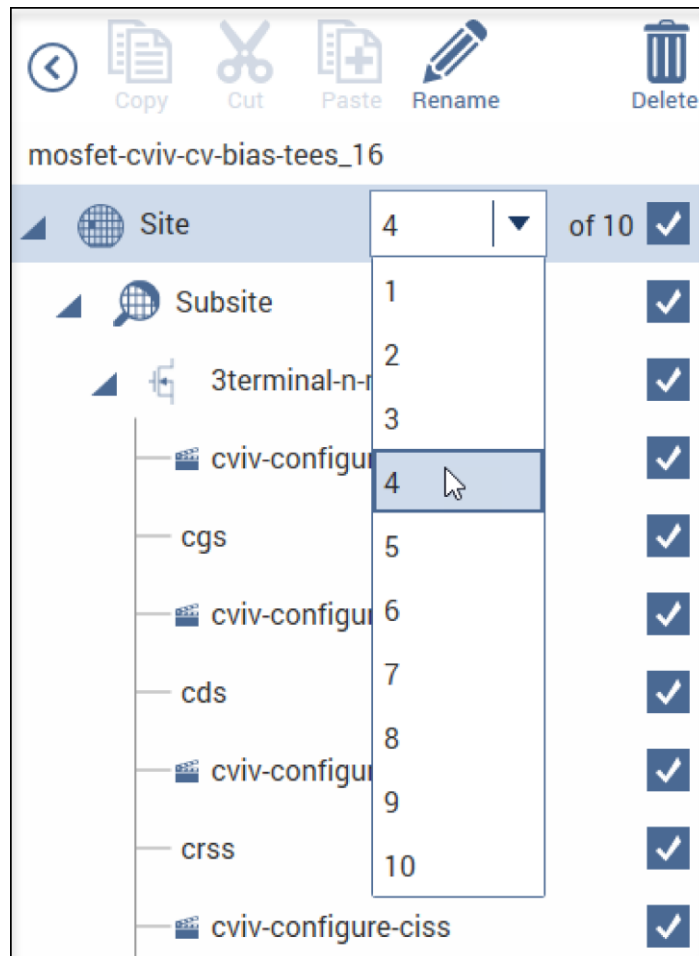
- **Number of Sites:** The maximum number of sites that can be tested. This is typically set to the number of sites that have been programmed in a prober controller.
- **Start Execution at Site:** The site where project execution starts. This is normally the same as the prober starting site number.
- **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**. If you are configuring your sites for Segment Stress mode, the maximum number of sites is 999.
4. Select **Save**.
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 144: 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 Analyze for the subsite.

The measured readings listed in the Analyze sheet 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 6-32) for details.

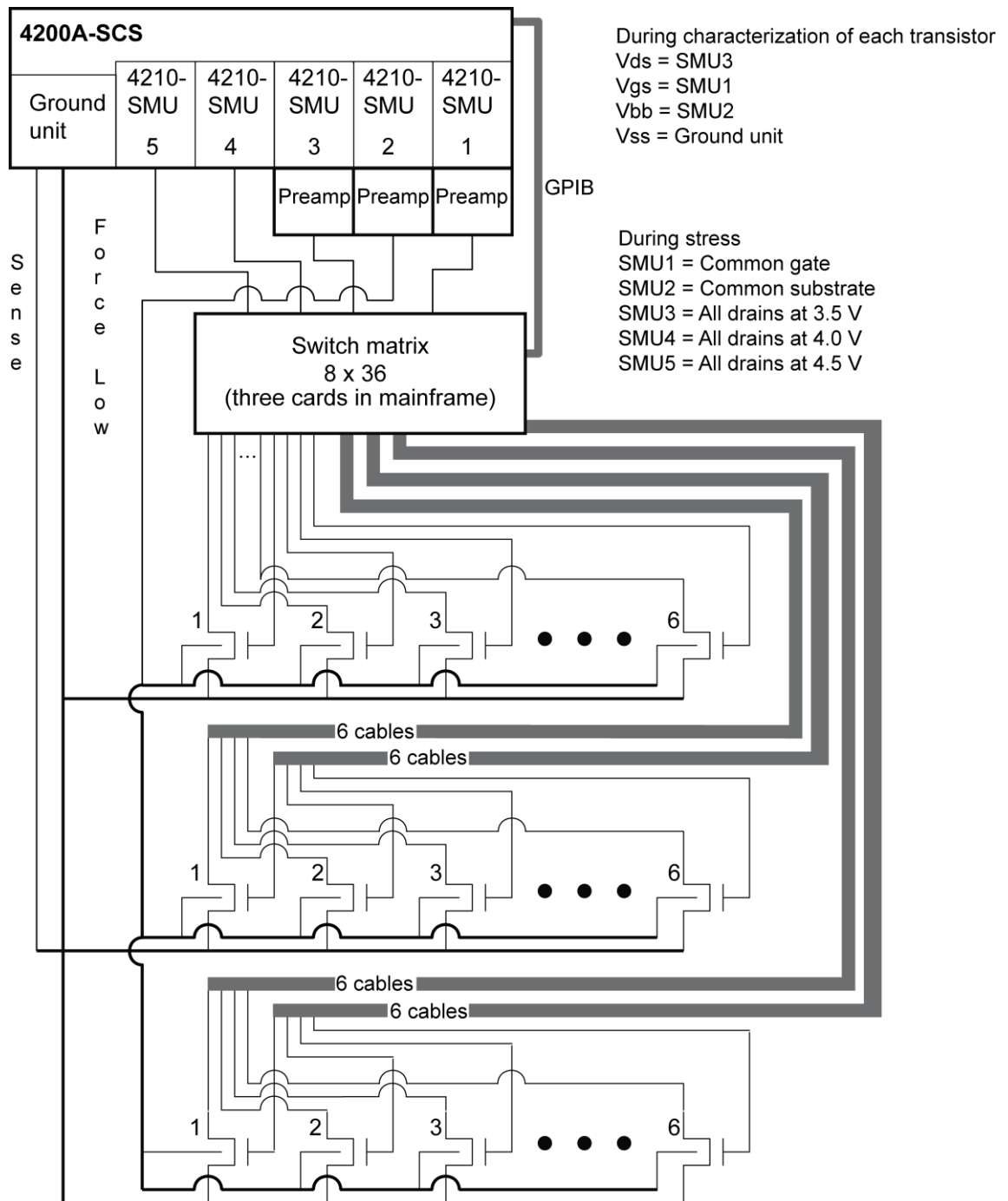
Stress mode integrates stressing with subsite cycling for testing. The first cycle is stress-free. For each subsequent cycle, the devices in the subsite are stressed with voltage or current for a specified period. After the stress period expires, the tests in the subsite are run. Device stressing includes dc voltage stress, dc current stress, ac voltage stress, and segment stressing. The 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. The 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. The dc bias voltage and current limit for the device can be provided by the SMUs in the system.

Connect devices for stress/measure cycling

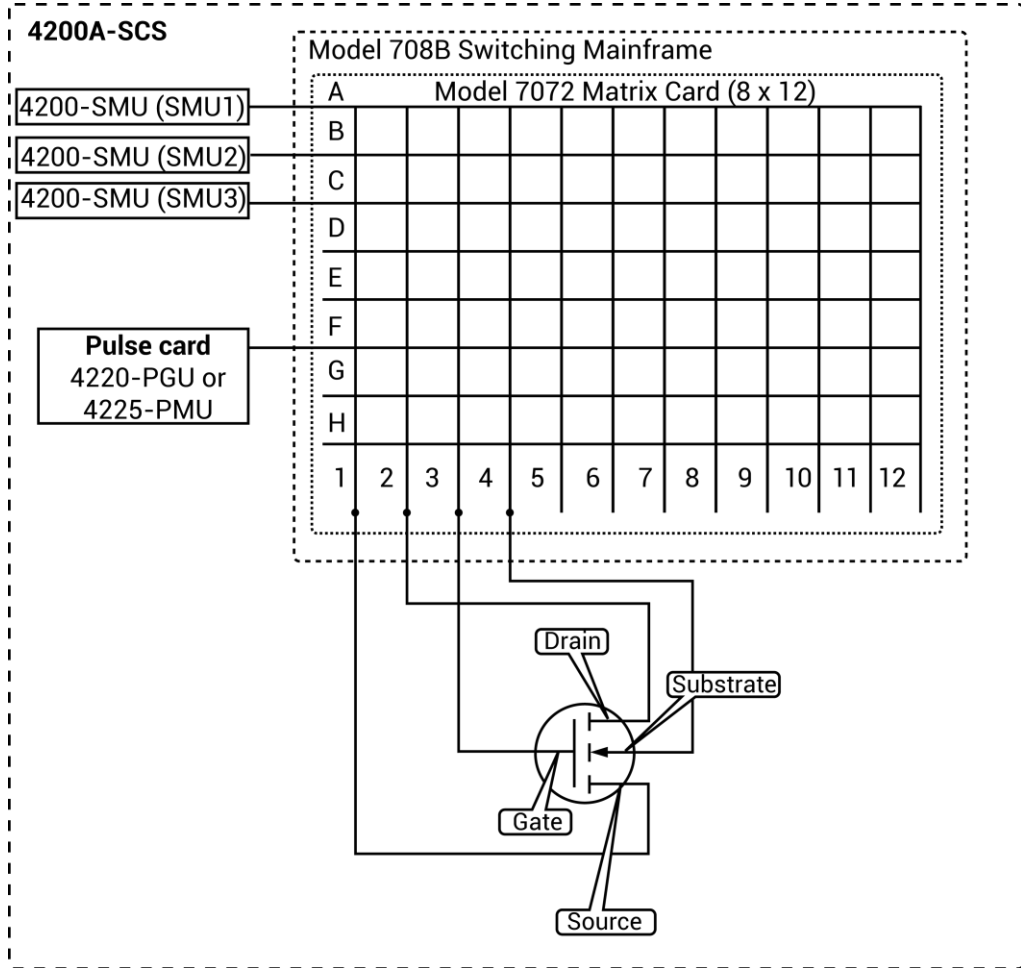
Devices that are stress/measure cycled in parallel are connected through a switching system. The following figure shows an example of connections for an HCI evaluation.

Figure 145: Stress / measure wiring example



Connections for matrix card

Figure 146: 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 147: AC pulse stress-measure — hardware setup block diagram for stress

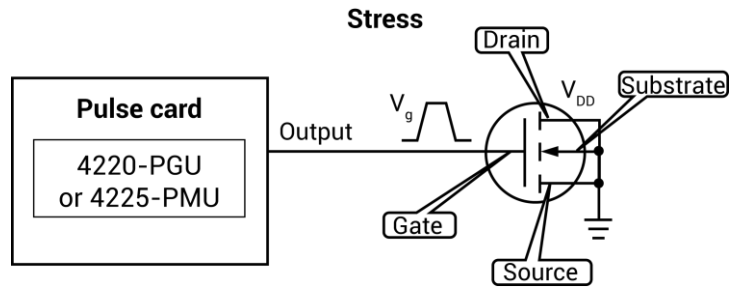
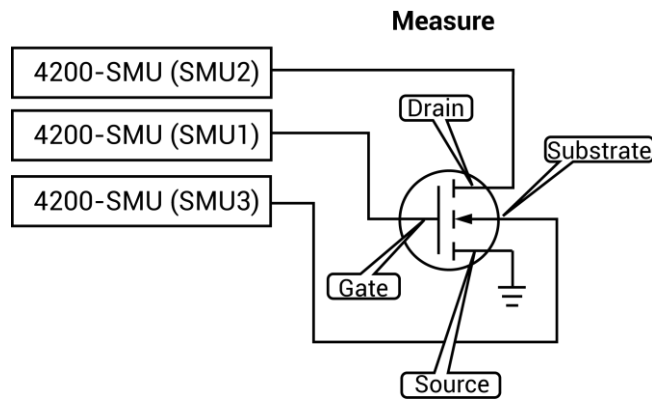
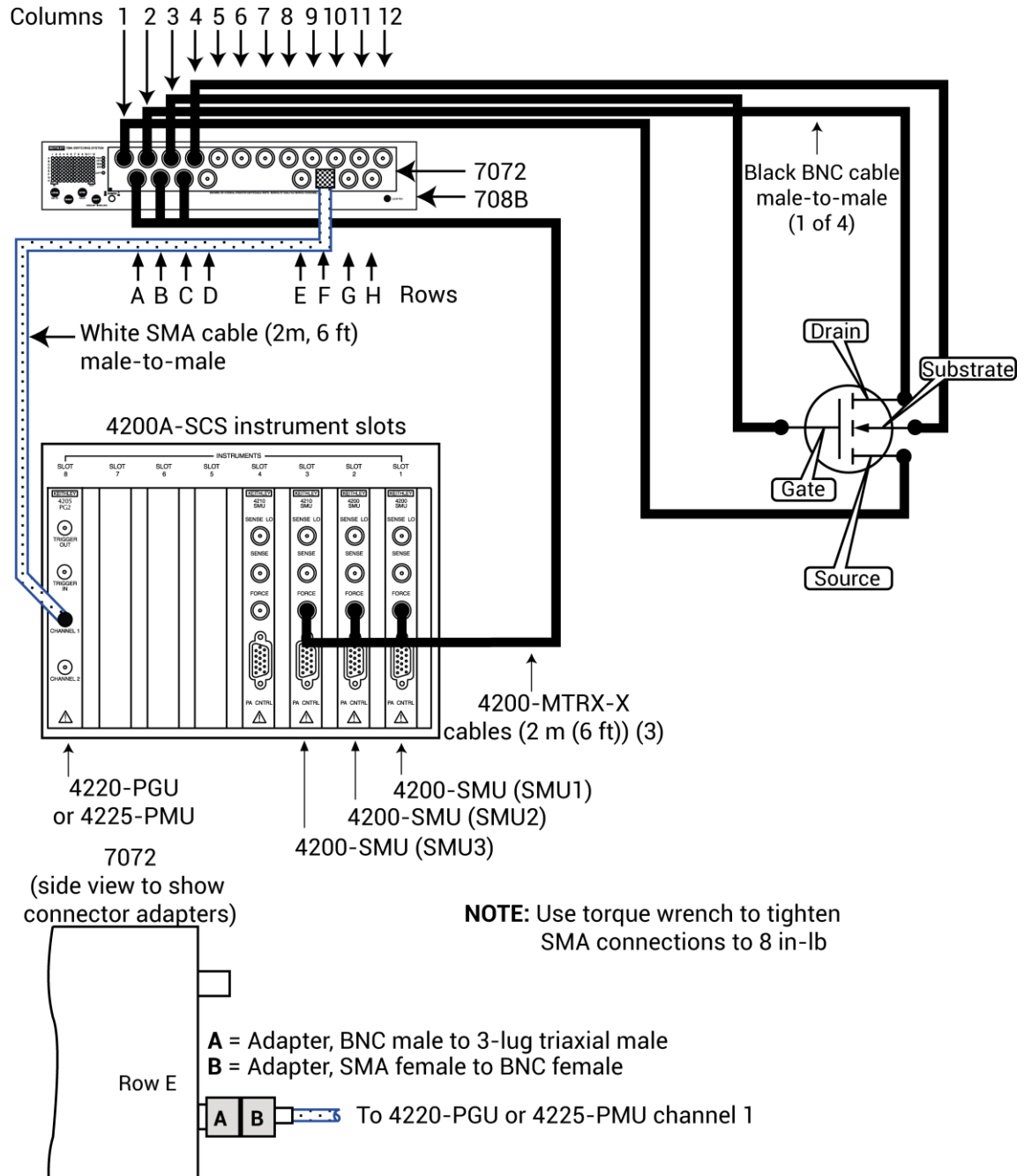


Figure 148: AC Pulse stress-measure — hardware setup block diagram for measure



Connections for system hardware

Figure 149: AC Pulse stress-measure — hardware connections



Set up the Subsite Operation

You can select one of the following Subsite Operations:

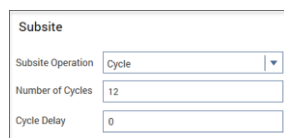
- **None:** No cycling or stressing operation is performed on the subsite.
- **Cycle:** Loops through the subsite tests without stressing the devices.
- **Stress:** Test-stress-test-stress cycles, such as hot-carrier injection (HCI) or negative bias temperature instability (NBTI) studies. You can use 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 150: Stress Mode Setup pane



The screenshot shows a 'Subsite' configuration window with three fields:

Subsite	
Subsite Operation	Cycle
Number of Cycles	12
Cycle Delay	0 s

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.

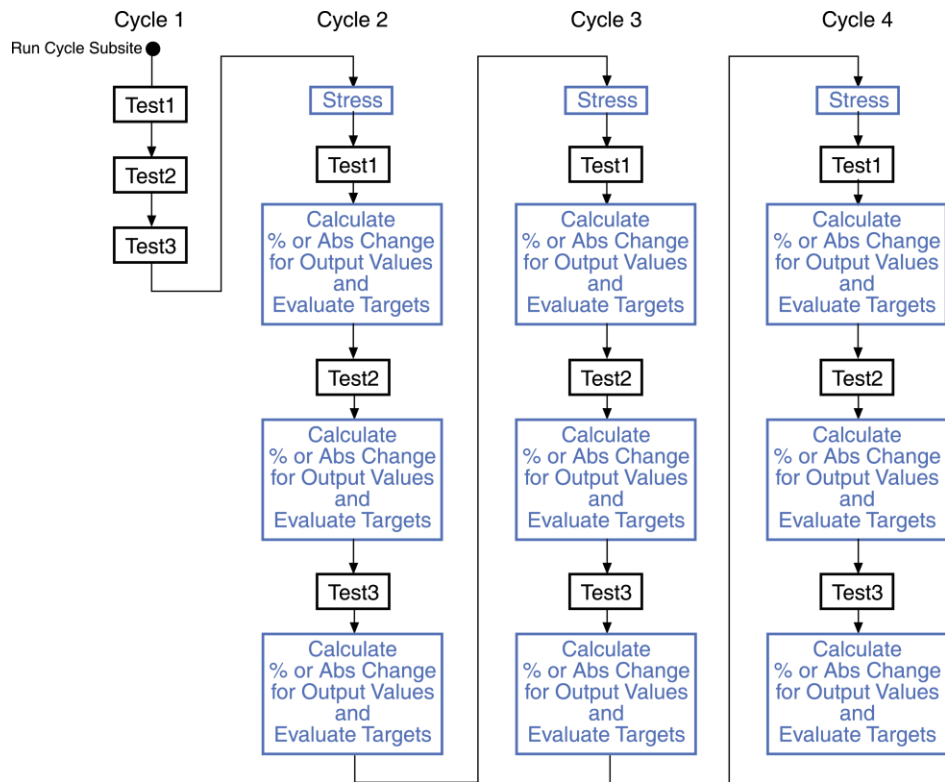
About the Stress operation

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 151: 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 6-19) 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 6-17) and [AC voltage stressing](#) (on page 6-18). Similar operations apply to other types of stress-measure studies.

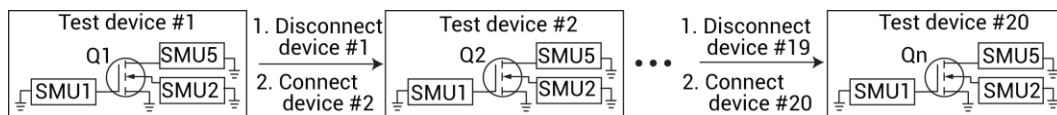
NOTE

For information about ac stress for wafer-level reliability, refer to [Wafer-level reliability testing](#) (on page 8-1).

Summary of an HCI evaluation:

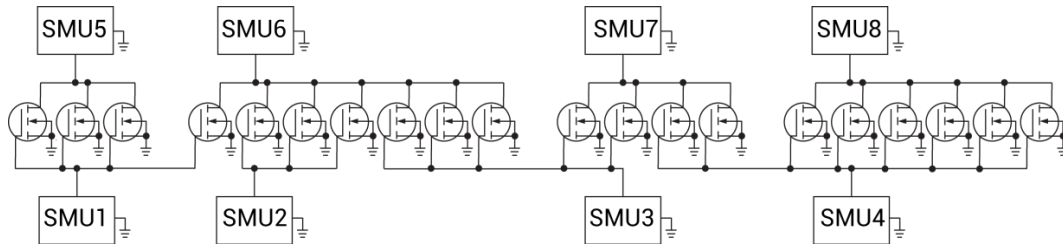
1. Use the switch matrix to automatically connect the SMUs to device 1.
2. Run pre-stress parametric tests on each device individually in device-number sequence, as shown in the following figure.

Figure 152: Pre-stress parametric tests



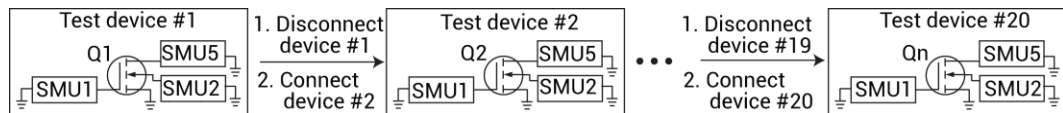
3. Disconnect all devices.
4. Use the switch matrix to automatically connect all devices to SMUs, as determined by the drain and gate voltages that were specified for each device.
5. Run stress cycle 1, which stresses all of the devices simultaneously, as shown in the following figure.

Figure 153: 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 154: Post-stress parametric tests

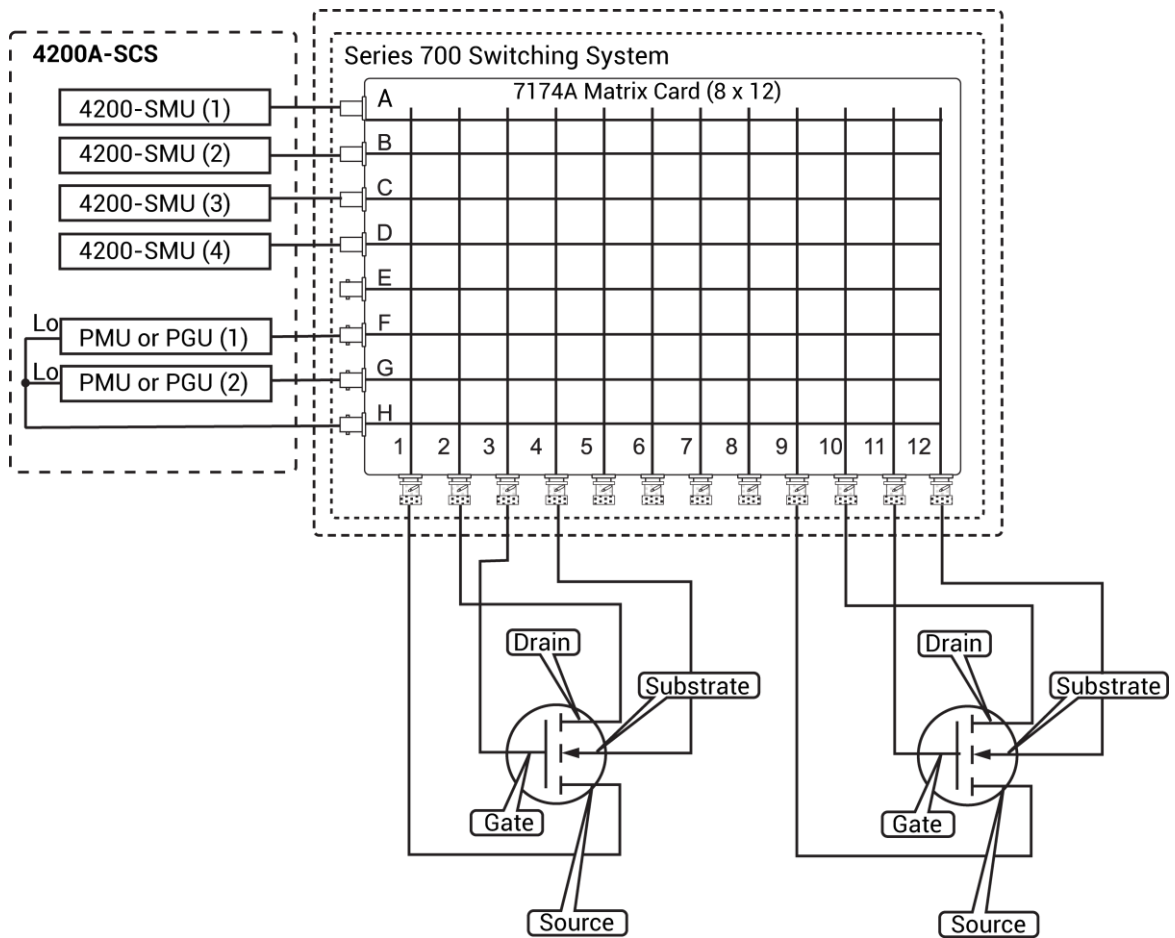


10. Disconnect all devices.
11. Reconnect all devices for stress cycle 2.
12. Run stress cycle 2.
13. Disconnect all devices.
14. Reconnect all devices for test cycle 2.
15. Wait for a 10 s to promote uniform pre-test decay for all devices.
16. Run test cycle 2.
17. Continue with stress and tests cycles (3, 4, ... n) until a device degrades to all enabled 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 following 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 155: Switch matrix for AC and DC voltage stress/measure system



Select a Subsite Mode

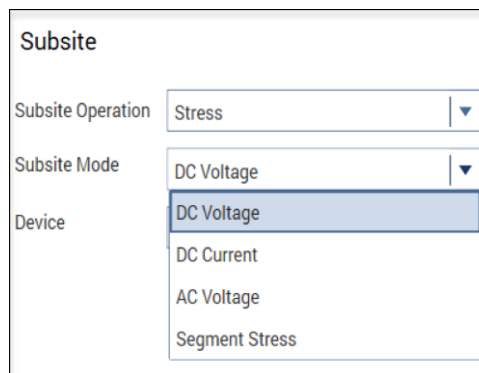
You can select four stress operation subsite modes. The modes are:

- 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 156: Selecting a subsite mode



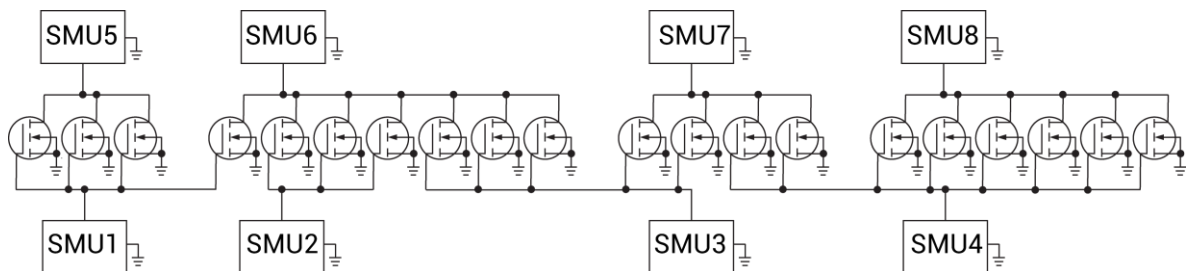
See the following topics in this section for a description of the subsite modes.

DC Voltage stressing

The stress algorithm built into Clarius 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 (V_g Stress) and a unique drain-stress bias voltage (V_d 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.

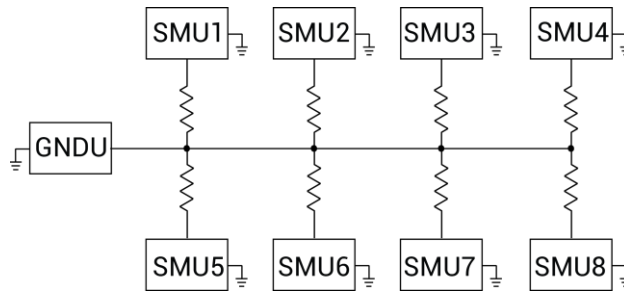
Figure 157: DC voltage stressing: 20 parallel-connected devices stressed at eight gate and drain voltages



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 158: EM test: Eight devices being current stressed by eight SMUs

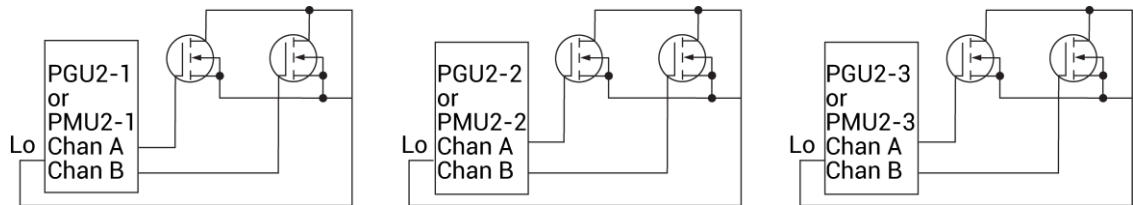


AC Voltage stressing

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 159: 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.

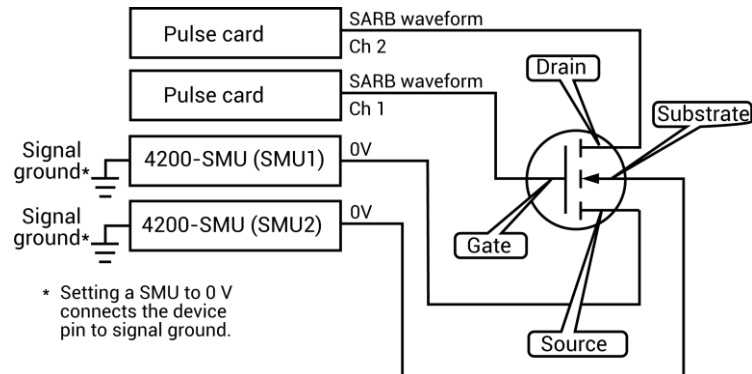
NOTE

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 or 4201-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 160: 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)

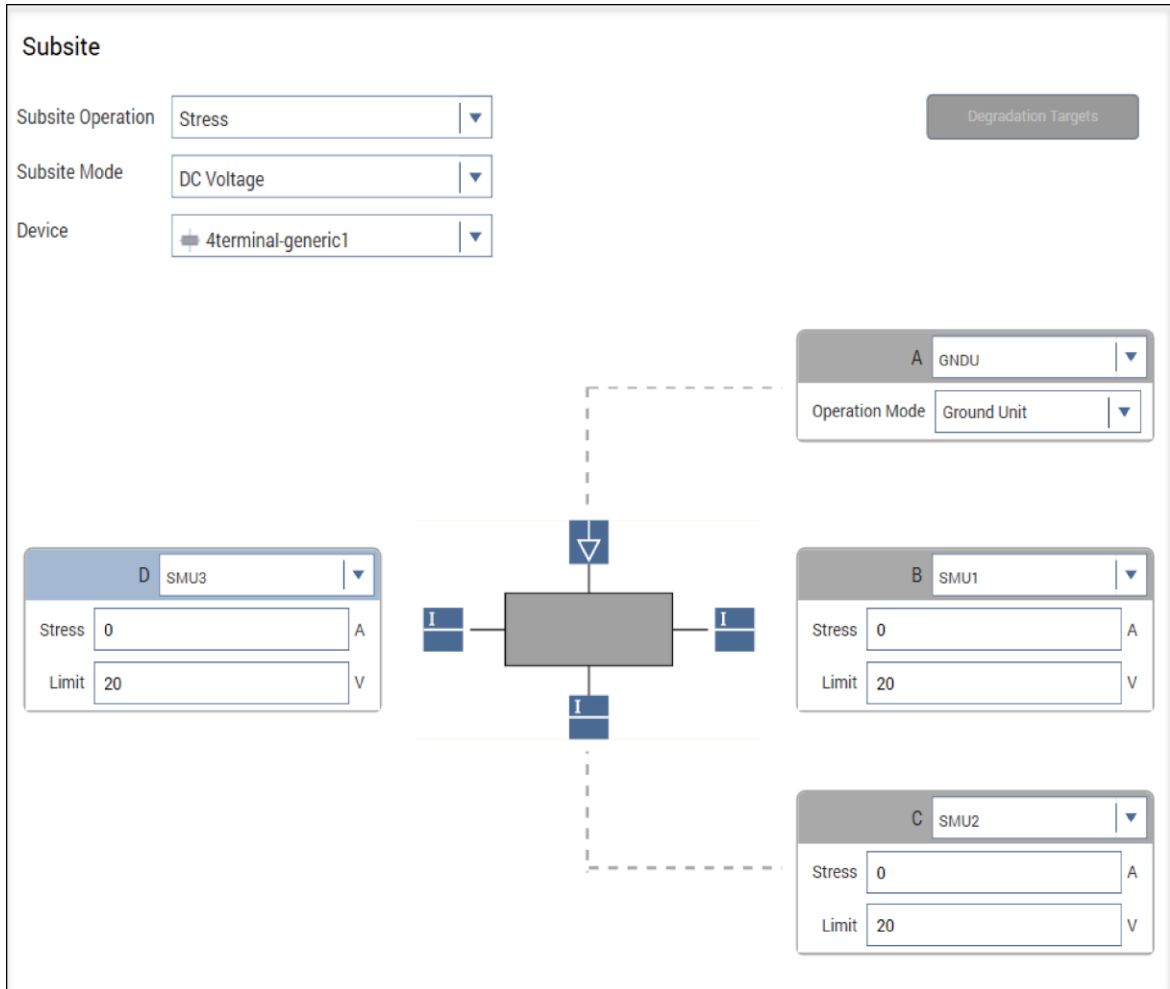
NOTE

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 161: Device with terminals assigned to instruments



2. Assign an instrument or function to each terminal of the device, as needed.

Figure 162: Selecting an instrument or function for the terminal

D	SMU1
Stress	0
Limit	0.1

- NONE
- GNDU
- SMU1
- SMU2
- SMU3
- SMU4
- PMU1-1
- PMU1-2
- PMU2-1
- PMU2-2

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 163: Warning for one instrument or function connected to multiple device terminals

Subsite Configuration Error:
Same instruments cannot be used more than once.

Subsite Operation: Stress

Subsite Mode: DC Voltage

Device: led

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 6-23).

NOTE

For additional device configuration options, see [Configure the Terminal Settings](#) (on page 6-29).

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.

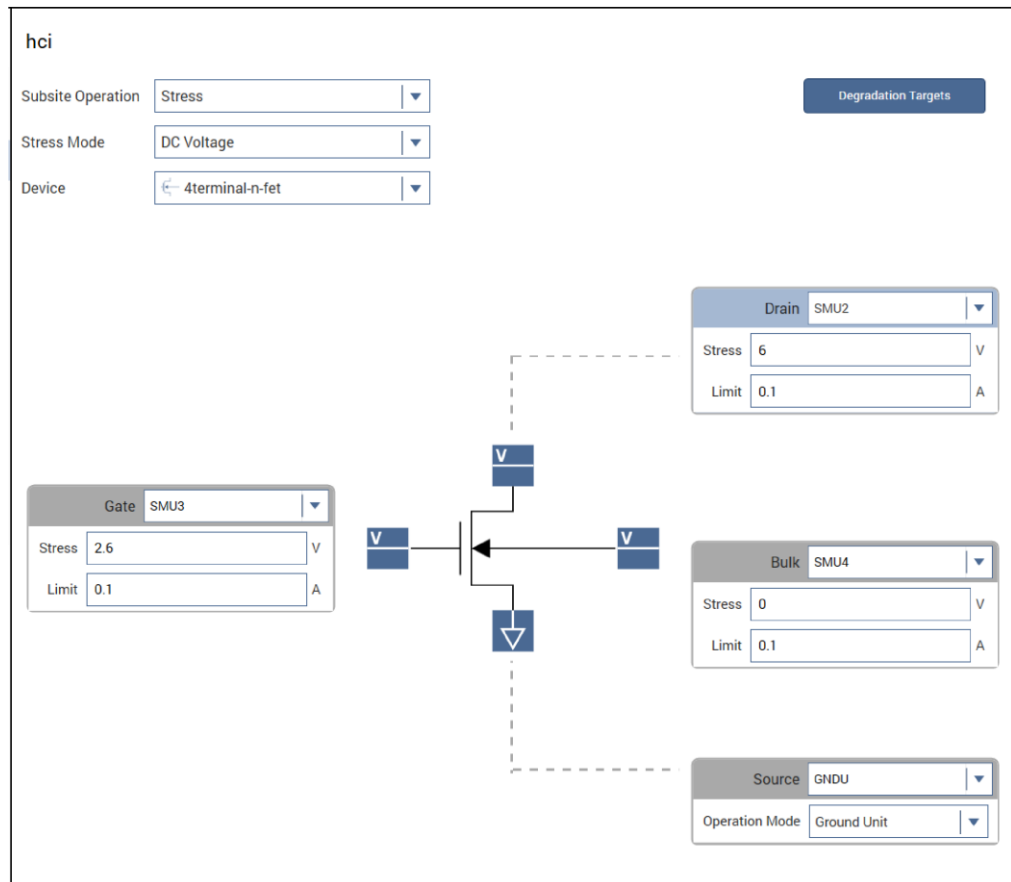
For example, if you select DC Voltage as a subsite mode and assign SMU1 to the Gate of a device terminal, you can adjust the stress in volts and limit in amperes. However, if you select the DC Current subsite mode, you will specify the stress in amperes and limit in volts.

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.

The following figure shows an example of DC voltage stress operation using SMUs.

Figure 164: DC voltage stressing using SMUs



Device pin connections for matrix cards

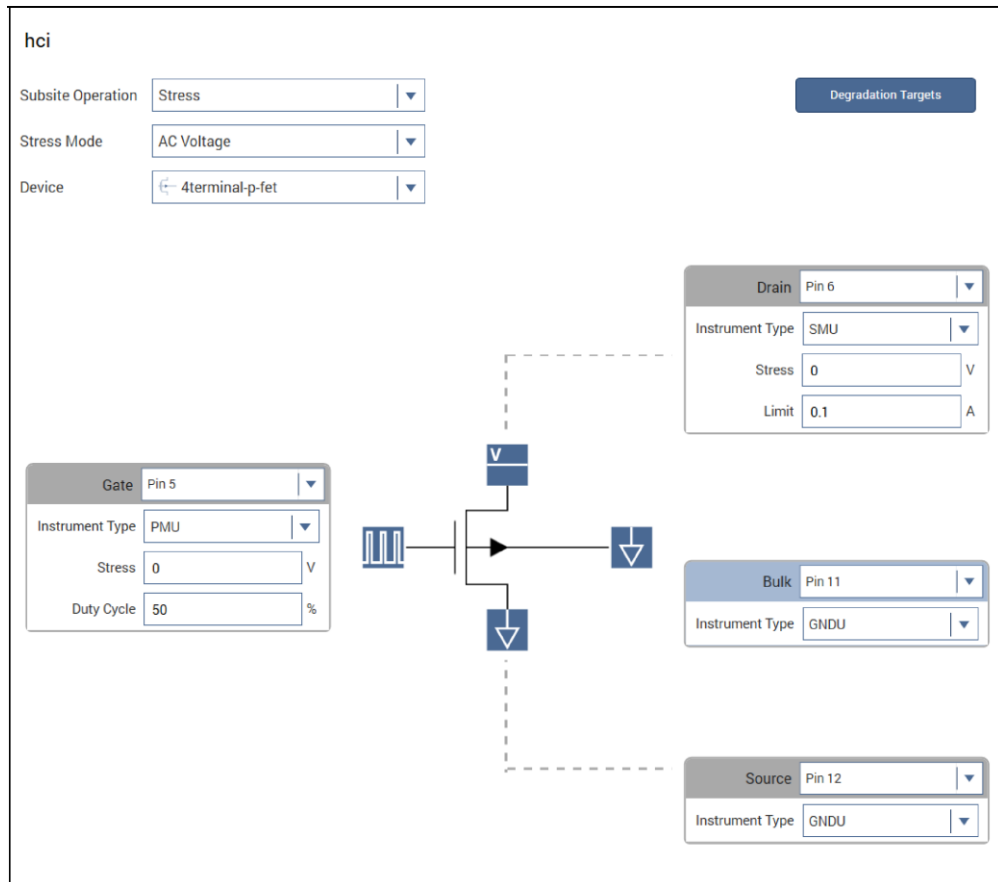
If you have a switch matrix connected to your test system, you will select the pin connection numbers and instrument type when you configure the DUT. The pin number assignments for the device must match the physical connections to the matrix card.

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 165: AC stress operation using a PMU connected through 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.

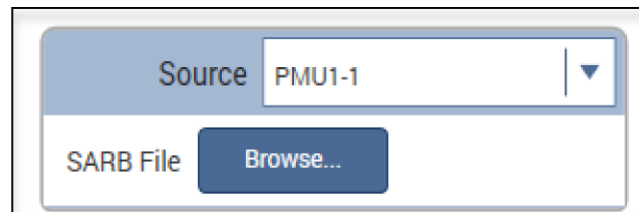
SARB waveform files have the extension `.ksf` and are normally stored at the following location:

```
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 166: 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.

Set up Linear and Log timing modes

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:
 - If you have selected **Linear** timing, set the **Number of Stresses**. This is the total number of stresses, up to 128.
 - 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 167: Set timing for linear or log measure mode

The screenshot shows the 'Stress Settings' pane with the following data:

Cycle	Cycle Stress Time	Stress Time
1	0.0 s	0.0 s
2	1.0 s	1.0 s
3	0.9 s	1.9 s
4	0.9 s	2.8 s
5	0.9 s	3.7 s
6	0.9 s	4.6 s

Below the table, the following settings are visible:

- Timing: Linear
- First Stress Time: 1 s
- Final Stress Time: 100 s
- Number of Stresses: 112
- Post Stress Delay: 0 s

Set up the List timing mode

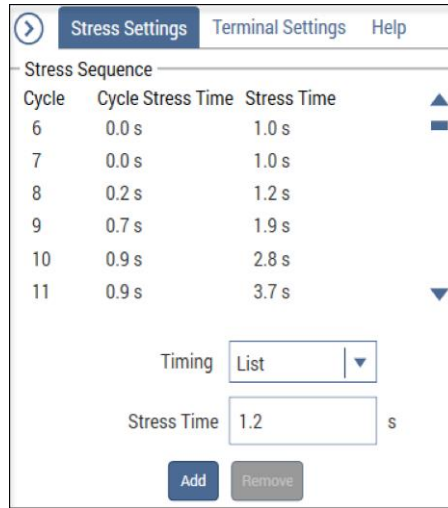
Information that is entered when Linear or Log timing is selected is shown when List is selected. You can use those settings as a starting point when you set up your list.

To set up the 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 168: Set timing for list measure mode

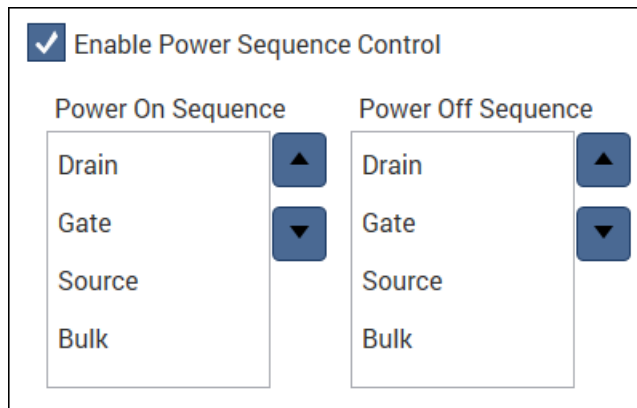


Power On and Off Sequence

When Subsite Operation is set to Stress, you can set the sequence that the instruments follow for powering on and powering off your test device.

To set the sequence, select **Enable Power Sequence Control**, as shown in the following figure.

Figure 169: Adjusting the power on and power off sequence for a four-terminal device



To change the sequence, select a terminal and use the arrows to the right of the box to move it up or down in the sequence.

NOTE

The devices must be connected to match the order selected.

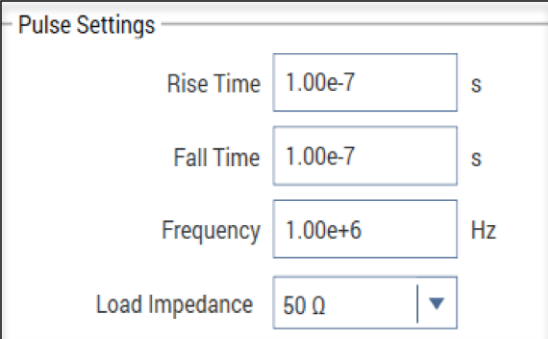
Pulse Settings

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 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 170: Pulse Settings dialog box



Parameter	Value	Unit
Rise Time	1.00e-7	s
Fall Time	1.00e-7	s
Frequency	1.00e+6	Hz
Load Impedance	50 Ω	

Configure the Terminal Settings

The Terminal Settings pane lets you further configure the terminals of a device. 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 6-25) for more information.

See the following graphics for examples.

Figure 171: Terminal settings for a diode assigned to a PMU in DC Voltage subsite mode

The screenshot shows the 'Terminal Settings' tab for an 'Anode' terminal. Under the 'Force' section, the 'Stress' is set to 1 V and the 'Limit' is set to 0.1 A. Under the 'Measure' section, the 'Current' checkbox is checked, and the 'Range' is set to 'Auto'.

Figure 172: Terminal settings for a diode assigned to a PMU in AC Voltage subsite mode

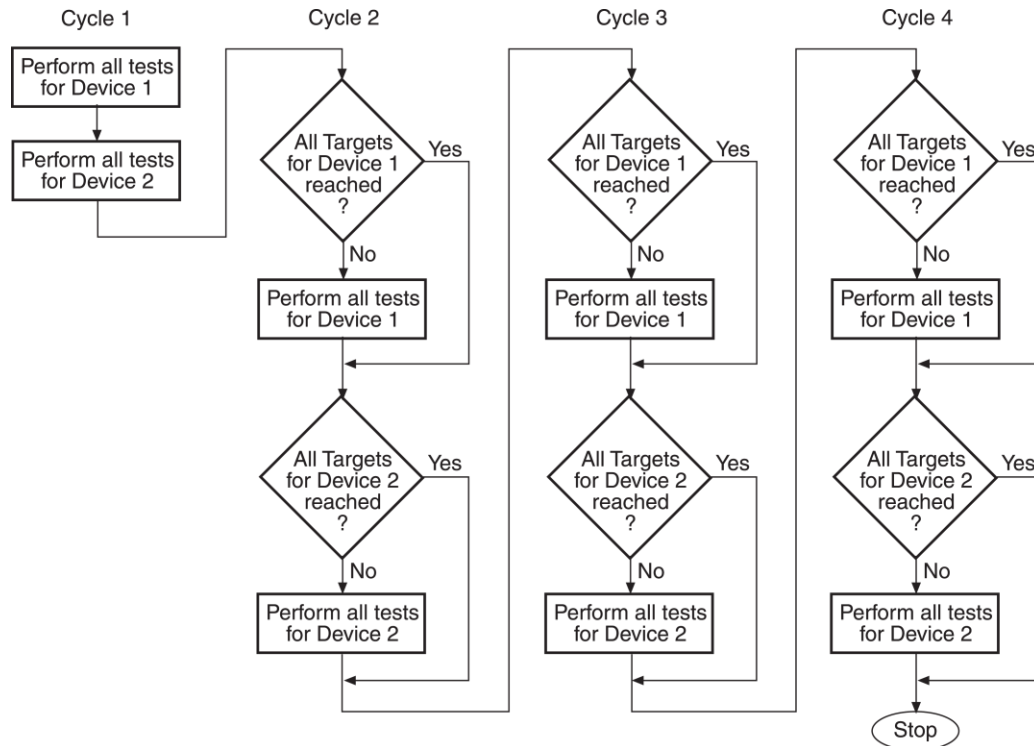
The screenshot shows the 'Terminal Settings' tab for a 'Cathode' terminal. Under the 'Force' section, the 'Stress' is set to 1 V and the 'Stress Low' is set to .2 V. The 'Duty Cycle' is set to 50%.

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.

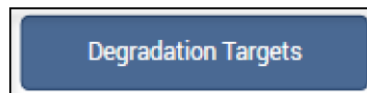
Figure 173: Target evaluation process (example for two devices, four cycles)



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 6-32).

When you have defined at least one output value, you can select Degradation Targets from the Subsite panel.

Figure 174: 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 6-31).

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 an individual subsite

When you run an individual subsite, only the components that are assigned to it run.

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.

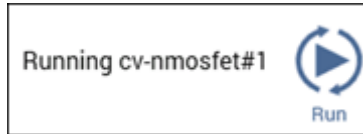
NOTE

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 175: 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.

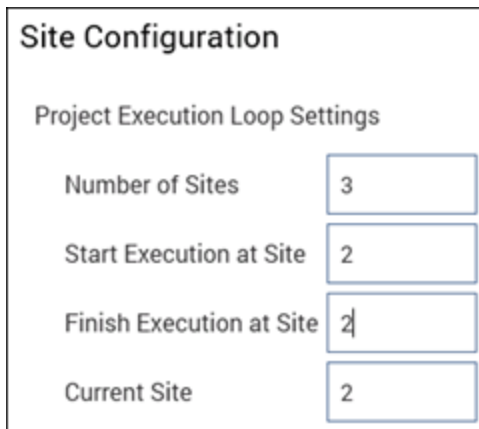
Run a single site

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:

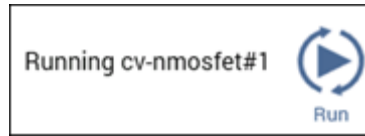
1. In the project tree, select the site.
2. Select **Configure**.
3. Set **Start Execution at Site** and **Finish Execution at Site** to the site you want to run. In the following example, only Site 2 is run when you select **Run**.

Figure 176: 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 177: 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 6-5).
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 6-5).
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:

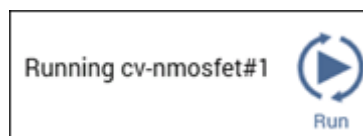
1. In the project tree, select the site.
2. Select **Configure**.
3. Set **Start Execution at Site** and **Finish Execution at Site** to the sites you want to run. In the following example, executing the site will run sites 3, 4, and 5.

Figure 178: Multi-site test sequence

Site Configuration	
Project Execution Loop Settings	
Number of Sites	<input type="text" value="10"/>
Start Execution at Site	<input type="text" value="3"/>
Finish Execution at Site	<input type="text" value="5"/>
Current Site	<input type="text" value="1"/>

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 179: 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.

User library descriptions

In this section:

Introduction	7-2
AVMControl user library	7-2
BeepLib user library	7-2
chargepumping user library	7-3
cvvulib user library	7-3
cvucompulib user library	7-4
cvuulib user library	7-4
DLCP user library	7-5
dmm-6500-7510-temp-ulib user library	7-5
flashulib user library	7-6
GateCharge user library	7-6
generic_gpib_ulib user library	7-7
generic_visa_ulib user library	7-8
hivcvulib user library	7-9
Hotchuck_Temptronics3010B user library	7-9
Hotchuck_Triotek user library	7-9
HP4284ulib user library	7-10
HP4294ulib user library	7-10
HP8110ulib user library	7-11
ki340xulib user library	7-11
KI42xxulib user library	7-11
KI590ulib user library	7-12
KI595ulib user library	7-12
ki622x_2182ulib user library	7-13
ki82ulib user library	7-14
LS336ulib user library	7-15
Matrixulib user library	7-15
MultiSegmentSweep_ulib user library	7-15
nvm user library	7-16
OVPCControl user library	7-17
parlib user library	7-18
pmuCompulib	7-18
pmuulib user library	7-19
PMU_examples_ulib user library	7-19
PMU_freq_time_ulib user library	7-20
PMU_PCRAM_ulib	7-21
PRBGEN user library	7-21
QSCVulib user library	7-22
RPM_ILimit_Control user library	7-22
utilities_ulib	7-22
van der Pauw user library	7-23
VLowFreqCV user library	7-24
wrlib user library	7-25
Winulib user library	7-25
AbortRetryIgnoreDialog user module	7-26
InputOkCancelDialog user module	7-28
OkCancelDialog user module	7-30
OkDialog user module	7-32
RetryCancelDialog user module	7-34

YesNoCancelDialog user module	7-36
YesNoDialog user module.....	7-38

Introduction

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 following table lists and briefly describes the user module.

AVMControl user module

User module	Description
SetAVMLevel	Sets the 4200-SMU, 4201-SMU, 4210-SMU, or 4211-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 following 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 following table lists and briefly describes the user modules.

chargepumping user modules

User module	Description
<code>AmplitudeSweep</code>	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.
<code>AmplitudeSweep_2SMU</code>	Same as the <code>AmplitudeSweep</code> user module, except that it uses a second SMU to apply a dc bias voltage to the source/drain terminals.
<code>BaseSweep</code>	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.
<code>BaseSweep_2SMU</code>	Same as <code>BaseSweep</code> user module, except it uses a second SMU to apply a DC bias voltage to the source/drain terminals.
<code>FallTimeLinearSweep</code>	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.
<code>FreqFactorSweep</code>	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.
<code>FreqLinearSweep</code>	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.
<code>RiseTimeLinearSweep</code>	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 following table lists and briefly describes the user modules.

cvivulib user modules

User module	Actions in Clarius	Description
<code>cviv_configure</code>	<code>cviv-configure</code>	This user module configures the CVIV relays and the display for each channel.
<code>cvu_cviv_comp_collect</code>	<code>cvu-cviv-comp-collect</code>	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 following table lists and briefly describes the user modules.

NOTE

If your configuration includes a 4200A-CVIV, use the [cvivulib](#) (on page 7-3) user library instead of this one.

cvucompulib user modules

User module	Description
<code>cvu_ConstantsFileSelect</code>	Selects the constants file that is used for a CVU ITM or UTM. The file must be created using <code>cvu_OSComp_collect</code> .
<code>cvu_OSComp_collect</code>	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.

cvuulib user library

The `cvuulib` user library contains user modules that perform V ac, V dc, and frequency sweeps for the 4210-CVU and 4215-CVU. The next table lists and briefly describes the user modules.

cvuulib user modules

User module	Description
<code>Sampling</code>	Samples a number of readings at a fixed ac voltage, dc voltage, and frequency.
<code>SweepACV</code>	Sweeps ac voltage on a CVU Instrument.
<code>SweepDCV</code>	Sweeps dc voltage on a CVU Instrument.
<code>SweepF_Log</code>	4215-CVU only: Logarithmic sweep of frequency.
<code>SweepF</code>	Sweeps frequency on a CVU Instrument.

DLCP user library

The DLCP user library contains a user module for making C-V measurement for drive-level capacitance profiling (DLCP). The following 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 or 4215-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 $\frac{1}{2}$ the p-p ac voltage.

dmm-6500-7510-temp-ulib user library

The dmm-6500-7510-temp-ulib user library contains modules that allow users to communicate with a Keithley DMM6500 or DMM7510 digital multimeter through USB to measure temperature. The library uses VISA to communicate with the external DMM. The following table briefly describes each module.

dmm-6500-7510-temp-ulib user modules

User module	Description
meas_smuI_dmmTemp	Measures the current on a two-terminal device using two SMUs or one SMU and the GNDU while the external DMM measures the temperature using a thermocouple.
meas_time_dmmTemp	Measures the temperature using an external DMM and records the time elapsed between measurements.

flashulib user library

The `flashulib` user library contains user modules for flash memory testing. The following table lists and briefly describes the user modules.

flashulib user modules

User module	Description
<code>configure_dc_flash</code>	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.
<code>double_pulse_flash</code>	Defines and outputs 1 to 8 waveforms that consist of two pulses that have independent 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.
<code>pmu_configure_dc_flash</code>	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.
<code>pmu_double_pulse_flash</code>	Defines and outputs 1 to 8 waveforms that consist of two pulses that have independent 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.
<code>pmu_single_pulse_flash</code>	Defines and outputs 1 to 8 waveforms that consists of two pulses that have independent 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.
<code>single_pulse_flash</code>	Defines and outputs 1 to 8 waveforms that consist of two pulses that have independent widths and levels. The waveforms are defined using line segments (Segment Arb mode of the pulse card). 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.

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 following table lists and describes the user module.

GateCharge user module

User module	Description
<code>gate_charge</code>	This module measures the gate charge of a power MOSFET using two source measure units (SMUs).

generic_gpib_ulib user library

The `generic_gpib_ulib` user library contains several modules that allow users to send a command or string of commands to an external instrument from Clarius. The external instrument must be connected using a GPIB cable and configured to communicate using GPIB. The following table briefly describes each module.

generic_gpib_ulib user modules

User module	Description
<code>query_double_gpib</code>	Sends a command to the external instrument and immediately reads the response. The response is assumed to be comma-separated double-format data and is parsed into up to four columns in the Clarius Analyze sheet.
<code>query_string_gpib</code>	Sends a command to the external instrument and immediately reads the response. The response is assumed to be a string and is returned to the Clarius Analyze sheet and the message console.
<code>read_double_gpib</code>	Reads from the output buffer of the external instrument. The response is assumed to be comma-separated double-format data and is parsed into up to four columns in the Clarius Analyze sheet.
<code>read_string_gpib</code>	Reads from the output buffer of the external instrument. The response is assumed to be a string and is returned to the Clarius Analyze sheet and output to the message console.
<code>write_string_gpib</code>	Writes a string command or commands to the external instrument. No response is read.

generic_visa_ulib user library

The `generic_visa_ulib` user library contains several modules that allow users to send a command or string of commands to an external instrument from Clarius. The external instrument must be connected using a USB cable and configured to communicate through USB. There is also a module to find VISA resource strings. The following table briefly describes each module.

generic_visa_ulib user modules

User module	Description
<code>find_visa_resource</code>	Returns the VISA resource strings for all external instruments connected to the 4200A-SCS through USB.
<code>query_double_visa</code>	Sends a command to the external instrument and immediately reads the response. The response is assumed to be comma-separated double-format data and is parsed into up to four columns in the Clarius Analyze sheet.
<code>query_string_visa</code>	Sends a command to the external instrument and immediately reads the response. The response is assumed to be a string and is returned to the Clarius Analyze sheet and the message console.
<code>read_double_visa</code>	Reads from the output buffer of the external instrument. The response is assumed to be comma-separated double-format data and is parsed into up to four columns in the Clarius Analyze sheet.
<code>read_string_visa</code>	Reads from the output buffer of the external instrument. The response is assumed to be a string and is returned to the Clarius Analyze sheet and output to the message console.
<code>write_string_gpib</code>	Writes a string command or commands to the external instrument. No response is read.

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 following table lists and briefly describes the user modules.

hivcvulib user modules

User module	Description
<code>CsRs_SweepV</code>	This module returns Cp-Gp and Cs-Rs parameters. It can be run using a CVU, SMU, and 4200A-CVIV, or using a CVU, SMU, and a 4205-RBT.
<code>multipleSMU_SweepV</code>	This module allows you to make high voltage C-V measurements up to 400 V using a CVU, SMUs, 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 is highly recommended that the SMUs at the gate and source have the same start and stop voltages to prevent damage to the device.
<code>CvsT</code>	Provides capacitance measurements as a function of time at a user-specified dc bias. You can measure capacitance up to 200 V dc bias with one 4205-RBT and one SMU. Additionally, you can measure capacitance up to 400 V dc bias with two 4205-RBTs and two SMUs.
<code>SweepV</code>	Uses one 4205-RBT to sweep a dc voltage across the DUT using the 4200-SMU, 4201-SMU, 4210-SMU, or 4211-SMU and measure the capacitance using the 4210-CVU or 4215-CVU. If two 4205-RBTs are used with the <code>SweepV</code> 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
<code>Settemp</code>	This routine controls the Temptronics thermal controller 3010B and other compatible models.

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
<code>SetChuckTemp</code>	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	Makes a single capacitance measurement.
CvSweep4284	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 probe to an OPEN calibration structure.
2. Call PhaseCal4294.
3. Call OpenCal4294.
4. Move probe to a SHORT calibration structure.
5. Call ShortCal4294.
6. Move probe 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)” in the *Model 4200A-SCS Setup and Maintenance User's Manual*.

NOTE

Details on Keysight 4294 operations are provided in the documentation provided by Keysight for the IMP meter.

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
<code>PguInit8110</code>	<code>pgul-init</code>	Initializes the pulse generator to the default setup.
<code>PguSetup8110</code>	<code>pgul-setup</code>	Sets the output pulse parameters.
<code>PguTrigger8110</code>	<code>pgu-trigger</code>	Specifies pulse count and trigger start of output.

ki340xulib user library

Used with the Keithley Instruments Series 3400 pulse/pattern generators.

ki340xulib user modules

User module	Description
<code>PguInit340x</code>	Initializes the 3401 or 3402 pulse generator to a specific state.
<code>PguSetup340x</code>	Defines the pulse timing and voltage settings. Once defined, the pulse can be triggered using the <code>PguTrigger340x</code> command.
<code>PguTrigger340x</code>	Triggers the pulse (or pulses) defined by the <code>PguSetup340x</code> 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
<code>Rdson42XX</code>	<code>rdson</code>	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 following table. 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	cable-compensate in the ivcvswitch project	Performs cable compensation using known capacitance source values.
Cmeas590	590-cmeas	Makes a single capacitance measurement.
CtSweep590	590-ctswEEP	Makes a capacitance versus time measurement.
CvPulseSweep590	590-cvpulsesweep	Makes capacitance versus voltage measurements using a pulse sweep.
CvSweep590	590-cvsweep	Makes capacitance versus voltage measurements using a staircase sweep.
DisplayCableCompCaps590	display-cap-file in the ivcvswitch project	Places capacitance source values in a spreadsheet.
LoadCableCorrectionConstants	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	save-cap-file in the ivcvswitch project	Saves entered capacitance source values to 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 following table.

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.

ki622x_2182ulib user library

The user modules in this library connect to a Keithley Model 6220 or 6221 Current Source and Model 2182 or 2182A Nanovoltmeter to make delta resistance measurements or differential conductance measurements.

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.

ki622x_2182ulib user modules

User module	Test name	Description
DeltaMeas	622x-2182a-delta-meas	Makes delta resistance measurements using a current source and nanovoltmeter. Delta measurements are a series of differential voltage measurements between two current values (high and low current). The 622X alternates the output current to form a square wave. The 2128/2182A makes three voltage measurements during the square waves and averages them together. This process eliminates the effects of thermal EMFs.
DiffCondSweep	None	Performs a differential conductance sweep using a current source and nanovoltmeter. This measurement uses a process similar to the delta measurement, a three-point moving average, to eliminate the effects of thermal EMFs.

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 following 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	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 <code>atexit()</code> function.
CableCompensate82	cable-compensate cablecomp	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.
CTsweep82	ct sweep	Measures capacitance as a function of time at a certain bias.
DisplayCableCompCaps82	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	qt sweep	Performs a quasistatic measurement sweep.
SaveCableCompCaps82	save-cap-file savecablecompfile	Saves entered capacitance source values in a file.
SIMCVsweep82	system82-cvsweep cvsweep	Performs simultaneous C-V sweep.

LS336ulib user library

The `LS336ulib` provides user modules that control the Lake Shore Cryotronics 336 Temperature Controller.

LS336ulib user modules

User module	Description
<code>heaterOff</code>	Turns off heater 1 and heater 2 and disables setpoint ramping on both outputs.
<code>setDelay_Dialog</code>	Either displays a window that contains the message you specified and an OK button, or performs the delay set by <code>WaitTime</code> .
<code>setSweepParams</code>	Generates the list of temperatures used by <code>setTemp</code> when the <code>setTemp</code> parameter <code>FlagMode</code> 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.
<code>setTemp</code>	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 <code>heaterOff</code> and <code>setSweepParam</code> 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
<code>ConnectPins</code>	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 4200-SMU and 4210-SMU instruments.

MultiSegmentSweep_ulib user modules

User module	Description
<code>MultiSegmentSweepI</code>	This module runs up to a four-segment current linear sweep.
<code>MultiSegmentSweepV</code>	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 following 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
<code>dcSweep</code>	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.
<code>doubleSweep</code>	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.
<code>doubleSweepSeg</code>	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.
<code>flashEndurance</code>	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.
<code>flashProgramErase</code>	Defines waveform for Programming and Erasing pulse for both drain and gate.
<code>getRes2</code>	This function returns the resistance of a two-terminal resistor. Voltage <code>v_force</code> is forced on the top side of the device; 0 V is forced to the low side. Measure current and reports resistance (V/I).
<code>pramEndurance</code>	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.
<code>pramSweep</code>	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.
<code>pulse_test</code>	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.
<code>pundEndurance</code>	This routine performs a device endurance test that runs fatigue pulse trains in between multiple PUND tests. A preliminary PUND test measurement 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 VFAT and one negative pulse to -VFAT. 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.

nvm user modules

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 <code>reramEndurance</code> routine performs a series of double sweeps using the same parameters used for the single sweeps, as described in the <code>reramSweep</code> 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 <code>reramSweep</code> 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.

SetOVPLLevel user module

User module	Description
SetOVPLLevel	Sets the 4200-SMU, 4201-SMU, 4210-SMU, or 4211-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
<code>beta</code>	Measure beta of bipolar transistor at the specified IE and VCB.
<code>fnddat</code>	Find data based on an x search or a y search.
<code>fntrg</code>	Decide if TRIGL or TRIGH should be used.
<code>gamma</code>	Returns the value of the body-effect parameter gamma obtained from two measurements of the threshold voltage at different substrate bias voltages.
<code>gm</code>	Estimate FET conductance (dI_d/dV_g) at V_{ds} and V_{gs} .
<code>gummel</code>	This test makes measurements that are similar to the <code>gummel</code> test in the Demo project for 3-terminal pnp BJTs.
<code>igvg</code>	This test makes measurements that are similar to the <code>ig-vg</code> test in the Demo project for 4-terminal MOSFETs.
<code>vceic</code>	This test makes measurements that are similar to the <code>vce-ic</code> test in the Demo project for 3-terminal BJTs.
<code>vdsid</code>	This test makes measurements that are similar to the <code>vds-id</code> test in the Demo project for 4-terminal MOSFETs.
<code>vgsid1</code>	This test makes measurements that are similar to the <code>vgs-id</code> test in the Demo project for 4-terminal MOSFETs.
<code>vtext</code>	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
<code>pmu_Offset_Current_Comp</code>	Collects offset current compensation data for both channels of the 4225-PMU.

pmuulib user library

The `pmuulib` user library contains user modules for configuring the 4225-RPM for the designated PMU channel. The following table lists and briefly describes the user modules.

pmuulib user modules

User module	Description
<code>RPM_configure</code>	This user module configures the 4225-RPM for the designated PMU channel.
<code>RPM_switch</code>	This user module was deprecated. Use the LPT command <code>rpm_config</code> for any RPM mode switching.

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 7-17) allows you to set the maximum voltage of the SMU.

PMU_examples_ulib user modules

User module	Description
<code>PMU_10ns_Pulse_Example</code>	This module sets up the PMU to continuously output pulses with 10 ns pulse widths. The PMU is in standard pulse mode with the pulse levels set at -1 V and 1 V.
<code>PMU_1Chan_Sweep_Example</code>	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.
<code>PMU_1Chan_Waveform_Example</code>	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.
<code>PMU_IV_sweep_Example</code>	This module is a functional programming reference to illustrate the basic LPT commands that are needed to perform a single V_d - I_d 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).
<code>PMU_PulseWaveform_FileSave_Example</code>	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 <code>*.csv</code> 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_examples_ulib user modules

User module	Description
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_8ch	This module configures multi-sequence, multi-segment waveform generation (Segment ARB) on eight channels using four 4225-PMU cards 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 up to four SMUs.
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.
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 <code>PMU_IV_Sweep_Example</code> .

PMU_freq_time_ulib user library

The user modules in this library are used to take evenly spaced measurements with the PMU for use with fast Fourier transform (FFT) computations.

PMU_freq_time_ulib user modules

User module	Description
PMU_gateWaveform	Outputs a defined number of pulsed waveforms from channel 1 of the PMU and measures the resulting current with channel 2.
PMU_sampleRate	Biases constant voltage using PMU channels 1 and 2 and measures the voltage on channel 1 and the current on channel 2. The number of samples returned depends on the entered sampling time and sample rate

PMU_PCRAM_ulib

The user modules in the `PCM_PCRAM_ulib` library provide examples of how the PMUs can be implemented in the characterization of PRAM elements.

PMU_PCRAM_ulib user modules

User module	Description
<code>pram_pulse_ilimit</code>	Simplifies the generation of segments when using the PMU. Forced voltage and current values are collected from the ForceCh and MeasureCh channels.
<code>pram_sweep_ilimit</code>	Provides an example of how the PMUs can be implemented in the characterization of PRAM elements. It allows specification of four pulses in one waveform. The parameters of these pulses are determined by the user and the SET pulse current values can be swept to generate RI and IV charts. It also demonstrates output debug information on voltage and currents for both PMU channels for any iteration of the sweep.
<code>pram_sweep</code>	Provides an example of how the PMUs can be implemented in the characterization of PRAM elements. It allows specification of four pulses in one waveform. The parameters of these pulses are determined by the user and the SET pulse amplitude can be swept to generate RI and IV charts. It also demonstrates output debug information on voltage and currents for both PMU channels for any iteration of the sweep.

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
<code>PrChuck</code>	<code>prober-contact</code>	Directs the prober to have the probe pins make contact with the wafer or separate the pins from the wafer.
<code>PrInit</code>	<code>prober-init</code>	Initializes the prober with die size, first coordinate (X and Y), units (mm or mils), and mode information.
<code>PrMovNxt</code>	<code>prober-move</code>	In learn mode, the <code>PrMovNxt</code> command causes the prober to move to the next site after inking.
<code>PrSSMovNxt</code>	<code>prober-ss-move</code>	In learn mode, the <code>PrSSMovNxt</code> 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
<code>meas_qscv</code>	<code>ramprate-cvsweep</code>	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
<code>Do_RPM_ILimit_Cal</code>	Performs a short-term calibration of the current clamp of properly-equipped 4225-RPMs.
<code>Get_RPM_ILimit_DAC_Value</code>	Do not set individually. This is used by <code>Set_RPM_ICompliance</code> .
<code>isLimitSupported</code>	Do not set individually. Used by <code>Do_RPM_ILimit_Cal</code> , <code>Get_RPM_ILimit_DAC_Value</code> , <code>Set_RPM_ICompliance</code> , and <code>OpenLimit</code> .
<code>OpenLimit</code>	Do not set individually.
<code>Set_RPM_ICompliance</code>	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
<code>Delay_second</code>	Enter delay time in seconds.

van der Pauw user library

The `vdplib` user library contains user modules for measuring the surface resistivity and volume resistivity of semiconductor material using the van der Pauw (vdp) technique.

vdplib user modules

User module	Description
<code>hall_coefficient</code>	Determines the Hall coefficient (RH) and mobility (μH) of a material using four SMUs.
<code>hall_coefficient_cviv</code>	Determines the Hall coefficient (RH) and mobility (μH) of a material using four SMUs and using the 4200A-CVIV.
<code>resistivity_surface</code>	Measures surface resistivity using four SMUs.
<code>resistivity_surface_cviv</code>	Measures surface resistivity using four SMUs and using the 4200A-CVIV.
<code>resistivity_volume</code>	Measures volume resistivity using four SMUs.
<code>resistivity_volume_cviv</code>	Measures volume resistivity using four SMUs and using the 4200A-CVIV.

VLowFreqCV user library

The VLowFreqCV user library contains user modules that are used for very low frequency C-V characterization. The next list 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 following table.

wlrlib user modules

User module	Description
<code>llsql</code>	Performs simple linear regression.
<code>qbd_rmpv</code>	Performs a charge-to-breakdown test using the QBD V-ramp test.
<code>qbd_rmpj</code>	Performs a charge-to-breakdown test using the QBD J-ramp test.

For more information, refer to [Wafer-Level Reliability Testing](#) (on page 8-1).

Winulib user library

The `Winulib` user library provides 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 7-26)	<code>abortretryignoredialog</code>	This user module creates a dialog box with Abort, Retry, and Ignore decision prompts.
InputOkCancelDialog (on page 7-28)	<code>inputokcanceledialog</code>	This user module creates a dialog box that can prompt for up to four input parameters.
OkCancelDialog (on page 7-30)	<code>okcanceledialog</code>	This user module creates a dialog box that provides OK or Cancel decisions.
OkDialog (on page 7-32)	<code>okdialog</code>	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 7-34)	<code>retrycanceledialog</code>	This user module creates a dialog box that presents Retry or Cancel decisions.
YesNoCancelDialog (on page 7-36)	<code>yesnocanceledialog</code>	This user module creates a dialog box that contains up to four lines of text and Yes, No, or Cancel decisions.
YesNoDialog (on page 7-38)	<code>yesnodialog</code>	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.

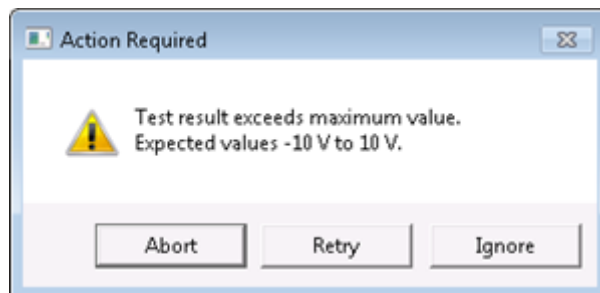
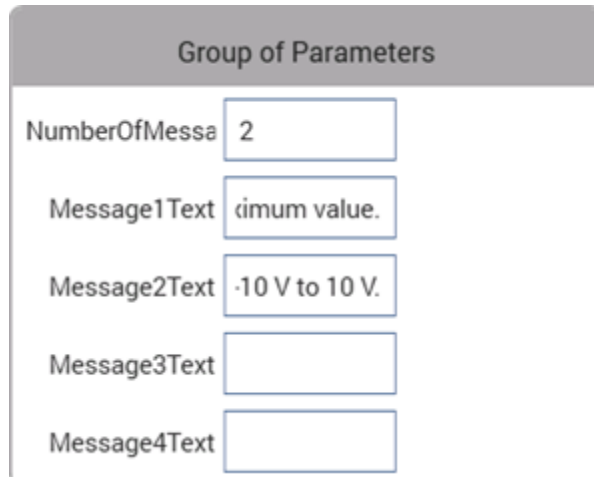
Usage

```
status = AbortRetryIgnoreDialog(int NumberOfMessages, char *Message1Text,
    char *Message2Text, char *Message3Text, char *Message4Text);
```

<i>status</i>	Returned values; see Details
<i>NumberOfMessages</i>	The number of text lines to display
<i>Message1Text</i>	The text to display on the first line of the dialog box; this line must be less than 40 characters
<i>Message2Text</i>	The text to display on the second line of the dialog box; this line must be less than 40 characters
<i>Message3Text</i>	The text to display on the third line of the dialog box; this line must be less than 40 characters
<i>Message4Text</i>	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

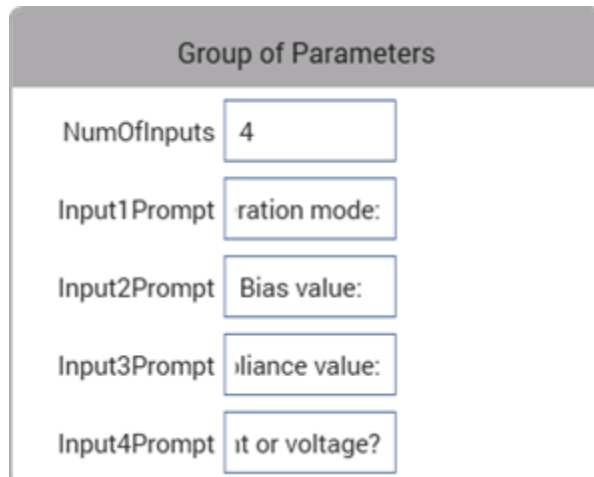
```
status = InputOkCancelDialog(int NumOfInputs, char *Input1Prompt, char *Input1,
    char *Input2Prompt, char *Input2, char *Input3Prompt, char *Input3, char
    *Input4Prompt, char *Input4);
```

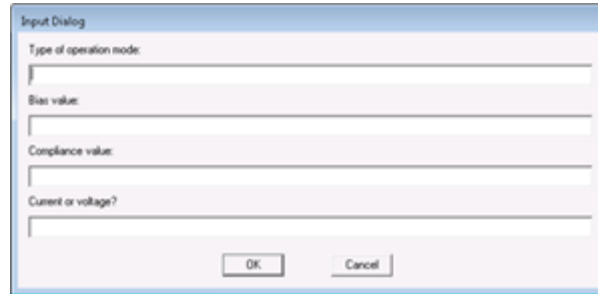
<i>status</i>	Returned values; see Details
<i>NumberOfInputs</i>	The number of text lines to display
<i>Input1Prompt</i>	The text to display on the first line of the dialog box; this line must be less than 40 characters
<i>Input1</i>	A character buffer for the first user input field; any text that the user inputs in the first displayed field is stored here
<i>Input2Prompt</i>	The text to display on the second line of the dialog box; this line must be less than 40 characters
<i>Input2</i>	A character buffer for the second user input field; any text that the user inputs in the second displayed field is stored here
<i>Input3Prompt</i>	The text to display on the third line of the dialog box; this line must be less than 40 characters
<i>Input3</i>	A character buffer for the third user input field; any text that the user inputs in the third displayed field is stored here
<i>Input4Prompt</i>	The text to display on the fourth line of the dialog box; this line must be less than 40 characters
<i>Input4</i>	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.

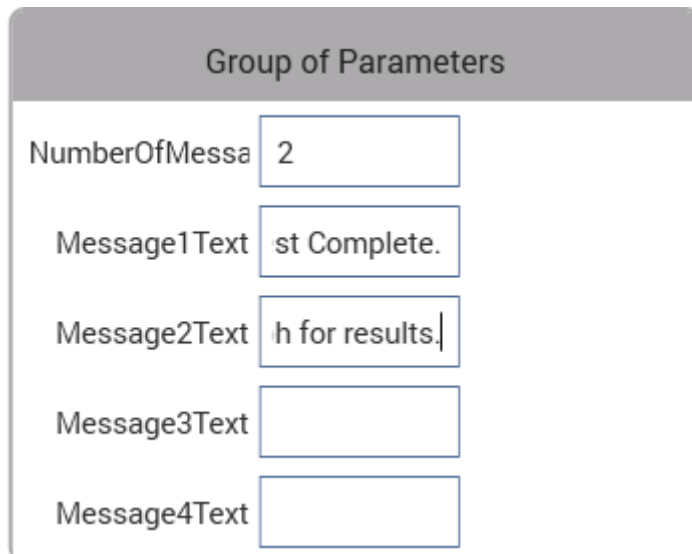
Usage

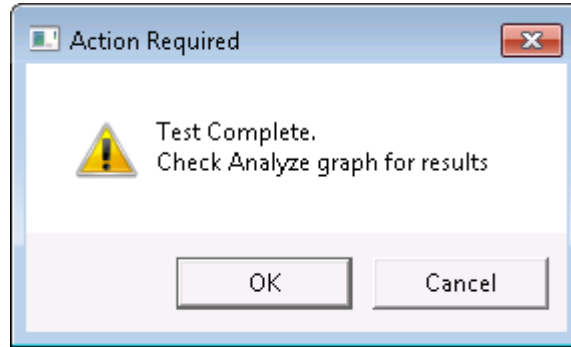
```
status = OkCancelDialog(int NumberOfMessages, char *Message1Text, char *Message2Text,
    char *Message3Text, char *Message4Text);
```

<i>status</i>	Returned values; see Details
<i>NumberOfMessages</i>	The number of text lines to display
<i>Message1Text</i>	The text to display on the first line of the dialog box; this line must be less than 40 characters
<i>Message2Text</i>	The text to display on the second line of the dialog box; this line must be less than 40 characters
<i>Message3Text</i>	The text to display on the third line of the dialog box; this line must be less than 40 characters
<i>Message4Text</i>	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

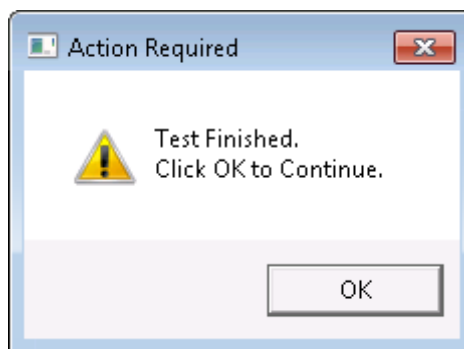
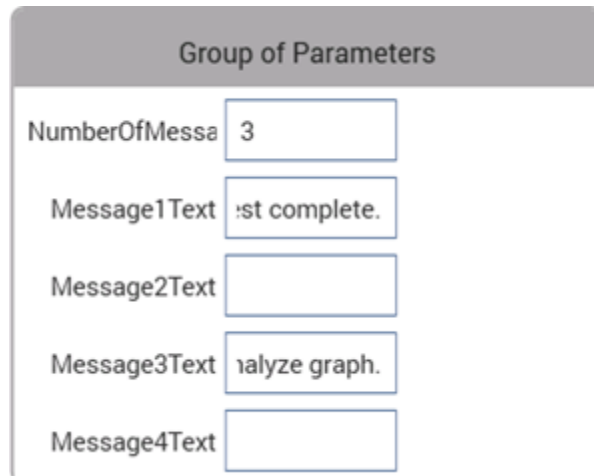
```
status = OkDialog(int NumberOfMessages, char *Message1Text, char *Message2Text, char *Message3Text, char *Message4Text);
```

<i>status</i>	Returned values; see Details
<i>NumberOfMessages</i>	The number of text lines to display
<i>Message1Text</i>	The text to display on the first line of the dialog box; this line must be less than 40 characters
<i>Message2Text</i>	The text to display on the second line of the dialog box; this line must be less than 40 characters
<i>Message3Text</i>	The text to display on the third line of the dialog box; this line must be less than 40 characters
<i>Message4Text</i>	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

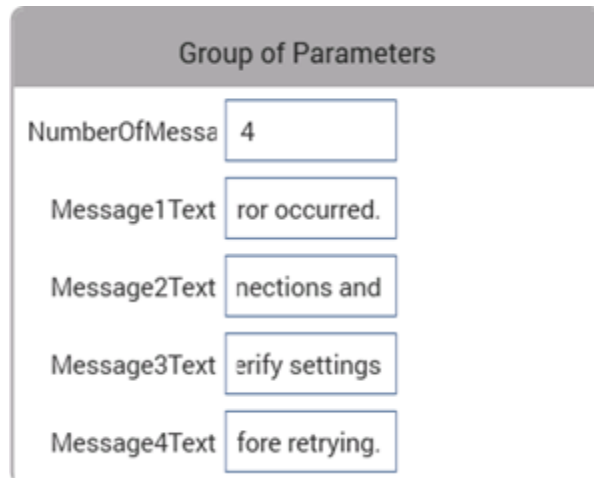
```
status = RetryCancelDialog(int NumberOfMessages, char *Message1Text, char
    *Message2Text, char *Message3Text, char *Message4Text);
```

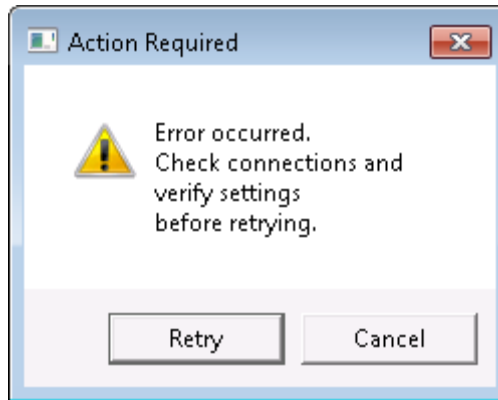
<i>status</i>	Returned values; see Details
<i>NumberOfMessages</i>	The number of text lines to display
<i>Message1Text</i>	The text to display on the first line of the dialog box; this line must be less than 40 characters
<i>Message2Text</i>	The text to display on the second line of the dialog box; this line must be less than 40 characters
<i>Message3Text</i>	The text to display on the third line of the dialog box; this line must be less than 40 characters
<i>Message4Text</i>	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.

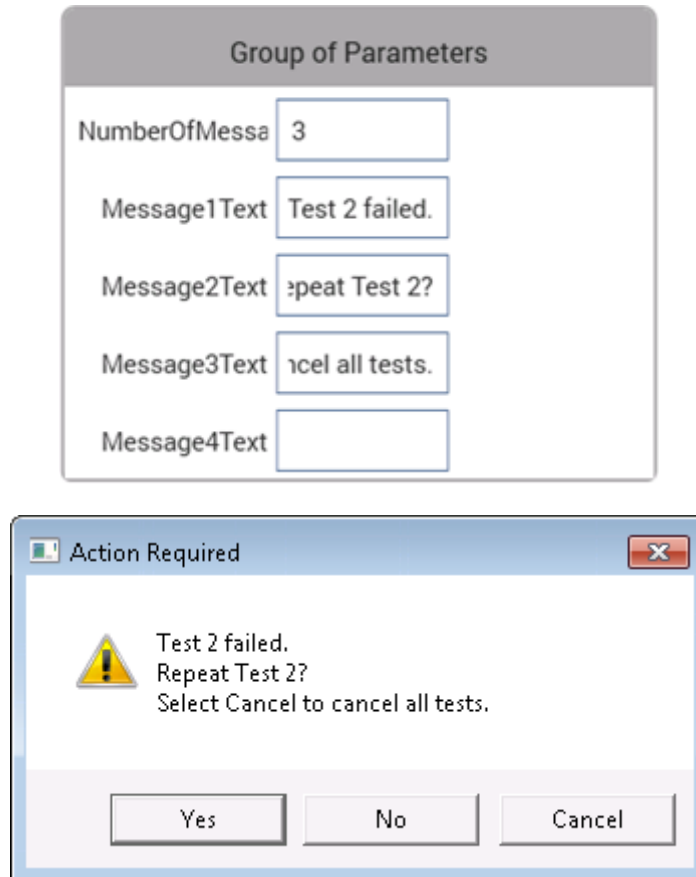
Usage

```
status = YesNoCancelDialog(int NumberOfMessages, char *Message1Text, char
    *Message2Text, char *Message3Text, char *Message4Text);
```

<i>status</i>	Returned values; see Details
<i>NumberOfMessages</i>	The number of text lines to display
<i>Message1Text</i>	The text to display on the first line of the dialog box; this line must be less than 40 characters
<i>Message2Text</i>	The text to display on the second line of the dialog box; this line must be less than 40 characters
<i>Message3Text</i>	The text to display on the third line of the dialog box; this line must be less than 40 characters
<i>Message4Text</i>	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);
```

<i>status</i>	Returned values; see Details
<i>NumberOfMessages</i>	The number of text lines to display
<i>Message1Text</i>	The text to display on the first line of the dialog box; this line must be less than 40 characters
<i>Message2Text</i>	The text to display on the second line of the dialog box; this line must be less than 40 characters
<i>Message3Text</i>	The text to display on the third line of the dialog box; this line must be less than 40 characters
<i>Message4Text</i>	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.

Group of Parameters

NumberOfMessa	<input type="text" value="2"/>
Message1Text	<input type="text" value="un complete."/>
Message2Text	<input type="text" value="second run?"/>
Message3Text	<input type="text"/>
Message4Text	<input type="text"/>



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

Wafer-level reliability testing

In this section:

JEDEC standards.....	8-1
Introduction	8-2
HCI and WLR projects	8-3
HCI degradation: Background information	8-7
Configuration sequence for subsite cycling.....	8-7
V-ramp and J-ramp tests.....	8-8

JEDEC standards

NOTE

The following descriptions for the JESD28-A and JESD35-A standard procedures were acquired from the JEDEC website. This is JEDEC copyright-protected material. The JEDEC standard procedures are available on the [JEDEC website \(jedec.org\)](http://jedec.org). Registration is free, but you must register before you can access the standards.

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.

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.

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 6-34).

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-MOSFETs and two 4-terminal p-MOSFETs.

The `hci-1-dut` project is shown in the following figure.

Figure 180: Project tree showing hci-1-dut project

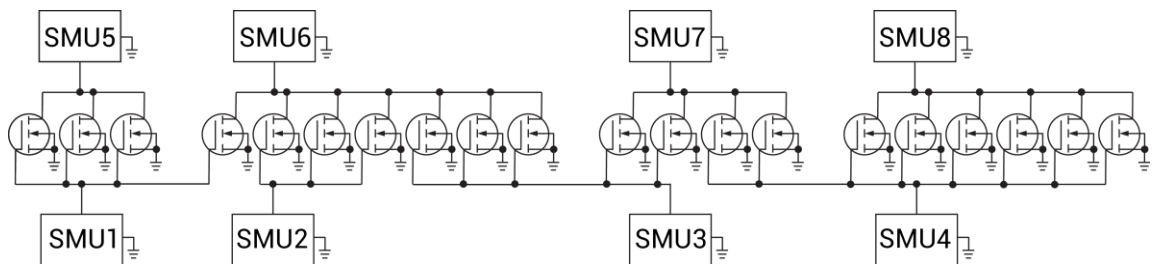


For the `hci-1-dut` project, the `hci` subsite is set up for subsite cycling using voltage stressing on the single n-channel MOSFET device (`4terminal-n-fet`). After the first pre-stress cycle to perform characterization tests, subsequent cycles voltage stress the device for a specified time before repeating 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 181: HCI and NBTI tests: 20 parallel-connected devices stressed by voltage



Negative Bias Temperature Instability project

The Negative Bias Temperature Instability (`nbt-1-dut`) project performs NBTI testing on a p-MOSFET with temperature and DC stress. The following figure shows the project tree when the `nbt-1-dut` project is selected.

Figure 182: Project tree for nbt-1-dut

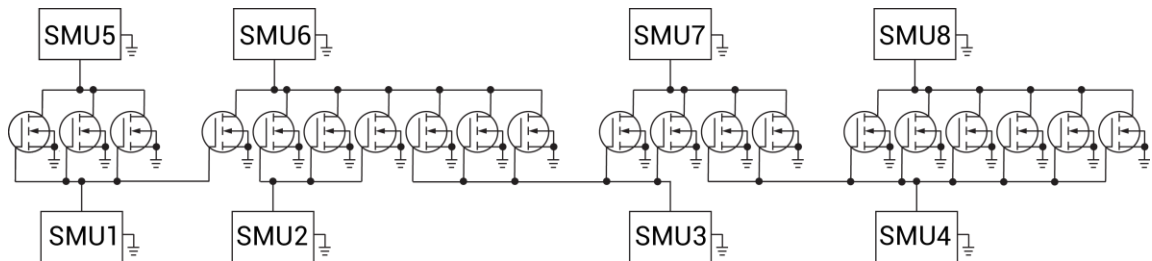


The `nbt_i` subsite is configured for subsite cycling using voltage stressing for a p-channel MOSFET (PMOS) device.

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.

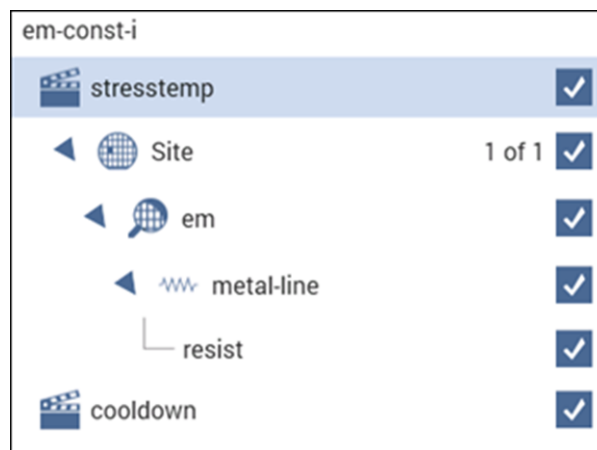
Figure 183: HCI and NBTI tests: 20 parallel-connected devices stressed by voltage



Electromigration project

The Electromigration project (`em-const-i`) is shown in the figure below.

Figure 184: em-const-i project tree



The subsite (`em`) is configured for subsite cycling using current stressing on a single device (`metal-line`).

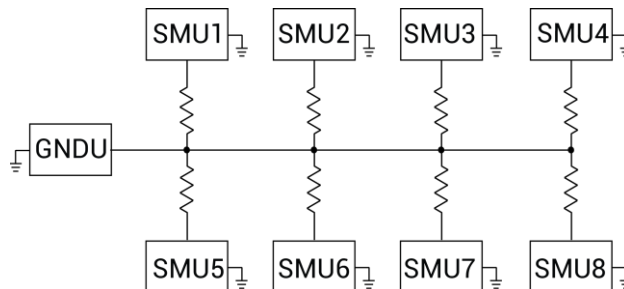
This project includes actions to control the temperature of the chuck. The subsite will not start cycling until the chuck reaches the specified temperature. After the first pre-stress cycle to perform a characterization test on the device, subsequent cycles current stress the device for a specified time 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 nonzero value.

Figure 185: EM test: Eight devices being current stressed by eight SMUs

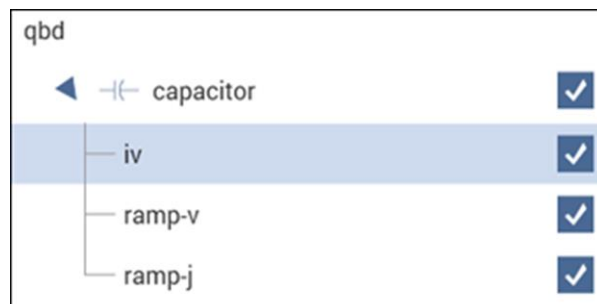


Charge-to-Breakdown Test of Dielectrics project

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 8-8).

Figure 186: qbd project tree



HCI degradation: Background information

Hot-carrier injection (HCI) degradation is one of the most important device issues facing the semiconductor industry. Small gate length and process variations in 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. 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 carriers 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 a 10% change in measured device parameters can occur within a few days.

The devices of today are 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.

Configuration sequence for subsite cycling

The following projects use subsite cycling:

- hci-1-dut
- hci-4-dut
- nbti-1-dut
- em-const-i

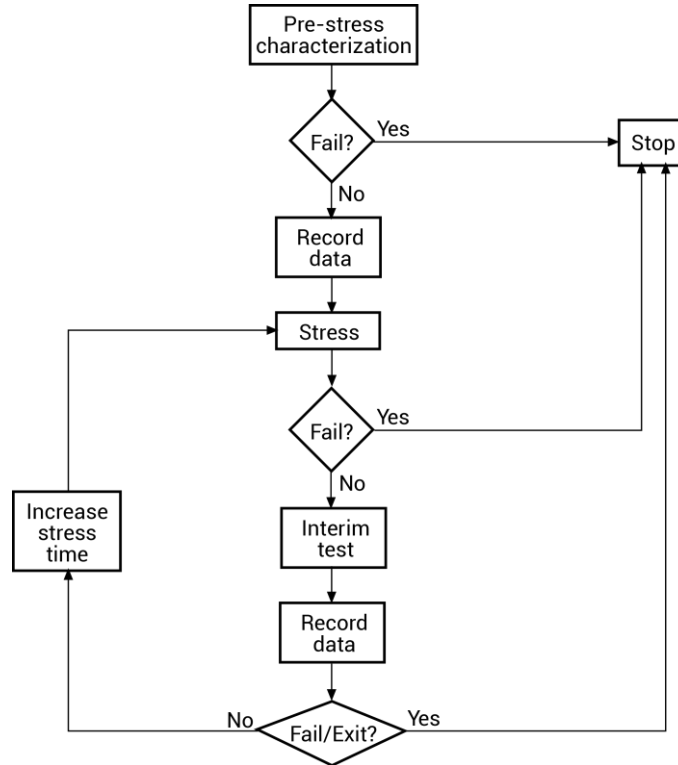
The process flow for these projects is shown in the figure below.

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 2-1).

To configure the subsite for subsite cycling, refer to [Configure Subsite Cycling](#) (on page 6-5).

Figure 187: 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 following table.

wlrlib user modules

User module	Description
<code>llsql</code>	Performs simple linear regression.
<code>qbd_rmpv</code>	Performs a charge-to-breakdown test using the QBD V-ramp test.
<code>qbd_rmpj</code>	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 8-1), 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

<code>status</code>	Returned values are placed in the Analyze sheet
<code>hi_pin</code>	High pin (usually the gate pin) (-1 to 72); enter -1 to not connect
<code>lo_pin1</code> <code>lo_pin2</code> <code>lo_pin3</code>	Usually for source drain and substrate connection; depending on device structure, some of those pins are optional; enter -1 to not connect
<code>HiSMUId</code>	ID string of the SMU outputting the stress
<code>LoSMUId1</code> <code>LoSMUId2</code> <code>LoSMUId3</code>	ID string of the SMU connected to ground terminal; these three IDs can be same
<code>v_use</code>	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)
<code>I_init</code>	Oxide breakdown failure current when biased at <code>v_use</code> ; see Details
<code>hold_time</code>	Time in ms to hold the first stress (<code>v_start</code>)
<code>v_start</code>	Starting voltage (V) for voltage ramp; typical value is <code>v_use</code> (Ref. JESD35-A)
<code>v_step</code>	Voltage (V) ramp step height; maximum of 0.1 MV/cm; refer to Details
<code>t_step</code>	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)
<code>measure_delay</code>	Time delay in ms for measurement after each voltage stress step; should be less than <code>t_step</code> (ms)

<i>I_crit</i>	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>I_box</i>	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 μ A (Ref. JESD35-A)
<i>I_max</i>	Oxide breakdown criteria; <i>I_bd</i> is obtained from I-V curves and is the oxide current at the step just prior to breakdown (Ref. JESD35-A)
<i>exit_curr_mult</i>	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)
<i>exit_slope_mult</i>	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)
<i>q_max</i>	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)
<i>t_max</i>	Maximum stress time allowed in seconds; reaching this limit will result in test to finish (s)
<i>v_max</i>	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
<i>area</i>	Area of oxide structure (cm ²)
<i>exit_mode</i>	Failure criteria mode; refer to Details
<i>V_size</i>	Size of data array; maximum 65535
<i>I_size</i>	Size of data array; maximum 65535
<i>T_size</i>	Size of data array; maximum 65535
<i>q_size</i>	Size of data array; maximum 65535

Output variables

<i>V_stress</i>	Voltage stress array
<i>I_stress</i>	Measured current array
<i>T_stress</i>	Time stamp array indicating when current is measured
<i>q_stress</i>	Accumulated charge array
<i>I_use_pre</i>	Measured oxide current at <i>v_use</i> , before starting the ramp (Ref. JESD35-A)
<i>I_use_post</i>	Measured oxide current at <i>v_use</i> , after the ramp finished (Ref. JESD35-A)
<i>Q_bd</i>	Charge-to-breakdown; cumulative charge passing through the oxide before breakdown (C) (Ref. JESD35-A)
<i>q_bd</i>	Charge-to-breakdown density (C/cm ²) (Ref. JESD35-A)
<i>v_bd</i>	Applied voltage at the step just before oxide breakdown (Ref. JESD35-A)
<i>I_bd</i>	Measured current at <i>v_bd</i> , just before oxide breakdown
<i>t_bd</i>	Time stamp when measuring <i>I_bd</i>
<i>v_crit</i>	Applied voltage at the step when the oxide current exceeds <i>I_crit</i> (Ref. JESD35-A)
<i>v_box</i>	Applied voltage at the step when the oxide current exceeds <i>I_box</i> (Ref. JESD35-A)
<i>failure_mode</i>	<ul style="list-style-type: none"> ■ 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)
<i>test_status</i>	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 8-12).

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 $\mu\text{A}/\text{cm}^2$ 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 $T_{ox} \cdot 0.1$ MV/cm, where T_{ox} 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 $T_{ox} \cdot 30$ 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 criterion 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 criterion 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 criterion.

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

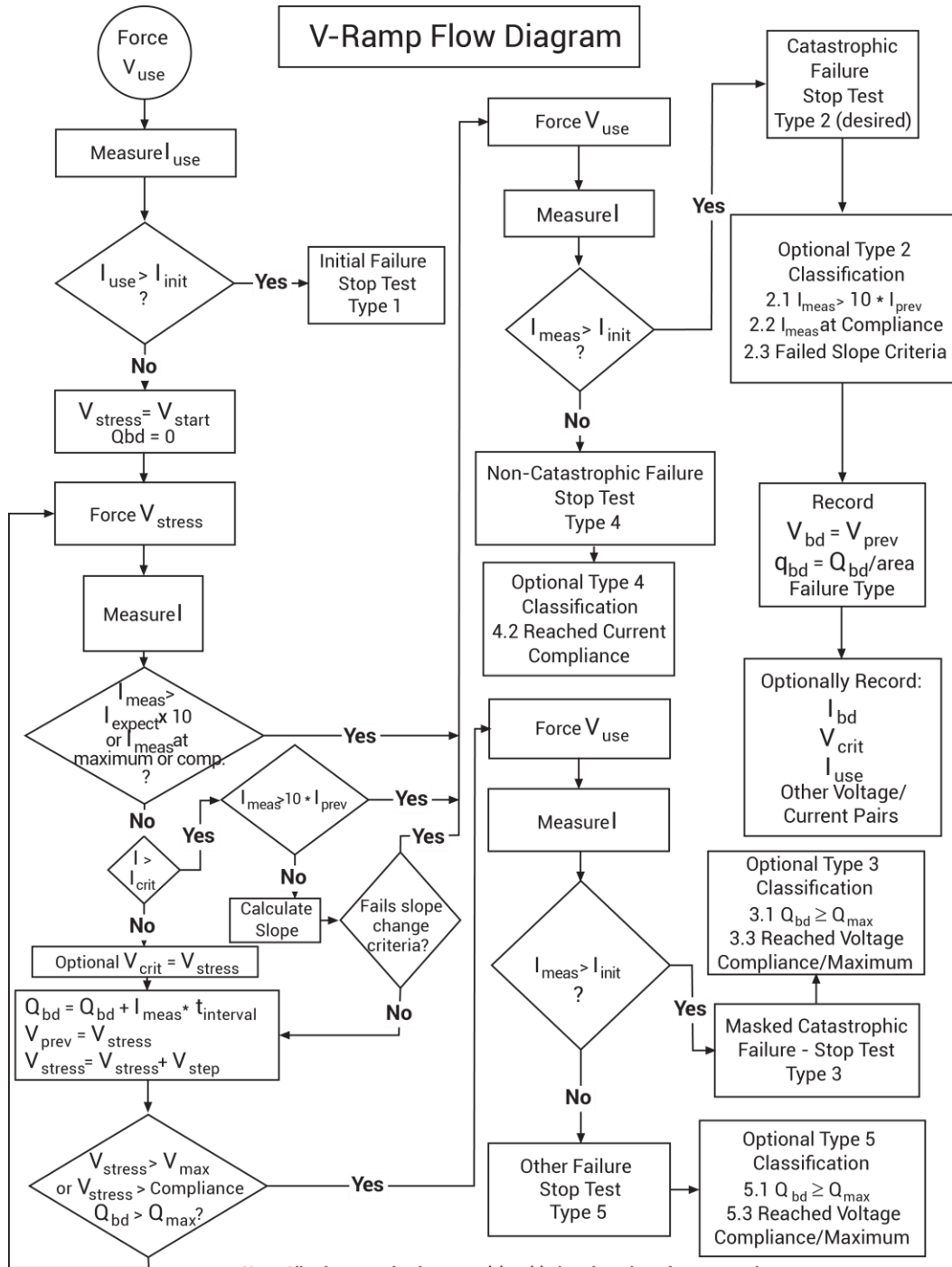
Invalid Test Result – Result = 1e21.

V-Ramp Flow Diagram

NOTE

The following diagram from JESD35-A has been reproduced with permission from JEDEC. The flowchart is from JEDEC.

Figure 188: 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

<code>status</code>	Returned values are placed in the Analyze sheet
<code>hi_pin</code>	High pin (usually the gate pin) (-1 to 72); enter -1 to not connect
<code>lo_pin1</code> <code>lo_pin2</code> <code>lo_pin3</code>	Usually for source drain and substrate connection; depending on device structure, some of those pins are optional; enter -1 to not connect
<code>HiSMUId</code>	ID string of the SMU outputting the stress
<code>LoSMUId1</code> <code>LoSMUId2</code> <code>LoSMUId3</code>	ID string of the SMU connected to ground terminal; these three IDs can be same
<code>v_use</code>	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)
<code>I_init</code>	Oxide breakdown failure current when biased at <code>v_use</code> ; typical value is 10 $\mu\text{A}/\text{cm}^2$ and may change depending on oxide area; see Details (on page 8-15)
<code>I_start</code>	Starting current (A) for current ramp; typical value is <code>I_init</code> (Ref. JESD35-A)
<code>F</code>	Current multiplier between two successive current steps (Ref. JESD35-A)
<code>t_step</code>	Current ramp step time (s) (Ref. JESD35-A)
<code>exit_volt_mult</code>	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
<code>I_max</code>	Maximum ramp current (A) (Ref. JESD35-A)
<code>q_max</code>	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)
<code>area</code>	Area of oxide structure (cm^2)
<code>V_size</code>	Size of data array; maximum 65535
<code>I_size</code>	Size of data array; maximum 65535
<code>T_size</code>	Size of data array; maximum 65535
<code>q_size</code>	Size of data array; maximum 65535

Output variables

<i>V_stress</i>	Voltage stress array
<i>I_stress</i>	Measured current array
<i>T_stress</i>	Time stamp array indicating when current is measured
<i>q_stress</i>	Accumulated charge array
<i>Q_bd</i>	Charge-to-breakdown; cumulative charge (C) passing through the oxide before breakdown (Ref. JESD35-A)
<i>q_bd</i>	Charge-to-breakdown density (C/cm ²) (Ref. JESD35-A)
<i>v_bd</i>	Applied voltage at the step just before oxide breakdown (Ref. JESD35-A)
<i>I_bd</i>	Measured current at <i>v_bd</i> , just before oxide breakdown
<i>t_bd</i>	Time stamp when measuring <i>I_bd</i>
<i>failure_mode</i>	<ul style="list-style-type: none"> ▪ 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)
<i>test_status</i>	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 8-16).

See JEDEC standard JESD35-A "Procedure for Wafer-Level Testing of Thin Dielectrics," April 2011, referenced in Signatone CM500 Prober.

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

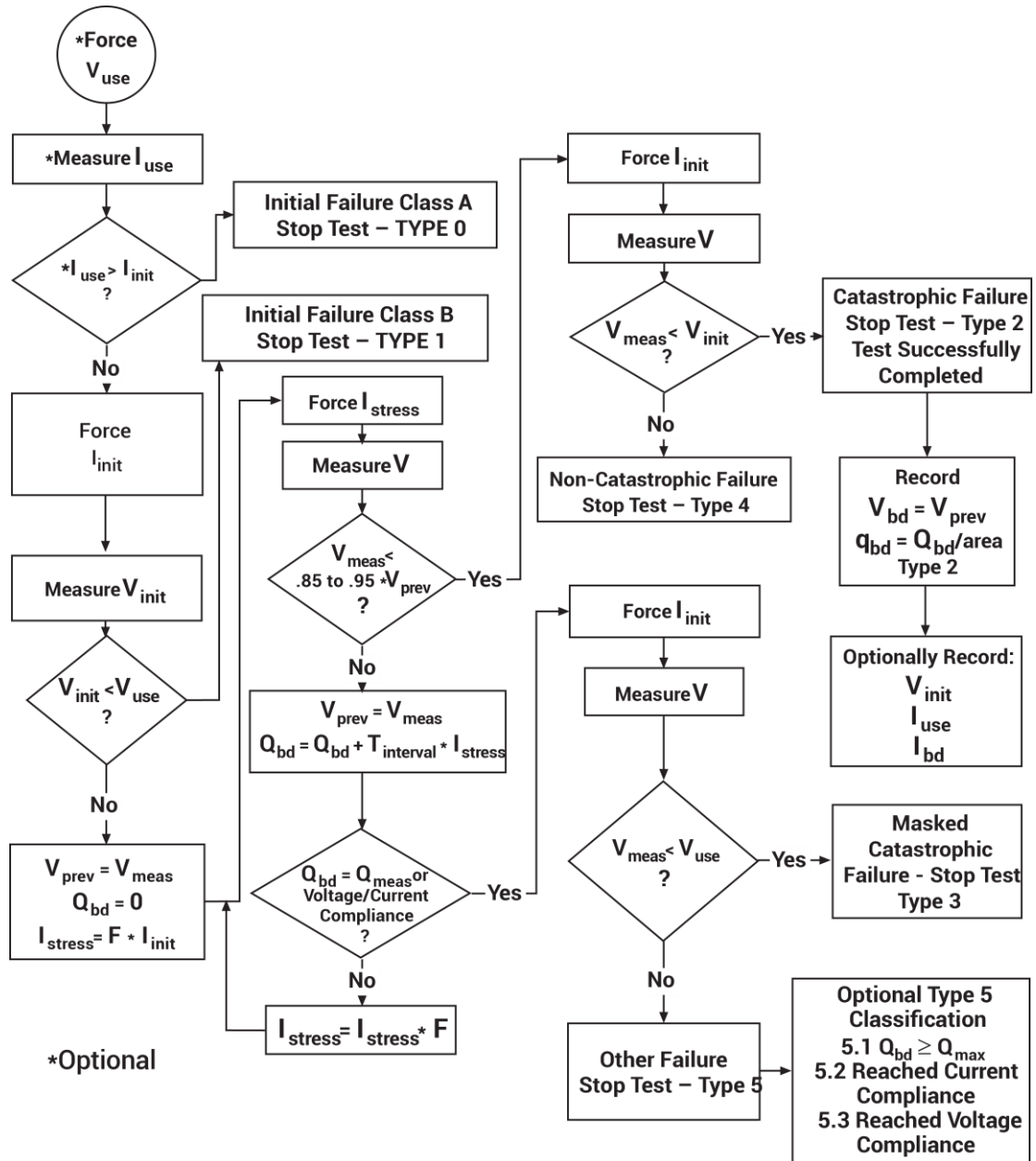
Invalid Test Result - Result = 1e21.

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 189: J-ramp flow diagram



NOTE

All values are absolute – no (+) or (-) signs have been incorporated.

Specifications are subject to change without notice.
All Keithley trademarks and trade names are the property of Keithley Instruments.
All other trademarks and trade names are the property of their respective companies.

Keithley Instruments

Corporate Headquarters • 28775 Aurora Road • Cleveland, Ohio 44139 • 440-248-0400 • 1-800-833-9200 • tek.com/keithley

