Section 6

# Measuring diode off-state breakdown

### In this section:



## Introduction

This example demonstrates how to use the Model 2657A to measure the reverse breakdown characteristics of a high-voltage diode. The Model 2657A measures the leakage current as the reverse voltage is swept to the specified breakdown voltage of the diode.

There are two examples in this section. The first example demonstrates the simplest method. This method configures a sweep using a programmatic "for" loop.

The second example demonstrates a more advanced method for configuring this measurement by using the Model 2657A trigger model to run the sweep. This method is useful in the following situations:

- If you require very precise timing either for source or measurement.
- If you are using a Model 2657A in a multi-instrument setup and you need to receive or output trigger signals to other instruments or device handlers.

## Equipment required

Equipment required:

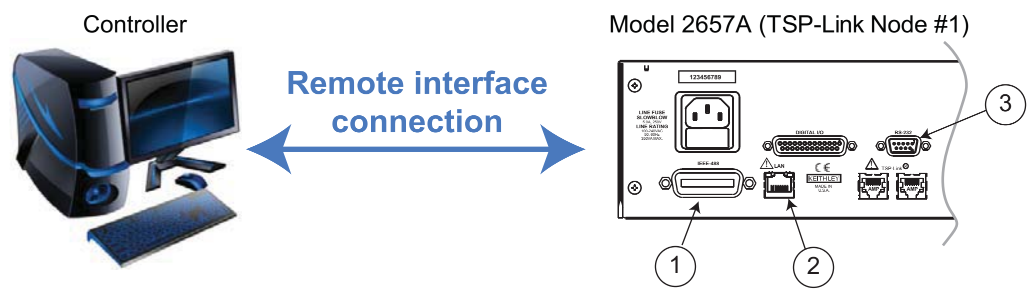
- One Model 2657A High Power System SourceMeter® instrument
- One GPIB or Ethernet cable to connect the Model 2657A to a computer
- One high voltage diode enclosed in a safe test fixture, such as the Keithley Instruments Model 8010 High Power Test Fixture Device
- Appropriate cabling to connect the Model 2657A to the test fixture

# Set up communication

The communication setup is illustrated in the following figure. This application can be run using any of the supported communication interfaces for the instrument.

For additional detail about remote communications, see "Communications interfaces" in the Model 2657A Reference Manual.

**Figure 25: Communication connections**



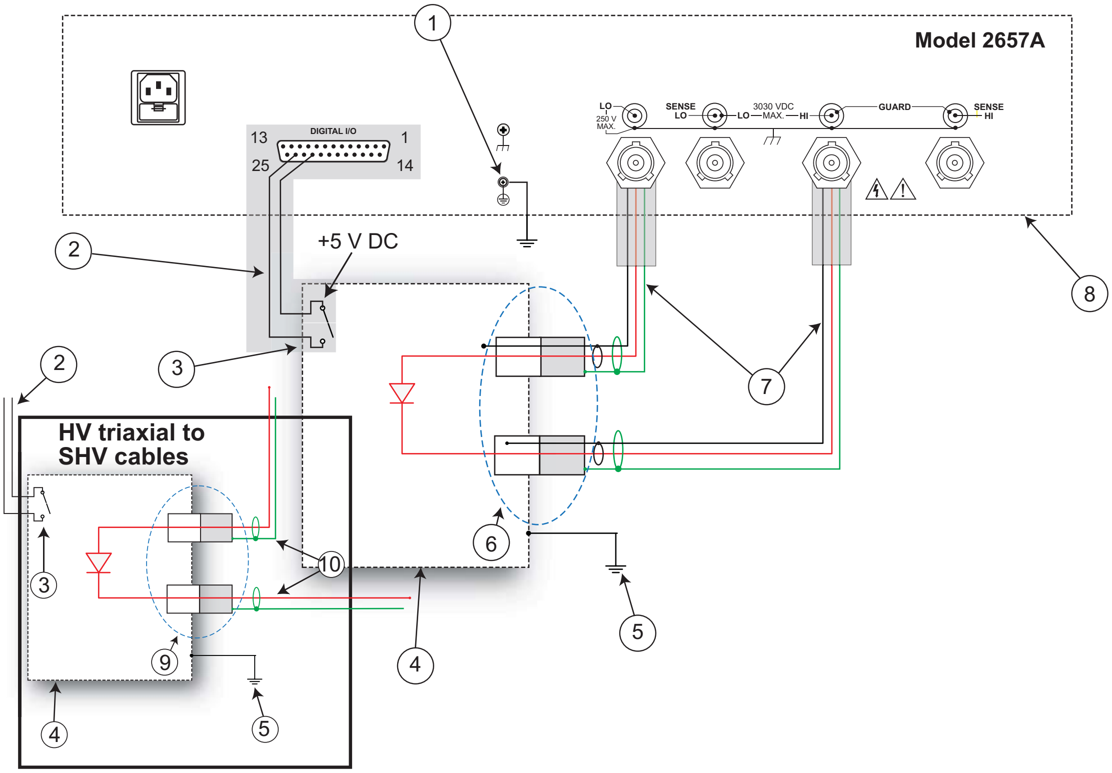

| Item | Description | Qty | Notes |
|---|---|---|---|
| 1 | IEEE-488 connection | 1 | GPIB. Model 2657A is IEEE Std 488.1 compliant. |
| 2 | CA-180-3A LAN connection | 1 | Model 2657A is LXI version 1.4 Core 2011 compliant. It supports TCP/IP and complies with IEEE Std 802.3 (ethernet). 10 or 100 Mbps. |
| 3 | RS-232 connection | 1 | |

# Device connections

Refer to the following figure to connect the diode in a safe test fixture.

See Safe configuration and test setup (on page 2-1) for information on safely connecting the Model 2657A to a device under test (DUT).

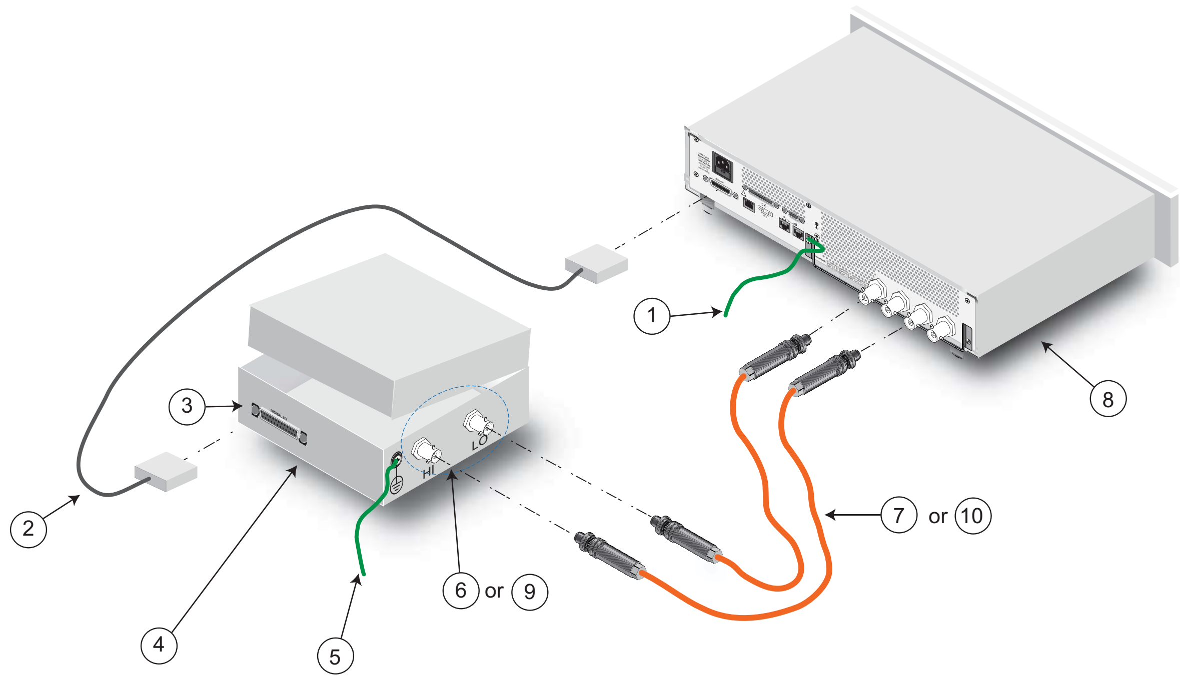
**Figure 26: Two-wire diode connections**



| Item | Description | Qty | Notes |
|---|---|---|---|
| 1 | Model 2657A protective earth (safety ground) | 1 | Additional connections for redundant protective earth may be required.<br>Keithley Instrument's Model CA-568 is a protective earth (safety ground) cable assembly. |
| 2 | Interlock connection | 1 | Model 2657A digital I/O; pin 24 (INT) and pin 22 (5 VDC) connected to the test fixture lid switch.<br>Keithley Instrument's Model 7709-308 is a 25-pin interlock male connector that can be used for custom connections.<br>Interlock switch is shown in the disengaged (lid open) position. |
| 3 | Test fixture interlock switch connection | 1 | |
| 4 | Interlocked metal safety enclosure | 1 | A safety enclosure with an interlock that has a normally-open (NO) switch. |
| 5 | Test fixture protective earth (safety ground) | | Redundant grounds may be required for specific test setups. |
| 6 | For HV connections, Model HV-CA-571-3 High-Voltage Triaxial Panel-Mount Connector to Unterminated Cable Assembly | 2 | See Using high-voltage triaxial connectors (on page 2-7). |
| 7 | Model HV-CA-554 High-Voltage Triaxial Cable | 2 | |
| 8 | Model 2657A | 1 | |
| 9 | Panel-mount SHV connectors | 2 | Customer supplied. |
| 10 | Model SHV-CA-553 High-Voltage Triaxial to SHV Cable | 2 | |

If you are using a Model 8010 High Power Test Fixture, see the Interconnect Reference Guide drawing "Two-terminal axial DUT with a Model 2657A connected" for connections and the Model 8010 User's Manual drawing "Model 2657A connections."

**Figure 27: Two-wire diode physical connections**




## Simple reverse voltage sweep

> **NOTE**
>
> The example code is designed to be run from Test Script Builder or TSB Embedded. It can also be run from other programming environments, such as Microsoft® Visual Studio® or National Instruments LabVIEW™. However, you may need to make changes to the example code to use other programming environments.

This example demonstrates a simple method that you can use to configure and execute a reverse voltage sweep on a diode and collect current measurements.

This example uses a Test Script Processor (TSP®) script to perform the measurement. The script includes two separate functions that configure the System SourceMeter Instrument and return the raw current and voltage readings that are stored in the reading buffer.

The script is written using Test Script Processor (TSP) functions rather than as a single block of inline code. TSP functions are similar to functions in other programming languages, such as Microsoft® Visual C® or Visual Basic®. They must be called before the code in them is executed. Because of this, running the script alone will not execute the test. To execute the test, you need to run the script to load the functions into test script memory. You then call the functions.

Refer to the documentation for Test Script Builder or TSB Embedded (on page 4-4) for directions on how to run scripts and enter commands using the instrument console.

## Example code

```
--[[
   DiodeRL(vstart, vstop, vstep, irange, ilimit)

   Description: This function uses a "for" loop to configure a
   reverse voltage sweep on a diode and collect leakage current
   measurements.

   This demonstrates a very simple method of performing a linear
   voltage sweep on the Model 2657A.

   Parameters:
   vstart: The starting voltage of the diode reverse voltage sweep.
   vstop: The stopping voltage of the diode reverse voltage sweep.
   vstep: The step voltage of the diode reverse voltage sweep
      (how much the voltage changes per step).
   irange: current measurement range, set to a value or set to
      "auto" to enable autorange.
   ilimit: The current limit of the voltage source.

   Additional notes:
      Source delay is automatically added to each point of this sweep
      because the smua.source.levelv command is used.
      Source delay varies by voltage range.
      If the timing of the source is critical to your application,
      please use the DiodeRL_Swp function instead.

   Example usage:
   DiodeRL(0, 1000, 100, 100e-9, 0.01)
--]]

function DiodeRL(vstart,vstop,vstep,irange,ilimit)

   --Reset and initialize instrument.
   reset()
   status.reset()
   errorqueue.clear()

   --Configure source function as 2W DCVOLTS.
   smua.source.func = smua.OUTPUT_DCVOLTS
   smua.sense = smua.SENSE_LOCAL

   --Define local variable to store the number of points in the sweep.
   local l_numPoints
```

```
--[[
     Calculate the number of points based on the start and
     stop values of the sweep.
--]]
if math.abs(vstart) > math.abs(vstop) then
   smua.source.rangev = vstart
   l_numPoints = (vstart - vstop) / vstep + 1
else
   smua.source.rangev = vstop
   l_numPoints = (vstop - vstart) / vstep + 1
end
--Set up current compliance.
smua.source.limiti = ilimit

--Sets up current range.
if irange == "auto" then
   smua.measure.autorangei = smua.AUTORANGE_ON
else
   smua.measure.autorangei = smua.AUTORANGE_OFF
   smua.measure.rangei = irange
end

--Set the measurement integration time.
smua.measure.nplc = 1
smua.measure.delay = 0.05

--Configure the reading buffers.
smua.nvbuffer1.clear()
smua.nvbuffer1.appendmode = 1
smua.nvbuffer1.collecttimestamps = 1
smua.nvbuffer1.collectsourcevalues = 0
smua.nvbuffer1.fillmode = smua.FILL_ONCE
smua.nvbuffer2.clear()
smua.nvbuffer2.appendmode = 1
smua.nvbuffer2.collecttimestamps = 1
smua.nvbuffer2.collectsourcevalues = 0
smua.nvbuffer2.fillmode = smua.FILL_ONCE

--Define local variable for index of the "for" loop.
local l_i

--Turn on the output.
smua.source.output = 1
```

```
   --[[
       Use a "for" loop to run the linear voltage sweep and
       make current and voltage measurements.
   --]]
   for l_i  = 0, (l_numPoints-1) do
      --Update the source level.
      smua.source.levelv = vstart + l_i*(vstep)
      --Make simultaneous voltage and current measurements.
      smua.measure.iv(smua.nvbuffer1, smua.nvbuffer2)
   end
   --Set the voltage level back to 0 V.
   smua.source.levelv = 0

   --Turn off the output.
   smua.source.output = 0
end

--[[
    This function prints the data from the smua.nvbuffer1 and
    smua.nvbuffer2 reading buffers into three separate
    tab-delimited columns.
--]]
function printData()
   if smua.nvbuffer1.n == 0 then
      print("No readings in buffer")

   else
      print("Timestamps\tCurrent\tVoltage")
      for i = 1, smua.nvbuffer1.n do
         print(string.format("%g\t%g\t%g", smua.nvbuffer1.timestamps[i],
           smua.nvbuffer1.readings[i], smua.nvbuffer2.readings[i]))
      end
   end
end
```

## Example usage

The functions in this script allow updates to the test parameters without rewriting or re-running the script. To run the test, call the `DiodeRL()` function, passing in the appropriate values for test parameters.

| DiodeRL() parameters | | |
|---|---|---|
| **Parameter** | **Units** | **Description** |
| `vstart` | volts | Start voltage for the linear voltage sweep |
| `vstop` | volts | Stop voltage for the linear voltage sweep |
| `vstep` | volts | Step voltage for the sweep |
| `irange` | current | Current measurement range |
| `ilimit` | current | Current limit (compliance) |

An example of how to call this function is shown here:

```
DiodeRL(0, 1000, 10, 100e-9, 0.01)
```

This call results in a 101-point linear voltage sweep starting at 0 V and proceeding to 1000 V in 10 V increments. The current measurement range is set to 100 nA and the current limit is set to 10 mA.

You can use the `printData()` function to return the stored voltage and current measurements and the timestamps. The data is formatted into three columns. Here is an example of how to call the `printData()` function:

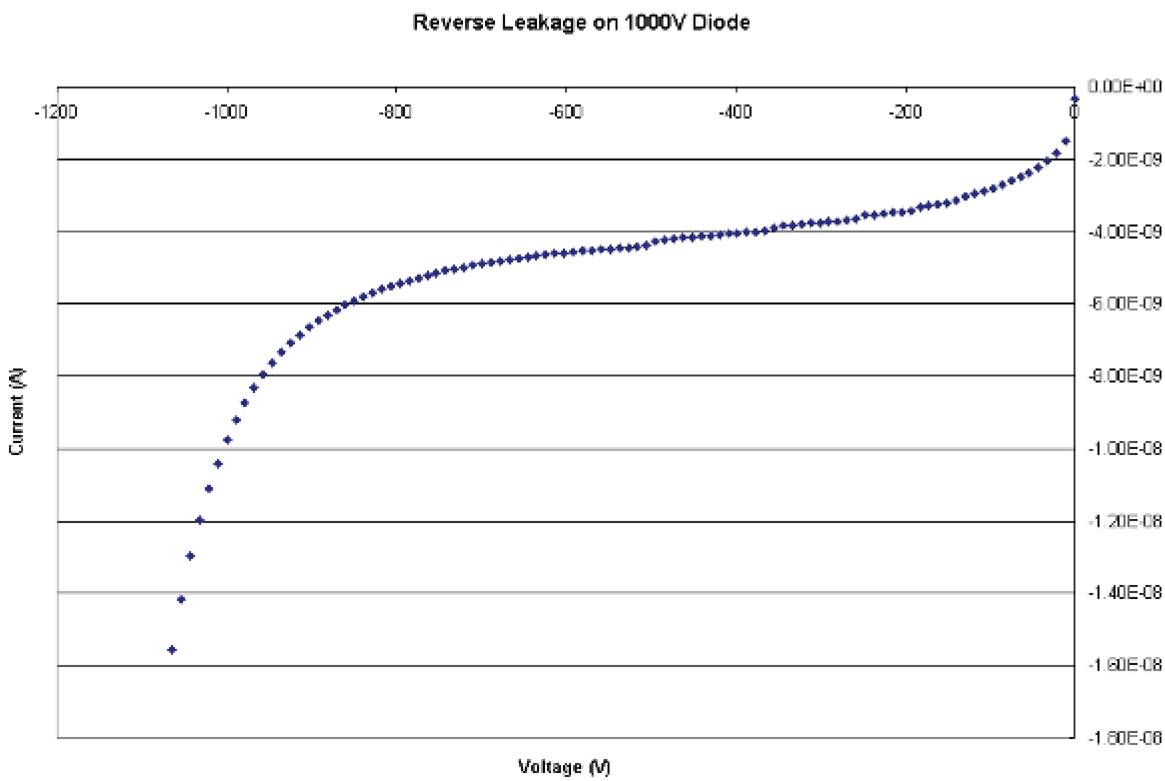
```
printData()
```

An example of the output is shown here.

**NOTE**

The connections that were made to the device were set up so that when a positive voltage was sourced, the device was reverse biased. The voltage readings were negated before graphing.

**Figure 28: Results of sample data for diode breakdown characteristics**




# Advanced reverse voltage sweep

This example demonstrates how to use the trigger model of the Model 2657A to perform a reverse linear voltage sweep on a diode and collect current measurements. The trigger model is useful when critical timing is required or when the Model 2657A must send or receive triggers from an external instrument or device handler.

This example does not configure any incoming or outgoing trigger signals. For more information on the trigger model, see the Model 2657A Reference Manual.

## Example code

```
--[[
   DiodeRL_Swp(vstart, vstop, vstep, irange, ilimit, sourceDelay)

   Description: This function uses the trigger model and built-in
   sweeping function to create a linear voltage sweep to measure
   the reverse leakage of a diode.

   Using this method is useful when there is a need to send or
   receive external triggers from another instrument or
   device handler.

   Parameters:
   vstart: The starting voltage of the diode reverse voltage sweep.
   vstop: The stopping voltage of the diode reverse voltage sweep.
   vstep: The step voltage of the diode reverse voltage sweep
    (how much the voltage changes per step).
   irange: current measurement range, set to value or set to "auto"
    to enable autorange.
   ilimit: The current limit of the voltage source.
   sourceDelay: The delay between the start of source and the
    source complete event.


   Example usage:
   DiodeRL_Swp(0, 1000, 10, 100e-9, 0.01, 0.05)
--]]

function DiodeRL_Swp(vstart,vstop,vstep,irange,ilimit,sourceDelay)

   --Reset and initialize instrument.
   reset()
   status.reset()
   errorqueue.clear()

   --Configure source function as 2W DCVOLTS
   smua.source.func = smua.OUTPUT_DCVOLTS
   smua.sense = smua.SENSE_LOCAL

   --Define a local variable to store the number of points in the sweep.
   local l_numPoints

   --[[
       Calculate the number of points in the sweep based on the
       start and stop values.
   --]]
   if math.abs(vstart) > math.abs(vstop) then
      smua.source.rangev = vstart
   else
      smua.source.rangev = vstop
   end
   l_numPoints = math.abs((vstop - vstart) / vstep) + 1
```

```
--Set up source delay.
smua.source.delay = sourceDelay

--Set up current compliance.
smua.source.limiti = ilimit

--Set up current measurement range.
if irange == "auto" then
   smua.measure.autorangei = smua.AUTORANGE_ON
else
   smua.measure.autorangei = smua.AUTORANGE_OFF
   smua.measure.rangei = irange
end

--Set the integration time.
smua.measure.nplc = 1

--Configure the reading buffers.
smua.nvbuffer1.clear()
smua.nvbuffer1.appendmode = 0
smua.nvbuffer1.collecttimestamps = 1
smua.nvbuffer1.collectsourcevalues = 0
smua.nvbuffer1.fillmode = smua.FILL_ONCE
smua.nvbuffer2.clear()
smua.nvbuffer2.appendmode = 0
smua.nvbuffer2.collecttimestamps = 1
smua.nvbuffer2.collectsourcevalues = 0
smua.nvbuffer2.fillmode = smua.FILL_ONCE

--Configure the source sweep.
smua.trigger.source.linearv(vstart, vstop,l_numPoints)
smua.trigger.source.action = smua.ENABLE
smua.trigger.source.stimulus = 0

--Configure measurements during the sweep.
smua.trigger.measure.action = smua.ENABLE
smua.trigger.measure.stimulus = 0
smua.trigger.measure.iv(smua.nvbuffer1,smua.nvbuffer2)

--Configure trigger model parameters.
smua.trigger.count = l_numPoints
smua.trigger.arm.count = 1

--Turn on the output.
smua.source.output = 1
```

```
   --[[
       Initiate the sweep and wait until sweep is complete before
       proceeding to next command.
   -]]
   smua.trigger.initiate()
   waitcomplete()

   --Turn off the output.
   smua.source.output = 0

end


--[[
   This function prints the data from the smua.nvbuffer1 and
   smua.nvbuffer2 reading buffers into three separate columns.
--]]
function printData()
   if smua.nvbuffer1.n == 0 then
      print("No readings in buffer")

   else
      print("Timestamps\tCurrent\tVoltage")
      for i = 1, smua.nvbuffer1.n do
         print(string.format("%g\t%g\t%g",
           smua.nvbuffer1.timestamps[i], smua.nvbuffer1.readings[i],
           smua.nvbuffer2.readings[i]))
      end
   end
end
```

## Example usage

The functions in this script allow updates to the test parameters without rewriting or re-running the script. To run the test, call the `DiodeRL_Swp()` function, passing in the appropriate values for test parameters.

| **DiodeRL_Swp() parameters** | | |
|---|---|---|
| **Parameter** | **Units** | **Description** |
| `vstart` | volts | Start voltage for the linear voltage sweep |
| `vstop` | volts | Stop voltage for the linear voltage sweep |
| `vstep` | volts | Step voltage for the sweep |
| `irange` | current | Current measurement range |
| `ilimit` | current | Current limit (compliance) |
| `sourceDelay` | seconds | The delay between the source complete event and the start of the measurement |

An example of how to call this function is shown here:

```
DiodeRL_Swp(0, 1000, 10, 100e-9, 0.01, 0.05)
```

This call results in a 101-point linear voltage sweep starting at 0 V and proceeding to 1000 V in 10 V increments. The current measurement range is set to 100 nA and the current limit is set to 10 mA. A 50 ms delay occurs after each source step is programmed. The source complete event of the trigger model is held off until that delay is complete. Once the delay, and therefore the source event, is complete, the measure action can take place.

For more information on the trigger model, please see the Model 2657A Reference Guide.

You can use the `printData()` function to return the stored voltage and current measurements and the timestamps. The data is formatted into three columns. An example of how to call the `printData()` function is:

```
printData()
```

The results of this test are the same as those produced by the simple reverse voltage sweep. See Simple reverse voltage sweep (on page 6-5) for a graph of the results.