Boost Test Automation with On-Instrument Scripting

What is TSP?

APPLICATION NOTE

KEITHLEY
A Tektronix Company

Tektronix®

Introduction

Like many instrument operators, you may have a comfortable preference toward SCPI as the foundation for your instrument automation needs. If, however, you are looking for a means to reduce your test time and cost and are open to suggestions to help you accomplish your goals, then this technical brief might lead you to exactly what you are looking for.

Test Script Processor (TSP®) technology from Keithley has many proven benefits, and we are very excited to share the power of this technology, so let us start with the basics. Simply put, there are three key concepts we should cover before you start your TSP journey: TSP is a scripting engine, a command set, and a means for promoting system scalability.

TSP is a Scripting Engine

TSP is an embedded script execution environment built on top of Lua, a permissive, free license script-based programming language [1]. The TSP + Lua combination localizes all the benefits of a true programming language within your instrument including:

- Global and local variables
- Functions or subroutines
- Loops and conditionals
- Typical arithmetic, relational and logical operators
- Advanced math and string operation libraries
- String concatenation
- Commenting
- Aliasing
- File I/O and management
- Powerful, flexible data structuring in the form of tables

The TSP command language is implemented in tables, and this becomes more evident as you get acquainted with the Lua syntax and how other custom, built-in data structures use the dot-based dereferencing to access their deeper levels. The code snippet shown in **Figure 1** helps to illustrate a variety of the programming features you might use in a TSP script.

```
------------- Keithley TSP Function -------------
function res_test(smu,irange,ilevel,srcdelay,vcmpl)
    -- Source current on selected SMU channel and
    --   measure voltage.
    -- Calculate and return resistance value.

    -- default to smua if none specified
    if smu == nil then smu = smua end

    -- Temporary variables used by this function
    local l_vmeasured,l_isourced

    -- 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

    -- Enable output
    smu.source.output = smu.ON

    -- Wait before making measurement
    delay(srcdelay)

    -- Measure current and voltage
    l_isourced, l_vmeasured = smu.measure.iv()

    -- Disable output
    smu.source.output = smu.OFF

    return l_isourced, l_vmeasured
end
```

Figure 1: TSP programming example.

In short, the embedded programming language allows you to virtually create your own instrument command set. You can create a wrapper around a command tailored to your preference. If you want to encapsulate a series of commands and computations within a function identified by a short name then execute it and get just the results, you can do that, too. The example shown in **Figure 1** can be uploaded to the instrument, your program can invoke **`"res _ test()"`** and extract the sourced current and measured voltage values when they are available in the output queue. This puts you in control of how the returned data are formatted. While the sourced current then measured voltage are returned in that order, you could just as well swap the two.

To illustrate the potential for simplification, consider **Table 1** which establishes the same test setup to extract current and voltage measurements upon testing a resistor.

Implementation	Issued by the PC	Returned Data
SCPI Commands	:SOUR:FUNC CURR :SOUR:CURR:RANG 1e-3 :SOUR:CURR:LEV 200e-6 :SOUR:CURR:VLIM 2.0 :SENS:VOLT:RANG 2.0 :SENS:FUNC “CURR” :OUTP ON *<user-specified delay of 100e-6 s>* :READ? :SENS:FUNC “VOLT” :READ? :OUP OFF	

TSP Commands	smu.source.func = smu.OUTPUT_DCAMPS
	smu.source.rangei = 1e-3
	smu.source.leveli = 200e-6
	smu.source.limitv = 2.0
	smu.measure.rangev = 2.0
	smu.source.output = smu.ON
	<user-specified delay of 100e-6 s>
	print(smu.measure.iv())
	smu.source.output = smu.OFF	

TSP Function Call	print(res_test(smua, 1e-3, 200e-6, 100e-6, 2.0))	

Table 1: Three different ways of doing the same thing.

In looking at the “TSP Function Call” sample in the table above, you could just as easily have embedded the print() statement within the **`res _ test()`** function itself and removed the need to add it as a wrapper, allowing it to operate more like a SCPI query: send the command, get the data.

The preceding details are used to help illustrate what a script might be composed of. Even more important is where all this next-level programming capability is called into action: at the instrument level. An embedded scripting engine allows you to add features or customize the behavior of the TSP-enabled instrument and run even without a PC connected.

The value of executing the script on the instrument is that you are able to lower the processing burden on the PC. TSP technology localizes complete test programs within the instrument for best-in-class system-level throughput. This may be inclusive of any advanced math operations that need to be performed on the acquired data. Offloading processing to the instrument, reducing communication bus traffic, TSP-enabled instruments have run production tests as much as 60% faster [2].

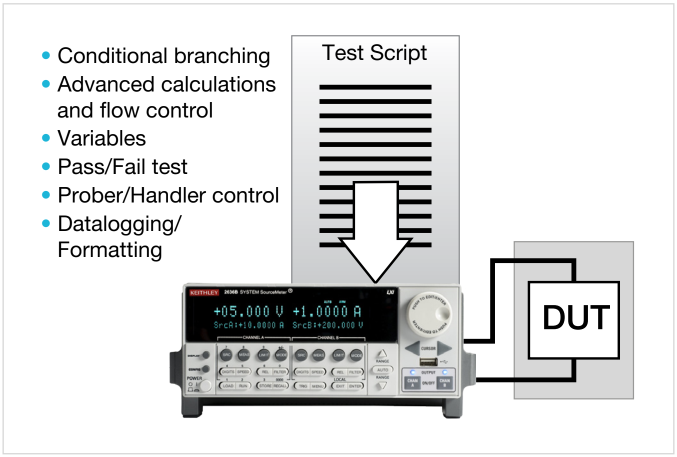

Figure 2: Locating and executing functional scripts at the instrument level helps to share the processing responsibility with the user PC.

An example of the TSP throughput improvement potential can be found in the application note entitled Increasing Production Throughput with Data Acquisition Systems [3]. With just a single instrument we show a >16% reduction in test time using the embedded scripting method over traditional transactional command issuing.

Figure 3: Learn how the Keithley team is able to improve test time through embedded scripting.

TSP is a Command Set

Another important thing you should understand is that TSP is a command set – a textual framework composed of specific program messages, expected instrument responses, and data formats. While we recommend you take full advantage of the available embedded scripting capability, you still have the option of sending individual TSP commands to a TSP-enabled instrument the same way you would when sending SCPI commands using any other instrument.

While SCPI allows for short command string formatting, one of the strategies of TSP is to enhance readability by enforcing the use of the full commands such as shown in the following string:

```
smu.source.levelv = 10.0
```

One other difference you might immediately recognize when using TSP is how it returns instrument data to the user. Consider the entries in the following table:

SCPI	TSP
`:READ?`	`print(smu.measure.v())`

While visually different, both commands perform the same functionality: they initiate a reading acquisition, place it in the output queue, and send the textual measurement back to the calling program.

For interested readers, more information and examples on these two command sets can be found in the How to Transition Code to TSP from SCPI application note on tek.com.

TSP Promotes System Scalability

If you are asking how such efficiencies can be realized with just a single instrument, we ask you to think bigger. Rarely is a test system composed of a single instrument which is exactly what the teams at Keithley had in mind. In addition to TSP as a scripting engine and command set, a multitude of instrumentation "nodes" can be interconnected over the TSP-Link system expansion interface.

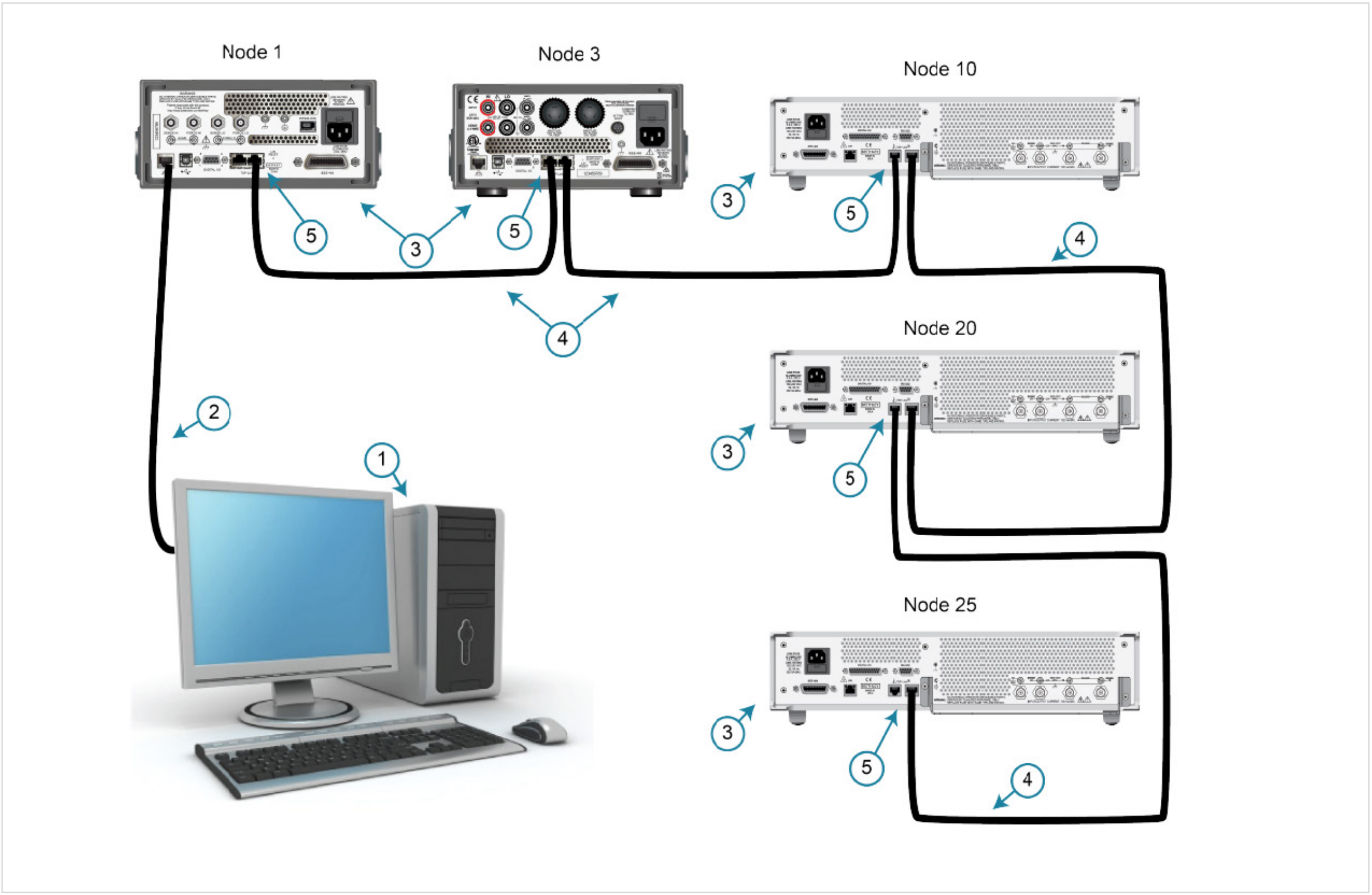

Figure 3: Example TSP configuration composed of: (1) the optional PC, (2) a communication connection, (3) instrument nodes, (4) LAN crossover cables, and (5) TSP-Link interface connections.

Keithley Instruments TSP-Link® is a high-speed trigger synchronization and communication bus that test system builders can use to connect multiple instruments in a master and subordinate configuration. Once connected, all the instruments that are equipped with TSP-Link in a system can be programmed and operated under the control of the master instrument or instruments. This allows the instruments to run tests more quickly because they can be decoupled from frequent computer interaction. The test system can have multiple master and subordinate groups, which can be used to handle multi-device testing in parallel. Combining TSP-Link with a flexible programmable trigger model ensures speed.

Conclusion

Instruments that are enabled for Test Script Processor (TSP®) operate like conventional instruments by responding to a sequence of commands sent by the controller. You can send individual commands to the TSP-enabled instrument the same way you would when using any other instrument.

Unlike conventional instruments, TSP-enabled instruments can execute automated test sequences independently, with or without an external controller. You can load a series of TSP commands into the instrument using a remote computer or the front-panel port with a USB flash drive. You can store these commands as a script that can be run later by sending a single command message to the instrument. In either case, the processing power of your PC can focus on other tasks while the instrument's embedded Test Script Processing engine conducts elaborate test executions at the instrument level, across multiple instruments, and over the TSP-Link® expansion bus.

You do not have to choose between using conventional control or script control. You can combine these forms of instrument control in the way that works best for your test application.

Visit www.tek.com for more information on TSP and TSP-enabled test equipment.

References

[1] lua.org, "About", 2022. [Online]. Available: https://www.lua.org/about.html

[2] Tektronix, "Source Measure Unit (SMU) Instruments Selector Guide," August 2021. [Online] Available: https://www.tek.com/en/documents/product-selector-guide/source-measure-unit-smu-instruments-selector-guide?msclkid=5dde5c8fac7b11ecbfa2c1fc1a49f670

[3] Tektronix, "Increasing Production Throughput with Data Acquisition Systems", April 2018. [Online] Available: https://www.tek.com/en/documents/application-note/increasing-production-throughput-data-acquisition-systems

Contact Information:

Australia	1 800 709 465
Austria*	00800 2255 4835
Balkans, Israel, South Africa and other ISE Countries	+41 52 675 3777
Belgium*	00800 2255 4835
Brazil	+55 (11) 3530-8901
Canada	1 800 833 9200
Central East Europe / Baltics	+41 52 675 3777
Central Europe / Greece	+41 52 675 3777
Denmark	+45 80 88 1401
Finland	+41 52 675 3777
France*	00800 2255 4835
Germany*	00800 2255 4835
Hong Kong	400 820 5835
India	000 800 650 1835
Indonesia	007 803 601 5249
Italy	00800 2255 4835
Japan	81 (3) 6714 3086
Luxembourg	+41 52 675 3777
Malaysia	1 800 22 55835
Mexico, Central/South America and Caribbean	52 (55) 88 69 35 25
Middle East, Asia, and North Africa	+41 52 675 3777
The Netherlands*	00800 2255 4835
New Zealand	0800 800 238
Norway	800 16098
People's Republic of China	400 820 5835
Philippines	1 800 1601 0077
Poland	+41 52 675 3777
Portugal	80 08 12370
Republic of Korea	+82 2 565 1455
Russia / CIS	+7 (495) 6647564
Singapore	800 6011 473
South Africa	+41 52 675 3777
Spain*	00800 2255 4835
Sweden*	00800 2255 4835
Switzerland*	00800 2255 4835
Taiwan	886 (2) 2656 6688
Thailand	1 800 011 931
United Kingdom / Ireland*	00800 2255 4835
USA	1 800 833 9200
Vietnam	12060128

*** European toll-free number. If not accessible, call:** +41 52 675 3777

Rev. 02.2022

KEITHLEY
A Tektronix Company

Find more valuable resources at TEK.COM

071922 SBG 1KW-73926-0