tek.com/keithley

Series 3700A System Switch/Multimeter

Application Manual

3700AS-904-01 Rev. C March 2021

3700AS-904-01C

KEITHLEY
A Tektronix Company

Series 3700A

System Switch/Multimeter

Application Manual

These are the original instructions in English.

Document number: 3700AS-904-01 Rev. C March 2021

KEITHLEY
A Tektronix Company

Safety precautions

The following safety precautions should be observed before using this product and any associated instrumentation. Although some instruments and accessories would normally be used with nonhazardous voltages, there are situations where hazardous conditions may be present.

This product is intended for use by personnel who recognize shock hazards and are familiar with the safety precautions required to avoid possible injury. Read and follow all installation, operation, and maintenance information carefully before using the product. Refer to the user documentation for complete product specifications.

If the product is used in a manner not specified, the protection provided by the product warranty may be impaired.

The types of product users are:

Responsible body is the individual or group responsible for the use and maintenance of equipment, for ensuring that the equipment is operated within its specifications and operating limits, and for ensuring that operators are adequately trained.

Operators use the product for its intended function. They must be trained in electrical safety procedures and proper use of the instrument. They must be protected from electric shock and contact with hazardous live circuits.

Maintenance personnel perform routine procedures on the product to keep it operating properly, for example, setting the line voltage or replacing consumable materials. Maintenance procedures are described in the user documentation. The procedures explicitly state if the operator may perform them. Otherwise, they should be performed only by service personnel.

Service personnel are trained to work on live circuits, perform safe installations, and repair products. Only properly trained service personnel may perform installation and service procedures.

Keithley products are designed for use with electrical signals that are measurement, control, and data I/O connections, with low transient overvoltages, and must not be directly connected to mains voltage or to voltage sources with high transient overvoltages. Measurement Category II (as referenced in IEC 60664) connections require protection for high transient overvoltages often associated with local AC mains connections. Certain Keithley measuring instruments may be connected to mains. These instruments will be marked as category II or higher.

Unless explicitly allowed in the specifications, operating manual, and instrument labels, do not connect any instrument to mains.

Exercise extreme caution when a shock hazard is present. Lethal voltage may be present on cable connector jacks or test fixtures. The American National Standards Institute (ANSI) states that a shock hazard exists when voltage levels greater than 30 V RMS, 42.4 V peak, or 60 VDC are present. A good safety practice is to expect that hazardous voltage is present in any unknown circuit before measuring.

Operators of this product must be protected from electric shock at all times. The responsible body must ensure that operators are prevented access and/or insulated from every connection point. In some cases, connections must be exposed to potential human contact. Product operators in these circumstances must be trained to protect themselves from the risk of electric shock. If the circuit is capable of operating at or above 1000 V, no conductive part of the circuit may be exposed.

Do not connect switching cards directly to unlimited power circuits. They are intended to be used with impedance-limited sources. NEVER connect switching cards directly to AC mains. When connecting sources to switching cards, install protective devices to limit fault current and voltage to the card.

Before operating an instrument, ensure that the line cord is connected to a properly-grounded power receptacle. Inspect the connecting cables, test leads, and jumpers for possible wear, cracks, or breaks before each use.

When installing equipment where access to the main power cord is restricted, such as rack mounting, a separate main input power disconnect device must be provided in close proximity to the equipment and within easy reach of the operator.

For maximum safety, do not touch the product, test cables, or any other instruments while power is applied to the circuit under test. ALWAYS remove power from the entire test system and discharge any capacitors before: connecting or disconnecting cables or jumpers, installing or removing switching cards, or making internal changes, such as installing or removing jumpers.

Do not touch any object that could provide a current path to the common side of the circuit under test or power line (earth) ground. Always make measurements with dry hands while standing on a dry, insulated surface capable of withstanding the voltage being measured.

For safety, instruments and accessories must be used in accordance with the operating instructions. If the instruments or accessories are used in a manner not specified in the operating instructions, the protection provided by the equipment may be impaired.

Do not exceed the maximum signal levels of the instruments and accessories. Maximum signal levels are defined in the specifications and operating information and shown on the instrument panels, test fixture panels, and switching cards.

When fuses are used in a product, replace with the same type and rating for continued protection against fire hazard.

Chassis connections must only be used as shield connections for measuring circuits, NOT as protective earth (safety ground) connections.

If you are using a test fixture, keep the lid closed while power is applied to the device under test. Safe operation requires the use of a lid interlock.

If a  screw is present, connect it to protective earth (safety ground) using the wire recommended in the user documentation.

The  symbol on an instrument means caution, risk of hazard. The user must refer to the operating instructions located in the user documentation in all cases where the symbol is marked on the instrument.

The  symbol on an instrument means warning, risk of electric shock. Use standard safety precautions to avoid personal contact with these voltages.

The  symbol on an instrument shows that the surface may be hot. Avoid personal contact to prevent burns.

The  symbol indicates a connection terminal to the equipment frame.

If this  symbol is on a product, it indicates that mercury is present in the display lamp. Please note that the lamp must be properly disposed of according to federal, state, and local laws.

The **WARNING** heading in the user documentation explains hazards that might result in personal injury or death. Always read the associated information very carefully before performing the indicated procedure.

The **CAUTION** heading in the user documentation explains hazards that could damage the instrument. Such damage may invalidate the warranty.

The **CAUTION** heading with the  symbol in the user documentation explains hazards that could result in moderate or minor injury or damage the instrument. Always read the associated information very carefully before performing the indicated procedure. Damage to the instrument may invalidate the warranty.

Instrumentation and accessories shall not be connected to humans.

Before performing any maintenance, disconnect the line cord and all test cables.

To maintain protection from electric shock and fire, replacement components in mains circuits — including the power transformer, test leads, and input jacks — must be purchased from Keithley. Standard fuses with applicable national safety approvals may be used if the rating and type are the same. The detachable mains power cord provided with the instrument may only be replaced with a similarly rated power cord. Other components that are not safety-related may be purchased from other suppliers as long as they are equivalent to the original component (note that selected parts should be purchased only through Keithley to maintain accuracy and functionality of the product). If you are unsure about the applicability of a replacement component, call a Keithley office for information.

Unless otherwise noted in product-specific literature, Keithley instruments are designed to operate indoors only, in the following environment: Altitude at or below 2,000 m (6,562 ft); temperature 0 °C to 50 °C (32 °F to 122 °F); and pollution degree 1 or 2.

To clean an instrument, use a cloth dampened with deionized water or mild, water-based cleaner. Clean the exterior of the instrument only. Do not apply cleaner directly to the instrument or allow liquids to enter or spill on the instrument. Products that consist of a circuit board with no case or chassis (e.g., a data acquisition board for installation into a computer) should never require cleaning if handled according to instructions. If the board becomes contaminated and operation is affected, the board should be returned to the factory for proper cleaning/servicing.

Safety precaution revision as of June 2017.

Table of contents

Section 1

Introduction

In this section:

Welcome

Thank you for choosing a Keithley Instruments product. The Series 3700A System Switch/Multimeter features scalable, instrument grade switching and multi-channel measurement solutions that are optimized for automated testing of electronic products and components. The Series 3700A includes four versions of the Model 3706A system switch mainframe, along with a growing family of plug-in switch and control cards. When the Model 3706A mainframe is ordered with the high performance multimeter, you receive a tightly-integrated switch and measurement system that can meet the demanding application requirements in a functional test system or provide the flexibility needed in stand-alone data acquisition and measurement applications.

Introduction to this manual

This manual provides application examples that demonstrate how to use the instrument in specific situations.

For detailed information about the Series 3700A, refer to the *Series 3700A Reference Manual*.

About the Series 3700A examples

This section shows you how to use the Series 3700A for a variety of uses.

To use the Test Script Processor (TSP®) commands and run the examples in this manual, you can use Keithley Test Script Builder (TSB) software or TSB Embedded.

TSB is a software tool that simplifies building test scripts for Keithley Instruments that are enabled to use Test Script Processor (TSP®). For more information about using the TSB software, refer to "Using Test Script Builder (TSB)" in the *Series 3700A Reference Manual*. You can download the TSB software from the Product Support and Downloads web page (tek.com/product-support).

TSB Embedded is a script management tool that is available through the web interface of the instrument. You can use TSB Embedded to create, modify, and save test scripts, and to send individual commands. TSB Embedded provides some of the features of Test Script Builder (TSB). For more information about using TSB Embedded, see "Using the web interface" in the *Series 3700A Reference Manual*.

Contact information

If you have any questions after you review the information in this documentation, please contact your local Keithley Instruments office, sales partner, or distributor. You can also call the Tektronix corporate headquarters (toll-free inside the U.S. and Canada only) at 1-800-833-9200. For worldwide contact numbers, visit tek.com/contact-us.

Extended warranty

Additional years of warranty coverage are available on many products. These valuable contracts protect you from unbudgeted service expenses and provide additional years of protection at a fraction of the price of a repair. Extended warranties are available on new and existing products. Contact your local Keithley Instruments office, sales partner, or distributor for details.

Safety precautions for connections

⚠ WARNING

Connection information for switching cards is intended for qualified service personnel. Do not attempt to connect DUT or external circuitry to a switching card unless qualified to do so.

To prevent electric shock that could result in serious injury or death, comply with these safety precautions:

Before making or breaking any connections to the switching card, make sure the Series 3700A is turned off and power is removed from all external circuitry.

Do not connect signals that will exceed the maximum specifications of any installed switching card.

If both the rear analog backplane connector of the Series 3700A and the switching card terminals are connected at the same time, the test lead insulation must be rated to the highest voltage that is connected. For example, if 300 V is connected to the analog backplane connector, the test lead insulation for the switching card must also be rated for 300 V.

Dangerous arcs of an explosive nature in a high-energy circuit can cause severe personal injury or death if contacted. If the multimeter is connected to a high-energy circuit when set to a current range, low-resistance range, or any other low-impedance range, the circuit is virtually shorted.

Dangerous arcing can result (even when the multimeter is set to a voltage range) if the minimum voltage spacing is reduced in the external connections. For details about how to safely make high-energy measurements, see "High-energy circuit safety precautions" in the *Series 3700A Reference Manual*.

As described in the International Electrotechnical Commission (IEC) Standard IEC 664, the Series 3700A is Installation Category I and must not be connected to mains.

Section 2

Scanning examples

In this section:

Scanning examples

The examples in this section demonstrate how to set up scanning scripts.

NOTE

In the examples, to clear a trigger stimulus after setting, set the stimulus to `0`, which returns the stimulus setting to its factory default value, which may or may not be `0`.

Scan the card with no measurement (switch-only)

This example assumes a Keithley Instruments Model 3720 card is installed in slot 3 of a Series 3700A. This example scans all channels in a slot in a switch-only application (no measurements are made). Digital I/O line 1 is used to initiate a background scan.

Code description

Reset the Series 3700A to factory defaults.

Create a scan for all channels on the card installed in slot 3.

Set up digital I/O line 1 to detect a falling-edge trigger.

Use the digital I/O event as the stimulus of the arm layer of the trigger model.

Start the scan and run it in the background.

Program code example

```
reset()
scan.create("slot3")
digio.trigger[1].mode = digio.TRIG_FALLING
scan.trigger.arm.stimulus = digio.trigger[1].EVENT_ID
scan.background()
```

Scan the card and make measurements

The following example assumes a Keithley Instruments Model 3720 card is installed in slot 3 of a Series 3700A.

The program code scans the entire card while measuring dc volts on each channel and stored readings in a buffer called `DCVbuffer`.

Code description

Reset the Series 3700A to factory defaults.

For the DMM function:

- Set the DMM function to measure dc volts.
- Set the dc volt range to 10 volts.
- Set the number of power line cycles (NPLC) over which a measurement is integrated to 0.1. This helps improve measurement quality by canceling line noise.
- Save the DMM configuration as `measureDCV`.
- Make a buffer named `DCVbuffer` and configure it to store up to 1000 readings.

Set up digital I/O line 1 to detect a falling-edge trigger.

For the scan:

- Use a digital I/O event as the stimulus to close each channel.
- Set bypass to off so that first channel needs to see a trigger before closing.
- Create a scan for channels 1 to 60 on the card installed in slot 3.
- Start the scan and run the scan in the background. Readings are saved to the `DVCbuffer` buffer.

Example program code

```
reset()

dmm.func = "dcvolts"
dmm.range = 10
dmm.nplc = 0.1
dmm.configure.set("measureDCV")
DCVbuffer = dmm.makebuffer(1000)

digio.trigger[1].mode = digio.TRIG_FALLING

scan.trigger.channel.stimulus = digio.trigger[1].EVENT_ID
scan.bypass = scan.OFF
scan.create("3001:3060", "measureDCV")
scan.background(DCVbuffer)
```

Make 4-wire ohm measurements using a background scan

The following example assumes a Keithley Instruments Model 3720 card is installed in slot 4 of Series 3700A. This script scans all the channels in slot 4 and makes 4-wire ohm measurements using a background scan.

Code description

Reset the Series 3700A to factory defaults.

For the DMM functions, set the DMM function and the configuration for all channels in slot 4 to four-wire ohm measurements.

Create a scan for all channels on the card installed in slot 4.

Set digital I/O lines 1 and 2 to detect a falling-edge trigger.

Set up the scan as follows:

- Close each channel with a measurement complete event.
- Pulse digital I/O line 2 when a channel ready event occurs.
- Trigger each measurement with a digital I/O line 1 event trigger.
- Set bypass to on so that the first channel closes without making a measurement.

Create a buffer named `4WBuffer` that stores up to 1000 readings.

Start the scan and run the scan in the background. Save readings to the `4WBuffer` buffer.

The example program code to use a background scan to make 4-wire ohm measurements:

```
reset()

dmm.func = "fourwireohms"
dmm.setconfig("slot4", "fourwireohms")

scan.create("slot4")

digio.trigger[1].mode = digio.TRIG_FALLING
digio.trigger[2].mode = digio.TRIG_FALLING

scan.trigger.channel.stimulus = scan.trigger.EVENT_MEASURE_COMP
digio.trigger[2].stimulus = scan.trigger.EVENT_CHANNEL_READY
scan.trigger.measure.stimulus = digio.trigger[1].EVENT_ID
scan.bypass = scan.ON

4WBuffer = dmm.makebuffer(1000)

scan.background(4WBuffer)
```

Scan, measure, and store

The following example assumes a Keithley Instruments Model 3723 card is installed in slot 3 of a Series 3700A.

This program code:

- Scans the entire Model 3723 card.
- Measures dc volts on each channel.
- Stores readings in a buffer called `DCvoltBuffer`.

NOTE

For the Model 3723, the channels are reed relays and the analog backplane relays are electromechanical relays. Therefore, to have the scan run faster, this example sets the scan mode to fixed ABR, which closes the backplane relays before scanning starts and keeps them closed during the entire scan.

Code description

Reset the Series 3700A to factory defaults.

For the DMM functions:

- Set the DMM function to measure dc volts.
- Set the dc volt range to 10 volts.
- Set the number of power line cycles (NPLC) over which a measurement is integrated to 0.1. This helps improve measurement quality by canceling line noise.
- Save the DMM configuration as `DCVreadings`.
- Make a buffer named `DCvoltBuffer` and configure it to store up to 1000 readings.

Set up digital I/O line 1 to detect a falling-edge trigger.

For the scan:

- Set each channel so it closes with a digital I/O line 1 event trigger.
- Set bypass to off so that the first channel needs to see the trigger before closing.
- Set the mode to fixed ABR so that the backplane relays are closed at the start of the scan and maintained closed throughout the scan without being opened or closed.
- Create a scan list of channels 1 to 60 on slot 3.
- Start the scan to execute in the background and save readings to a buffer called `DCvoltBuffer`.

Program code example

```
reset()

dmm.func = "dcvolts"
dmm.range = 10
dmm.nplc = 0.1
dmm.configure.set("DCVreadings")
DCvoltBuffer = dmm.makebuffer(1000)

digio.trigger[1].mode = digio.TRIG_FALLING

scan.trigger.channel.stimulus = digio.trigger[1].EVENT_ID
scan.bypass = scan.OFF
scan.mode = scan.MODE_FIXED_ABR
scan.create("3001:3060", "DCVreadings")
scan.background(DCvoltBuffer)
```

Optimize scan for speed

The following example assumes a Keithley Instruments Model 3723 card is installed in slot 1 of a Series 3700A.

Some cards, such as the Model 3723, use relays that are optimized for switching speed and reliability. However, these cards still use backplane relays, which are slow. To achieve the full speed and reliability of the card, avoid scan modes (`scan.mode`) that intelligently open and close backplane relays, such as `scan.MODE_OPEN_SELECTIVE`. Instead, you can set the scan mode to `scan.MODE_FIXED_ABR`, which closes all required backplane relays before the start of the scan and keeps them closed until you program them to open.

The following is an example of a Series 3700A configured for fast scanning with the Model 3723 card. Sixty channels are scanned ten times on 200 V dc.

NOTE

The NPLC setting is at 0.006 in the example. The fastest NPLC setting supported in a Series 3700A is 0.0005. Another speed improvement option is to set the channel connect rule to OFF (`channel.connectrule = channel.OFF`). Using this setting allows channels to open and close at the same time if the application supports this operation.

Code description

Reset the Series 3700A to factory defaults.

For the DMM function:

- Set the DMM function to measure dc volts.
- Turn autorange off.
- Set the range to 200 volts.
- Turn autozero off.
- Set the NPLC to 0.006.
- Turn auto delay off.
- Create a reading buffer named `reading_buffer` that can hold up to 600 readings.
- Save the current DMM settings as the configuration `Chan1to60dcvolts`.
- Assign the configuration `Chan1to60dcvolts` to channels 1 to 60 on slot 1.

For the scan:

- Set the scan mode to fixed ABR.
- Create a scan list of channels 1 to 60 on slot 1.
- Set the scan count to 10.
- Scan in the foreground.
- Write the data out to a file on a USB flash drive.

Program code example

```
reset()

dmm.func = "dcvolts"
dmm.autorange = dmm.OFF
dmm.range = 200
dmm.autozero = dmm.OFF
dmm.nplc = 0.006
dmm.autodelay = dmm.OFF
reading_buffer = dmm.makebuffer(600)
dmm.configure.set("Chan1to60dcvolts")
dmm.setconfig("1001:1060", "Chan1to60dcvolts")

scan.mode = scan.MODE_FIXED_ABR
scan.create("1001:1060")
scan.scancount = 10
scan.execute(reading_buffer)
dmm.savebuffer("reading_buffer", "/usb1/mydata.csv")
```

Fast dc volt measurement

The following example script samples a 500 Hz 70.7 mV sine wave into the 100 mV dc range at 0.0005 PLC with autozero and autodelay disabled.

NOTE

You can cut and paste the output data from the script into a text editor. From the text editor, you can import the data into Microsoft® Excel®.

Code description

Assign the script name `test_dcv_time`.

For the DMM function:

- Set the DMM function to measure dc volts.
- Set the range to 100 mV.
- Turn line synchronization off.
- Set the NPLC to 0.0005.
- Turn auto delay off.
- Turn autozero off.
- Set the measure count to 1.
- Output the last measurement.

Create a buffer and save 30 measurements:

- Create a buffer named `buf` that holds 30 readings.
- Clear the buffer.
- Set the buffer to add data to the buffer without clearing existing buffer data.
- Set the DMM to make 30 readings.
- Send the measurements to the buffer.

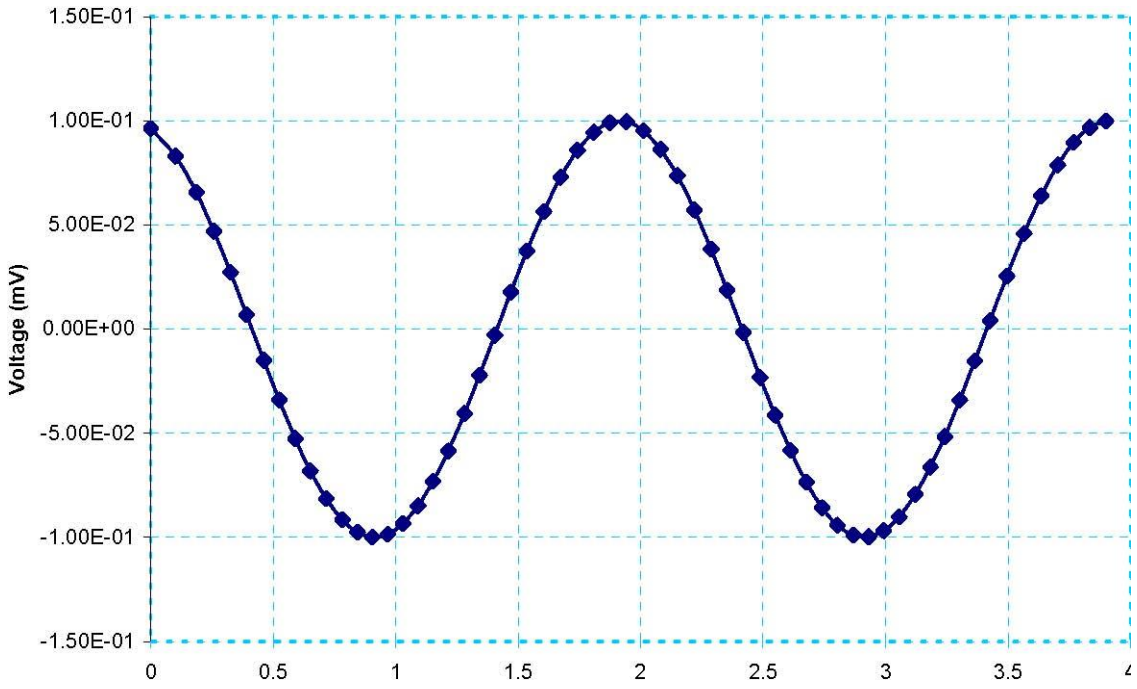
Program code example

```
loadscript test_dcv_time

dmm.func = "dcvolts"
dmm.range = 100e-3
dmm.linesync = 0
dmm.nplc = 0.0005
dmm.autodelay = 0
dmm.autozero = 0
dmm.measurecount = 1
print(dmm.measure())
print("wait nplc delay")

buf = dmm.makebuffer(30)
buf.clear()
buf.appendmode = 1
dmm.measurecount = 30
dmm.measure(buf)
   for x=1,buf.n do printbuffer(x,x,buf, buf.relativetimestamps)
   end
endscript
```

Figure 1: 3706A ACV Input on DCV at 0.0005 plc



AC volts Autodelay Once script

The following example script creates a scan that makes ac voltage measurements with a delay after the first measurement only.

Code description

Assign the script name `test_acv_autodelay`.

Reset the Series 3700A to the factory defaults.

Set the connect rule off, which causes the instrument to close relays as efficiently as possible to improve speed performance without applying a rule.

For the DMM function:

- Set the DMM function to measure dc volts.
- Set the range to 1 volt.
- Set the detector bandwidth of 300.
- Set the NPLC to 0.06. When bandwidth set to 300, NPLC can be programmed from 0.0005 PLC to 12 PLC at 60 Hz or 15 PLC at 50 Hz.
- Turn autozero off.
- Set autodelay to once. The instrument will include a single 50 ms delay before the first measurement only after channel closure.
- Set the DMM to make 25 readings on the same channel.
- Define this group of DMM settings as `my-1Vac`.
- Assign the configuration for channels 4 and 24 on slot 4 to `my-1Vac`.

For the buffer:

- Create a buffer named `buf` with a buffer size of 200 readings.
- Clear the buffer.
- Set the readings to be appended to the existing buffer content.

For the scan:

- Create a scan list that includes channels 4 and 24 on slot 4. Backplane channels 4911 and 4921 are automatically paired.
- Set the scan to loop 4 times.
- Start the scan.
- Send the information to the buffer. Note that `x,x` prints reading and time vertically so you can copy and paste the information into Microsoft Excel.

Program code example

```
loadscript test_acv_autodelay

reset()
channel.connectrule = channel.OFF

dmm.func = "acvolts"
dmm.range = 1
dmm.detectorbandwidth = 300
dmm.nplc = 0.06
dmm.autozero = 0
dmm.autodelay = dmm.AUTODELAY_ONCE
scan.measurecount = 25
dmm.configure.set("my-1Vac")
dmm.setconfig("4004, 4024", "my-1Vac")

buf = dmm.makebuffer(200)
buf.clear()
buf.appendmode = 1

scan.create("4004, 4024")
scan.scancount = 4
scan.execute(buf)
   for x=1,buf.n do printbuffer(x,x,buf, buf.relativetimestamps)
   end
endscript
```

This table illustrates how to optimize ac volt or ac current measurements for input signal frequency, reading rate, autodelay, and measure count.

Setting	Detector band-width	Frequency band	Aperture time	Fixed number of samples per reading	Auto zero	Autodelay (ms)		Measure count	Total measure count time (s)	Average reading (s)
SLOW	3	3 Hz to 300 kHz	1 ms, fixed	2000	N/A	200	Off	10	21.41167	0.467
							Once		21.61796	0.463
							On		23.41259	0.427
MED	30	30 Hz to 300 kHz	1 ms, fixed	200	N/A	200	Off	20	6.00595	3.33
							Once		6.22548	3.213
							On		10.0174	1.997
FAST	300	300 Hz to 300 kHz	8.33 µs (10 µs) min	1	Off	50	Off	100	0.00765	13,070
							Once		0.15072	663.5
							On		5.08854	19.65
FAST	300◄	300 Hz to 300 kHz	16.67 ms (20 ms)	1	On◄	50	Off	100	2.35427	42.48
							Once◄		2.50335	39.95
							On		7.08160	14.12

Default setting ◄ at 50 Hz power line frequency

Fast alternating scan, dc volts, and two-wire ohms

The following example illustrates how to configure a Series 3700A and a Model 3723 switch card for fast alternating function scans. The example shows channel 1 measuring +7.5 V dc and channel 2 measuring a 1 KΩ resistor. Scanning, which includes a relay close, DMM configuration, DMM measure, and channel open, is achieved at rates of less than 1.6 ms per channel.

Code description

Assign the script name `test_func_chg`.

Reset the Series 3700A to factory defaults.

Set the connect rule off. The instrument closes relays as efficiently as possible to improve speed performance without applying a rule.

For the first set of DMM settings:

- Set the DMM function to measure dc volts.
- Set the range to 10 volts.
- Set the NPLC to 0.0005.
- Turn autozero off.
- Save the settings to a configuration named `fastDCV`.

For the second set of DMM settings:

- Set the DMM function to 2-wire ohms.
- Set the range to 1000 Ω.
- Se the NPLC to 0.0005.
- Turn autozero off.
- Turn autodelay off.
- Save the settings to a configuration named `my-2w`.

Set the scan measure count to 1.

Use the `fastDCV` configuration for channel 4 on slot 4.

Use the `my-2w` configuration for channel 24 on slot 4.

For the buffer:

- Create a buffer named `buf` and set the buffer size to 20 readings.
- Clear the buffer.
- Set the readings to be appended to the existing buffer content.

For the scan:

- Create a scan list with channels 4 and 24 on slot 4. Backplane channel 4911 and 4921 will be automatically paired.
- Set the scan to loop 10 times.
- Start the scan.
- Print the reading and relative timestamps from the start of the scan. `buf.relativetimestamps` includes the date and real time. `x,x` prints reading and time vertically so you can copy and paste the information into Microsoft Excel.
- Open all channels on all slots.

Program code example

```
loadscript test_func_chg

reset()

channel.connectrule = channel.OFF

dmm.func = "dcvolts"
dmm.range = 10
dmm.nplc = 0.0005
dmm.autozero = 0
dmm.configure.set("fastDCV")
dmm.func = "twowireohms"

dmm.range = 1000
dmm.nplc = 0.0005
dmm.autozero = 0
dmm.autodelay = dmm.OFF
dmm.configure.set("my-2w")
scan.measurecount = 1

dmm.setconfig("4004", "fastDCV")
dmm.setconfig("4024", "my-2w")

buf = dmm.makebuffer(20)
buf.clear()
buf.appendmode = 1

scan.create("4004, 4024")
scan.scancount = 10
scan.execute(buf)
   for x=1,buf.n do printbuffer(x,x,buf, buf.relativetimestamps)
   end
channel.open("allslots")
endscript
```

Thermocouple measurement using the front panel

To make temperature measurements from the front panel:

1. To select the temperature measurement function, press the **CONFIG** key, then press the **DMM** key. FUNC flashes on, then off. Press the **ENTER** key or wheel. **Function?** is displayed on the first line of the display and the second line displays available functions. Use the left or right arrow keys or the knob to select **TEMP**.
2. Set thermocouple device attributes:
   - Turn the navigation wheel to scroll to the **THERMO** menu item and press the navigation wheel or the **ENTER** key.
   - Turn the navigation wheel to scroll to the **THERMOCOUPLE** temperature connection and press the **ENTER** key.
   - Turn the navigation wheel to select the thermocouple type (J, K, T, E, R, S, B, or N) and press the navigation wheel or the **ENTER** key.
3. Set thermocouple device reference junction type:
   - Turn the navigation wheel to scroll to the **REFJUNCT** menu item and press the navigation wheel or the **ENTER** key.
   - Select the Reference Junction: SIMULATED, INTERNAL, or EXTERNAL. See "Reference junctions" in the *Series 3700A Reference Manual* for more information.
   - Press the navigation wheel or the **ENTER** key to set the selection.
4. If a SIMULATED reference junction was selected:
   - Turn the navigation wheel to scroll to the **SIMREF** menu item and press the navigation wheel or the **ENTER** key.
   - Using the navigation wheel, select the reference temperature (default values are units dependent: 023.00 °C, 296.15 °K, and 073.40 °F).
   - Press the navigation wheel or the **ENTER** key to set the selection.
5. Press the **EXIT** key to leave the THERMO MENU.
6. Change temperature attributes as needed.
7. Press the **EXIT** key to leave the TEMP ATTR MENU.
8. Press **TRIG** to make measurements.

Thermocouple measurement using a remote interface

This example demonstrates how to make thermocouple measurements through the remote interface.

Code description

Create a script named `test_temp`.

Reset the Series 3700A to the factory defaults.

For the DMM function:

- Set the DMM function to measure temperature using a J-type thermocouple.
- Set open lead detection on.
- Set the units to Fahrenheit.
- Set the reference junction to internal.
- Define this group of settings as the configuration `my_temp_j`.

Assign the configuration settings to channels 1 to 10 on slot 4.

Set the scan to measure once.

For the buffer:

- Create a buffer named `buf` with a buffer size of 20 readings.
- Clear the buffer.
- Set the readings to be appended to the existing buffer content.

For the scan:

- Create a scan of channels 1 to 10 on slot 4.
- Set the scan to run twice.
- Start the scan and store the readings in the buffer `buf`.
- Retrieve the readings. `x,x` prints reading and time vertically so you can copy and paste the information into Microsoft Excel.
- Open all channels in all slots.

Program code example

```
loadscript test_temp

reset()

dmm.func = dmm.TEMPERATURE
dmm.transducer = dmm.TEMP_THERMOCOUPLE
dmm.thermocouple = dmm.THERMOCOUPLE_J
dmm.opendetector = dmm.ON
dmm.units = dmm.UNITS_FAHRENHEIT
dmm.refjunction = dmm.REF_JUNCTION_INTERNAL
dmm.configure.set("my_temp_j")
dmm.setconfig("4001:4010", "my_temp_j")

scan.measurecount = 1

buf = dmm.makebuffer(20)
buf.clear()
buf.appendmode = 1

scan.create("4001:4010")
scan.scancount = 2
scan.execute(buf)
   for x = 1, buf.n do printbuffer (x,x,buf)
   end
channel.open("allslots")
endscript
```

Section 3

TSP-Link and interactive triggers

In this section:

TSP-Link and interactive triggers

This example uses a TSP script to configure and run the test. The script includes two separate functions for configuring the Series 3700A and System SourceMeter® Instrument and a third function for running the test. TSP-Link trigger lines are used to coordinate the actions between the two instruments.

NOTE

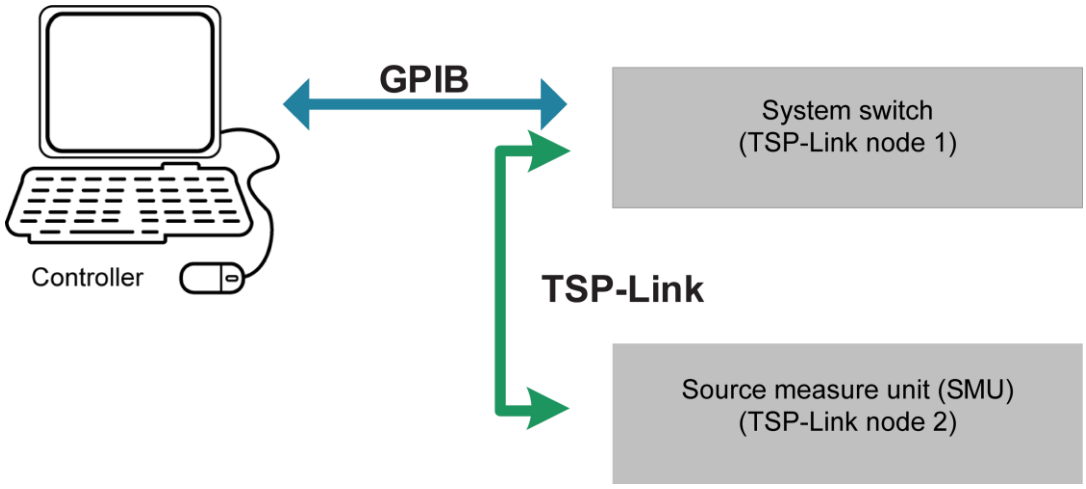
A TSP script is a sequence of instrument commands and programming statements. The Series 3700A can store the TSP script for later use. Using scripts allows the Series 3700A to perform simple and complex tasks with less interaction with the remote interface controller (the controlling computer). For more information on using scripts, please see "Instrument programming" in the *Series 3700A Reference Manual*.

When the test is run, the instruments assert trigger signals after completing their respective actions. After connecting the diode, the Series 3700A sends a trigger signal to the System SourceMeter instrument. The SMU then performs a single I-V sweep on the diode and sends a trigger signal back to the Series 3700A to indicate that the test is completed. Hardware handshaking continues in this fashion until all six diodes have been tested. Data from the SMU is returned to the controlling computer at the end of all the tests.

Set up communication

The communication setup is illustrated in the diagram below. GPIB is used as an example, but this application can be run using any of the supported communication interfaces for the instruments. A TSP-Link connection enables communication between the two instruments and commands for the SourceMeter (on TSP-Link node 2) are sent over the TSP-Link interface.

Figure 2: GPIB communication interface with TSP-Link



To configure the TSP-Link communication interface, each instrument must have a unique TSP-Link node number. Configure the node number for the Series 3700A to 1 and the SourceMeter Instrument to 2.

To set the TSP-Link node number using the front panel interface of either instrument:

1. Press **MENU**.
2. Select **TSPLink**.
3. Select **NODE**.
4. Use the navigation wheel ◔ to adjust the node number.
5. Press **ENTER** to save the TSP-Link node number.

On the Series 3700A, perform a TSP-Link reset to alert the Series 3700A to the presence of the SMU:

1. Press **MENU**.
2. Select **TSPLink**.
3. Select **RESET**.

NOTE

You can also perform a TSP-Link reset from the remote command interface by sending `tsplink.reset()` to the Series 3700A. This command is included in the following example program code.

NOTE

If error code 1205, "TSP-Link initialization failed (no remote nodes found)," is generated during the TSP-link reset, ensure that the System SourceMeter instrument has a unique TSP-Link node number.

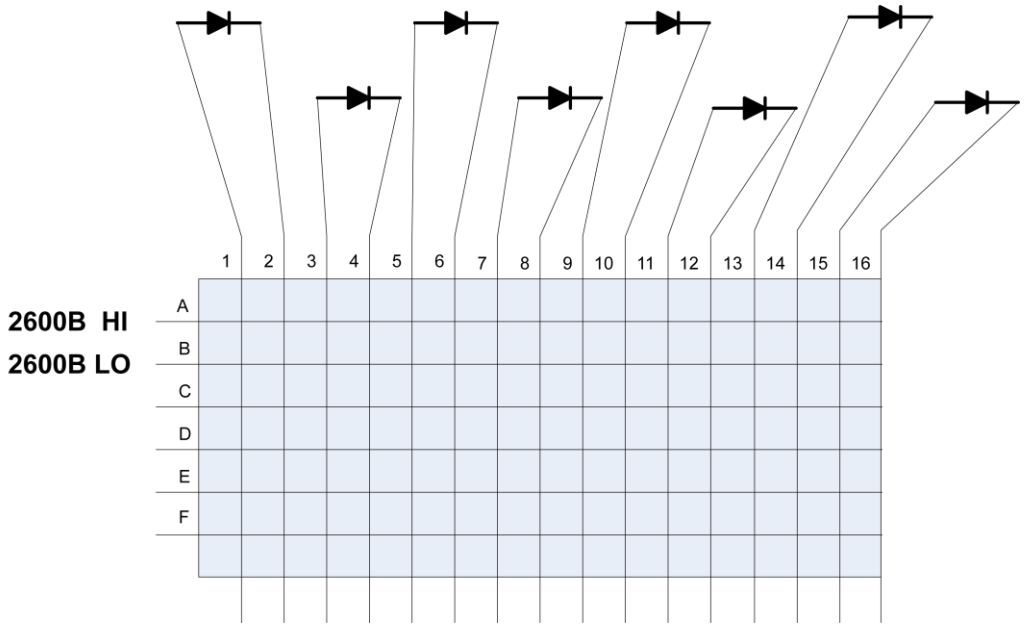
In this example, in addition to the TSP-Link interface serving as a communication bus, it will also serve as triggering bus to enable hardware handshaking between the System SourceMeter instrument and Series 3700A.

There are three digital lines in the TSP-Link cable that may be used for triggering. This example uses trigger lines 1 and 2. The Series 3700A outputs a trigger signal on TSP-Link line 1 and waits for a trigger signal on TSP-Link line 2 before proceeding with the next set of crosspoint closures. The System SourceMeter instrument outputs a trigger signal on TSP-Link line 2 after completing its test and waits for a trigger signal on TSP-Link line #1 before testing.

Logical block diagram of test connections

The following figure shows the logical block diagram of the test connections.

Figure 3: Series 3700A test connections



Example program code

In the following, all commands are sent from the computer to the Series 3700A. This example can be run using TSB Embedded. The Series 3700A sends all commands in the shaded rows to the System SourceMeter instrument through the TSP-Link interface.

3700A diodeTest code description

Create the script `diodeTest` to contain test configuration and execution instructions.

Create a function to configure the Series 3700A.

Reset TSP-Link.

Reset Series 3700A.

Clear errors.

Define table that details crosspoints to close to connect each diode to the SMU.

Add entries to the table.

Set TSP-Link lines 1 and 2 to send and receive falling-edge triggers.

Clear any latched triggers on TSP-Link trigger lines 1 and 2.

Return the table as an output of the function.

diodeTest program code example

```
loadscript diodeTest
function configSwitch()
   tsplink.reset()
   localnode.reset()
   errorqueue.clear()
   matrixChannels = {}
   matrixChannels[1] = "1001, 1002"
   matrixChannels[2] = "1003, 1004"
   matrixChannels[3] = "1005, 1006"
   matrixChannels[4] = "1007, 1008"
   matrixChannels[5] = "1009, 1010"
   matrixChannels[6] = "1011, 1012"
   tsplink.trigger[1].mode = tsplink.TRIG_FALLING
   tsplink.trigger[2].mode = tsplink.TRIG_FALLING
   tsplink.trigger[1].clear()
   tsplink.trigger[2].clear()
   return matrixChannels
end
```

SMU code description

```
Create a function to configure the System SourceMeter instrument.
Reset the instrument.
Clear all errors and reset status bits.
Clear existing data from data buffer.
Set source function to current source.
Set source current range to 10 mA.
Set voltage measurement range to  6 V.
Set integration rate.
Enable buffer append mode.
Set voltage compliance during sweep.
Configure linear staircase sweep.
Enable the source action during the sweep.
Configure sweep voltage measurements.
Enable voltage measurements during the sweep.
Set trigger count to the number of steps in  the sweep.
Set arm count to six, the total number of sweeps that will be run for all diodes.
Set TSP-Link trigger lines 1 and 2 to send and receive falling edge trigger signals.
Clear any latched triggers on TSP-Link trigger lines 1 and 2.
Configure sweep to start when signal received on TSP-Link line 1.
Configure System SourceMeter to output trigger signal on TSP-Link line 2 when sweep is complete.
```

SMU program code example

```
function configSMU()
   node[2].reset()
   node[2].errorqueue.clear()
   node[2].smua.nvbuffer1.clear()
   node[2].smua.source.func = node[2].smua.OUTPUT_DCAMPS
   node[2].smua.source.rangei = 0.01
   node[2].smua.measure.rangev = 6
   node[2].smua.measure.nplc = 1
   node[2].smua.nvbuffer1.appendmode = 1
   node[2].smua.trigger.source.limitv = 10
   node[2].smua.trigger.source.lineari(0, 0.01, 11)
   node[2].smua.trigger.source.action = node[2].smua.ENABLE
   node[2].smua.trigger.measure.v(node[2].smua.nvbuffer1)
   node[2].smua.trigger.measure.action = node[2].smua.ENABLE
   node[2].smua.trigger.count = 11
   node[2].smua.trigger.arm.count = 6
   node[2].tsplink.trigger[1].mode = node[2].tsplink.TRIG_FALLING
   node[2].tsplink.trigger[2].mode = node[2].tsplink.TRIG_FALLING
   node[2].tsplink.trigger[1].clear()
   node[2].tsplink.trigger[2].clear()
   node[2].smua.trigger.arm.stimulus = node[2].tsplink.trigger[1].EVENT_ID
   node[2].tsplink.trigger[2].stimulus =
    node[2].smua.trigger.SWEEP_COMPLETE_EVENT_ID
end
```

3700A runDiodeTest function description

This function sequences the crosspoint closures and triggering.

Call the function to configure the switch storing the result in the variable `xptTable`.

Call function to configure the SMU and turn on its output.

Initiate the System SourceMeter so that it is ready to sweep once a trigger signal is received (the output turns on but no power is applied).

The `for` loop iterates through the channel list, waiting three seconds for a trigger on TSP-Link line 2 before closing a channel. The first channel of the list does not wait for a trigger. After channel closure is complete, an output trigger is asserted on TSP-Link line 1.

After all switch and measure actions have completed, turn off the SMU output, open all crosspoints, and print `done`. The computer controller can query for `done` to determine when the test is complete.

Ends the function `runDiodeTest`.

Ends the script `diodeTest`.

3700A runDiodeTest program code example

```
function runDiodeTest()
   xptTable = configSwitch()
   configSMU()
   node[2].smua.source.output = 1
   node[2].smua.trigger.initiate()
   for i = 1, table.getn(xptTable) do

      if i > 1 then
         triggered = tsplink.trigger[2].wait(3)
         if triggered == true then
            channel.exclusiveclose(xptTable[i])
            tsplink.trigger[1].assert()
         else
            print("Trigger not received")
         end
      else
         channel.exclusiveclose(xptTable[i])
         tsplink.trigger[1].assert()
      end
   end
   waitcomplete()
   node[2].smua.source.output = 0
   channel.open("allslots")
   print("done")
end
endscript
```

Program code to run the test

The example program code stores the program in the instrument for later use. This code runs the script to define the functions, then executes the test by calling the function that configures the instruments and sequences their actions.

Program code example

```
diodeTest()
runDiodeTest()
```

To retrieve data from the System SourceMeter instrument through TSP-Link, use the `printbuffer()` command to print the data to the computer communication interface. The following example code prints all readings from the SMU data buffer. `readings` is a buffer attribute.

Program code example

```
printbuffer(1, node[2].smua.nvbuffer1.n, node[2].smua.nvbuffer1.readings)
```

Section 4

Using the scanning and triggering model

In this section:

Using the scanning and triggering model

This example demonstrates how to use the scanning and triggering model of the Series 3700A to maximize the speed of the test by preconfiguring the sequence of crosspoints used.

This example uses a TSP script to configure the instruments for testing. Once the instruments are configured, the Series 3700A internally manages all close and open actions and any triggering signals to interface with the System SourceMeter instrument. No separate script is required to sequence these actions.

As with the previous examples, a full I-V sweep is performed on each of the six diodes. Data from these tests is available from the System SourceMeter instrument after all tests have been completed.

Set up communication

In this example, the TSP-Link interface serves as a communication bus and as a triggering bus to enable hardware handshaking between the System SourceMeter instrument and Series 3700A.

The communication setup is identical to the setup in the previous example; see TSP-Link and interactive triggers (on page 3-1).

There are three digital lines in the TSP-Link cable that you can use for triggering. This example uses trigger lines 1 and 2. The Series 3700A outputs a trigger signal on TSP-Link line 1 and waits for a trigger signal on TSP-Link line 2 before proceeding with the next set of crosspoint closures. The System SourceMeter instrument outputs a trigger signal on TSP-Link line 2 after completing its test and waits for a trigger signal on TSP-Link line #1 before testing.

Program code

In the following code, all commands are sent from the computer to the Series 3700A. This example can be run using TSB Embedded. The Series 3700A sends all the SMU commands to the System SourceMeter instrument through the TSP-Link interface.

3700A diodeTest code description

Create the script `diodeTest` to contain test configuration and execution instructions.

Create a function to configure the Series 3700A.

Reset TSP-Link.

Reset Series 3700A.

Clear errors.

Define table that details crosspoints to close to connect each diode to the SMU. Add entries to the table.

Use the values in the table to create a scan list.

Set TSP-Link lines 1 and 2 to send and receive falling-edge triggers.

Clear any latched triggers on TSP-Link trigger lines 1 and 2.

For the scan:

- Configure the scanning and triggering mode of Series 3700A.
- Bypass is enabled, so the channel stimulus event is bypassed on the first scan.
- The scan is armed immediately.
- The channel close action is triggered with receipt of trigger on TSP-Link line 2.
- After the channel close action occurs, the trigger is output on TSP-link line 1.

Loop through the scan list once.

End of the `configSwitch` function.

diodeTest program code example

```
loadscript diodeTest
function configSwitch()
   tsplink.reset()
   localnode.reset()
   errorqueue.clear()

   muxChannels = {}
   muxChannels[1] = "1001, 1202"
   muxChannels[2] = "1003, 1204"
   muxChannels[3] = "1005, 1206"
   muxChannels[4] = "1007, 1208"
   muxChannels[5] = "1009, 1210"
   muxChannels[6] = "1011, 1212"

   for i = 1, 6 do
      scan.addimagestep(muxChannels[i])
   end

   tsplink.trigger[1].mode = tsplink.TRIG_FALLING
   tsplink.trigger[2].mode = tsplink.TRIG_FALLING

   tsplink.trigger[1].clear()
   tsplink.trigger[2].clear()

   scan.bypass = scan.ON
   scan.trigger.arm.stimulus = 0
   scan.trigger.channel.stimulus = tsplink.trigger[2].EVENT_ID
   tsplink.trigger[1].stimulus = scan.trigger.EVENT_CHANNEL_READY
   scan.scancount = 1
end
```

SMU code description

Create a function to configure the System SourceMeter instrument.

Reset the instrument.

Clear all errors and reset status bits.

Clear existing data from data buffer.

For the SMU:

- Set source function to current source.
- Set source current range to 10 mA.
- Set voltage measurement range to 6 V.
- Set integration rate.
- Enable buffer append mode.

For the SMU sweep:

- Set the voltage compliance during the sweep.
- Configure linear staircase sweep.
- Enable the source action during the sweep.
- Configure sweep voltage measurements.
- Enable voltage measurements during the sweep.
- Set trigger count to the number of steps in the sweep.
- Set arm count to six, the total number of sweeps that will be run for all diodes.

Set TSP-Link trigger lines 1 and 2 to send and receive falling edge trigger signals.

Clear any latched triggers on TSP-Link trigger lines 1 and 2.

Configure sweep to start when signal received on TSP-Link line 1.

Configure System SourceMeter to output trigger signal on TSP-Link line 2 when sweep is complete.

End of the function `ConfigSMU`.

SMU program code example

```
function configSMU()
   node[2].reset()
   node[2].errorqueue.clear()
   node[2].smua.nvbuffer1.clear()
   node[2].smua.source.func = node[2].smua.OUTPUT_DCAMPS
   node[2].smua.source.rangei = 0.01
   node[2].smua.measure.rangev = 6
   node[2].smua.measure.nplc = 1
   node[2].smua.nvbuffer1.appendmode = 1

   node[2].smua.trigger.source.limitv = 10
   node[2].smua.trigger.source.lineari(0, 0.01, 11)
   node[2].smua.trigger.source.action = node[2].smua.ENABLE
   node[2].smua.trigger.measure.v(node[2].smua.nvbuffer1)
   node[2].smua.trigger.measure.action = node[2].smua.ENABLE
   node[2].smua.trigger.count = 11
   node[2].smua.trigger.arm.count = 6

   node[2].tsplink.trigger[1].mode = node[2].tsplink.TRIG_FALLING
   node[2].tsplink.trigger[2].mode = node[2].tsplink.TRIG_FALLING
   node[2].tsplink.trigger[1].clear()
   node[2].tsplink.trigger[2].clear()
   node[2].smua.trigger.arm.stimulus = node[2].tsplink.trigger[1].EVENT_ID
   node[2].tsplink.trigger[2].stimulus =
    node[2].smua.trigger.SWEEP_COMPLETE_EVENT_ID
end
```

Run the runDiodeTest() function from 3700A

Send the following code to sequence the crosspoint closures and triggering:

```
function runDiodeTest()
```

Call the 3700A from the SMU

Send the following code from the SMU to call the function to configure the Series 3700A:

```
configSwitch()
```

Run the configSMU function from the 3700A

Call the `configSMU` function, which configures the System SourceMeter instrument and turns on its output.

Initiate the System SourceMeter so that it is ready to sweep once a trigger signal is received. Note that the output turns on but no power is applied.

Initiate the System Switch to start a foreground scan. Waits until Model 2635B finishes all measurements. Remote operation pauses until all six tests are complete. For other options on running a scan, see Using background scans for longer scan lists (on page 4-6).

The code to run is:

```
   configSMU()
   node[2].smua.source.output = node[2].smua.OUTPUT_ON
   node[2].smua.trigger.initiate()
   scan.execute()
   waitcomplete()
```

Turn off the SMU output and open all crosspoints

The following code turns off the SMU output and opens all crosspoints. It also ends the `runDiodeTest` function and ends the `diodeTest` script.

```
   node[2].smua.source.output = node[2].smua.OUTPUT_OFF
   channel.open('allslots')
end
endscript
```

Program code to run the test

The example program code stores the program in the instrument for later use. This code runs the script to define the functions, then executes the test by calling the function that configures the instruments and sequences their actions.

Program code example

```
diodeTest()
runDiodeTest()
```

To retrieve data from the System SourceMeter instrument through TSP-Link, use the `printbuffer()` command to print the data to the computer communication interface. The following example code prints all readings from the SMU data buffer. `readings` is a buffer attribute.

Program code example

```
printbuffer(1, node[2].smua.nvbuffer1.n, node[2].smua.nvbuffer1.readings)
```

Using background scans for longer scan lists

The previous example illustrates use of the scanning and triggering model to create a foreground scan.

While running a foreground scan, you must wait for the scan to complete or you must abort the scan before you can query the instrument state or any reading buffers. For a scan with a few channels or crosspoints in the scan list, this might not be a problem. However, when there are many channels in the scan list, or when scans are run over long periods, it can be useful to determine instrument state. In these situations, you can run a background scan. A background scan allows you to query settings during a scan.

This example demonstrates how to use a background scan. This example also sequences the crosspoint closures and triggering.

Program code description

Create the script `runDiodeTest` to contain test configuration and execution instructions.

Call the `configSwitch()` function to configure the Series 3700A.

Call the `configSMU()` function that configures the System SourceMeter instrument and turns on its output.

Initiate the System SourceMeter so that it is ready to sweep once a trigger signal is received. Note that the output turns on but no power is applied.

Initiate the System Switch to start a background scan.

Delay 2 seconds to allow the scan to start.

Use variables to hold the scan state response.

Print the response to scan state, which you can use to determine when the scan is complete.

Query the scan state every second to determine when the scan completes.

When the scan is complete (scan state = 6), exit the loop.

Turn off the SMU output.

Open all crosspoints.

End the `runDiodeTest` function.

Program code example

```
function runDiodeTest()
   configSwitch()
   configSMU()
   node[2].smua.source.output = node[2].smua.OUTPUT_ON
   node[2].smua.trigger.initiate()
   scan.background()
   delay(2)
   scanState, scanCount, stepCount = scan.state()
   print(scanState, " ", scanCount, " ", stepCount, " ")
   while scanState ~=6 do
      delay(1)
   end
   node[2].smua.source.output = node[2].smua.OUTPUT_OFF
   channel.open('allslots')
end
```

Section 5

IEEE-1588 in Series 3700A-based systems

In this section:

IEEE-1588 in Series 3700A-based systems

This section provides alarm scheduling examples using IEEE Std 1588-2008 precision time protocol (PTP).

Scheduling alarms

You can schedule alarms to request the Series 3700A to perform actions at a specific time and date or at a specific time interval. You can schedule alarms in UTC or PTP time.

NOTE

It is important to be consistent in defining the alarms using the same time format. Otherwise, the alarms will fire on the networked devices at different times, with a time difference equal to the difference between PTP and UTC.

You can set a maximum of two alarms for each Series 3700A.

To schedule an alarm, first convert the alarm time to UTC seconds. You can perform this conversion using `os.time`. If you are specifying alarms in UTC time, you can use this value with `schedule.alarm[N].seconds` to schedule an alarm, where *N* represents the tag number of the alarm that you configure.

You can use the Lua function `os.time` to return the present time or convert a local date and time to UTC-based seconds elapsed since January 1, 1970. When used without parameters, `os.time` returns the present date and time. When used with parameters, the syntax is `os.time{year = <n>, month = <n>, day = <n>, hour = <n>, sec = <n>, isdst = <b>}`. `<n>` is a number and `<b>` is a Boolean where true is Daylight Savings Time. It is not necessary to specify all parameters.

The following example demonstrates how to use `os.time`.

Program code description

Retrieve current UTC time in seconds since 1/1/1970.

Convert 3:00 pm March 1, 2021 to UTC seconds since 1/1/1970.

Create start time to occur 60 seconds after current time.

Program code example

```
print(os.time)
local l_start_Time
l_start_Time = os.time{year=2021, month=3, day=1, hour=15}
local l_start_Time
l_start_Time = os.time() + 60
print(l_start_Time)
```

To specify alarms in PTP format, convert UTC seconds to PTP seconds by adding the value returned by `ptp.utcoffset` to the UTC time. The Series 3700A does not differentiate between PTP and UTC time. Use the converted PTP time in setting values for `schedule.alarm[N].ptpseconds`, where *N* represents the number of the alarm you configure.

You can also schedule alarms to occur at a fractional second using either PTP or UTC format with `schedule.alarm[N].fractionalseconds`.

After defining the alarm, configure the number of times you would like to repeat this alarm using `schedule.alarm[N].repetition`.

To set the time (in seconds) between firings of the alarm, use `schedule.alarm[N].period`. To fire the alarm once, set `schedule.alarm[N].period` to zero. If you want the alarm to repeat forever, set `schedule.alarm[N].period` to a non-zero value and set `schedule.alarm[N].repetition` to zero.

To enable an alarm, set `schedule.alarm[N].enable` to `1`. To disable an alarm, set it to `0`. To disable all alarms, send `schedule.disable()`.

For more detail on the schedule alarm commands, see the descriptions in the "Command Reference" section of the *Series 3700A Reference Manual*.

Scheduling alarms on a stand-alone Series 3700A

To configure a single Series 3700A to perform an event at a particular date and time, you must schedule alarms, but you do not need to enable IEEE-1588. Therefore, you can send these commands over any remote interface.

To initiate a specific action at the firing of the alarm, you must use the event identifier for the scheduled alarm, `schedule.alarm[N].EVENT_ID`, as the stimulus of one of the control sources defined in the trigger model.

The following example demonstrates how to configure a scan of five channels to run once every hour starting at 3 am on September 1, 2019.

Program code description

Convert to UTC time.

Convert to PTP time.

Configure the alarm.

Configure the alarm repetition count.

Set the alarm period to 1 hour = 60 seconds x 60 minutes.

Enable the alarm.

Associate a DMM configuration and configure a scan.

Set 5 scans of 5 channels.

Command the scan to start when alarm 1 fires.

Set scan count and initiate execution of background scan.

Program code example

```
Start_time = os.time{year=2019, month=9, day=1, hour=3}
Start_time = Start_time + ptp.utcoffset
schedule.alarm[1].ptpseconds = Start_time
schedule.alarm[1].fractionalseconds = 0
schedule.alarm[1].repetition = 5
schedule.alarm[1].period = 60 * 60
schedule.alarm[1].enable = 1
dmm.setconfig("1001:1005", "dcvolts")
scan.create("1001:1005")
buf = dmm.makebuffer(25)
scan.trigger.arm.stimulus = schedule.alarm[1].EVENT_ID
scan.scancount = 5
scan.background(buf)
```

Section 6

Measurement examples

In this section:

External DMM and switch triggering

This example demonstrates how to perform DMM and switch triggering with legacy DMMs, such as the Model 2000, or SMUs, such as the Model 2400.

Program code description

Create a script called `Ext_Trig_Dmm` that contains the following commands.

Display the main screen on the front panel of the Series 3700A.

Open all slots.

Set the instrument to close relays as efficiently as possible to improve speed performance without applying a rule.

For one set of DMM measurements:

- Set the DMM function to dc volts, 5½ display digits, a range of 10, NPLCs of 0.0005, relative offset enable off, autozero off, and autodelay off.
- Name this DMM configuration `dc_10V` and set it as the DMM configuration for measurements on channel 14 on slot 1.

For another set of DMM measurements:

- Keep the DMM settings at dc volts, 5½ display digits, NPLCs of 0.0005, relative offset enable off, autozero off, and autodelay off.
- Change the range to be 100 mV.
- Name this DMM configuration `dc_100mV`.
- Set this configuration as the DMM configuration for measurements on channel 12 on slot 1.

For the scan:

- Create a scan that includes channels 14 and 12 of slot 1.
- Set the scan mode so that all required backplane relays close before the start of the scan and remain closed.
- Set the scan to take 100 measurements for each channel closure.
- Set the scan to loop 3 times.

For the triggers:

- Clear and remove any Series 3700A latched triggers.
- Define the Series 3700A digital I/O as inputs, then redefined by trigger model triggers.
- Set the trigger event for each trigger layer.
- Set the arm and channel trigger events to pass through immediately.

Define the Series 3700A digital I/O input trigger.

Set scan bypass on, which does not gate the channel trigger layer. `scan.execute` is the first trigger, closing the first channel in the channel list.

Set the scan to detect the rising-edge input triggers and automatically latch and drive the trigger line low. Asserts a TTL-low pulse as an output trigger.

NOTE

You must use digital I/O 10 through 14 with legacy DMMs. The pull-up of the Series 3700A is too strong on lines 1 to 9.

Allows the trigger model to continue to the next layer immediately.

Wait to receive a VMC signal from the external DMM. The measure stimulus gates every DMM reading. Every DMM reading is triggered by external digital I/O.

Define the digital I/O output trigger. Send a trigger to the external DMM.

Define the Series 3700A digital I/O output 3 trigger.

Define the Series 3700A digital I/O output 4 trigger.

Initialize the buffers to a variable buffer size.

Manual digital I/O triggering is required to ensure that the next open or close channel operation does not occur while the external DMM is measuring.

Send the scan status to the computer to wait for the scan to finish.

Wait up to 5 seconds for the last `CHANNEL_READY` (digital I/O 4) to be triggered.

Trigger measurement on an external DMM.

Ensure that the last VMC pulse cleared.

Wait up to 5 seconds for last DMM VMC pulse (digital I/O 3) to be triggered.

Software handshake to verify that Series 3700A scan completed properly. Wait for `scan_state()` to update to `SUCCESS`. Issues scan complete status to the computer.

Program code example

```
loadscript Ext_Trig_Dmm
display.screen = display.MAIN
channel.open("allslots")
channel.connectrule = 0

dmm.func = "dcvolts"
dmm.displaydigits = dmm.DIGITS_5_5
dmm.range = 10
dmm.nplc = 0.0005
dmm.rel.enable = dmm.OFF
dmm.autozero = dmm.OFF
dmm.autodelay = dmm.OFF
dmm.configure.set("dc_10V")
dmm.setconfig("1014", "dc_10V")

dmm.range = 100e-3
dmm.nplc = 0.0005
dmm.rel.enable = dmm.OFF
dmm.autozero = 0
dmm.autodelay = dmm.OFF
dmm.configure.set("dc_100mV")
dmm.setconfig("1012", "dc_100mV")

scan.create("1014, 1012")
scan.mode = scan.MODE_FIXED_ABR
scan.measurecount = 100
scan.scancount = 3
scan.trigger.arm.clear()
scan.trigger.channel.clear()
scan.trigger.sequence.clear()
scan.trigger.measure.clear()

digio.writebit(1, 1)
digio.writebit(2, 1)
digio.writebit(3, 1)
digio.writebit(4, 1)
digio.writebit(10, 1)

scan.trigger.arm.clear()
scan.trigger.arm.stimulus = 0
scan.trigger.channel.clear()
scan.trigger.channel.stimulus = 0
scan.bypass = scan.ON
```

```
digio.trigger[10].mode = digio.TRIG_RISING

scan.trigger.sequence.stimulus = scan.trigger.EVENT_CHANNEL_READY
scan.trigger.measure.stimulus = digio.trigger[10].EVENT_ID

digio.trigger[2].mode = digio.TRIG_FALLING
digio.trigger[2].pulsewidth = 5e-6
digio.trigger[3].stimulus = scan.trigger.EVENT_MEASURE_COMP
digio.trigger[3].mode = digio.TRIG_FALLING
digio.trigger[3].pulsewidth = 5e-6
digio.trigger[4].stimulus = scan.trigger.EVENT_CHANNEL_READY
digio.trigger[4].mode = digio.TRIG_FALLING
digio.trigger[4].pulsewidth = 5e-6

dmm_buffer_size = (scan.measurecount * scan.scancount * scan.stepcount)
buf_size_max = dmm_buffer_size
buf = dmm.makebuffer(buf_size_max)
buf.clear()
buf.appendmode = 1

chan_loop_cnt = scan.scancount * scan.stepcount
int_vmc_cnt = scan.measurecount -1

digio.trigger[1].clear()
digio.trigger[2].clear()
digio.trigger[3].clear()
digio.trigger[4].clear()
digio.trigger[10].clear()

scan.background(buf)

for z=1, chan_loop_cnt do
   print(scan.state())

digio.trigger[4].wait(5)
digio.trigger[2].assert()
digio.trigger[3].clear()

for y=1, int_vmc_cnt do
   digio.trigger[3].wait(5)
   digio.trigger[2].assert()
   end
end

while (scan_state() < scan.SUCCESS) do end
print(scan.state())
dmm.autozero = dmm.ON
endscript
```

DMM buffer statistics with interactive operation

Program code description

Sets the instrument to generate prompts in response to command messages and to send generated errors.

Assign the name `Generic_Noise_loop` to the script.

Display the main screen on the front panel.

Configure the DMM settings.

Prompt the user to enter the measurement count from the front panel within a range of 1000 to 50,000.

Create a buffer that can hold up to 10000 readings. stop = `dmm.measurecount()`.

Print the total time based on `for x=1, buf.n do value = buf.readings[x]`.

Create standard deviation numeric to string variable. Use `e` for exponential, `f` for floating point. The string variable is formatted with four leading digits and four digits after the decimal point.

Get buffer readings for use in Microsoft Excel.

Program code example

```
localnode.prompts = 1
localnode.showerrors = 1

loadscript Generic_Noise_loop

display.screen = display.MAIN

dmm.func = "fourwireohms"
dmm.displaydigits = dmm.DIGITS_7_5
dmm.range = 1
dmm.nplc = 0.0005
dmm.rel.enable = dmm.OFF
dmm.autozero = 0
dmm.linesync = 0
dmm.offsetcompensation = dmm.OFF
dmm.autodelay = dmm.OFF
dmm.measurecount = 1000

dmm.measurecount = display.prompt("00000", " Meas_Cnts", "Enter Meas_Count",
   dmm.measurecount, 1, 50000)
num_str7 = string.format(dmm.measurecount)
num_str9 = string.format(dmm.func)
display.clear()
display.settext("$BCalculating $N" .. num_str9.. " Noise "..num_str7.. " Rdg")
```

```
buf = dmm.makebuffer(10000)
delay(2.5)
display.screen=display.MAIN

buf.clear()
buf.appendmode = 1
hi_last_rdg = -1.2e8
lo_last_rdg = 1.2e8
sum_sq = 0
sum = 0
ave = 0
diff = 0
count = 1
dsply_cnt = 8
time = 0
timer.reset()
dmm.measure(buf)
time = timer.measure.t()
for x = 1,dmm.measurecount do value = buf.readings[x]
   if value > hi_last_rdg then hi_last_rdg = value end
      if value < lo_last_rdg then lo_last_rdg = value end
      sum = value + sum
      ave = sum /count
      diff = (value - ave ) * (value - ave )
      sum_sq = diff + sum_sq
      rdg_per_sec = (time / count)*1e6
      count = count + 1
      stdev = ( ( sum_sq ) / (count - 1) )^0.5
      peak_peak = (hi_last_rdg - lo_last_rdg)

num_str1 = string.format("%4.4e", stdev)
num_str2 = string.format("%4.4e", peak_peak)
num_str3 = string.format("%4.4e", ave)
num_str4 = string.format("%4.4e", buf.readings[x])
num_str8 = string.format(buf.units[1])

dsply_cnt = dsply_cnt + 1
   if dsply_cnt > 8 then
         display.clear()
         display.settext(num_str4.. " "  ..num_str8.. "$Npp=".. num_str2.. " Std="..
    num_str1)
         delay(0.125)
         dsply_cnt = 0
   end
end
print("finished Generic_Noise_AZ_Off () ")
 endscript
   for x = 1,buf.n do printbuffer(x,x,buf, buf.relativetimestamps)
end
```

Commonside ohm measurement with Model 3721

This example configures channels 38 and 39 in slot 1 to perform commonside four-wire ohm measurements on the 3721 card. Please refer to the *Series 3700A Switching and Control Cards Reference Manual* for the physical connections for a commonside ohms measurement.

Program code description

Create a script named `CommonSide4wOhms`.

Open all slots.

For the DMM:

- Set the DMM function to commonside ohms.
- Set autorange on.
- Set NPLC to 0.0005.
- Set autozero off.
- Set autodelay off.
- Name this DMM configuration `commonside4w` and set the configuration for measurements on channels 38 and 39 on slot 1.

For the scan:

- Initiate the scan to execute in the background.
- Backplane channels 1927 and 1928 are automatically closed with either channel 1038 or 1039.
- Measures two resistors in commonside ohms configuration.
- DMM makes 10 readings on each channel.

For the buffer:

- Make a buffer named `mybuffer` set to store up to 1000 readings.
- Clear the buffer.
- Start the scan and save the readings to `mybuffer`.
- Print the reading and relative time from start of scan. `mybuffer.timestamp` includes date and real time.

Program code example

```
loadscript CommonSide4wOhms

channel.open("allslots")

dmm.func = "commonsideohms"
dmm.autorange = dmm.ON
dmm.nplc = 0.0005
dmm.autozero = dmm.OFF
dmm.autodelay = dmm.OFF
dmm.configure.set("commonside4w")
dmm.setconfig("1038,1039","commonside4w")

scan.mode = scan.MODE_FIXED_ABR
scan.create("1038,1039")
scan.measurecount=10

mybuffer = dmm.makebuffer(1000)
mybuffer.clear()
mybuffer.appendmode = 1
scan.execute(mybuffer)
printbuffer(1, mybuffer.n, mybuffer,mybuffer.relativetimestamps)
endscript
```

Section 7

Next steps

In this section:

Next steps

This manual has prepared you to start using your new Series 3700A for your real-world applications. For sample applications, refer to the *Series 3700A Application Manual*, 3700AS-904-01.

For more detailed information about the Series 3700A, refer to the Keithley Instruments *Series 3700A Reference Manual*, part number 3700AS-901-01.

For information on Series 3700A cards, refer to the Keithley Instrument *Series 3700A Switching and Control Cards Reference Manual*.

Keithley Instruments
Corporate Headquarters • 28775 Aurora Road • Cleveland, Ohio 44139 • 1-800-833-9200 • tek.com/keithley

KEITHLEY
A Tektronix Company

07/2020