KEITHLEY

Application Note Series

Number 2998

Migrating from Series 2600 System SourceMeter® Instruments to Series 2600A

Shorter product life and increased technology drive the need for configurable test systems that maximize manufacturers' investments. The introduction of the Series 2600A System SourceMeter instruments updates the existing Series 2600 instruments to address the parallel test requirements and time-critical applications of today's test engineers.

Just what improvements were made? ***Table 1*** provides an overview of the significant differences between the Series 2600A and the Series 2600.

Feature	Capability/performance	
	Series 2600 (non-A)	Series 2600A
Built-in sweep generation	Using factory script only	Built-in ICLs to configure linear, log, and list sweeps. Tie directly to trigger model for precise timing control.
Flexible trigger model	No	Yes
Precision pulse timing	Minimum pulse width = 200µs. Pulsing synchronization limited to maximum of two channels. Pulse width of SMUA must be 40µs longer than pulse width of SMUB.	Minimum pulse width = 100µs. Includes ability to output synchronize pulses from multiple nodes without regard to pulse width.
Pulsing in extended operating area	Only on Models 2611, 2612	10A pulse capability on all models
Parallel test	Only using multiple TSP-Link networks. Synchronization possible only by external cabling between digital I/O ports.	Asynchronous and synchronous parallel test possible within a single TSP-Link network. No additional cabling needed for synchronization.
Single measure-only reading rate to memory	10,000 readings/second	20,000 readings/second
USB memory storage	No	Yes
Lower wideband source noise	Source noise varies by model. Models 2601/02/11/12 had higher source noise than 2635/36.	Lower source noise on Models 2601A/02A/11A/12A to meet 2635/36 specifications. Source noise for all models is less than 20mVp-p (typical).
LAN interface with software LXI triggering	Not available	Yes
Embedded web-based software for quick, easy testing	Not available	Yes, and it's free!

Table 1. Comparison of features/capabilities of the Series 2600 System SourceMeter Instruments features vs. Series 2600A.

Current users of the Series 2600 SourceMeter instruments will find migrating their applications to Series 2600A instruments easy. Nearly all the commands for the Series 2600A are backward compatible, so existing programs should run with little or no modification. Better still, with some code modification to support the new trigger model, the Series 2600A offers users the potential to achieve throughput improvements, as well as synchronization of source and measure operations across multiple units.

To highlight the advantages the Series 2600A offers, this application note considers new features in light of common semiconductor test applications.

Built-In Sweep Generation Capabilities Enable Faster Throughput and Precise Timing Control

Flexibility has been one of the advantages Series 2600 instruments provided. Such flexibility was due in part to their independence from a rigid trigger model, which was common to SCPI-based instrumentation. However, using a trigger model has the benefit of more precise timing control over source and measure operations. With the Series 2600A, Keithley introduces a trigger model with far more flexibility and better timing control than trigger models for other Keithley products and competitive products. Series 2600A trigger latency (i.e., the period from receipt of input trigger to source change) can be less than 10µs. Additionally, users can achieve sub-microsecond synchronization between source changes on a single unit or across a multi-unit network connected via TSP-Link. Also, programming flexibility is maintained because the user has the choice of operating outside of the trigger model.

Built-in sweep generation functions to program linear, log, and list sweeps easily in a single command statement are provided in addition to the trigger model. Similar sweep capability was available on the Series 2600 (non-A) but only by using factory script functions or user-created loops that ramped source levels and took measurements.

To understand the power of the new triggering and sweeping capabilities of the Series 2600A, this application note now considers the implementation of a Gummel test on a Bipolar Junction Transistor (BJT). A simplified schematic of the test is in ***Figure 1***. Let's first consider how the test would be performed using a Model 2636 (non-A) SourceMeter instrument. Refer to the appendix of this note for the test script to program a Model 2636 to perform a Gummel test.

The results of this test are plotted in ***Figure 2***. The set-up time is the time required to configure the source and measure parameters of the SMU prior to executing the sweep. Therefore, the set-up interval is required only once. Thereafter, the sweep can be run as many times as the user desires. In this test, the Model 2636 was configured to take measurements at very short integration times (NPLC = 0.001), but default source

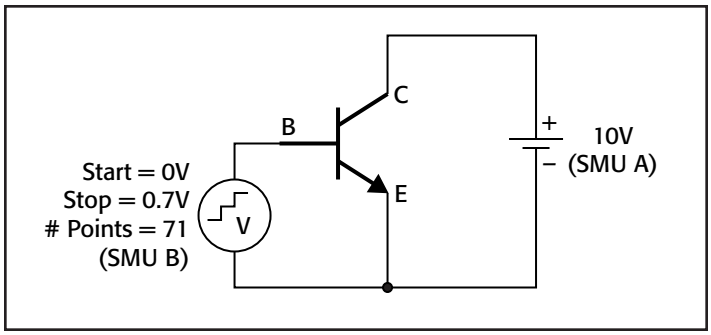

Figure 1. SMU configuration for Gummel test on BJT

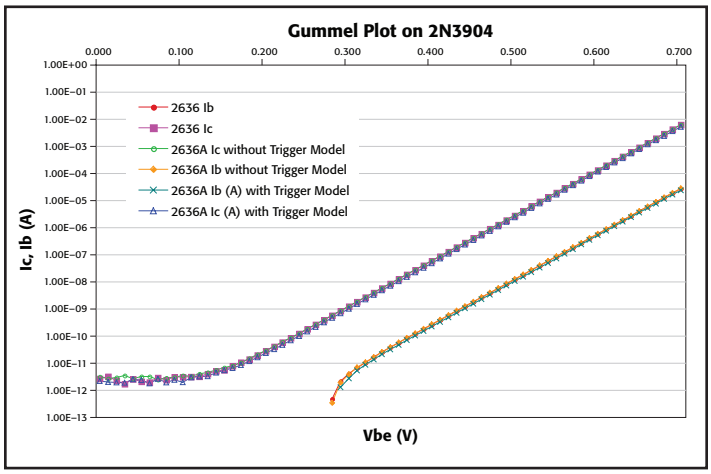

Figure 2. Gummel plot data using Models 2636 and 2636A.

and measure delays were maintained for necessary settling on low current ranges. For the Model 2636, the set-up time was approximately 78ms. The test execution time was 91.9ms.

The same code was also run on a Model 2636A. That data is also plotted in ***Figure 2***. Note the excellent correlation between the data produced by the two instruments. The Model 2636A set-up times and test execution times are comparable to those of the Model 2636.

Next, the Model 2636A is configured to perform a Gummel test using the new sweeping functions and trigger model. See the appendix for the test script to program the Model 2636A to execute a Gummel test using the trigger model.

The results from this test are also plotted in ***Figure 2***. Note again how well the data correlates with the data taken with the Model 2636.

The set-up time for the Model 2636A when using the trigger model was approximately 88ms, slightly longer (by about 9ms) when using the trigger model than without using the trigger model. Again, however, note that this set-up interval is non-recurring. Also, the increase in set-up time is offset by the improvement in test execution time. The test execution time when using the trigger model was 55.8ms, a test time reduction of nearly 40%, using the same default source and measure delays as the Model 2636. This example clearly shows the speed advantage of the Series 2600A trigger model, even when making low current measurements.

Achieve Precision Pulse Timing and DC Bias Turn-On Sequencing

Precision Timing of DC Bias Turn-On Sequence

In many ICs and ASICs, the turn-on sequence of multiple power supplies is critical. Equally important is the timing between the turn-on of the various supplies. In Series 2600 (non-A) instruments, the precision of the delay between the turn-on of various power supplies was limited by the command execution of sequential command lines. The following command sequence might be used to program a 200μs delay between the turn-on of SMUA and SMUB:

```
reset()
smua.source.levelv = 0
smub.source.levelv = 0
smua.source.output = smua.OUTPUT_ON
smub.source.output = smub.OUTPUT_ON
smua.source.levelv = 5
delay(200e-6)
smub.source.levelv = 3.5
```

Figure 3 shows the results of executing the preceding sequence on a Model 2602. The actual delay is nearly 1.5ms, more than 7 times the programmed value. Furthermore, the command execution time is multiplied for each node connected to the TSP-Link network.

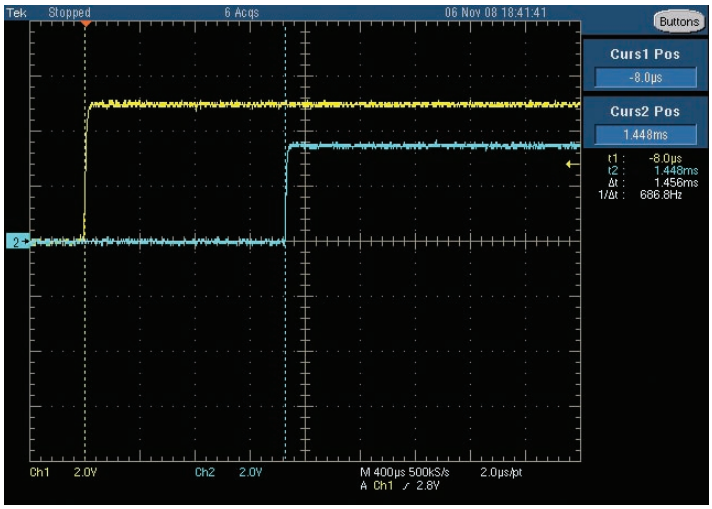

Figure 3. DC Bias Sequencing with Model 2602. Desired delay was 200µs. Actual delay was 1.45ms.

The Series 2600A trigger model allows the user to preprogram source levels and delay times with accuracies on the order of microseconds. The following command sequence programs a 200μs delay between turn-on of SMUA and SMUB on a Model 2602A.

```
reset()
smua.trigger.source.listv({5})
smua.trigger.source.action = smua.ENABLE
smub.trigger.source.listv({3.5})
smub.trigger.source.action = smub.ENABLE
--Program timer with a delay of 200us
trigger.timer[1].delay = 200e-6
trigger.timer[1].stimulus = smua.trigger.ARMED_EVENT_ID
trigger.timer[1].count = 1
trigger.timer[1].passthrough = false
```

```
smua.trigger.source.stimulus = 0
smub.trigger.source.stimulus = trigger.timer[1].EVENT_ID
smua.trigger.endsweep.action = smua.SOURCE_HOLD
smub.trigger.endsweep.action = smub.SOURCE_HOLD
smua.source.output = smua.OUTPUT_ON
smub.source.output = smub.OUTPUT_ON
smub.trigger.initiate()
smua.trigger.initiate()
```

The results of the preceding command sequence are captured in ***Figure 4***. Note that the delay between the channels is approximately 196.8µs (measured at 50% amplitude of SMUA to 50% amplitude of SMUB).

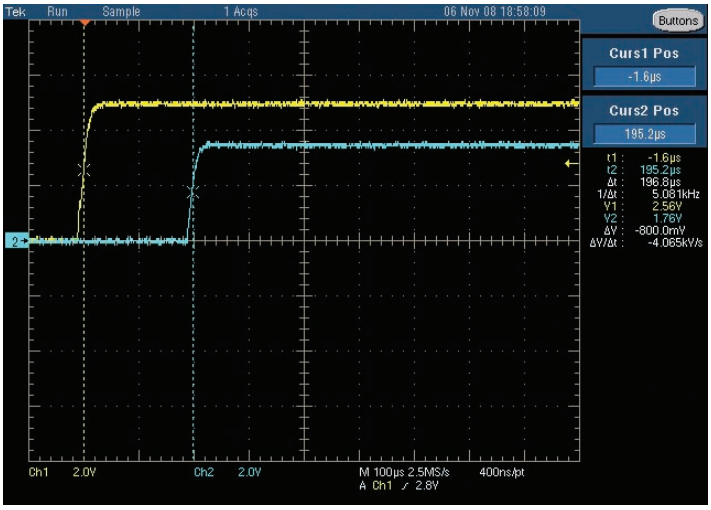

Figure 4. DC bias sequencing with Model 2602. Desired delay was 200µs. Actual delay was 196.8µs.

With Series 2600A instruments, a minimum delay time of 10µs seconds is possible.

Precision Pulsing in Diode Junction Temperature Measurements

The precision timing made possible when using the Series 2600A trigger model has advantages in applications that involve pulsing. Consider, for instance, measuring the junction temperature of diodes.

Junction temperature measurements are performed by correlating device temperature to diode forward voltage drop. The forward voltage is measured using very short current pulses in order to avoid additional heating of the junction to capture the true temperature of the junction. Short pulses are especially important if the drive current is high, as is often the case with high brightness LEDs.

All Series 2600A SourceMeter instruments provide 10A pulse capability, which the original Models 2601 and 2602 did not. Series 2600A instruments also cut the minimum pulse width of the Series 2600 instruments in half, allowing users to program pulses as short as 100µs. Moreover, pulses can now be precisely synchronized in time over all nodes over a Series 2600A network. Compare the dual channel pulse performance between Models 2612 and 2612A in ***Figures 5*** and ***6***.

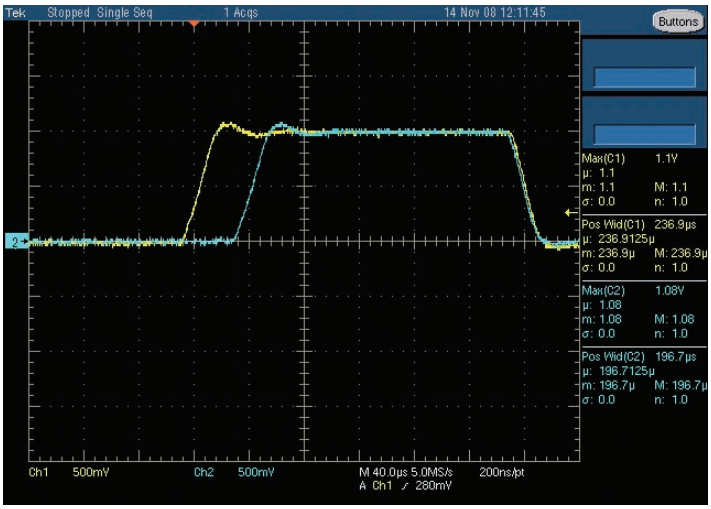

Figure 5. Two-channel 10A pulse into 100mΩ resistor. Shown are minimum programmable pulse widths of 200µs and 240µs, created using KIPulse scripts on a Model 2612.[1]

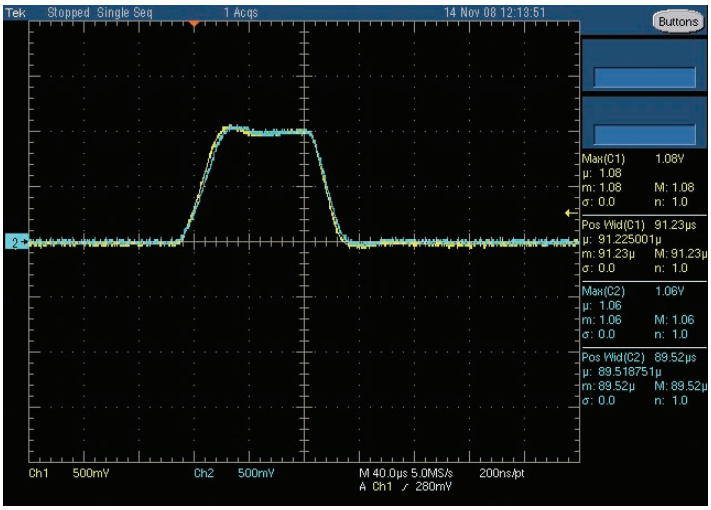

Figure 6. Two-channel 10A pulse into 100mΩ resistor. Minimum programmable pulse width of 100µs shown, created using trigger model on Model 2612A.

Other Significant Features/Enhancements

Asynchronous and Synchronous Parallel Testing

Parallel script execution is the ability to run scripts on remote nodes in a TSP-Link network. When the Series 2600 was originally introduced, scripts could only be executed on one node of a TSP-Link network. That one node (the master node), defined by a unique GPIB address, was capable of controlling the source and measure actions of other nodes (remote nodes), but such operations could only be performed sequentially. These limitations made it impossible to obtain true parallel test in a single TSP-Link network. Consequently, the use of TSP-Link was largely abandoned for parallel test applications in favor of separate GPIB communication cables to multiple nodes and

1 Previous software written for Series 2600 Instruments that uses KIPulse scripts works without modification on Series 2600A Instruments.

triggering by means of a separate connection to the Series 2600 digital I/O port.

Series 2600A addresses the limitations of parallel test over TSP-Link by enabling execution of scripts on remote nodes of the TSP-Link network. ***Figure 7*** illustrates one parallel test application. Nodes can be configured into groups and a group leader can execute scripts to control the operations of any node in its group. Such group assignments are dynamic, making it possible to test one device with a collection of SMUs and then later reassign the same SMUs to different groups in order to run tests on multiple devices independently. These assignments are made using only software commands. Furthermore, the Series 2600A trigger model makes it possible to synchronize the source and measure actions across all nodes within the TSP-Link network using hardware trigger signals in the TSP-Link connection.

This flexibility of test configurations maximizes resource utilization by allowing test engineers to regroup and reassign SMUs quickly in preparation for testing a new device. Dynamic configuration, coupled with the ability to perform tests in parallel, lowers the cost of test by minimizing the time spent changing hardware setups and allowing independent tests to run on all SMUs simultaneously.

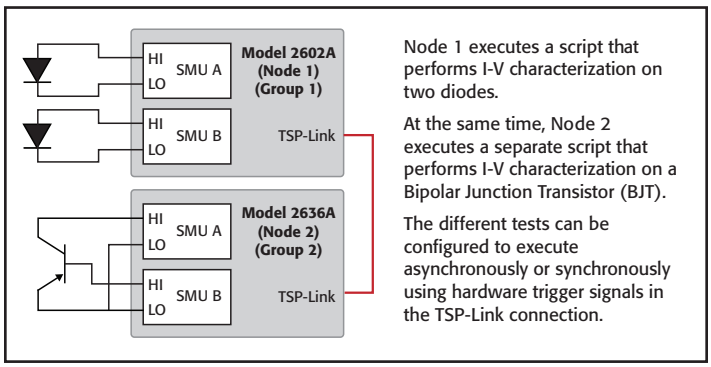

Figure 7. Example configuration for parallel script execution.

Easier Communication and Synchronization with External Instruments

As mentioned previously, Series 2600A instruments connected via TSP-Link offer enhanced triggering and synchronization capabilities. Such synchronization also easily extends to other TSP-based instruments using the three hardware synchronization lines in the TSP-Link cable.

The Series 2600A trigger model also enables precise triggering between the SMU and external non-TSP and non-Keithley instruments. This is done by (1) low trigger latency for digital I/O triggering and (2) LAN triggering:

1. Digital I/O Triggering: Trigger latency from receipt of digital input trigger to start of source or measure change has been reduced from 150μs on the Series 2600 to 10μs on the Series 2600A. This permits more precise alignment of SMU operations with operations on external instruments.
2. LAN triggering: Series 2600A instruments are LXI-C compatible. Beyond basic LXI-C specifications, however, Series 2600A includes software LAN triggers that permit handshaking with LXI instruments with triggering capability.

Command Compatibility with Programs Written for Series 2600 Instruments

One cost aspect of implementing new test equipment includes the cost for migrating existing, stable software programs to support new products. These types of costs are significantly minimized when migrating to Series 2600A SourceMeter Instruments because these instruments are nearly 100% command-compatible with software programs written for Series 2600 instruments. The minor differences between them are outlined in the following paragraphs.

Script Management Changes

With Series 2600A instruments, scripts can be stored in internal nonvolatile memory or in a memory device plugged into the front panel USB port. To enable this USB storage functionality, the `save()` function for scripts has been updated to allow the user to specify a directory and filename. For example, the following command can be used to save the script "myscript" to a folder on the USB port named "test1" as a file named "example1.tsp".

```
myscript.save("usb1/test1/example1.tsp")
```

It is no longer possible to save a script with a different name using the `save()` function, as was possible with the Series 2600 (non-A). To save a script to a different name, the script must be renamed before the `save()` function[2] is called.

Make it more difficult to access script source code

Occasionally, programmers want to discourage other users from accessing the source code of a script. In the Series 2600 instruments, the Binary Distribution example script (available on www.keithley.com) could be used to make it difficult to read the script source code. In the Series 2600A instruments, however, the source code of a script can be deleted and still maintain the ability to run the script. To delete the script source code, set the source attribute of a script to nil. Afterward, reading the script source code returns only the binary-encoded version of the code. The following is an example of how to delete the source code of the `myScript`. The results of attempts to retrieve the script source code are included.

```
myScript.source = nil
print(myScript.source)
```

2 A script can be renamed by assigning a string to the script's name attribute. The following example renames the script "myscript" to "exampleScript".

```
myscript.name = "exampleScript"
```

Renaming a script does change the name of any variables that reference the script. When a script is created using the `loadscript` function, the Series 2600/2600A SourceMeter Instrument also creates a global variable with the same name. Generally, this global variable is accessed in order to perform operations on the script. Therefore, in the preceding example, the variable `myscript` still references the script with the new name `"exampleScript"`.

Series 2612A returns the following:

```
loadstring(table.concat(
{
    "\27LuaP\0\4\4\4\6\8\9\9\8A}\245\23h\147\9\182\0\
0\ 0\1\0\0\0",
    "\0\0\0\0\0\2\0\0\0\12\0\0\0\1\0\0\0\1\0\0\0\1\0\
0\0\2\0\0\0\2",
    "\0\0\0\2\0\0\0\3\0\0\0\3\0\0\0\3\0\0\0\4\0\0\0\4
\0\0\0\4\0\0",
    "\0\0\0\0\0\0\0\0\8\4\0\0\0\5smua\0\4\0\0\0\
7source\0\4\0\0",
    "\0\5func\0\3?\240\0\0\0\0\0\0\4\0\0\0\7leveli\0\
4\0\0\0\8mea",
    "sure\0\3\0\0\0\0\0\0\0\4\0\0\0\6sense\0\0\0\0\
0\0\0\0\0\12\0",
    "\0\0\5\0\0>\198\0~?I\0\0\0\5\0\0>\198\0\127?I\0\
0\0\5\0\0?\198",
    "\0~@\9\0\0\0\5\0\128\191I\0\0\128\27"
}
))()
```

Data Storage Changes

The 2× increase of the maximum measurement to memory reading rate in the Series 2600A is sure to demand more reading buffer storage. In addition to increasing the internal reading buffer storage on the Series 2600A, a front panel USB port was added to facilitate user access to more memory. Users can now select from a variety of data storage devices in order to meet their memory requirements without being limited by the Series 2600A's internal memory allotment.

With the original Series 2600s, instrument data in dedicated reading buffers (`smua.nvbuffer1`, `smua.nvbuffer2`, `smub.nvbuffer1`, and `smub.nvbuffer2`) was automatically saved to non-volatile memory. Series 2600A instruments require a specific function call to save data in dedicated reading buffers. Use the `smuX.savebuffer` function to save reading buffers to internal memory or to a USB memory device. Specify a filename and format (.csv or .xml) with the `smuX.savebuffer` function to store to USB memory device. When `smuX.savebuffer` is called without any parameters, the buffer is saved to the Series 2600A internal nonvolatile memory.

Series 2600A reading buffers also have a different base timestamp than reading buffers in Series 2600 instruments. The base timestamp for Series 2600A instruments is in number of seconds from 12:00 AM January 1, 1970.[3] For Series 2600 reading buffers, the base time stamp is the number of seconds since the instrument was powered on.

Miscellaneous Changes

There are a few other differences of note between Series 2600 and Series 2600A instruments:

- **Double-precision math:** Series 2600 instruments use single-precision floating point math. Series 2600A instruments use double-precision floating point math. This is important to note if the existing software program was written in a way as to be susceptible to roundoff or truncation error. Typical ramifications include:
  - Different exit behaviors for conditional loops. For example, assume a loop is programmed to exit when (*voltage* <1.0). A measured value of 0.999999990 would continue looping in double-precision, but exit in single-precision.
  - Slightly different results for calculated parameters and mathematical values. This includes interpolated or extrapolated values (such as threshold voltage); values from numerical differential or slope calculations (such as transconductance); or numerically integrated values (such as charge calculated by integrating measured current).
- **`reset()` command:** Issuing a `reset()` on the Series 2600A is far more comprehensive than on Series 2600 instruments. Some properties have different default states. For example, the default trigger mode for digital I/O lines is now `TRIG_BYPASS`, which permits direct read and write to digital I/O lines. On firmware version 1.3.4 and earlier, the default mode for Series 2600 (non-A) instruments was `TRIG_FALLING`. To send and receive triggers on the Series 2600A digital I/O lines, the mode must be explicitly set for one of the supported hardware trigger modes.
- **Updating of factory scripts:** Any software programs that call functions created in Keithley factory scripts are backward compatible. Moreover, users can access the source code for all factory scripts, including KI Pulse functions. Many of the factory script functions have been rewritten to utilize new features of Series 2600A instruments, such as the trigger model. This source code can serve as an excellent set of examples for users who need to write similar functions and scripts for their own applications.

Conclusion

Keithley's release of Series 2600A represents a commitment to meet the ever-expanding needs of today's test engineers. The high level of command compatibility between the Series 2600 and Series 2600A allows test engineers to maintain their existing test stands but still invest in new technology. Engineers can prepare to unlock new Series 2600A capabilities at times convenient in their test processes. Implementing these enhancements allows for potential throughput enhancement, precise timing of source and measure actions, dynamic test configurations, and improved parallel test capabilities.

Appendix

Series 2600-Compatible Code for BJT Gummel Plot test

```
function Gummel(vbestart, vbestop, vbesteps, vcebias)
--[[Configure SMUB to perform a voltage sweep on
the base (Vbe) from start to stop in a user defined
number of steps while SMUA performs a fixed voltage
bias on the collector-emitter. Returns measured Ib,
```

3 The base timestamp of the reading buffer is different from timestamps element. The base timestamp is a real-time reference for the first reading stored in the reading buffer. The timestamps attribute of the reading buffer is an array of timestamps, in seconds for each reading of the reading buffer. These timestamps are relative to the base timestamp. Therefore, the timestamp for the first reading stored in the buffer is always zero seconds.

```
Ic, and Vbe.]]--

--Global variables
timer.reset() -- Reset timer to start setup timer.
local l_icmpl = 100E-3 --Source compliance

--Shared local variables
local l_nplc = 0.001 --Integration rate of
measurement

--Local sweep variables
local l_vbestart = vbestart --Base sweep start
voltage
local l_vbestop =  vbestop --Base sweep stop voltage
local l_vbesteps = vbesteps --Number of steps in
sweep

local l_vcebias = vcebias --Collector-emitter voltage

--Calculate Vbe step size
local l_vbestep = (l_vbestop - l_vbestart)/ (l_
vbesteps - 1)
local l_vbesource_val = l_vbestart --Source value
during sweep
local l_vbe_i = 1 --Iteration variable

--Data tables
local l_vbe = {} --Create data table for sourced
voltage
local l_ic = {} --Create data table for Ic
local l_ib = {} --Create data table for Ib

smua.reset() --Reset SMU
smub.reset() --Reset SMU

errorqueue.clear() --Clear the error queue

--Configure Collector/Emitter (SMUA) source and
measure settings
smua.source.func = smua.OUTPUT_DCVOLTS
smua.source.autorangev = smua.AUTORANGE_ON --Enable
source autorange
smua.source.levelv = 0
smua.source.limiti = l_icmpl
smua.measure.autorangei = smua.AUTORANGE_ON --Enable
measure autorange

smua.measure.autozero = smua.AUTOZERO_ONCE
smua.measure.nplc = l_nplc --Measurement integration
rate

smua.source.output = smua.OUTPUT_ON --Enable Output

--Configure Base (SMUB) source and measure settings
smub.source.func = smub.OUTPUT_DCVOLTS
smub.source.autorangev = smub.AUTORANGE_ON --Enable
source autorange
smub.source.levelv = 0
smub.source.limiti = l_icmpl
smub.measure.autorangev = smub.AUTORANGE_ON --Enable
measure autorange

smub.measure.autozero = smub.AUTOZERO_ONCE
smub.measure.nplc = l_nplc --Measurement integration
rate

smub.source.output = smub.OUTPUT_ON --Enable Output

smua.source.levelv = l_vce_bias

local setuptime = timer.measure.t() -- Capture setup
time
print("Setup Time = "..setuptime)
--End setup

timer.reset() -- Reset timer to start test execution
timer
--Execute sweep
for l_vbe_i = 1,l_vbesteps do

if (l_vbe_i == 1) then --Intialize start source value
      l_vbesource_val = l_vbestart
   end --if

   delay(0.01) --Delay
   l_vbe[l_vbe_i] = smub.measure.v() --Measure Vbe
   l_ib[l_vbe_i] = smub.measure.i() --Measure Ib
   l_ic[l_vbe_i] = smua.measure.i() --Measure Ic

   l_vbesource_val = l_vbesource_val + l_vbestep
--Calculate new source value


   if (l_vbe_i == l_vbesteps) then --Reinitialize
voltage value after last iteration
      l_vbesource_val = l_vbestart
   end --if

      smub.source.levelv = l_vbesource_val --Increment
source
end --for

smua.source.output = smua.OUTPUT_OFF --Disable output
smub.source.output = smub.OUTPUT_OFF --Disable output

smua.source.levelv = 0 --Return source to bias level
smub.source.levelv = 0 --Return source to bias level


local total_test_time = timer.measure.t()
print("Total Test Time           = "..total_test_time)

end--function Gummel()

--Use following command to run Gummel test
Gummel(0,0.7,71,10)
```

Series 2600A Code for BJT Gummel Plot Test (Uses Trigger Model)

```
function Gummel(vbestart, vbestop, vbesteps, vcebias)
--[[Configure SMUB to perform a voltage sweep on
the base (Vbe) from start to stop in a user defined
number of steps while SMUA performs a fixed voltage
bias on the collector-emitter. Returns measured Ib,
Ic, and Vbe.]]--

--Global variables
timer.reset() -- Reset timer
local l_icmpl = 100E-3 --Source compliance

--Shared local variables
local l_nplc = 0.001 --Integration rate of
measurement

--Local sweep variables
local l_vbestart = vbestart --Base sweep start
voltage
local l_vbestop =  vbestop --Base sweep stop voltage
local l_vbesteps = vbesteps --Number of steps in
sweep

local l_vcebias = vcebias --Collector-emitter voltage

local l_vbestep = (l_vbestop - l_vbestart)/ (l_
vbesteps - 1) --Vbe step size
local l_vbesource_val = l_vbestart --Source value
```

```
during sweep
local l_vbe_i = 1 --Iteration variable

smua.reset() --Reset SMU
smub.reset() --Reset SMU

errorqueue.clear() --Clear the error queue

trigger.blender[1].clear()
--[[ Initialize reading buffers by clearing and
setting them to append mode, collect time stamps and
source values ]]--
smua.nvbuffer1.clear()
smua.nvbuffer2.clear()
smua.nvbuffer1.appendmode = 1
smua.nvbuffer2.appendmode = 1
smua.nvbuffer1.collectsourcevalues = 1
smua.nvbuffer2.collectsourcevalues = 1
smua.nvbuffer1.collecttimestamps = 1
smua.nvbuffer2.collecttimestamps = 1
smua.makebuffer(l_vbesteps)

smub.nvbuffer1.clear()
smub.nvbuffer2.clear()
smub.nvbuffer1.appendmode = 1
smub.nvbuffer2.appendmode = 1
smub.nvbuffer1.collectsourcevalues = 1
smub.nvbuffer2.collectsourcevalues = 1
smub.nvbuffer1.collecttimestamps = 1
smub.nvbuffer2.collecttimestamps = 1
smub.makebuffer(l_vbesteps)

--[[Configure Sweeping SMUB]]--
smub.trigger.arm.stimulus = 0
smub.trigger.source.stimulus = 0
smub.trigger.endpulse.stimulus = trigger.blender[1].
EVENT_ID
smub.trigger.measure.stimulus = smub.trigger.SOURCE_
COMPLETE_EVENT_ID
smub.trigger.endpulse.action = smub.SOURCE_HOLD
smub.trigger.endsweep.action = smub.SOURCE_IDLE
smub.trigger.arm.count = 1
smub.trigger.count = l_vbesteps
smub.trigger.measure.action = smub.ENABLE
smub.trigger.source.action = smub.ENABLE

--[[Configure  SMUA]]—
smua.trigger.arm.stimulus = 0
smua.trigger.source.stimulus = 0
smua.trigger.endpulse.stimulus = 0
smua.trigger.measure.stimulus = smub.trigger.SOURCE_
COMPLETE_EVENT_ID
smua.trigger.endpulse.action = smua.SOURCE_HOLD
smua.trigger.endsweep.action = smua.SOURCE_IDLE
smua.trigger.arm.count = 1
smua.trigger.count = l_vbesteps
smua.trigger.measure.action = smua.ENABLE
smua.trigger.source.action = smua.ENABLE

trigger.blender[1].orenable = false  --{set to false
for AND mode }

trigger.blender[1].stimulus[1] = smua.trigger.MEASURE_
COMPLETE_EVENT_ID
trigger.blender[1].stimulus[2] = smub.trigger.MEASURE_
COMPLETE_EVENT_ID

--Configure Collector/Emitter (SMUA) source and
measure settings
smua.source.func = smua.OUTPUT_DCVOLTS
smua.source.autorangev = smua.AUTORANGE_ON --Disable
source autorange
smua.source.levelv = 0
smua.source.limiti = l_icmpl
smua.measure.autorangei = smua.AUTORANGE_ON --Disable
measure autorange
smua.measure.autozero = smua.AUTOZERO_ONCE
smua.measure.nplc = l_nplc --Measurement integration
rate
smua.measure.count = 1
smua.source.delay = 0
smua.sense = smua.SENSE_LOCAL

smua.trigger.source.linearv(l_vce_bias, l_vce_bias,
l_vbesteps)

--Configure Base (SMUB) source and measure settings
smub.source.func = smub.OUTPUT_DCVOLTS
smub.source.autorangev = smub.AUTORANGE_ON --Enable
source autorange
smub.source.levelv = 0.0
smub.source.limiti = l_icmpl
smub.measure.autorangei = smub.AUTORANGE_ON --Enable
measure autorange
smub.measure.autozero = smub.AUTOZERO_ONCE
smub.measure.nplc = l_nplc --Measurement integration
rate
smub.source.delay = 0
smub.measure.count = 1

smub.trigger.source.linearv(l_vbestart, l_vbestop,  l_
vbesteps)
smub.sense = smub.SENSE_LOCAL


--[[Turn SMUs ON, enable Output ]]--
smua.source.output = smua.OUTPUT_ON --Enable Output
smub.source.output = smub.OUTPUT_ON --Enable Output

smua.source.levelv = l_vce_bias

local setuptime = timer.measure.t()
print("Setup Time = "..setuptime)          --End setup

timer.reset() -- Reset timer

--[[  Initiates the steping and sweeping smus
]]--
smua.trigger.initiate()
smub.trigger.initiate()

--[[  Waits for the sweeps to complete ]]--
waitcomplete()

--[[  Diasble Output ]]          --
smua.source.output = smua.OUTPUT_OFF --Disable output
smub.source.output = smub.OUTPUT_OFF --Disable output

smua.source.levelv = 0 --Return source to bias level
smub.source.levelv = 0 --Return source to bias level


local total_test_time = timer.measure.t()
print("Total Test Time          = "..total_test_time)

end--function Gummel()

--Use following command to run Gummel test
Gummel(0,0.7,71,10)
```

KEITHLEY

A GREATER MEASURE OF CONFIDENCE

KEITHLEY INSTRUMENTS, INC. ■ 28775 AURORA ROAD ■ CLEVELAND, OHIO 44139-1891 ■ 440-248-0400 ■ Fax: 440-248-6168 ■ 1-888-KEITHLEY ■ www.keithley.com

BELGIUM
Sint-Pieters-Leeuw
Ph: 02-3630040
Fax: 02-3630064
info@keithley.nl
www.keithley.nl

CHINA
Beijing
Ph: 8610-82255010
Fax: 8610-82255018
china@keithley.com
www.keithley.com.cn

FINLAND
Espoo
Ph: 09-88171661
Fax: 09-88171662
finland@keithley.com
www.keithley.com

FRANCE
Saint-Aubin
Ph: 01-64532020
Fax: 01-60117726
info@keithley.fr
www.keithley.fr

GERMANY
Germering
Ph: 089-84930740
Fax: 089-84930734
info@keithley.de
www.keithley.de

INDIA
Bangalore
Ph: 080-26771071,-72,-73
Fax: 080-26771076
support_india@keithley.com
www.keithley.com

ITALY
Peschiera Borromeo (Mi)
Ph: 02-5538421
Fax: 02-55384228
info@keithley.it
www.keithley.it

JAPAN
Tokyo
Ph: 81-3-5733-7555
Fax: 81-3-5733-7556
info.jp@keithley.com
www.keithley.jp

KOREA
Seoul
Ph: 82-2-574-7778
Fax: 82-2-574-7838
keithley@keithley.co.kr
www.keithley.co.kr

MALAYSIA
Penang
Ph: 60-4-643-9679
Fax: 60-4-643-3794
chan_patrick@keithley.com
www.keithley.com

NETHERLANDS
Gorinchem
Ph: 0183-635333
Fax: 0183-630821
info@keithley.nl
www.keithley.nl

SINGAPORE
Singapore
Ph: 65-6747-9077
Fax: 65-6747-2991
koh_william@keithley.com
www.keithley.com.sg

SWEDEN
Stenungsund
Ph: 08-50904600
Fax: 08-6552610
sweden@keithley.com
www.keithley.com

SWITZERLAND
Zürich
Ph: 044-8219444
Fax: 044-8203081
info@keithley.ch
www.keithley.ch

TAIWAN
Hsinchu
Ph: 886-3-572-9077
Fax: 886-3-572-9031
info_tw@keithley.com
www.keithley.com.tw

UNITED KINGDOM
Theale
Ph: 0118-9297500
Fax: 0118-9297519
info@keithley.co.uk
www.keithley.co.uk

Printed in the U.S.A. No. 2998 4.17.09