KEITHLEY

Application Note Series

Number 2897

Comparing a Series 7000 SCPI Application to a Series 3700 Script Application

Introduction

For many years, instrument manufacturers have used "Standard Commands for Programmable Instrumentation" or SCPI to control programmable test and measurement devices in instrumentation systems. SCPI provides a uniform and consistent language for the control of test and measurement instruments. The same commands and responses control corresponding instrument functions in SCPI equipment, regardless of the manufacturer or the instrument type.

By design, Keithley's new Series 3700 System Switch/Multimeter instruments do not use SCPI commands. Instead, instruments on this platform use an internal Test Script Processor (TSP) to process and run programs called "scripts." The motivation to migrate toward the use of TSP was to address today's more demanding system throughput requirements. Communication with any TSP-based instrument is much the same as interfacing to a conventional instrument, using commercially available software applications and an application development environment. The user simply sends text strings containing either TSL (Test Script Language) or ICL (Instrument Control Library) commands to the instrument via the communication interface. Although Keithley provides an application suite called Test Script Builder (TSB) for use in developing scripts for TSP-based instruments, it is not intended to be the only tool used for communicating to the instrument, nor does not replace commercially available software applications.

A script is a collection of instrument control commands and/or program statements. Program statements control script execution and provide facilities such as variables, functions, branching, and loop control. TSB provides a programming interface in which users can create powerful, high speed, multi-channel tests to download into either the unit's volatile or nonvolatile memory. Using the downloaded script, the unit controls itself, independent of the system's host controller. This capability can free up the system controller to interface with the other equipment in the test rack more frequently, thereby increasing the overall system throughput. Series 3700 instruments have very deep memory; program memory can hold 50,000 lines of code and data memory can store at least 650,000 readings.

This application note compares SCPI commands with the new Series 3700 scripting approach and illustrates how to convert a Series 7000 SCPI-based application to a Series 3700 test script application.

The SCPI Instrument Model

Some measurements require direct control over an instrument's hardware. To provide this control, SCPI-based instruments contain command subsystems that control particular instrument functions and settings. These commands trade interchangeably for fine control. The ability to configure instruments and make measurements with different degrees of control is a major advantage of SCPI.

The SCPI Instrument Model controls the way instrument functionality is divided among the SCPI command subsystems. In the case of Series 7000 instruments, the command subsystems are broken down into the following categories:

1. DISPlay: Controls the display of the 700X.
2. OUTput:Controls the polarities of the digital output port.
3. ROUTe: Controls the signal routine through the switch system.
4. SENSe: Used to read the digital input ports.
5. SOURce: Controls the logic level(True or False) of each digital output line.
6. STATus: Controls the status registers.
7. SYSTem: Contains miscellaneous commands for scanner setup.
8. TRIGger: Contains a series of commands to configure the three layers of the Trigger Model.

TSP and Scripting: A More Efficient Programming Method

The TSP language of the Series 3700 offers an equivalent set of instrument commands. The following command set applies to the Series 3700:

1. **Beeper:** Commands used to control the built-in beeper.
2. **Bit:** Used to perform logic operations on one or two numbers.
3. **Channel:** Commands used to configure and control the channels of the switching cards.
4. **Delay:** Used to control trigger operations.
5. **Digital I/O:** Configures parameters of the digital I/O features (write/read, trigger and limits).
6. **Display:** Used to control display messaging on the front panel of the Series 3700 instrument.
7. **Error:** Used to read the entries in error event queue.
8. **Exit:** Used to terminate a script that is presently running.

9. **Format:** Used for data printed with the `printnumber` and `printbuffer` commands.
10. **GPIB:** Used to set the GPIB address.
11. **LAN:** Used to set Local Area Network parameters.
12. **Local node:** Used to set the power line frequency control (on/off) prompting and control (hide/show) error messages on the display.
13. **Reset:** Used to set all the logical instruments in the system to the default settings.
14. **Scan:** Used to specify and configure channels and/or change channel patterns to scan. As well as associated buffers, triggers, or other scanning aspects.
15. **Setup:** Used to save or recall stored set-up configurations and to set the power-on state.
16. **Timer:** The timer can be used to measure the time it takes to perform various actions.
17. **Trigger:** Used to control triggering.
18. **TSPLink:** Used to assign node numbers to a mainframe and initialize the TSP-Link system.
19. **UserString:** Used to store/retrieve user-defined strings in non-volatile memory.
20. **Print Buffer:** Used to print numbers and data.
21. **Slot[1]:** Used to program attributes in different slots in mainframe.

There are two simple ways to program and communicate with the Model 3706 Switch System: either by executing individual TSP commands (similar to sending individual SCPI commands) or by writing test scripts. Scripts are a collection (list) of instrument control commands and/or program statements. All commands and statements in the script are executed by the Model 3706 Switch system. Running a script at the switch system is faster than running the test program from the PC. The use of scripts can eliminate the transmission time from the PC to the switch system.

Table 1 compares the SCPI approach for a Series 7000 instrument to the scripting approach for a Model 3706 for performing a simple scan of the first 10 channels on slot one.

Note that there are only four steps in this example:

1. Reset the unit.
2. Create the scan list.
3. Configure the scan count, which is defined as how many times the instrument goes through the scan list.
4. Start the scan.

Memory Pattern Scan

This example takes advantage of the memory pattern feature of Series 7000 and Series 3700 instruments. The memory pattern feature allows multiple channel closures to be saved in a memory location, then the memory location is placed in the scan list. The scan list can be run as a standard sequential scan but with the memory patterns instead of single channels. This is a powerful feature that can be used in applications that require multiple switch closures in a system in order to have certain configurations for the specific test.

In this example, two channels are closed and saved in each memory location. Note that the Model 7002 requires the SCPI commands to be sent and processed. The Model 3706-S can do this using either of two techniques. The first technique is similar to that of the Series 7000 instrument, where commands, either SCPI or ICL, are simply sent to the instrument to create the memory pattern, memory location, and scan list. Even when using this technique, the Model 3706-S uses fewer commands, making the programming easier.

The second technique is the script technique in which the Model 3706-S is running the user-created "test program/script" rather than waiting to receive each command via the computer bus, which can be slowed significantly by command traffic. Notice that the channels are never closed and saved as in the Model 7002 SCPI codes. The `channel.pattern.setimage` command does this automatically. See the example script in ***Table 2***.

Model 2400 Sweep and Model 3706 Scan

Here is an example using the Model 700X as a scanner and a Model 2400 SourceMeter® instrument performing a voltage sweep from 1V to 10V in 1V steps. Each voltage step in the sweep is on a different channel of the 700X. ***Table 3*** also compares the Model 2400 and Model 3706 commands.

Table 1. Simple Scan Comparison

Series 7000 SCPI	Description	Model 3706 ICL	Description
`*RST`	Reset	`reset()`	Reset
`:ROUTe:SCAN (@1!1:1!10)`	Set scan list	`scan.create("1001:1010")`	Set scan list
`:TRIGger:COUNt 10`	Set trigger count to 10	`scan.scancount = 1`	Set to go through all scan list channels once
`:INIT`	Start scan	`scan.execute()`	Start scan

Table 2.

Model 7002 SCPI	Description	Model 3706-S ICL	Description
*RST	Reset	Reset()	Reset
:ROUTe:OPEN ALL	Open all channels	channel.pattern.setimage("1001, 1021,1031","mem1")	Set image and channel pattern for mem1
:ROUTe:CLOSe (@1!1,1!21)	Close channels 1 and 21	channel.pattern.setimage("1002, 1022,1032","mem2")	Set image and channel pattern for mem2
:MEMory:SAVe M1	Save pattern in memory 1	channel.pattern.setimage("1003, 1023,1033","mem3")	Set image and channel pattern for mem3
:ROUTe:OPEN ALL	Open all channels	scan.create("mem1,mem2,mem3")	Set scan list of mem1, mem2 and mem3
:ROUTe:CLOSe (@1!2,1!22)	Close channels 2 and 22	scan.scancount=10	Do scan list 10 times
:MEMory:SAVe M2	Save pattern in memory 2	scan.execute()	Start scan
:ROUTe:OPEN ALL	Open all channels	end	End program
:ROUTe:CLOSe (@1!3,1!23)	Close channels 3 and 23		
:MEMory:SAVe M3	Save pattern in memory 3		
:ROUTe:OPEN ALL	Open all channels		
:ROUTe:SCAN (@M1,M2,M3)	Set scan list of M1, M2 and M3		
:TRIGger:COUNt 9	One trigger for each channel		
TRIGger:DELay 1	Delay of one second for each channel		
:INIT	Start scan		

Script method

```
Reset()
Scan.create()
For x=1,10,do
   Chan1=1000+x
   Chan2=1020+x
   Chan3=1030+x
   Channel.pattern.setimage(chan1..",..chan2..",..chan3,"mem"..x)
   Scan.add("mem"..x)
End
Scan.scancount=1
Scan.execute
```

Table 3.

Model 700X SCPI	Description	Model 3706 ICL	Description
*RST	Reset	reset()	Reset
:ROUTe:OPEN ALL	Open all channels	channel.open("allslots")	Open all channels
:TRIGger:SOURce TLINk	Source of trigger is trigger link	digio.trigger[1].mode = digio.TRIG _ FALLING	Source of trigger is digio
:TRIGger:COUNt 100	100 trigger count	scan.scancount = 10	100 readings (10 channels × 10 scans)
:TRIGger:DELay 0.1	Delay after trigger 100msec	No trigger delay command	
:TRIGger:TCONTrol:DIRection SOURce	Drop through trigger layer on the first pass	scan.bypass = 1	Drop through scan layer first time (default)
:ROUT:SCAN (@1!1:1!10)	Scan list of ten channels	scan.create("1001:1010")	Scan list of ten channels
		digio.trigger[1].clear()	Clears event on line 1
		scan.trigger.channel.stimulus = digio. trigger[1].EVENT _ ID	Set trigger input event for digio trigger on line 1
		scan.trigger[2].stimulus = scan. trigger.EVENT _ CHANNEL _ READY	Set trigger output to line 2
		digio.trigger[2].mode = digio.TRIG _ FALLING	Falling edge trigger on line 2
		digio.trigger[2].pulsewidth=0.001	Set pulse width to 1msec.
Model 2400 SCPI	**Description**	**Model 2400 SCPI**	**Description**
*RST	Reset	*RST	Reset
:SOURCe:FUNCtion:MODE VOLT	Source voltage	:SOURCe:FUNCtion:MODE VOLT	Source voltage
:SOURce:SWEep:SPACe LINear	Linear sweep	:SOURce:SWEep:SPACe LINear	Linear sweep
:SOURce:VOLT:STARt 1	Start volt at 1V	:SOURce:VOLT:STARt 1	Start volt at 1V
:SOURce:VOLT:STOP 10	Stop at 10V	:SOURce:VOLT:STOP 10	Stop at 10V
:SOURce:VOLT:STEP 1	Step by 1V	:SOURce:VOLT:STEP 1	Step by 1V
:TRIGger:COUNt 100	100 triggers	:TRIGger:COUNt 100	100 triggers
:SOURce:VOLT:MODE SWEep	Sweep mode	:SOURce:VOLT:MODE SWEep	Sweep mode
:TRIGger:DIRection ACCeptor	Wait for first trigger	:TRIGger:DIRection ACCeptor	Wait for first trigger

Table 3 (continued)

Model 2400 SCPI	Description	Model 2400 SCPI	Description
:TRIGger:OUTPut SENSe	Output trigger after measurement	:TRIGger:OUTPut SENSe	Output trigger after measurement
:TRIGger:SOURce TLINk	Source of trigger TLINK	:TRIGger:SOURce TLINk	Source of trigger TLINK
:TRIGger:ILINe 2	Input trigger on line 2	:TRIGger:ILINe 2	Input trigger on line 2
:TRIGger:OLINe 1	Ouput trigger on line 1	:TRIGger:OLINe 1	Ouput trigger on line 1
:SENSe:FUNCtion 'CURRent'	Measure current	:SENSe:FUNCtion 'CURRent'	Measure current
:FORMat:ELEMents CURRent	Send only current data	:FORMat:ELEMents CURRent	Send only current data
:TRACe:CLEar	Clear buffer	:TRACe:CLEar	Clear buffer
:TRACe:POINts 100	Store 100 readings	:TRACe:POINts 100	Store 100 readings
:TRACe:FEED:CONTrol NEXT	Fill buffer and stop	:TRACe:FEED:CONTrol NEXT	Fill buffer and stop
:STATus:MEASure:ENABle 512	SRQ on buffer full	:STATus:MEASure:ENABle 512	SRQ on buffer full
*SRE 1	Enable SRQ	*SRE 1	Enable SRQ
:OUTPut ON	Enable output	:OUTPut ON	Enable output
:INIT	Initiate sequence	:INIT	Initiate sequence
Model 7001	**Description**	**Model 3706**	**Description**
:INIT	Start	scan.background()	Start scan
Wait for buffer full SRQ			
Model 2400	**Description**	**Model 2400**	**Description**
:TRACe:DATA?	Set up to read buffer data	:TRACe:DATA?	Set up to read buffer data
Read the data with GPIB read command		Read the data with GPIB read command	
:STATus:MEASure?	Query status	:STATus:MEASure?	Query status
Read status	Read to reset SRQ enable	Read status	Read to reset SRQ enable

It is important to note the trigger model of both units. The Model 2400 and Model 3706 must trigger each other. The scenario goes as follows:

1. Set Model 3706 to the number of channels to scan, scan list, external trigger, input and output trigger lines.
2. Set Model 2400 to sweep voltage, read current, configure trigger input and output, number of triggers/readings and set up the buffer to store the readings.
3. Send commands to enable the output on 2400 and arm it for the incoming trigger from the Model 3706.
4. Send the command "`scan.background()`" to the Model 3706 to begin the scan. At this point the 3706 closes the first channel, then sends a trigger the Model 2400 to go to the first step in the sweep.
5. Model 2400 does a Source Delay Measure cycle and outputs a trigger to the Model 3706 to go to the next channel in the scan list.
6. This continues until all the channels and readings are taken.
7. When the scan/read is complete the command "`trace:data?`" is sent to the Model 2400 and a GPIB read is sent, all of the buffer data is now transferred to the computer.

Conclusion

This application note touches on a few of the applications that can potentially be converted from a Series 7000 scanner SCPI program to a Series 3700 scanner TSP program.

Although the commands are somewhat similar, there are differences in two phases: the command syntax and command of features. The command syntax differences are shown in ***Table 4***. The command features are basically the features of the instrument. If one instrument has a feature the other instrument does not, the table indicates this feature is not applicable (N/A).

Once the command sequences have been mastered, the power of the Series 3700 TSP language is available to create reusable test sequence scripts that can solve applications ranging from the simple to the sophisticated.

Table 4. Model 3706 scanner commands vs. Model 700X scanner commands

TSP Command	Series 7000 Command(s)
Beeper Commands	
beeper.enable = 0/1	N/A
beeper.beep(duration,frequency)	N/A
Bit Commands	
bit.bitand(value1,value2)	N/A
bit.bitor(value1,value1)	N/A
bit.bitxor(value1,value2)	N/A
bit.clear(value1,index)	N/A
bit.get(value1,index)	N/A
bit.getfield(value1,index,width)	N/A
bit.set(value1,index)	N/A
bit.setfield(value1,index,width,fieldvalue)	N/A
bit.test(value1.index)	N/A
bit.toggle(value1,index)	N/A
Channel Commands	
channel.close(ch list)	ROUTe:CLOSe
channel.connectrule = rule	N/A
channel.connectsequential = channel.ON	N/A
channel.exclusiveclose(ch list)	N/A
channel.exclusiveslotclose(ch list)	NA
channel.getbackplane(ch _ list)	N/A
channel.getclose(ch _ list)	ROUTe:CLOSe?
channel.getcount(ch _ list)	N/A
channel.getdelay(ch _ list)	N/A
channel.getimage(ch _ list)	N/A
channel.getlabel(ch _ list)	N/A
channel.getpole(ch _ list)	N/A
channel.getstate(ch _ list)	ROUTe:CLOSe:STATe?
Query closed channels in specified list	
:ROUTe:MULTiple:CLOSe:STATe?	
Query closed channels in specified list	
channel.open(ch _ list)	:ROUTe:OPEN:ALL
channel.pattern.catalog()	N/A
channel.pattern.delete(name)	N/A
channel.pattern.getimage(name)	N/A
channel.pattern.setimage(ch _ list,name)	N/A
channel.pattern.snapshot(name)	N/A
channel.reset(ch _ list)	N/A
channel.setbackplane(ch _ list,abuslist)	N/A
channel.setdelay(ch _ list,value)	N/A
channel.setlabel(ch _ list,label)	N/A
channel.setpole(ch _ list,value)	N/A
Delay Command	
Delay *	TRIGger:DELay *
Digital I/O Commands	
digio.readbit(N)	OUTPut:TTL:LSENse?
digio.readport()	OUTPut:TTL:LSENse?
digio.trigger[N].assert()	N/A
digio.trigger[N].clear()	N/A
digio.trigger[N].mode	N/A
digio.trigger[N].overrun	N/A
digio.trigger[N].puslewidth	N/A
digio.trigger[N].release()	N/A
digio.trigger[N].stimulus	N/A

TSP Command	Series 7000 Command(s)
digio.trigger[N].wait (timeout)	N/A
digio.writebit	
digio.writeport	
digio.writeprotect	
Display Command	
display.clear()	DISPlay:WINDow1:TEXT:DATA
display.getannunciators()	N/A
display.getcursor()	N/A
display.getlastkey()	N/A
display.gettext()	DISPlay:WINDow1:TEXT:DATA
display.inputvalue(format)	N/A
display.loadmenu. add(displayname,chunk)	N/A
display.loadmenu. delete(displayname)	N/A
display.locallockout	N/A
display.menu	N/A
display.prompt(format, units, help)	N/A
display.screen	DISPlay:WINDow1:TEXT:STATe
display.sendkey(keycode)	SYSTem:KEY
display.setcursor(row,column)	N/A
display.settext("text")	DISPlay:WINDow1:TEXT:DATA
DISPlay:WINDow1:TEXT:STATe	
display.waitkey()	N/A
Error Queue	
errorqueue.clear()	N/A
errorqueue.count	N/A
errorqueue.next()	SYSTem:ERRor?
Exit Function	
exit	N/A
Format	
format.asciiprecision	N/A
format.byteorder	N/A
format.data	N/A
GPIB	
Gpib.address	*IDN?
Trigger	
For lan ones, N is 0 to 7	
lan.trigger[N].assert()	N/A
lan.trigger[N].clear()	N/A
lan.trigger[N].mode	N/A
lan.trigger[N].pseudostate	N/A
lan.trigger[N].overrun	N/A
lan.trigger[N].stimulus	TRIGger:SOURce
lan.trigger[N].protocol	N/A
lan.trigger[N].wait(timeout)	N/A
Local Node Attributes	
localnode.linefreq	N/A
localnode.model	*IDN?
localnode.prompts	N/A
localnode.reset()	N/A
localnode.revision	*IDN?
localnode.serialno	*IDN?
localnode.settime	N/A
localnode.setup.poweron	SYStem:POSetup
localnode.setup.recall	SYStem:POSetup
localnode.setup.save	*SAV
localnode.showerrors	System:ERRor?
Reset	
reset	*RST or SYSTem:PRESet

Table 4. Model 3706 scanner commands vs. Model 700X scanner commands (continued)

TSP Command	Series 7000 Command(s)
Scan	
scan.abort	:ABORT or ROUTe:OPEN ALL
scan.add(ch _ list,dmm _ config)	:ROUTe:SCAN (list)
scan.background	N/A
scan.bypass	TRIGger:TCONtrol:DIRection SOURce
scan.create(chlist,dmm _ config)	ROUTe:SCAN list
scan.execute	:INIT
scan.list	ROUTe:SCAN?
scan.mode	ROUTE:OPEN ALL
scan.reset()	*RST
scan.scancount	ARM:LAYer1:COUNt() or ARM:LAYer2:COUNt()
scan.state	N/A
scan.stepcount	TRIGger:COUNt?
scan.trigger.arm.clear()	
scan.trigger.arm.set()	
scan.trigger.arm.stimulus	ARM:LAYer1:SOURce () or
ARM:LAYer1:SOURce()	
scan.trigger.channel.clear	N/A
scan.trigger.channel.set()	N/A
scan.trigger.channel.stimulus	TRIGger:SOURce ()
scan.trigger.clear	N/A
scan.trigger.sequence.clear	N/A
scan.trigger.sequence.set	N/A
scan.trigger.sequence.stimulus	N/A
scan.trigger.count	
Set Up Functions	
setup.poweron	SYSTem:POSetup
setup.recall	*RCL
setup.save	*SAV
setup.cards	ROUTe:CONFigure:SLOTX:CTYPe?
slot[X] Attributes	
slot[X].commonsideohms where X is to 6 for slot number	N/A
slot[X].connectionmethod where X is to 6 for slot number	N/A
slot[X].digio where X is to 6 for slot number	N/A
slot[X].endchannel.amps where X is to 6 for slot number	N/A
slot[X].endchannel.isolated where X is to 6 for slot number	N/A
slot[X].endchannel.voltage where X is to 6 for slot number	N/A
slot[X].idn where X is to 6 for slot number	ROUTe:CONFigure:SLOTX:CTYPe?
slot[X].interlock.override	N/A
slot[X].interlock.state	N/A
slot[X].isolated where X is to 6 for slot number	N/A
slot[X].matrix where X is to 6 for slot number	N/A
slot[X].maxsettlingtime where X is to 6 for slot number	N/A
slot[X].maxvoltage where X is to 6 for slot number	N/A

TSP Command	Series 7000 Command(s)
slot[X].multiplexer where X is to 6 for slot number	N/A
slot[X].poles.four where X is to 6 for slot number	ROUTe:CONFigure:SLOTX:POLE?
slot[X].poles.one where X is to 6 for slot number	ROUTe:CONFigure:SLOTX:POLE?
print(slot[X].poles.two) where X is to 6 for slot number	ROUTe:CONFigure:SLOTX:POLE?
slot[X].pseudocard=<value>) where X is to 6 for slot number	ROUTe:CONFigure:SLOTX (name)
slot[X].startchannel.amps where X is to 6 for slot number	N/A
slot[X].startchannel.isolated where X is to 6 for slot number	N/A
slot[X].startchannel.voltage where X is to 6 for slot number	N/A
print(slot[X].tempsensor) where X is to 6 for slot number	N/A
Timer	
timer.measure.t	N/A
timer.reset	N/A
Trigger Functions	
trigger.clear	N/A
trigger.wait	N/A
TRIGGER.BLENDER	
trigger.blender[N].clear	N/A
trigger.blender[N].orenable	N/A
trigger.blender[N].overrun	N/A
trigger.blender[N].stimulus[N]	N/A
trigger.blender[N].wait where N is 1 to 4 for triggersource and N in 1 ? for blender	N/A
trigger.timer[N].clear	N/A
trigger.timer[N].count	N/A
trigger.timer[N].delay	N/A
trigger.timer[N].overrun	N/A
trigger.timer[N].passthrough	N/A
trigger.timer[N].stimulus	N/A
trigger.timer[N].wait where N is 1 to ?	N/A
TSPLINK Function	
tsplink.node	N/A
tsplink.reset	N/A
tsplink.state	N/A
tsplink.trigger[N].assert	N/A
tsplink.trigger[N].clear	N/A
tsplink.trigger[N].mode	N/A
tsplink.trigger[N].overrun	N/A
tsplink.trigger[N].release	N/A
tsplink.trigger[N].stimulus	N/A
tsplink.trigger[N].wait	N/A
userstring.add	N/A
userstring.catalog	N/A
userstring.delete	N/A
userstring.get	N/A

KEITHLEY

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: 02-363 00 40
Fax: 02-363 00 64
www.keithley.nl

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

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

FRANCE
Saint-Aubin
Ph: 01-64 53 20 20
Fax: 01-60-11-77-26
www.keithley.fr

GERMANY
Germering
Ph: 089-84 93 07-40
Fax: 089-84 93 07-34
www.keithley.de

INDIA
Bangalore
Ph: 080-26771071-73
Fax: 080-26771076
www.keithley.com

ITALY
Milano
Ph: 02-553842.1
Fax: 02-55384228
www.keithley.it

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

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

MALAYSIA
Penang
Ph: 60-4-656-2592
Fax: 60-4-656-3794
www.keithley.com

NETHERLANDS
Gorinchem
Ph: 0183-63 53 33
Fax: 0183-63 08 21
www.keithley.nl

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

SWEDEN
Solna
Ph: 08-50 90 46 00
Fax: 08-655 26 10
www.keithley.com

SWITZERLAND
Zürich
Ph: 044-821 94 44
Fax: 41-44-820 30 81
www.keithley.ch

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

UNITED KINGDOM
Theale
Ph: 0118-929 75 00
Fax: 0118-929 75 19
www.keithley.co.uk

Printed in the U.S.A. No. 2897 Dec. 07