KEITHLEY

Application Note Series

Number 2633

Diode Production Testing with Series 2600 System SourceMeter® Instruments

Introduction

Performing single-point pass/fail DC tests on packaged diodes is critical to ensure compliance with manufacturers' specifications and weed out defective devices before they are shipped. Most types of diodes undergo at least three basic DC parametric tests during this final inspection process: the Forward Voltage Test (V_F), Breakdown Voltage Test (V_R), and Leakage Current Test (I_R). While the reliability of these tests is essential to ensuring product quality, it's equally important that they be conducted quickly to maintain high production throughput.

At one time, test engineers needed several instruments to make these tests, such as a DMM, voltage source, and current source. Such multi-instrument systems consume valuable rack space and can limit test throughput rates. Using multiple measurement instruments and sources makes trigger timing more complex and can lead to increased triggering uncertainty. Coordinating the operation of separate instruments can extend the measurement cycle by increasing the amount of bus traffic required. Additionally, three separate instruments also mean there are three sets of commands to learn, complicating system programming and maintenance.

Keithley's Series 2400 SourceMeter instruments are widely used for diode production testing because they enable test engineers to configure a test system using a single instrument that can source and measure both current and voltage. Another application note (#1805) describes how to implement the diode tests using the Series 2400's source memory sweep feature, which is a powerful test sequencer featuring advanced math, automatic limit inspection, and conditional branching capabilities.

This application note describes how to implement a diode test system using Keithley's new Series 2600 System SourceMeter instruments. This next-generation SourceMeter family includes the single-channel Model 2601 and dual-channel Model 2602. With a built-in Test Script Processor (TSP™) and a new inter-unit communication interface (TSP-Link™), Series 2600 instruments offer even more power and flexibility than their predecessors.

This application note outlines the three basic parametric tests for diodes and algorithms for both the test system and SourceMeter remote operation. In addition to the three parametric tests (also referred to as functional tests), this note describes how to perform polarity testing on the diode, which is often required on many of the newer surface mount diode packages, such as the Small Outline Diode (SOD) package. Such diodes don't orient themselves automatically in the same direction in a component handler, so the polarity of the device must sometimes be determined before beginning the full functional test sequence. The results of the polarity test can be used in one of two ways: the functional test parameters can be adjusted to account for the orientation of the diode (i.e., using reversed polarity test signals and adjusting data inspection accordingly) or the handler can be triggered to rotate the device before functional testing begins. The choice between the two options usually depends on the capability of the handler.

The example program (script) discussed in this note is available from Keithley's web site. See the section on "Obtaining Example Scripts" to learn how to obtain a copy of this and other example scripts.

Test Descriptions

Figure 1 illustrates the test points for each of the tests described.

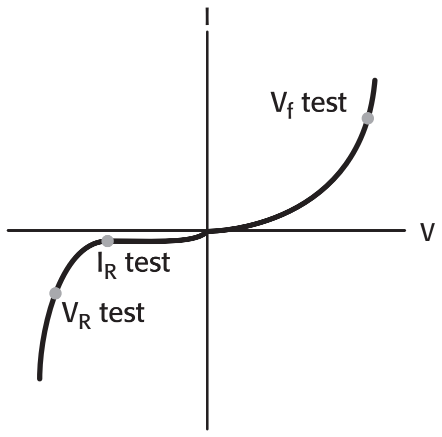

Figure 1. Typical Diode I-V Curve and Test Points (not to scale)

Polarity Test

The polarity test is designed to determine the orientation of the diode safely and quickly prior to performing functional tests on the device. The breakdown characteristics of the diode are used to generate an indication of the diode's polarity. By properly choosing a test current level and voltage compliance limit, it's possible to distinguish a correctly oriented forward-biased diode from an incorrectly oriented reverse-biased diode. If a positive current is sourced through a properly oriented diode, the diode will be forward-biased and the resulting forward voltage will generally be small (typically less than 1V). If the same test current is forced through a diode with reversed orientation, then the diode will be reverse-biased and the resulting reverse voltage will be much higher than the forward voltage and will easily exceed the compliance limit. Thus, by simply checking the compliance state of the instrument, it's possible to determine if the diode is in the forward or reverse orientation. If compliance is reached, the diode is in reversed orientation; otherwise, it's in forward orientation. It would be possible to perform this test by actually measuring the voltage drop across the diode and comparing it to forward- and reverse-biased limits, but simply checking for compliance is generally faster.

Forward Voltage Test (V_F)

This functional test involves sourcing a specified forward bias current within the normal operating range of the diode, then measuring the resulting voltage drop after a specified period of time (e.g., 1ms). To pass the test, the voltage must be within specified minimum and maximum values (e.g., $V_F = 0.8V \pm 5\%$).

Reverse Breakdown Voltage Test (V_R)

In this test, a specified reverse current bias is sourced and the resulting voltage drop across the diode is measured after a specified period of time (e.g., 1ms). To determine whether the diodes pass or fail, the measurements are compared to a specified minimum limit (e.g., $V_R \leq -100V$ or $|V_R| \geq 100V$).

Leakage Current Test (I_R)

The leakage test verifies the low level of current that leaks across the diode under reverse voltage conditions. For this test, a specified reverse voltage is sourced for a specified period of time (e.g., 10ms), and then the resulting leakage current is measured. Good diodes will have a leakage current that doesn't exceed a specified maximum value (e.g., $-10nA \leq I_R \leq 0$ or $|I_R| \leq 10nA$).

Test System Configuration

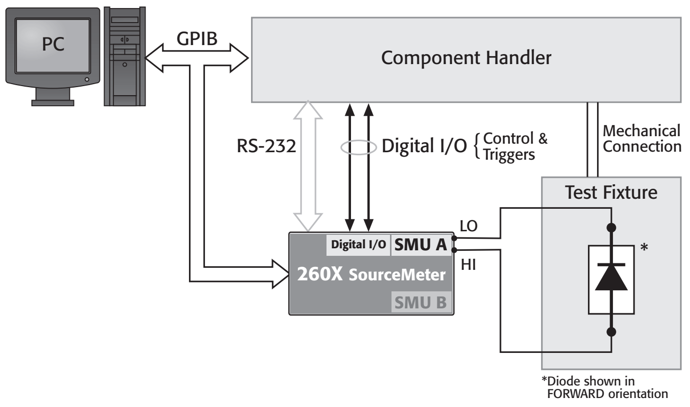

Figure 2. Block diagram of a SourceMeter-based system for diode production testing.

The diode or diode package is placed in a test fixture and connections are made to the input of the SourceMeter instrument. Depending on their packaging, many diodes are photosensitive. To prevent generating unwanted currents, use a test fixture that shields the diode from light. The SourceMeter instrument biases and measures the diode. The measurements are then compared to pre-specified limits in the instrument and pass/fail determinations are made. The 260X can be controlled like a typical programmable instrument by sending it discrete commands via the IEEE-488 bus (GPIB) or RS-232. However, for maximum throughput, a complete test script can be downloaded to the instrument's Test Script Processor, which can then perform the complete test virtually independent of the host PC (system controller).

Output signals from the SourceMeter instrument's digital I/O port are used for interfacing with the handler to initiate diode orientation and/or binning. The instrument is equipped with 14 digital input/output lines, which can be used for digital control or as input or output trigger lines. Each digital output code conveys a message, such as "part is good," "part is bad," "turn part," etc. As shown in ***Figure 2***, a 260X SourceMeter instrument can even communicate with a handler directly via its RS-232 serial port if necessary (assuming it isn't connected to the PC). The ability of the SourceMeter instrument to interface directly with the component handler frees the PC during handler control operations. This makes it possible for the computer to download and store test data while a new diode or diode package is being positioned in the test fixture. On the other hand, the 260X has very deep memory, which can eliminate the need to transfer data after each diode. Each SMU channel has two nonvolatile buffers, which can each hold up to 100,000 readings, and there is also volatile memory available for even more data storage.

The three algorithms that follow describe diode testing for three different scenarios, depending on handler capability and diode package.

Diode Polarity Known

The following algorithm describes the operation of a Model 260X-based diode production test system to perform the measurements described when the polarity of the diode is known prior to functional testing.

1. Operator indicates to the PC that a diode production lot is in place and ready for test.
2. The PC initiates 260X operation over the IEEE-488 bus (GPIB) or RS-232 (see "Remote Operation…"). Note that the 260X can be configured to allow the operator to make selections at its front panel, possibly precluding the need for the PC to initiate the action.
3. The 260X waits for a Start-of-Test (SOT) trigger from the handler.
4. When the first diode is in position, the handler sends an SOT trigger signal to the 260X, indicating the first diode is ready for testing.
5. The 260X runs the diode functional tests in the order dictated by the test executive, makes pass/fail determinations, and saves data for each test.
6. The 260X sends an overall pass/fail code and End-of-Test (EOT) signal to handler and sends test data to the PC (operations occur in parallel).
7. Repeat Steps 3 through 6 for the remainder of the diodes in the lot.

8. The 260X returns to the idle state. Operator installs a new lot of diodes in the handler.
9. Repeat Steps 1 through 8 as required.

Diode Polarity/Orientation Unknown – Component Handler Can Turn Device

The following algorithm describes the operation of a Model 260X-based diode production test system to perform the measurements described when the polarity of the diode is unknown prior to functional testing and the handler can turn the diode end-for-end.

1. Operator indicates to the PC that a diode production lot is in place and ready for test.
2. The PC initiates 260X operation over GPIB or RS-232 (see "Remote Operation…").
3. The 260X waits for a Start-of-Test (SOT) trigger from the handler.
4. When the first diode is in position, the handler sends an SOT trigger signal to the 260X, indicating the first diode is ready for testing.
5. The 260X executes the polarity test. If the diode is verified in forward polarity, the 260X proceeds with the functional tests (Step 6). If it's in reverse polarity, a signal is sent to the handler to turn the device and return to Step 4.
6. Once the diode is in the proper orientation, the 260X runs the diode functional tests in the order dictated by the test executive, makes pass/fail determinations, and saves data for each test.
7. The 260X sends an overall pass/fail code and End-of-Test (EOT) signal to the handler and sends test data to the PC (operations occur in parallel).
8. Repeat Steps 3 through 7 for the remainder of diodes in the lot.
9. The 260X returns to the idle state. Operator installs a new lot of diodes in the handler.
10. Repeat Steps 1 through 9 as required.

Diode Polarity/Orientation Unknown – Component Handler Can't Turn Device

This algorithm is slightly different from the previous one. It describes the operation of a diode production test system to perform the measurements described when the polarity of the diode is unknown prior to functional testing and the handler can't turn the device under test.

1. Operator indicates to the PC that a diode production lot is in place and ready for test.
2. The PC initiates 260X operation over the GPIB or RS-232 (see "Remote Operation…").
3. The 260X waits for a Start-of-Test (SOT) trigger from the handler.
4. When the first diode is in position, the handler sends an SOT trigger signal to the 260X, indicating the first diode is ready for testing.
5. The 260X executes the polarity test. If the diode is forward-biased, then the 260X proceeds with the functional tests using forward polarity parameters. If it's reverse-biased, then the 260X runs tests using reverse polarity parameters.
6. Depending on the results of the polarity test, the 260X runs either the forward polarity diode functional tests or the reverse polarity diode tests in the order dictated by the test executive. It makes pass/fail determinations and saves data for each test.
7. The 260X sends an overall pass/fail code, diode polarity indicator, and End-of-Test (EOT) signal to the handler and sends test data to the PC (operations occur in parallel).
8. Repeat Steps 3 through 7 for the remainder of diodes in lot.
9. The 260X returns to the idle state. Operator installs a new lot of diodes in the handler.
10. Repeat Steps 1 through 9 as required.

Remote Operation and Using the Test Script Processor

Series 2600 System SourceMeter instruments have a powerful embedded computer or Test Script Processor (TSP™), which offers capabilities never seen before in rack-and-stack instruments. A complete test program (script) can be downloaded to the TSP. As with other common programming languages, a well-designed script creates reusable functions or subroutines, which can be called by a test executive or other functions. It's possible to pass parameters to these functions. In the diode test example presented here, functions are created to perform the polarity test and various functional tests, to inspect the data, and to return the results. These functions can be called from a test executive in the system controller PC or, as in this example, from a test executive function, which resides in the TSP. The script, which creates these functions, must be downloaded to the 260X using either GPIB or RS-232. When the script is first downloaded, it's stored in volatile memory. However, it can be saved to non-volatile memory if desired. The script must be run to create the functions. These functions always reside in volatile memory, which means they must be recreated any time power is cycled. The creation script can be explicitly run at any time or it can be set to run automatically at power-up. The test executive function determines the order in which the various tests are executed. The executive is written in such a way that it will wait for an SOT trigger from the handler or the system controller PC. In response to the single trigger, the instrument executes the test sequence without system controller intervention, thereby saving communications time and increasing system throughput.

A script can be created using any text editor. However, Keithley provides a free application called Test Script Builder, which can be used to create, debug, and organize scripts. Test

Script Builder can download scripts to volatile instrument memory or save them in nonvolatile instrument memory. It can also run the scripts. Scripts can also be loaded and run using applications created in other languages, such as Visual Basic, Visual C/C++, or LabVIEW. Once the script is in memory, it can even be run from the front panel.

All measurements, calculations and inspections can be performed by the 2602, so it isn't necessary to send data to a host computer for processing. However, it's possible to do so if desired for recordkeeping or other purposes. As seen in the example script, "print" statements are used to send data back to the system controller (host computer). The data listed in the print statement is placed in the instrument's output queue for retrieval by the host. The output queue can hold up to 1000 print responses or a maximum of 32 kilobytes, whichever is less. If the output queue is allowed to fill up before all print actions have completed, program execution will be held up until there is room in the output queue for the additional data. Therefore, to ensure that a program will operate quickly, the output queue should be read often enough to avoid overloading it.

There are no set rules for partitioning a test program between the TSP and a system controller because one must consider the complete test system to determine how partitioning may simplify system control and improve system throughput. The significant thing is that the 260X family gives the test engineer the option to consider such partitioning, thus providing a new level of design flexibility for rack-and-stack instruments.

Diode Test Example Script

For the following example, it's assumed that it's necessary to perform a polarity test. As described earlier, the actions that must be taken after performing the polarity test depend on the capability of the handler. If the diode is in reverse orientation and the handler can physically turn the part, then an appropriate digital control signal would be sent to the handler to tell it to turn the diode. If the handler can't turn the part, then it's necessary to use test signals appropriate for the forward- and reversed-bias conditions. The example is based on the latter situation.

The diode test script creates the following functions:

1. ChkPolarity(smu, irange, ilevel, srcdelay, vcmpl)
2. Vfwd_Vrev(smu, irange, ilevel, srcdelay, vcmpl)
3. Ileakage(smu, vrange, vlevel, srcdelay, icmpl)
4. TestStatus(testvalue, lolim, hilim)
5. TestFail(teststatus)
6. PartStatus(tst1fail, tst2fail, tst3fail)
7. BinPart(tst1fail, tst2fail, tst3fail)
8. DisplayTestStatus(testname, teststatus)
9. DisplayPartStatus(partstatus, bins)
10. DiodeTest(smu, ndiodes, speed)

The script excerpts that follow don't necessarily list the entire function. Rather, they just show the basic commands and code structure required to perform a particular test or series of tests. The actual diode test script contains the complete functions, as well as many comments and notations, which explain what the script does. Comments are identified by a double dash (--).

As one might expect, the function ChkPolarity() performs the polarity check test. The parameters passed to the function enable the user to specify the source current range, the test current level, the delay before checking the compliance state, and the voltage compliance level. The user can also select the source-measure unit (SMU) to use for the test. The Model 2601 has only one SMU, "smua," and the Model 2602 has two SMUs, "smua" and "smub." After performing the compliance check, the function returns either "FWD" or "REV" to the calling function or host program, indicating whether the diode is in the forward orientation or the reverse orientation. The 260X SourceMeter instruments don't use SCPI (Standard Commands for Programmable Instruments). However, the commands included in the Instrument Control Language (ICL) are similar to many instrument driver attributes and functions. Their plain English syntax makes them easy to learn, even for first-time users of the instrument. Keithley Application Note #2616, "Converting a Series 2400 SourceMeter SCPI Application to a Series 2600 System SourceMeter Script Application," shows how straightforward it is to move from one command language to another.

```
function ChkPolarity(smu, irange, ilevel, srcdelay, vcmpl)

    -- Default to smua if no smu is specified.
    if smu == nil then smu = smua end

    -- Temporary variables used by this function.
    local l_testcurrent, l_incompliance, l_polarity

    -- Configure source and measure settings
    smu.source.func = smu.OUTPUT_DCAMPS
    smu.source.rangei = irange
    smu.source.leveli = ilevel
    smu.source.limitv = vcmpl

    -- Wait before making measurement
    delay(srcdelay)
    -- Check compliance attribute (i.e., state); returns boolean true or false
    l_incompliance = smu.source.compliance

    if l_incompliance then              -- if source is in compliance, then
       l_polarity = "REV"               -- diode is reverse biased (reversed orientation)
    else                                -- otherwise
       l_polarity = "FWD"               -- diode is forward biased (forward orientation)
    end --if

    return l_polarity

end --function ChkPolarity
```

The forward voltage and reverse voltage breakdown tests both require measuring the voltage drop across the diode developed in response to an applied test current. The most significant difference is that for a diode in the forward orientation, the forward voltage test requires a positive applied test current and the reverse voltage test requires a negative applied test current. For a diode in the reverse orientation, the exact opposite test current polarities are required. Therefore a single function, Vfwd_Vrev(), can be used for both the forward and reverse voltage tests. The passed parameters enable the user to specify the SMU, the source current range, the test current level, the delay before measuring the voltage, and the voltage compliance level. In this example, the compliance level is also used to determine the voltage measurement range, but another range could be selected if desired. Finally, the function returns both the measured value of the test current and the measured value of the forward or reverse voltage to the calling function or host program, where the voltage value is inspected. The test current is measured to demonstrate the "source readback" capability of the instrument. This enables the user to verify the actual value of the test current. However, if the source accuracy alone is sufficient to meet test requirements, then the current measurement can be eliminated.

```
function Vfwd_Vrev(smu, irange, ilevel, srcdelay, vcmpl)

    -- Default to smua if no smu is specified.
    if smu == nil then smu = smua end

    -- Temporary variables used by this function.
    local l_testcurrent, l_vmeasured

    -- Configure source and measure settings
    smu.source.func = smu.OUTPUT_DCAMPS
    smu.source.rangei = irange
    smu.source.leveli = ilevel
    smu.source.limitv = vcmpl
    smu.measure.rangev = vcmpl

    -- Wait before making measurement
    delay(srcdelay)

    -- Measure current and voltage
    l_testcurrent, l_vmeasured = smu.measure.iv()
    return l_vmeasured, l_testcurrent
end --function Vfwd_Vrev
```

The function Ileakage() performs the leakage current measurement when the diode is reverse-biased. The parameters passed to the function specify the SMU, the source voltage range, the test voltage level, the delay before measuring the current and the current compliance level. The compliance level is used to determine the measurement range, but as before, another current range could be selected. Finally, if the source accuracy alone is sufficient to meet test requirements, then the voltage measurement can be eliminated.

```
function Ileakage(smu, vrange, vlevel, srcdelay, icmpl)
    -- Default to smua if no smu is specified.
    if smu == nil then smu = smua end

    -- Temporary variables used by this function.
    local l_testvoltage, l_imeasured

    -- Configure source and measure settings
    smu.source.func = smu.OUTPUT_DCVOLTS
    smu.source.rangev = vrange
    smu.source.levelv = vlevel
    smu.source.limiti = icmpl
    smu.measure.rangei = icmpl

    -- Wait before making measurement
    delay(srcdelay)

    -- Measure current and voltage
    l_imeasured, l_testvoltage = smu.measure.iv()

    --Set source output to 0
    smu.source.levelv = 0

    return l_testvoltage, l_imeasured

end --function Ileakage
```

The functions TestStatus() and PartStatus() are used to inspect the results of the functional tests and determine if the diode under test is good or bad. TestStatus() determines the PASS/FAIL status of an individual test within the test sequence, and creates a Boolean "fail" flag, which is TRUE if the subject test fails. The parameters passed to this function include the result of an individual test and the minimum and maximum acceptable values for the subject test result. PartStatus() determines the GOOD/BAD status of the part based on the results of all of the tests in the test sequence. Any individual test failure causes the part to be BAD. The Boolean "fail" flags for the functional tests are passed to PartStatus() to determine the overall part status.

```
function TestStatus(testvalue, lolim, hilim)

    -- Temporary variables used by this function.
    local l_status
    local l_testfail

    if (testvalue >= lolim) and (testvalue <= hilim) then
        l_status = "PASS"
        l_testfail = false
    else
        l_status = "FAIL"
        l_testfail = true
    end --if

    return l_status, l_testfail

end --function TestStatus

function PartStatus(tst1fail, tst2fail, tst3fail)

    -- Temporary variables used by this function.
    local l_status = "GOOD"

    if tst1fail or tst2fail or tst3fail then l_status = "BAD" end

    return l_status

end --function PartStatus
```

The BinPart() function increments a "Part Bins" table, simulating the bins a component handler may use. For this example, the three functional tests are always performed, even if the diode-under-test fails one of the tests. Upon completion of the tests, the part is binned based on the first test failure. If no failures occur, the part is put into the "good" bin. Although the component bins are simulated, this function actually does write binning codes to the instrument's digital I/O port, bits 1 through 4. It's assumed that all other bits or lines are protected or their state is a “don’t care.” Note that parameters passed to a function are passed by reference, which allows the table “bins” to be modified by this function.

```
function BinPart(tst1fail, tst2fail, tst3fail, bins)

     if tst1fail then                    -- If part first failed Forward Voltage Test
        bins[2] = bins[2] + 1
        digio.writeport(2)               -- Write bit pattern 0010 (decimal 2) to DIO

     elseif tst2fail then                -- If first failed Reverse Leakage Current Test
        bins[3] = bins[3] + 1
        digio.writeport(4)               -- Write bit pattern 0100 (decimal 4) to DIO

     elseif tst3fail then                -- If first failed Reverse Voltage Breakdown Test
        bins[4] = bins[4] + 1
        digio.writeport(8)               -- Write bit pattern 1000 (decimal 8) to DIO

     else                                -- Part is GOOD; does not fail any tests
        bins[1] = bins[1] + 1
        digio.writeport(1)               -- Write bit pattern 0001 (decimal 1) to DIO

     end --if

end -- function BinPart
```

The functions DisplayTestStatus() and DisplayPartStatus() write test results to the front panel display of the 260X SourceMeter instrument, primarily to show that such a capability exists. See the complete diode test script to look at these functions. These functions can simply be deleted if this functionality isn’t required for a specific test application.

Finally, the function DiodeTest() is the “test executive” function, which calls all of the other functions and dictates the order in which tests and other actions occur. The system controller might call this function whenever a new batch of diodes is installed in the handler and is ready to be tested. The pass parameters for this function include the SMU to use for the tests, the number of diodes to be tested, and a speed flag, which is either “SLOW” or “FAST.” The “SLOW” setting slows the tests down so the user can see the progression of the tests on the instrument display. To demonstrate the type of throughput that can be achieved with a 260X SourceMeter instrument, the “FAST” setting maximizes test speeds without regard for the impact on measurement accuracy.. A review of the complete diode test script will show that the speed flag affects the settings for the measurement integration time (NPLC), the state of the autozero function, the source (signal settling) delays, other program delays, and the state of the display. Test speed and measurement accuracy are generally inversely related, which means that speed and accuracy requirements usually have to be traded off for a real test application. The test script is easily modified to eliminate the speed flag.

The DiodeTest() function also defines the test variables for all of the functional tests, including source and measure ranges, test signal levels and polarities, inspection limits, and so on. It performs some initial setup of the 260X, which is applicable to all tests. It also configures the digital I/O port, setting up the binning control bits and the input and output trigger lines for the SOT trigger and the EOT trigger. For each diode being tested, the SourceMeter waits for an SOT from the handler to begin the tests and then issues an EOT signal after the binning code is written to the DIO port (i.e., the handler). Finally, it displays the results on the front panel displays and prints the test data the instrument’s output queue, where it can be retrieved by the host computer. Each print statement requires a corresponding “enter” statement in the host program to pull the data from the output queue.

The first test is the polarity check. This test determines if the diode is in forward or reverse orientation. If it’s reversed, then the test functions are called using the negative of the test current or voltage (e.g., Vfwd_Vrev(smu, irange, *-ilevel*, srcdelay, vcmpl)); otherwise, they are called using the normal polarity (e.g., Vfwd_Vrev(smu, irange, *ilevel*, srcdelay, vcmpl)). Using the negative of the test signal causes the polarity of the test result to be the opposite of what is expected for a diode in the forward orientation. Therefore, the negative of the test result is passed to the TestStatus() function when the diode is reversed (e.g., TestStatus(*-testvalue*, lolim, hilim)). This is simpler than adjusting the limits for the reversed orientation case. For the forward orientation case, the TestStatus() function is called using the measured value as is (e.g., TestStatus(*testvalue*, lolim, hilim))

```
function DiodeTest(smu, ndiodes, speed)
     •  Declare local variables including test parameters and inspection limits
     •  Set parameters based on speed flags
     •  Declare local variables to hold test data
     •  Perform initial setup of 260X applicable to all tests

     -- Configure Digital I/O Port
     digio.writeprotect = 0                -- Unprotect all bits
     digio.writeport(48)                   -- Set bits/lines 5 and 6 high; set all other bits low
     digio.writeprotect = 48               -- Write protect lines 5 and 6, which are used for triggers
     digio.writeport(0)                    -- Set all unprotected bits to zero

     -- Configure trigger lines 5 (input SOT) and 6 (output EOT)

     digio.trigger[5].mode = digio.TRIG_FALLING   -- Detect falling edge as input
     digio.trigger[5].clear()                     -- Clear any "latched" triggers

     digio.trigger[6].mode = digio.TRIG_FALLING   -- Assert TTL-low pulse for output
     digio.trigger[6].pulsewidth = 10E-6          -- Guaranteed minimum time output
                                                  -- trigger line will be asserted;
                                                  -- 10us is default value
     smu.source.output = smu.OUTPUT_ON

     for l_i = 1, ndiodes do

        l_sot_received = digio.trigger[5].wait(10E-3)      -- Wait for SOT; timeout after 10ms
        -- if not(l_sot_received) then
           -- Take appropriate action if times out before getting SOT trigger
           -- In this example, simply proceed to next command
        -- end --if

        -- Perform Polarity Check Test (PC)

        l_pc_bias_dir[l_i] = ChkPolarity(smu, l_pc_isrc_rng, l_pc_isrc_lev, l_pc_src_del, l_pc_vcmpl)

        if l_pc_bias_dir[l_i] == "REV" then-- If diode is in reversed orientation, then

           -- Perform Forward Voltage Test (Vf)

           l_vf_data[l_i], l_vf_test_curr[l_i] = Vfwd_Vrev(smu, l_vf_isrc_rng, -l_vf_isrc_lev, l_vf_
src_del, l_vf_vcmpl)
           l_vf_status[l_i], l_vf_fail[l_i] = TestStatus(-l_vf_data[l_i], l_vf_lowlim, l_vf_highlim)
           if l_disp_on then DisplayTestStatus("Vf = ", -l_vf_data[l_i], "V", l_vf_status[l_i], l_
delay) end

           -- Perform Reverse Leakage Current Test (Ir)

           -- Perform Reverse Voltage Breakdown Test (Vr)

        else   -- If diode is in forward orientation, then

           -- Perform Forward Voltage Test (Vf)

           l_vf_data[l_i], l_vf_test_curr[l_i] = Vfwd_Vrev(smu, l_vf_isrc_rng, l_vf_isrc_lev, l_vf_
src_del, l_vf_vcmpl)
           l_vf_status[l_i], l_vf_fail[l_i] = TestStatus(l_vf_data[l_i], l_vf_lowlim, l_vf_highlim)
           if l_disp_on then DisplayTestStatus("Vf = ", l_vf_data[l_i], "V", l_vf_status[l_i], l_
delay)
     end

           -- Perform Reverse Leakage Current Test1 (Ir)

           -- Perform Reverse Voltage Breakdown Test (Vr)

        end --if

        -- Determine cumulative part status

        l_part_status[l_i] = PartStatus(l_vf_fail[l_i], l_ir_fail[l_i], l_vr_fail[l_i])

        -- Bin the part

        BinPart(l_vf_fail[l_i], l_ir_fail[l_i], l_vr_fail[l_i], l_bins)

        -- Output EOT trigger

        digio.trigger[6].assert()

        -- Clear binning code (set all unprotected bits to zero)
```

```
            digio.writeport(0)

            if l_disp_on then DisplayPartStatus(l_part_status[l_i], l_bins) end
            delay(3 * l_delay)

        end --for

        smu.source.output = smu.OUTPUT_OFF


        -- Print test data to output queue

        print("D#", "POL", " Vf", " Ir", " Vr") -- Headings
        for l_i = 1, ndiodes do
            x = string.format("%03d,%s,%+2.4f,%+3.3E,%+2.4f",l_i, l_pc_bias_dir[l_i], l_vf_data[l_i], l_ir_
    data[l_i], l_vr_data[l_i])
            print(x)

        end --for

    end --function DiodeTest
```

Running the example diode test script using Test Script Builder or other application only creates the functions; it doesn't perform any tests. Actually executing the diode tests requires calling the DiodeTest() function. For example, to test 100 diodes, the system controller must only send the command "DiodeTest(smua, 100, "FAST")." In response to this command, the 260X will execute the diode test sequence 100 times using SMU A and the high speed test parameters. Test data will be put into the 260X output queue, so the system controller must read it back, either during the test or upon completion of the test. Imagine performing this test series using a conventional programmable instrument, where many of the commands must be sent over and over to the instrument via GPIB or RS-232. There is a significant speed improvement due simply to the reduced number of commands that must be sent to the instrument. Additionally, the system controller is freed up to interface with other instruments in the rack more frequently, which can further increase the overall system throughput.

Switching Multiple Diodes

For diode arrays or multi-die packages, switching can be used to connect a single SourceMeter instrument to each of the individual elements. The configuration discussed in Application Note #1805 applies equally well to a Series 2600 System SourceMeter instrument. The Series 2600 also provides another option. For little more than the cost of adding a switch to the test system, two Model 2602s can be configured as shown in ***Figure 3***. The configuration pictured is made possible by the Series 2600's Test Script Processor and new TSP-Link bus. The Test Script Processor isn't only an embedded computer; with TSP-Link, it is also a scalable computer. TSP-Link is an inter-unit communication and trigger synchronization bus—essentially an external backplane. With TSP-Link, users can connect multiple Series 2600 instru-

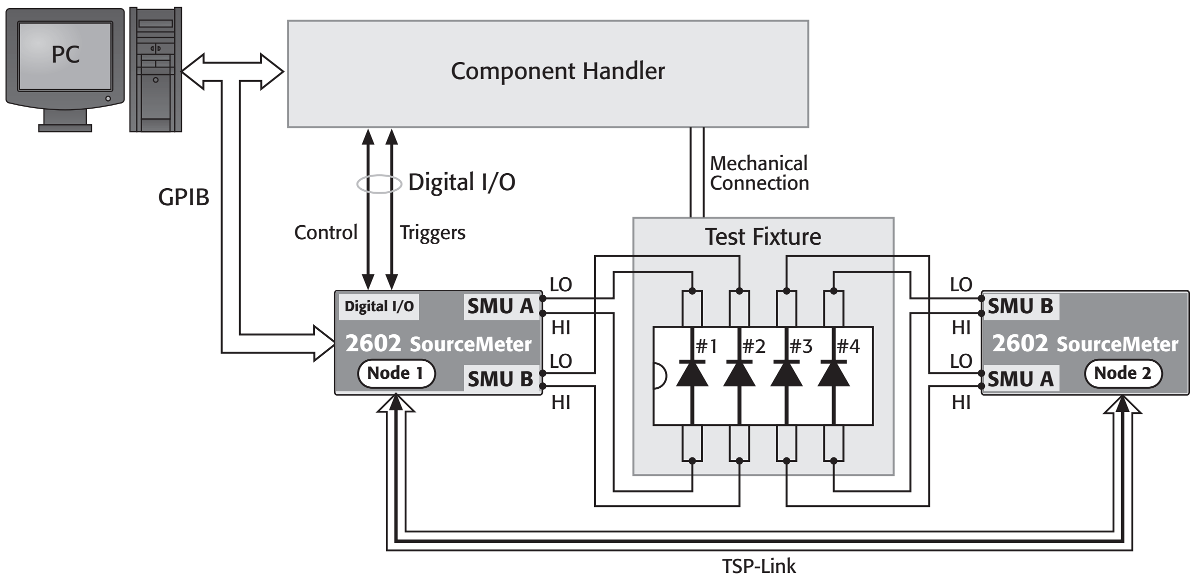

Figure 3. Using multiple Series 2600 SourceMeter instruments with TSP-Link™ to test multiple diodes

ments and program them as a single instrument, just as though they were housed in the same chassis. The simple programming interface allows creating powerful, high speed, multichannel tests quickly. There's no mainframe to restrict them, so users can create seamlessly integrated test systems with up to 64 nodes. In this example, each node is a dual-channel Model 2602, so the system could be scaled up to 128 SMU channels. Actual systems can be configured for any number of diode elements and for various electrical specifications. Switching is still easily combined with multiple Series 2600 instruments when appropriate for the application.

In ***Figure 3***, each 2602 is assigned a unique node number. Node 1, which is connected to the system controller in this example, is the "master" unit and Node 2 is the "slave" unit. A program running on Node 1 controls both nodes. The script for the single-diode, single-instrument example can be easily modified for use in a multi-node system. All commands simply require the inclusion of a node number. For example, the command for setting the current level in a single node system is "smua.source.leveli = ilevel." In a multi-node system, it would be "node[1].smua.source.leveli = ilevel" or "node[2].smua.source.leveli = ilevel." See the Series 2600 User and Reference Manuals for more information about setting up a multi-node system.

Typical Sources of Error

Lead Resistance

A common source of voltage measurement error is the series resistance from the test leads running from the instrument to the diode. This series resistance is added into the measurement when making a two-wire connection (***see Figure 4***). The effects of lead resistance are particularly detrimental when long connecting cables and high currents are used, because the voltage drop across the lead resistance becomes significant compared to the measured voltage.

To eliminate this problem, use the four-wire remote sensing method rather than the two-wire technique. With the four-wire method (***Figure 5***), a current is forced through the diode using one pair of leads and the voltage across the diode is measured through a second set of leads. As a result, only the voltage drop across the diode is measured.

Leakage Current

Stray leakage in cables and fixtures can be a source of error in measurements involving very low currents, such as for leakage currents. To minimize this problem, construct test fixturing with high resistance materials.

Another way to reduce leakage currents is to use the built-in guard of the Series 2600 instrument. The guard is a low impedance point in the circuit that is nearly the same potential as the high impedance point to be guarded. This is best illustrated by example (***Figure 6***).

In this example, the diode to be measured is mounted on two insulated standoffs (R_L). Guard is used in this circuit to

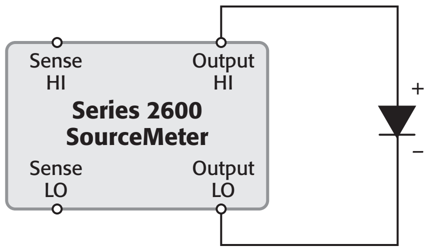

Figure 4. Two-wire connection

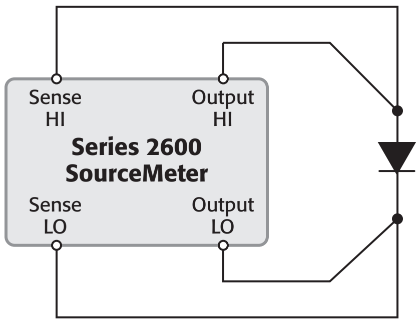

Figure 5. Four-wire connection

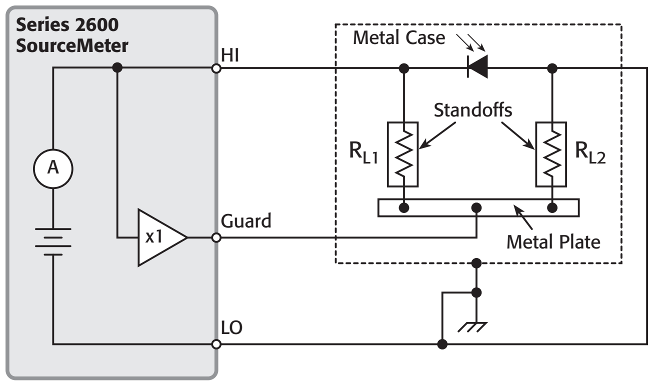

Figure 6. Series 2600 guarding technique

ensure that all the current flows through the diode and not through the standoffs. In general, a guard should be used when sourcing or measuring currents of less than 1μA. This circuit is guarded by connecting one of the Guard terminals of the instrument to the metal plate. This puts the bottom of insulator RL1 at almost the same potential as the top. Both ends of the insulator are at nearly the same potential, so no significant current flows through it. Thus, all the current will flow through the diode as desired.

WARNING: Guard is at the same potential as output HI. Therefore, if hazardous voltages are present at output HI, they are also present at the Guard terminal.

Electrostatic Interference

High resistance measurements may be affected by electrostatic interference, which occurs when an electrically charged object is brought near an uncharged object. To reduce the effect of electrostatic fields, a shield can be built to enclose the circuit being measured. As shown in ***Figure 6***, a metal shield connected to ground surrounds the diode under test. The LO of the SourceMeter instrument must be connected to the metal shield to avoid noise due to common mode and other interference. This also acts as a safety shield because the metal plate is at guard potential.

Test System Safety

Many electrical test systems or instruments are capable of measuring or sourcing hazardous voltage and power levels. It's also possible, under single fault conditions (e.g., a programming error or an instrument failure), to output hazardous levels even when the system indicates no hazard is present. These high voltage and power levels make it essential to protect operators from any of these hazards at all times. Protection methods include:

- Design test fixtures to prevent operator contact with any hazardous circuit.
- Make sure the device under test is fully enclosed to protect the operator from any flying debris. For example, capacitors and semiconductor devices can explode if too much voltage or power is applied.
- Double insulate all electrical connections that an operator could touch. Double insulation ensures the operator is still protected, even if one insulation layer fails.
- Use high reliability, fail-safe interlock switches to disconnect power sources when a test fixture cover is opened.
- Where possible, use automated handlers so operators do not require access to the inside of the test fixture or have a need to open guards.
- Provide proper training to all users of the system so they understand all potential hazards and know how to protect themselves from injury. It's the responsibility of the test system designers, integrators, and installers to make sure operator and maintenance personnel protection is in place and effective.

Equipment List

The following equipment is required to assemble a diode production test system and run the example scripts available from Keithley:

1. Keithley Model 2601 or 2602 SourceMeter instrument.
2. Component handler with test fixture
3. IEEE-488 (GPIB) Interface Card (KUSB-488, KPCI-488 or equivalent)
4. Keithley 7007 IEEE-488 interface cables
5. Custom DB-25 digital I/O handler interface cable to interface the instrument to the handler
6. Test leads to connect the instrument to the test fixture

Obtaining Example Scripts

The example script discussed in this Application Note is available for download on Keithley's web site (www.keithley.com). The script can be viewed, edited, loaded and executed using Test Script Builder.

Two diode test demonstration scripts are also included with Test Script Builder. These scripts are similar to the example script, but they don't include any SOT/EOT triggering. However, these scripts add front panel menu selections, which allow them to be run from the front panel. They also include a function to perform a Dynamic Impedance test, which is another common type of diode test, especially for zener diodes.

KEITHLEY

Keithley Instruments, Inc. 28775 Aurora Road • Cleveland, Ohio 44139 • 440-248-0400 • Fax: 440-248-6168
1-888-KEITHLEY (534-8453) • www.keithley.com

No. 2633
0605