TEKTRONIX
INNOVATION FORUM
Engineering the Future

Being Productive from Python with a Keithley Test Script Processor (TSP) Enabled Product

What Products Have TSP?

SMUs

2600B Series

(8 System SourceMeters)

2650A Series

(2 High-Power SMUs)

2450+ Series

(4 Touch-screen SMUs)

DMMs

DMM6500

DMM7510

and DMM7512

Switches

707B, 708B

3706A-S

3706A

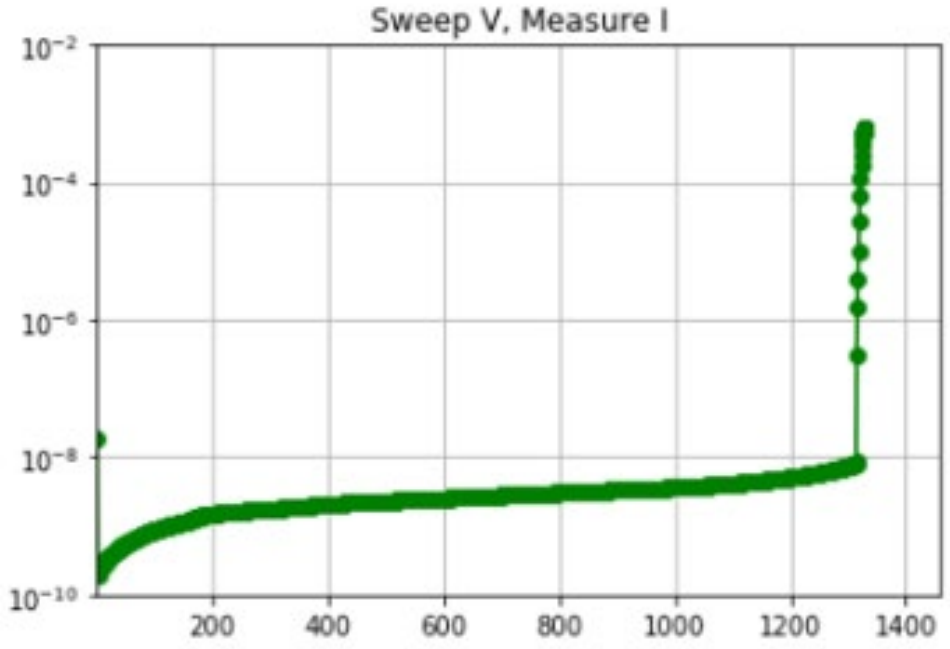
DAQs

DAQ6510

Practical Example: Breakdown Tests

- IGBT, FET, Diodes

- Shape of Curve is sometimes important
- Sometimes just want the Max V at $I_{breakdown}$
- Device Wobble/Repeatability: AC Waveform
- Fast : measure the V at $I_{breakdown}$



- Speed of Test:
  - Dynamic Range of Current
  - Exit Condition/Current Limiting
  - Step Sizes/Number of Steps in the Sweep

What is TSP and TSP vs. SCPI

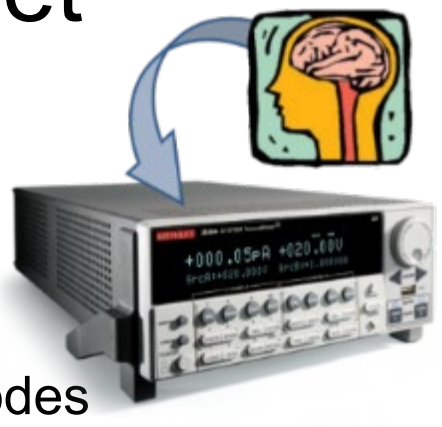
TSP = TEST SCRIPT PROCESSING

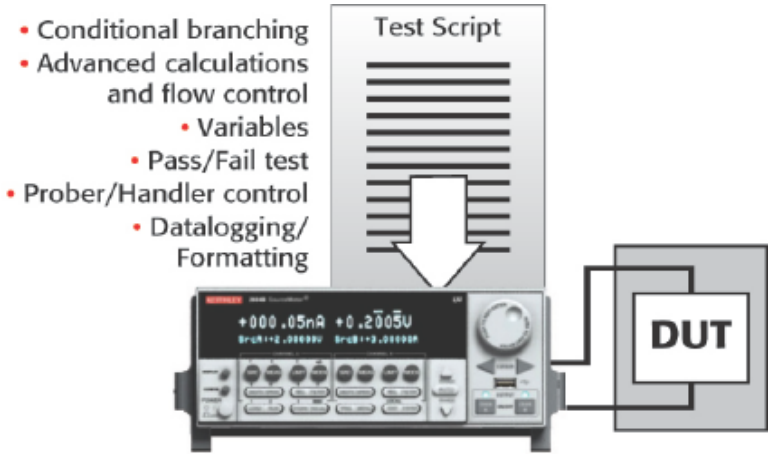
- SCPI = Standard Commands for Programmable Instruments
- In both cases, they are just strings of ASCII characters
- TSP command set can be treated the same way
  - But you are not taking full advantage of your instrument

SCPI Commands	Comments	TSP Script Commands
*RST	Restore GPIB defaults.	reset()
:SOUR:FUNC VOLT	Select voltage source.	smua.source.func = smua.OUTPUT_DCVOLTS
:SOUR:VOLT:LEV 10	Source output = 10V.	smua.source.levelv = 10
:SENS:CURR:PROT 10E-3	10mA compliance.	smua.source.limiti = 0.01
:SENS:FUNC "CURR"	Current measure function.	
:SENS:CURR:RANG 10E-3	10mA measure range.	smua.measure.rangei = 0.01
:OUTP ON	Output on before measuring.	smua.source.output = smua.OUTPUT_ON
:READ?	Trigger, acquire reading.	READING = smua.measure.i()
:OUTP OFF	Output Off	smua.source.output = smua.OUTPUT_OFF

TSP is Much More than ASCII Command Set

- TSP is the brain of the Instrument – Lua based scripting environment
- This brain allows the Instrument to make decisions without PC intervention
  - Loop Until
- Function Encapsulation = parameters, code reuse, implement emulation modes
- Reduced BUS traffic = faster throughput, reduced test time
- Analysis “on the box” = data becomes information
  - Send back just hFE or BVdss

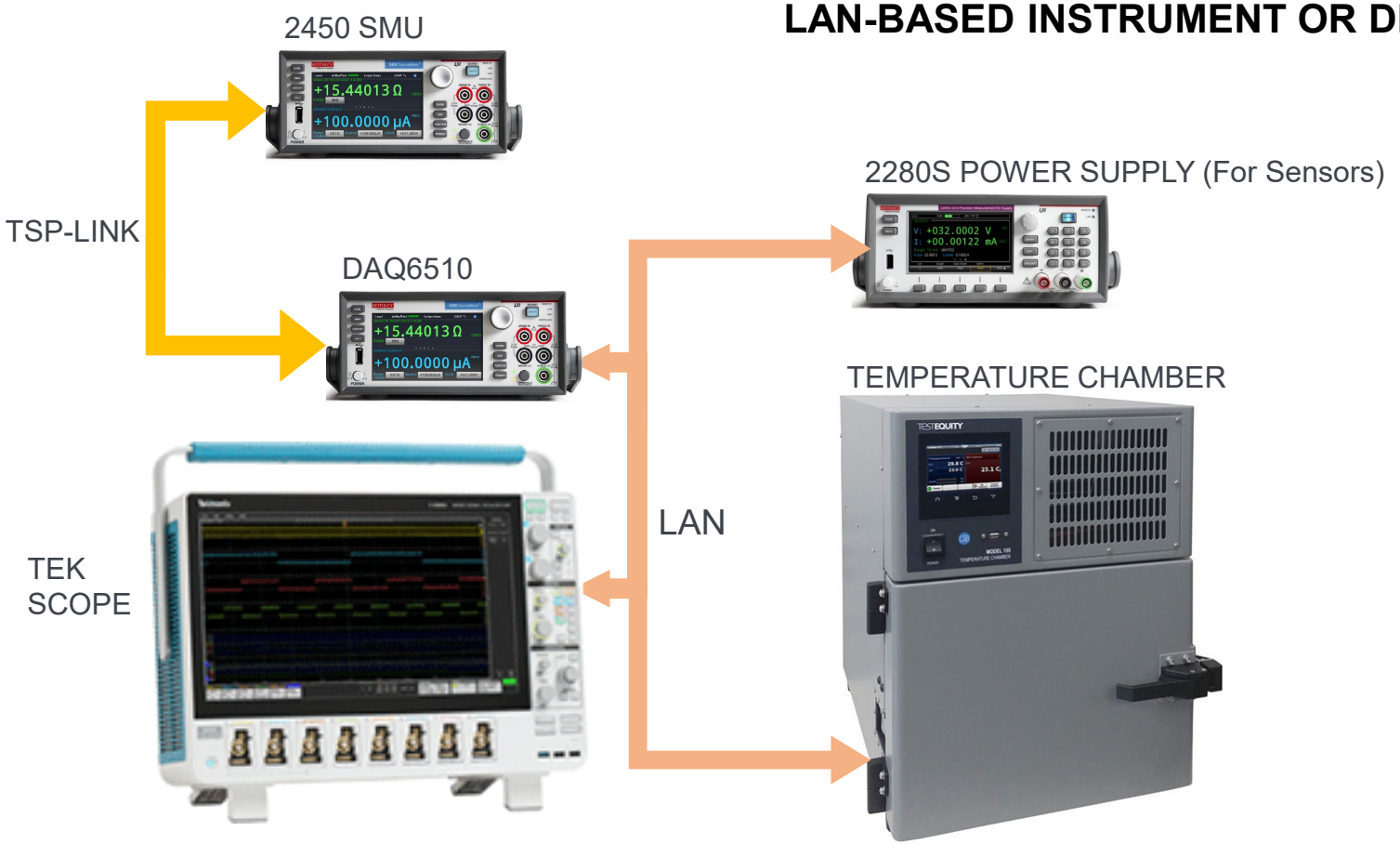




Think of Powerful Integration

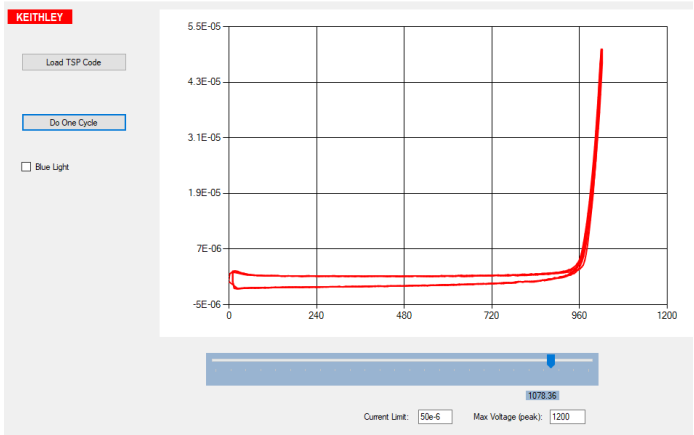
TSP-LINK TO OTHER TSP ENABLED PRODUCTS: SWITCHING, DMMS, ETC.

A TSP INSTRUMENT CAN COMMUNICATE WITH A LAN-BASED INSTRUMENT OR DEVICE VIA TSP SCRIPT (TSP-NET)

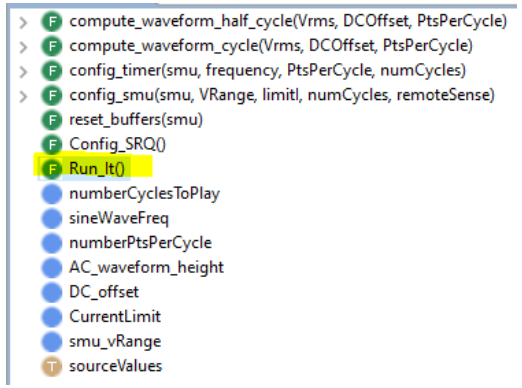


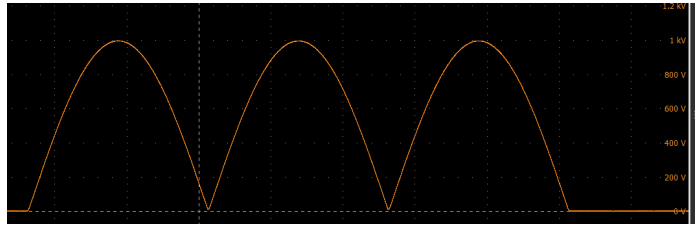
First Favorite Feature of TSP:

- Function Encapsulation
  - Vastly reduces bus traffic between PC and Test Equipment
    - Less traffic = Test time reduction
    - Is not the same as functions on the PC side
      - TSP functions are loaded once
  - Allows me to segregate the development tasks:
    - GUI or Operator Interface part
    - The instrument or TSP functions called by the GUI




```
myInstr.WriteString("Run_It()")
```





TSP Function: Simple Example

- Use Internal Beeper of the Instrument: beeper.beep()
  - Define a function that takes two parameters

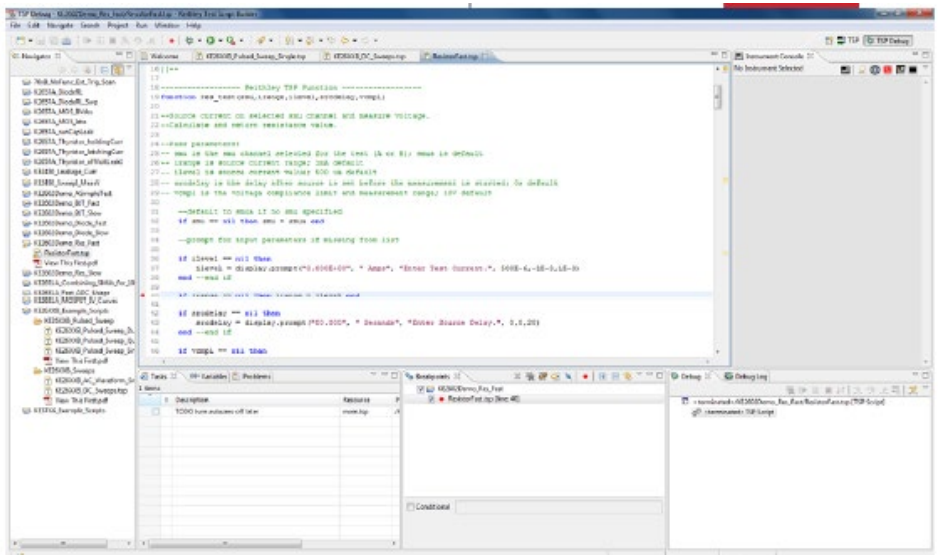
```
#Load a function into TSP Runtime Memory
my_instr.write("loadscript myScriptName")
my_instr.write("function myBeepFunction(duration, freq)")
my_instr.write("beeper.beep(duration, freq)")
my_instr.write("end")  #function definition
my_instr.write("endscript")
#run the script to load into runtime memory
my_instr.write("myScriptName.run()")
```

  - When needing the feature, call the function

```
#call our function
my_instr.write("myBeepFunction(1, 1200)")
time.sleep(0.1);
my_instr.write("myBeepFunction(1, 800)")
```

Second Favorite Feature: Test Script Builder

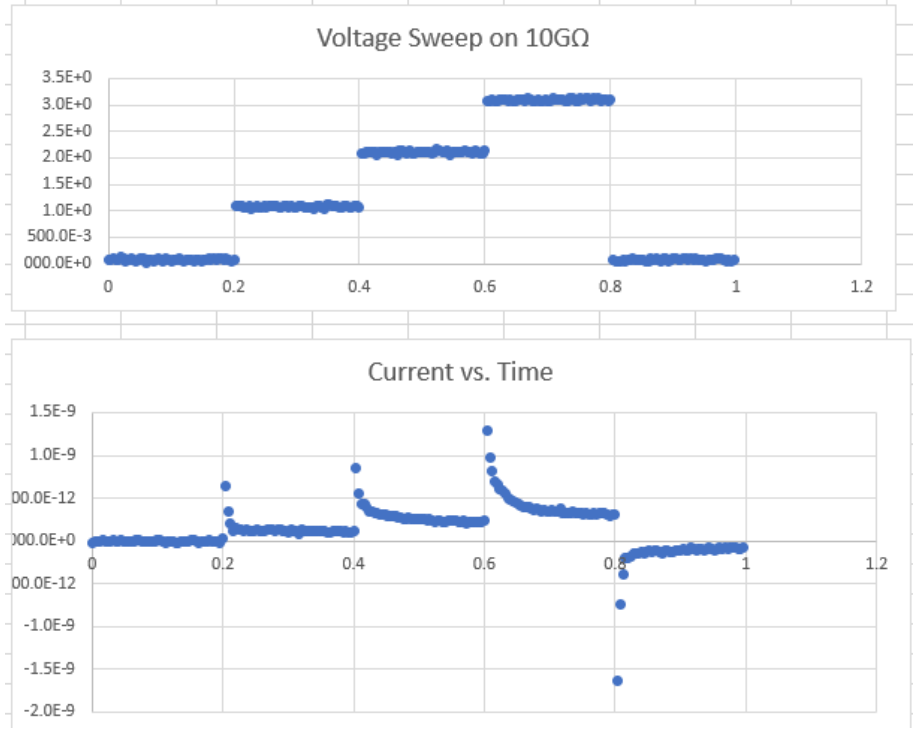
- Keithley's Test Script Development Tool
  - Windows OS
- Create, Modify, Debug, Organize scripts
  - Debug: breakpoints, stepping, watches, etc.
- Connects to Instrument
  - Enables to run the scripts on the hardware



Let’s Look at an Example

- Breakdown tests often involve low currents and staircase sweeps
- Stepping the voltage (dV/dt) will produce a displacement current
- I = C * dV/dt

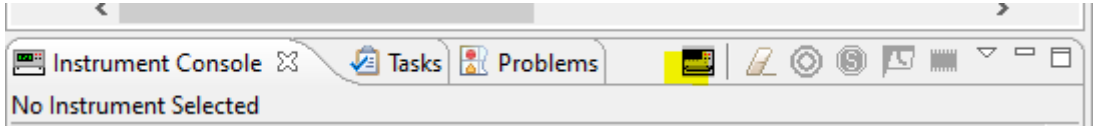
- Used ASYNC trigger model
  - Dual A/D: measure V and I vs. time
  - Sample at 250Hz
  - Hold each Source Level for 200msec



Migrate From Test Script Builder to Python

- Important: Close the connection in Test Script Builder

- If you are seeing “TSP>” responses in Python, close TSB



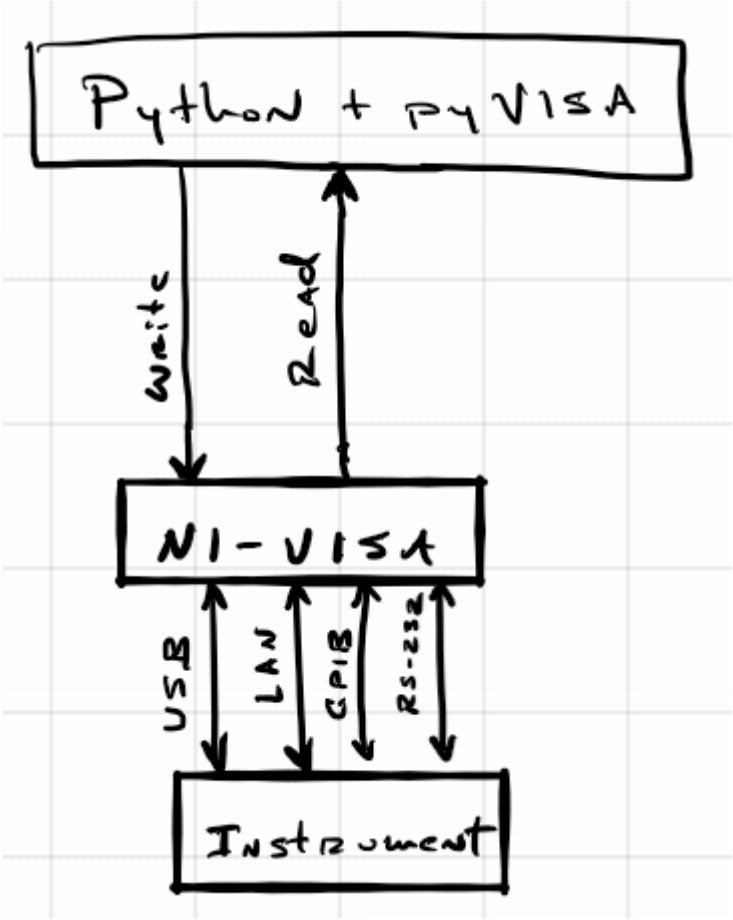
Migrate to Python: Use pyVISA

```
import visa
import time

instrument_resource_string = "TCPIP0::192.168.1.39::inst0::INSTR"
resource_mgr = visa.ResourceManager()

my_instr = resource_mgr.open_resource(instrument_resource_string)
```

```
my_instr.write("*IDN?\n")
print(my_instr.read())
```

```
Keithley Instruments Inc., Model 2602B, 4408370, 3.2.2
```

```
[16]: #put instrument back to local and close connection
      my_instr.clear()
      my_instr.close()
```



Migrate to Python: Load Functions From File

- Functional TSP from Test Script Builder
- Use loadscript and endscript
- Loads the functions to Runtime Memory of instrument
  - Read a line from the TSP file
  - VISA write that line to the instrument

```
file_path = "C:\\Users\\aclary\\Keithley Test Script Builder\\Workspaces\\workspace\\_0_Cool_new_Project\\"
file_name = "Abort_Sweep_on_Compliance_functions.tsp"
file_path_and_name = file_path + file_name

print(file_path_and_name)

my_instr.write("loadscript myWorkers")
with open(file_path_and_name) as fp:
    #read the TSP function definition file line by line
    for line in fp:
        #print(line)
        #write the TSP command to the instrument, line by line
        my_instr.write(line)

my_instr.write("endscript")
#run the script to place it into runtime memory
my_instr.write("myWorkers.run()")

#close the script file
fp.close()

print('done loading functions')
```

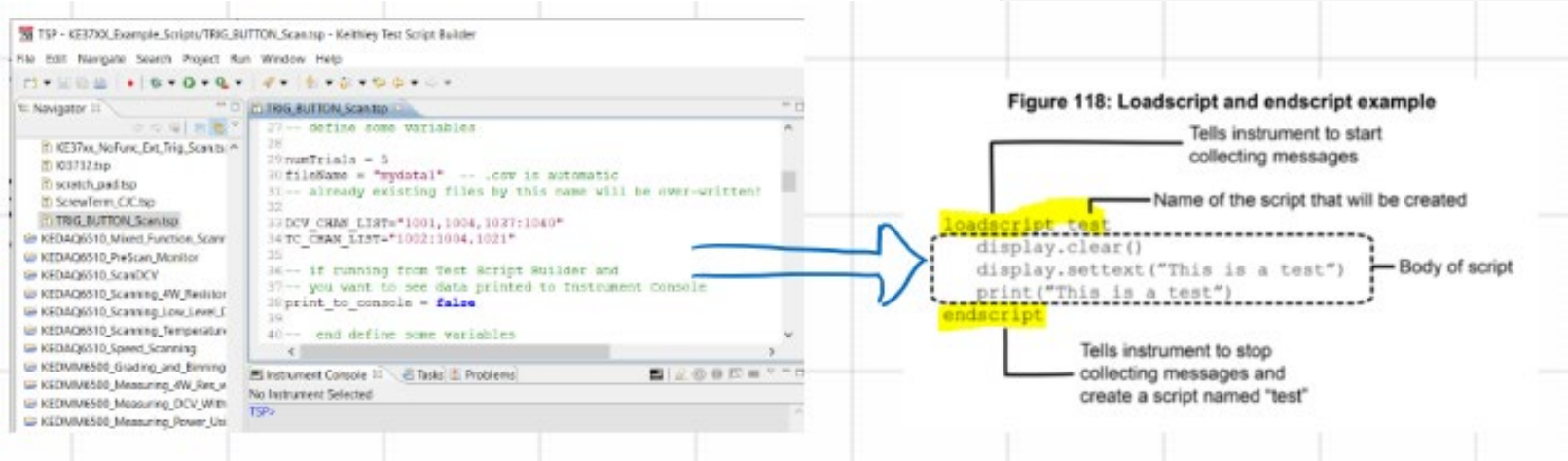


Figure 118: Loadscript and endscript example

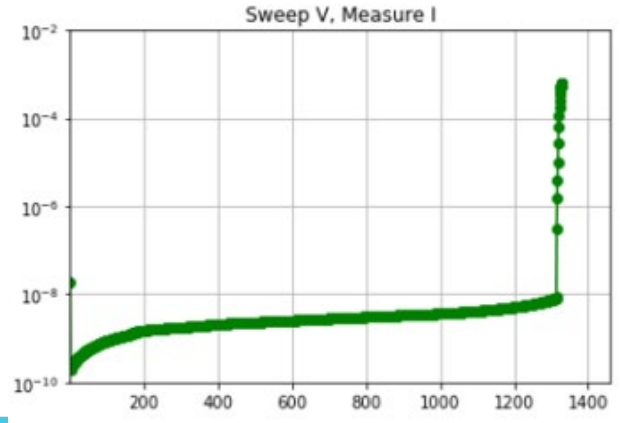
TSP Command: waitcomplete()

- Consider this sequence of events:
  - On the system controller, from Python (or C# or whatever)
    - Get a VISA session with SMU
    - Set VISA session timeout to 10 seconds (default typically)
    - Command SMU to perform Sweep that will require more than 10 seconds to complete
    - Part of the SMU command was use of waitcomplete()
      - smua.trigger.init() ← starts the sweep
      - waitcomplete() ← tells TSP engine to wait
    - Immediately send printbuffer() command and use VISA Read to obtain buffer data.
- What error results? Why? What to do about it?

Breakdown Test Using 2657A SMU

- From a recent customer interaction
- He was making use of KISweep Factory Script (installed on Instrument)
- Enhancement Request: Have the sweep exit early if breakdown occurs
    - Hitting the current limit is typical signal that this occurs

- Depending on various factors, Breakdown tests can be slow
    - A lot of points (small dV/dt)
    - Allow setting time for low currents (RC time constant)

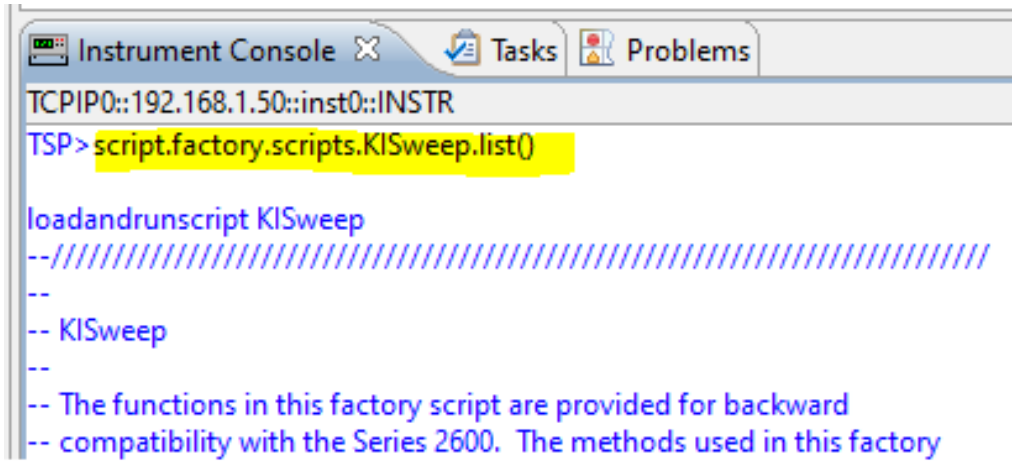


Factory Scripts

The Keithley Instruments Model 2657A High Power System SourceMeter® instrument is shipped with one or more factory scripts saved in its flash firmware memory. A factory script is made up of a number of functions. Some of them can be called from the front-panel LOAD TEST menu. All of them can be called using remote programming.

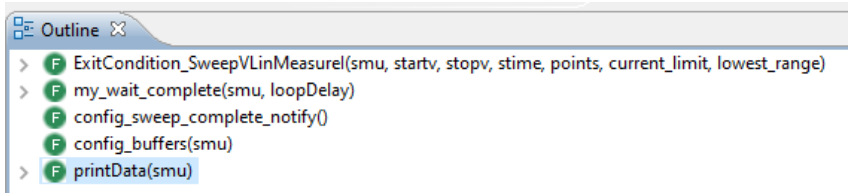
SweepILinMeasureV() (on page 7-323)
SweepIListMeasureV() (on page 7-324)
SweepILogMeasureV() (on page 7-325)
SweepVLinMeasureI() (on page 7-326)
SweepVListMeasureI() (on page 7-327)
SweepVLogMeasureI() (on page 7-328)

- The source can be extracted with the list() command



High Level Steps – Use Test Script Builder

- Extract the KISweep Source code to a TSP file
- Modify the logic



- Validate the changes from pseudo code in Test Script Builder

```
errorqueue.clear()

-- ***************  call the functions
--ExitCondition_SweepVLinMeasureI(smu, startv, stopv, stime, points, current_limit, lowest_range)
ExitCondition_SweepVLinMeasureI(smua, 1, 800, 0, 800, 10e-6, 1e-9)
```

- Migrate the solution to Python

```
#ExitCondition_SweepVLinMeasureI(smua, start, stop, meas_delay, noSteps, current_limit, lowest_range)
cmd_list = ["errorqueue.clear()",
            "ExitCondition_SweepVLinMeasureI(smua, 1, 800, 0, 800, 10e-6, 1e-9)"]

for cmd in cmd_list:
    my_instr.write(cmd)
```

What About the Logic Did We Change?

- We want the test to complete early if the current limit is encountered
    - Breakdown occurred = current limit

- What if breakdown does not occur?
    - The SMU can tell you when the sweep is finished

- Need two exit conditions:
    - When the sweep is finished normally
    - Compliance Limit Encountered – exit early

Exit Condition Logic – Implemented in TSP

- Factory Script:
  - Uses waitcomplete() which blocks
  - PC could not interrogate about compliance

```
-- Run the sweep and then turn the output off.
smu.source.output = smu.OUTPUT_ON
smu.trigger.initiate()
waitcomplete()
smu.source.output = smu.OUTPUT_OFF
```

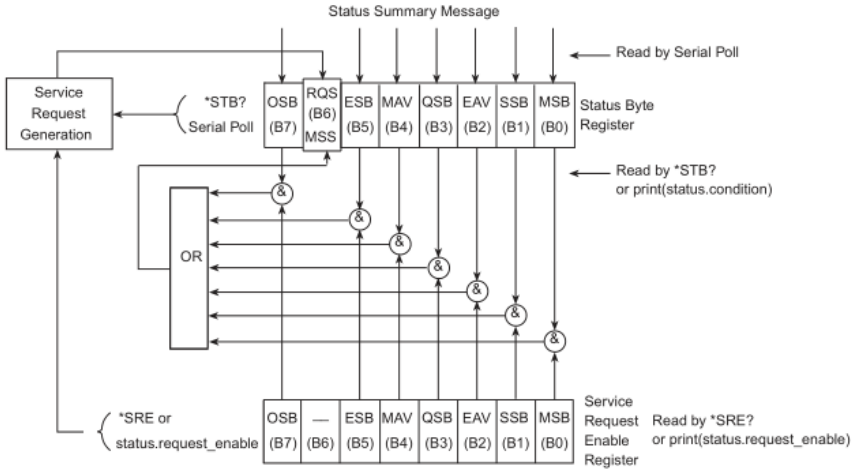
- Revised Script:
  - Custom wait complete also blocks
  - But it incorporates the required logic

```
-- get our status subsystem ready
config_sweep_complete_notify()

smu.source.output = smu.OUTPUT_ON
smu.trigger.initiate()

--waitcomplete()   this blocks

-- replace with a custom version that looks for current limit or sweep done
--my_wait_complete(smu, loopDelay)
my_wait_complete(smu, 0.05)

smu.source.output = smu.OUTPUT_OFF
```

```
function my_wait_complete(smu, loopDelay)

    repeat
        delay(loopDelay)

        if (bit.test(status.measurement.current_limit.condition, 2)) == true then
            smu.abort()
            -- add cmds to take smu to known state
            -- set source level, etc.
            smu.source.levelv = 0
            smu.measure.i()  -- new measurement to clear current_limit bit
        end  -- if
        if debug == 1 then print("Abort due to compliance detect") end

    until bit.test(status.condition, 8) == true  -- until sweeping bit falling edge

end  -- function
```

Python Side: Detecting SRQ

- SRQ: Service request raised by the instrument
- NI-VISA has functions for status byte polling



Bit position	B7	B6	B5	B4	B3	B2	B1	B0
Binary value	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
Decimal Weights	128 (2^7)	64 (2^6)	32 (2^5)	16 (2^4)	8 (2^3)	4 (2^2)	2 (2^1)	1 (2^0)

```
#detect the Sweep is finished
#repeat until the SRQ bit is set
still_running = True
status_byte = 0
debug = 1


while still_running:
    status_byte = int(my_instr.read_stb())
    if debug: print(status_byte)
    if (status_byte and 64) == 64:
        still_running = False
    time.sleep(0.25)  #250msec pause before asking again
```

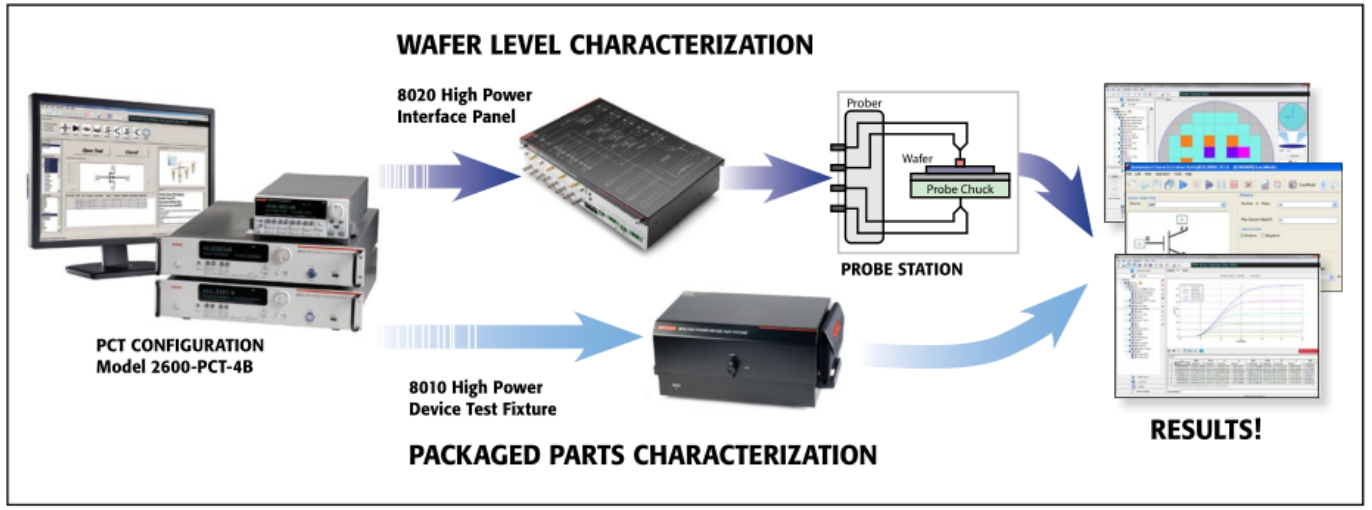
Getting the Data Back

- The .n property on the buffer = number of readings in buffer
- Alternately, make a data analysis function
    - If BVdss is the parameter of interest, just pass back the answer rather than all the IV data pairs
    - Data vs. Information

```
#ask for voltage and current; buffer2= measured voltage, buffer1 = measured current
my_instr.write("printbuffer(1, smua.nvbuffer1.n, smua.nvbuffer2.readings, smua.nvbuffer1.readings)")
#one long comma delimited string
raw_data = my_instr.read()

# an array of strings: voltage, current, voltage, current
raw_data_array = raw_data.split(",")

volts = []
current = []
abs_current = []

# step through the array of strings, step size 2
# place the V and I into their own array of floats
for i in range(0, len(raw_data_array),2):
    volts.append(float(raw_data_array[i]))
    current.append(float(raw_data_array[i+1]))
    #absolute value of currents for log scale
    abs_current.append(abs(float(raw_data_array[i+1])))
```

Equipment Used Today

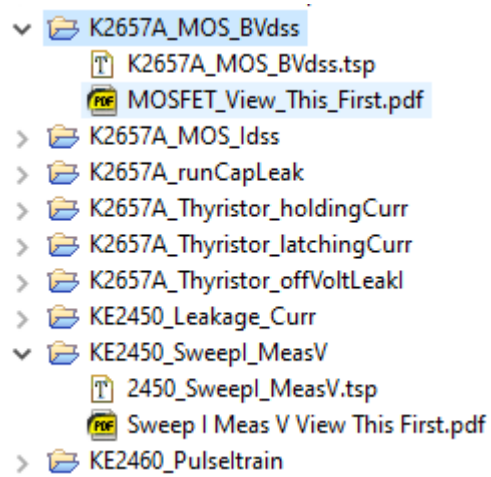
- 2657A – 3KV SourceMeter
- High Voltage Triax Cables
- 8010 Test Fixture



- We also have 8020 Interface Panel for Wafer Level
- We also have software solutions

Resources

- Test Script Builder: Distributed with a lot of examples
- Support Forum: https://forum.tek.com/
  - TSP for Breakdown Test: https://forum.tek.com/viewtopic.php?f=14&t=142633
  - TSP code for ASYNC Sampling: https://forum.tek.com/viewtopic.php?f=14&t=142669
- GitHub: https://github.com/tektronix/keithley
- Python Install and Getting Started: https://www.youtube.com/watch?v=W5Brxiwnp5g
- Contact Technical Support: https://www.tek.com/support



TEKTRONIX INNOVATION FORUM

Engineering the Future

Tektronix®

KEITHLEY

A Tektronix Company