Section 9

Measuring thyristor DC characteristics

In this section:

Introduction

These examples demonstrate how to use the Model 2657A High Power System SourceMeter® instrument to characterize several DC characteristics of gated thyristors.

There are three examples in this section.

The first example performs measurements of peak forward off-state blocking voltage (V_{drm}) and its corresponding leakage current (I_{drm}). It can also be used to measure the peak reverse block voltage (V_{rrm}) and its corresponding leakage current (I_{rrm}).

The second example determines the holding current of the thyristor.

The third example determines the latching current of the thyristor.

Equipment required for this example

Equipment required:

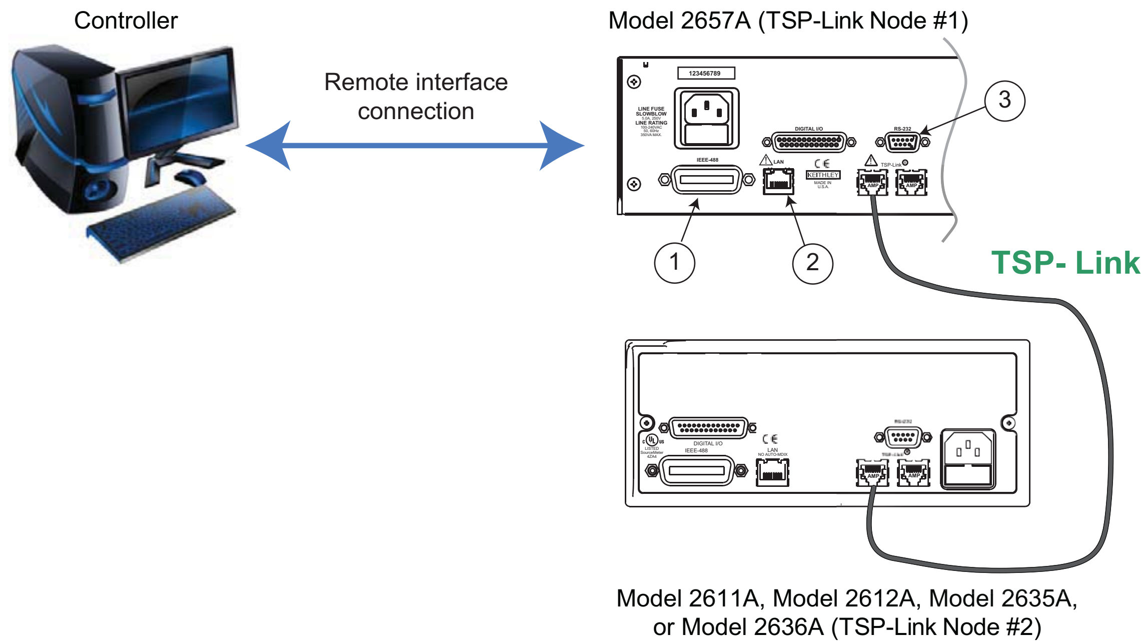
- One Model 2657A High Power System SourceMeter® instrument
- One Series 2600A System SourceMeter® Instrument (Model 2611A, Model 2612A, Model 2635A, or Model 2636A)
  - If needed, cables to connect the Series 2600A SourceMeter Instrument to the protection module
  - If needed, cables to connect the protection module to the test fixture
  - If a Model 2611A or Model 2612A is being used, one Model 2600-TRIAX adapter
- If you are not using a Model 8010 High Power Device Test Fixture, you need to use a Model 2657A-PM-200 Protection Module to ensure that the lower voltage Series 2600A SMU is protected if the device fails
- One GPIB or Ethernet cable to connect the Model 2657A to a computer
- One gated thyristor (for example, SCR or triac) enclosed in a safe test fixture
- Appropriate cabling to connect the SMUs to the test fixture, such as the Keithley Instruments Model 8010

Set up communication

The communication setup is illustrated in the following diagram. This application can be run using any of the supported communication interfaces for the instruments.

For additional detail about remote communications, see "Communications interfaces" in the Model 2657A Reference Manual.

Figure 38: Remote interface and TSP-Link communications setup



Item	Description	Qty	Notes
1	IEEE-488 connection	1	GPIB. Model 2657A is IEEE Std 488.1 compliant.
2	CA-180-3A LAN connection	1	Model 2657A is LXI version 1.4 Core 2011 compliant. It supports TCP/IP and complies with IEEE Std 802.3 (ethernet). 10 or 100 Mbps.
3	RS-232 connection	1	

NOTE

For the first example, no TSP-Link connections are required because only the Model 2657A at TSP-Link node 1 is used.

Device connections

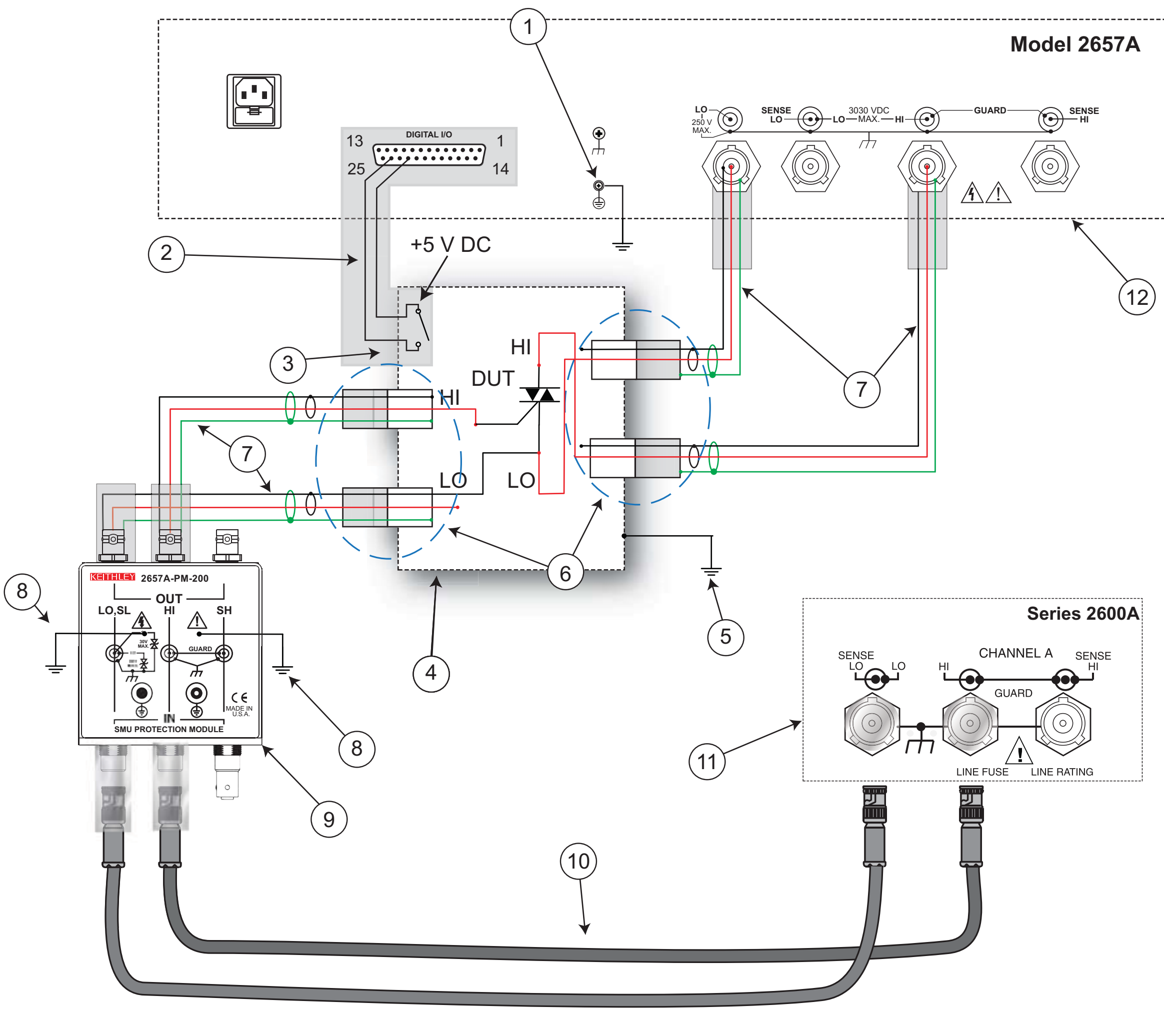
Refer to the following figures to connect the thyristor in a safe test fixture.

See Safe configuration and test setup (on page 2-1) for information on safely connecting the Model 2657A to a device under test (DUT).

CAUTION

If a device under test fails, high voltage may be present at a terminal to which Series 2600A instruments are connected. This could damage a Series 2600A instrument. To prevent damage to the Series 2600A instrument, use a Model 2657A-PM-200 Protection Module. Failure to use a protection module could result in equipment damage.

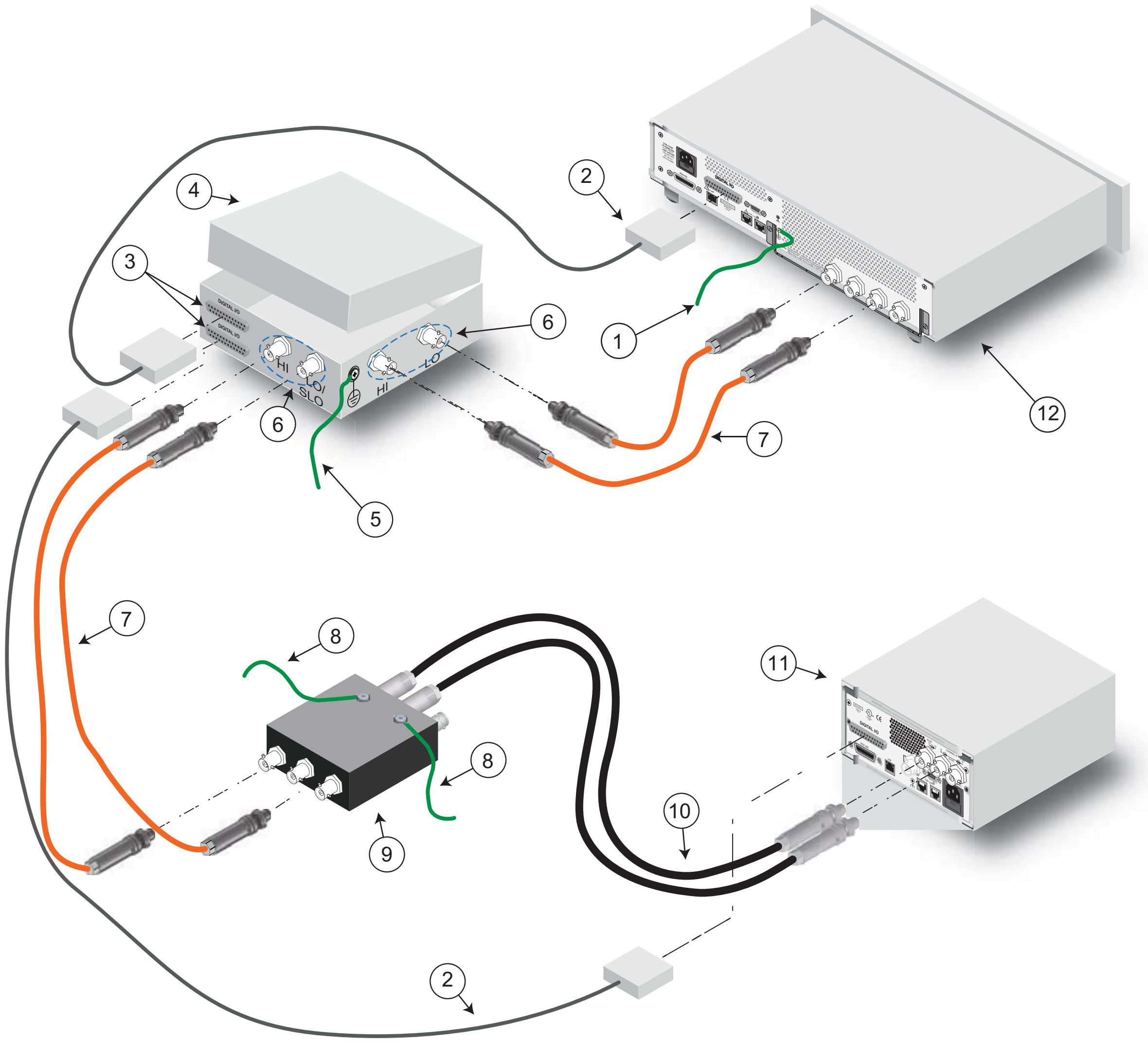
Figure 39: Schematic for gated thyristor application example



Item	Description	Qty	Notes
1	Model 2657A protective earth (safety ground)	1	Additional connections for redundant protective earth may be required. Keithley Instrument's Model CA-568 is a protective earth (safety ground) cable assembly.
2	Interlock connection	1	Model 2657A digital I/O; pin 24 (INT) and pin 22 (5 VDC) connected to the test fixture lid switch. Keithley Instrument's Model 7709-308 is a 25-pin interlock male connector that can be used for custom connections. Interlock switch is shown in the disengaged (lid open) position.
3	Test fixture interlock switch connection	1	
4	Interlocked metal safety enclosure	1	A safety enclosure with an interlock that has a normally-open (NO) switch.
5	Test fixture protective earth (safety ground)	1	Redundant grounds may be required for specific test setups.
6	Model HV-CA-571-3 High-Voltage Triaxial Panel-Mount Connector to Unterminated Cable Assembly	4	See Using high-voltage triaxial connectors (on page 2-7).
7	Model HV-CA-554 High-Voltage Triaxial Cable	4	
8	Protective earth (safety ground) for the Model 2657A-PM-200	2	
9	Model 2657A-PM-200 Protection Module	1	Refer to documentation provided with Model 2657A-PM-200 for more information.
10	Model 7078-TRX Triaxial Cable Assembly	2	
11	Series 2600A System SourceMeter® Instrument	1	Model 2611A, Model 2612A, Model 2635A, or Model 2636A.
12	Model 2657A	1	

If you are using a Model 8010 High Power Test Fixture, see the Interconnect Reference Guide drawing "Three-terminal DUT with a Model 2657A". Also see the Model 8010 User's Manual for Model 2657A, Model 2611A, and Model 2612A connections.

Figure 40: Connections for gated thyristor application example



Vdrm and Idrm thyristor measurements

NOTE

The example code is designed to be run from Test Script Builder or TSB Embedded. It can also be run from other programming environments, such as Microsoft® Visual Studio® or National Instruments LabVIEW™. However, you may need to make changes to the example code to use other programming environments.

This example performs the V_{drm} and I_{drm} measurements, and may also be used to determine V_{rrm} and I_{rrm}. The Model 2657A is connected from anode to cathode and performs a voltage sweep while capturing current measurements at each point of the sweep.

This example uses a TSP script to perform the measurement. The script includes two separate functions for configuring the System SourceMeter Instrument and returning the raw current and voltage readings from the reading buffer.

The script is written using Test Script Processor (TSP) functions rather than as a single block of inline code. TSP functions are similar to functions in other programming languages, such as Microsoft® Visual C® or Visual Basic®. They must be called before the code in them is executed. Because of this, running the script alone will not execute the test. To execute the test, you need to run the script to load the functions into test script memory. You then call the functions.

Refer to the documentation for Test Script Builder or TSB Embedded (on page 4-4) for directions on how to run scripts and enter commands using the instrument console.

Example code

```
--[[
   offVoltLeakI(startV, stopV, numSteps, measDelay, measRange, iLimit, numNPLC)

   Description: This function can be used to determine Vdrm and Idrm or
   Vrrm and Irrm of a thyristor. In this function, the Model 2657A will
   perform a voltage sweep on the anode and measure the corresponding
   leakage current at each step. As no voltage or current bias is required
   on the gate terminal of the thyristor, this function can be used with
   gated and non-gated devices.

   To determine Vdrm and Idrm, connect the Model 2657A from anode to
   cathode of the device, and program positive start and stop voltages
   for the sweep.

   To determine Vrrm and Irrm, connect the Model 2657A from the anode to
   the cathode of the device, and program negative start and stop voltages
   for the sweep.

   Parameters:
      startV: Starting drain voltage (Vds).
      stopV: Final drain voltage (Vds).
      numSteps: Number of points in the drain voltage sweep.
      measDelay: Measurement delay.
      measRange: Current measurement range for the drain current measurements.
      iLimit: Current limit (compliance) for the drain current.
      numNPLC: Integration time in number of power line cycles.

   Example usage:

   for Vdrm and Idrm:
   offVoltLeakI(0, 800, 501, 0.010, 100e-6, 1e-3, 1)

   for Vrrm and Irrm:
   offVoltLeakI(0, -800, 501, 0.010, 100e-6, 1e-3, 1)

--]]
```

```
function offVoltLeakI(startV, stopV, numSteps, measDelay, measRange, iLimit,
    numNPLC)

   --Initialize SMU.
   reset()
   errorqueue.clear()
   status.reset()

   --Configure reading buffers.
   smua.nvbuffer1.clear()
   smua.nvbuffer1.appendmode = 1
   smua.nvbuffer1.collecttimestamps = 1

   smua.nvbuffer2.clear()
   smua.nvbuffer2.appendmode = 1
   smua.nvbuffer2.collecttimestamps = 1

   --Configure source parameters for the drain SMU.
   smua.source.func = smua.OUTPUT_DCVOLTS
   smua.source.levelv = 0
   smua.source.limiti = iLimit
   smua.source.rangev = stopV

   --Configure measurement parameters for the drain SMU.
   smua.measure.rangei = measRange
   smua.measure.nplc = numNPLC
   smua.measure.delay = measDelay
   smua.measure.autozero = smua.AUTOZERO_ONCE

   step = (stopV - startV)/(numSteps - 1)
   voltage = startV

   --Run the test.
   smua.source.output = 1


   for i = 1, numSteps do
      smua.measure.iv(smua.nvbuffer1, smua.nvbuffer2)
      smua.source.levelv = voltage + step
      voltage = voltage + step
   end

   --Run the test.
   smua.source.levelv = 0
   smua.source.output = 0

   printData()

end
```

```
function printData()

   if  smua.nvbuffer1.n == 0 then
      print("No reading in buffer")
   else
      print("Timestamps\tVoltage\tCurrent")
      for i = 1, smua.nvbuffer1.n do
         print(string.format("%g\t%g\t%g", smua.nvbuffer1.timestamps[i],
   smua.nvbuffer2.readings[i], smua.nvbuffer1.readings[i]))
      end
   end
end
```

Example usage

The function in this script allows updates to the test parameters without rewriting or re-running the script. To run the test, call the `offVoltLeakI()` function, passing in the appropriate values for test parameters.

offVoltLeakI() parameters		
Parameter	**Units**	**Description**
`startV`	volts	Start value for the voltage sweep performed at the anode
`stopV`	volts	Stop value for the voltage sweep performed at the anode
`numSteps`	not applicable	Number of steps in the voltage sweep
`measDelay`	seconds	Measurement delay prior to making a measurement (allows for source and system settling)
`measRange`	amps	Current range on which to perform the leakage current measurements
`iLimit`	amps	Current limit (compliance) for the Model 2657A
`numNPLC`	not applicable	Integration time, specified as the number of power line cycles

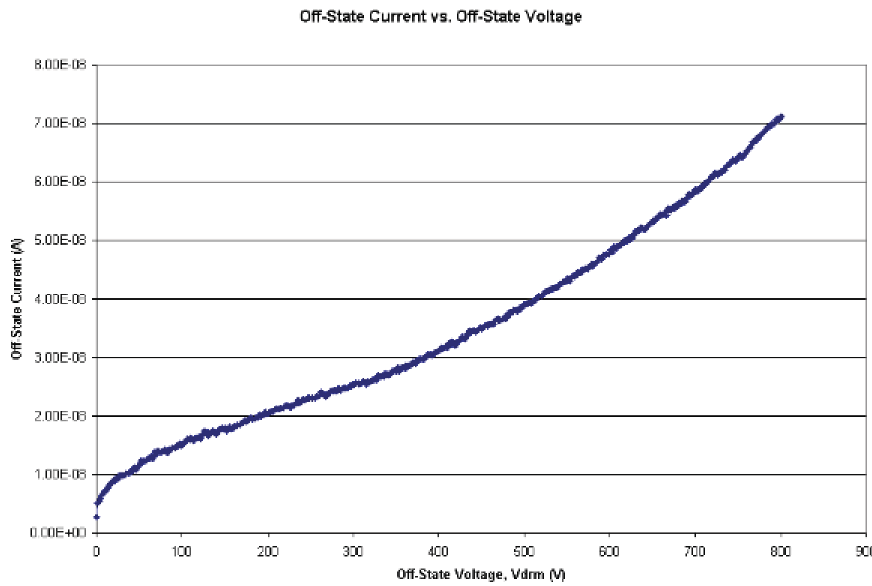
An example of how to call this function is shown here:

```
--for Vdrm and Idrm:
offVoltLeakI(0, 800, 501, 0.01, 100e-6, 1e-3, 1)
```

This function call programs the Model 2657A to sweep from 0 V to +800 V in 501 steps. After each source step, a 10 ms delay occurs and then the Model 2657A makes a 1 NPLC current measurement on the 100 μA range. The current limit is set to 1 mA. All current and voltage measurements are automatically printed to the communication interface at the conclusion of the test.

The data that results is shown in the following graphic.

Figure 41: Thyristor measurement sample output



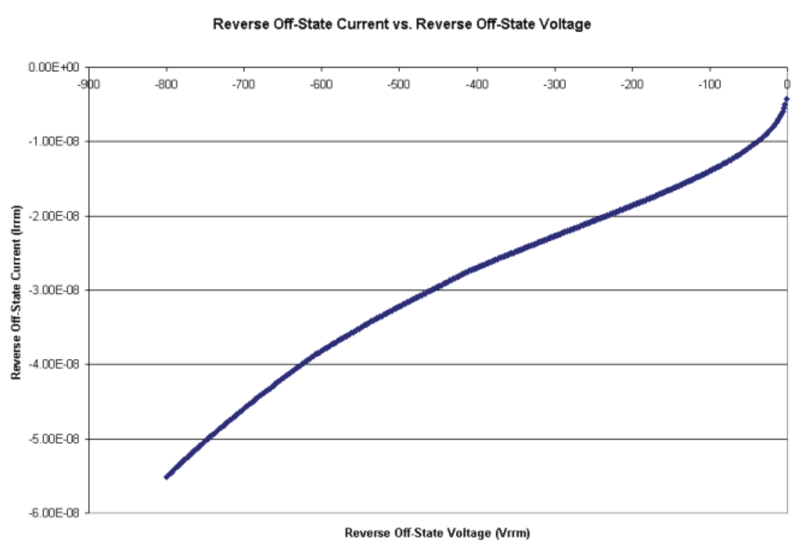
Another example of how to call this function is shown here:

```
 --for Vrrm and Irrm:
offVoltLeakI(0, -800, 501, 0.010, 100e-6, 1e-3, 1)
```

This function call programs the Model 2657A to sweep from 0 V to –800 V in 501 steps. After each source step, a 10 ms delay occurs and then the Model 2657A makes a 1 NPLC current measurement on the 100 µA range. The current limit is set to 1 mA. All current and voltage measurements are automatically printed to the communication interface at the conclusion of the test.

An example of the output of this test is shown in the graphic below.

Figure 42: Sample output Vrrm



Determine the holding current of a thyristor

This example determines the holding current of a thyristor. The holding current is the minimum DC current at which the thyristor turns off after the gate signal is removed.

Two SourceMeter instruments are required for this example. One SourceMeter instrument triggers the thyristor to turn on by supplying a current to the thyristor's gate terminal. Once triggered, the gate signal is removed. The Model 2657A applies a voltage bias to the thyristor's anode with an initial current limit greater than or equal to the specified latching current of the thyristor. This ensures that it stays in the on-state after the gate signal is removed. The current limit is gradually decreased until the device turns off. This final current is the holding current.

Example code

```
--[[
   holdingCurr(gateBiasCurr, gateVoltLimit, gateOnTime, anodeBiasVolt,
    anodeLatchingCurr, anodeCurrStep, numNPLC)

   Description: This function can be used to determine Ih of a thyristor.
   Two SMUs are required for this function. One SMU (node[2].smua) is
   connected to the gate terminal and is used to trigger the device.
   The Model 2657A (smua) is connected to the anode and is applying
   a voltage bias.
   After the device is turned on, the gate signal is removed and the
   current limit of the Model 2657A is varied until the device turns off.
   The current at which the device turns off is the holding current.

   Parameters:
      gateBiasCurr = Current to apply to the gate to trigger the device on.
       Should be greater than or equal to the gate trigger current.
      gateVoltLimit = Voltage limit for the gate SMU. Should be greater
       than or equal to the gate trigger voltage.
      gateOnTime = Number of seconds to apply the gate trigger
       (single pulse event).
      anodeBiasVolt = Model 2657A applies this voltage to the anode terminal.
      anodeLatchingCurr = Specified latching current of the device.
        This will be the initial current limit of the Model 2657A.
      anodeCurrStep = How much to vary the anode current in order to
       arrive at the holding current.
      numNPLC = Integration time in number of power line cycles.

   Example usage:

   holdingCurr(0.05, 10e-3, 2, 12, 0.06, 0.005, 1)

--]]
```

```
function holdingCurr(gateBiasCurr, gateOnTime, gateVoltLimit, anodeBiasVolt,
    anodeLatchingCurr, anodeCurrStep, numNPLC)

   --Initialize instruments and clear errors and status registers.
   tsplink.reset()
   reset()
   errorqueue.clear()
   status.reset()

   --Configure gate SMU.
   node[2].smua.source.func = node[2].smua.OUTPUT_DCAMPS
   node[2].smua.source.rangei = gateBiasCurr
   node[2].smua.source.leveli = gateBiasCurr
   node[2].smua.source.limitv = gateVoltLimit
   node[2].smua.source.delay = 0


   --Configure anode SMU source parameters.
   smua.source.func = smua.OUTPUT_DCVOLTS
   smua.sense = smua.SENSE_LOCAL
   smua.source.rangev = anodeBiasVolt
   smua.source.levelv = anodeBiasVolt
   smua.source.limiti = anodeLatchingCurr
   --[[
       This is the initial current limit to ensure that the device stays
       on after the gate.
   --]]
```

```
   --Configure anode SMU measure parameters.
   smua.measure.rangei = anodeLatchingCurr
   smua.measure.nplc = numNPLC
   smua.measure.autozero = smua.AUTOZERO_ONCE

   numSteps = ((anodeLatchingCurr - 0) / (anodeCurrStep)) + 1

   if anodeCurrStep >= anodeLatchingCurr then
      print("Wrong value: The current step cannot be greater than or equal to the
   expected latching current.")
   else
      smua.source.output = 1
      node[2].smua.source.output = 1
      delay(gateOnTime)
      node[2].smua.source.output = 0

      for i = 1, numSteps do
         smua.measure.i()
         cmplCheck = smua.source.compliance
         if cmplCheck == false then
            break
         else
            smua.source.limiti = anodeLatchingCurr - (i*anodeCurrStep)
         end

      end

      --Complete the test and output the result.
      smua.source.output = 0
      finalIh = smua.source.limiti
      print("Holding Current (Amps): ", finalIh)
   end

end
```

Example usage

The functions in this script allow updates to the test parameters without rewriting or re-running the script. To run the test, call the `holdingCurr()` function, passing in the appropriate values for test parameters.

holdingCurr() parameters		
Parameter	**Units**	**Description**
`gateBiasCurr`	amps	Gate current to apply with the Series 2600A SourceMeter instrument.
`gateVoltLimit`	volts	Voltage limit (compliance) for the Series 2600A connected to the thyristor's gate terminal.
`gateOnTime`	seconds	Amount of time to apply the gate signal.
`anodeBiasVolt`	volts	Anode voltage provided by the Model 2657A.
`anodeLatchingCurr`	amps	Specified latching current of the thyristor. This is also the initial current limit (compliance) of the Model 2657A connected to the anode terminal. The latching current is typically more than or equal to the holding current. Ensure that this value is larger than holding current for this test.
`anodeCurrStep`	amps	The step size of the change in anode current.
`numNPLC`	not applicable	Integration time, specified as number of power line cycles.

An example of how to call this function is shown here:

```
holdingCurr(0.05, 10e-3, 2, 12, 0.06, 0.005, 1)
```

This function call uses the Series 2600A SourceMeter Instrument to apply 50 mA for 2 s to trigger the thyristor to turn on. Meanwhile, the Model 2657A applies 12 V to the anode with an initial current limit of 60 mA. Since the device is in the on-state, the Model 2657A should be in current limit. The current limit is decreased in 5 mA increments. The program monitors the current limit bit of the status register in order to determine when the Model 2657A returns to normal voltage sourcing conditions, which indicates that the device has turned off. This final programmed current limit is the holding current.

The result of the function call for this example is:

```
Holding Current (Amps):     2.40000e-02
```

Determine the latching current of a thyristor

This example determines the latching current of a thyristor. The latching current is the minimum DC anode current at which the device remains in the on-state after the gate signal is removed.

Two SourceMeter instruments are required for this example. The Series 2600A SourceMeter Instrument provides a repetitive square-wave like signal to the gate to trigger and re-trigger the device. The Model 2657A supplies current to the thyristor's anode. Each time the gate is removed, the program checks the Model 2657A voltage limit to determine if the device is on. If the device has not stayed on, the anode current is increased and the device is re-triggered. This process continues until the thyristor remains on even after the gate signal is removed. The first anode current at which this occurs is the latching current.

Example code

```
--[[
   latchingCurr(gateBiasCurr, gateVoltLimit, gateOnTime, anodeStartCurr,
    anodeMaxCurr, anodeStepCurr, anodeVoltLimit, numNPLC)

   Description: This function can be used to determine Ilatch of a
   thyristor. Two SMUs are required for this function. One SMU
   (node[2].smua) is connected to the gate terminal and is used to
   trigger the device. The Model 2657A (smua) is connected to the anode
   and is used to source the current.
   The gate is triggered while a current bias is applied to the anode.
   The gate current is removed and measurements are made to verify that the
   device is on. If the device is off, the current at the anode is
   increased and the gate is re-triggered. This continues until the
   device remains on even after the gate signal is removed.
   The current at which the device remains on is the latching current.

   In order to complete this task, custom list sweeps are configured on
   the gate and anode SMUs.

   Parameters:
      gateBiasCurr = Current to apply to the gate to trigger the device on.
       Should be greater than or equal to the gate trigger current.
      gateVoltLimit = Voltage limit for the gate SMU. Should be greater
       than or equal to the gate trigger voltage.
      gateOnTime = Number of seconds to apply the gate trigger
       (single pulse event).
      anodeStartCurr = Start value for the anode current sweep.
       Typically slightly less than or equal to the holding current value.
      anodeMaxCurr = Stop value for the anode current sweep. May not reach
       this value, but this is the maximum value to source.
      anodeStepCurr = Step value for the anode current sweep.
      anodeVoltLimit = Voltage limit for the anode SMU. Should be equal
       to Vdrm.
      numNPLC = Integration time in number of power line cycles.

   Example usage:

   latchingCurr(0.060, 10, 1e-3, 0.020, 0.120, 0.005, 600, 0.01)

--]]
```

```
function latchingCurr(gateBiasCurr, gateVoltLimit, gateOnTime, anodeStartCurr,
   anodeMaxCurr, anodeStepCurr, anodeVoltLimit, numNPLC)

   --Initialize instruments and clear errors and status registers.
   tsplink.reset()
   reset()
   errorqueue.clear()
   status.reset()

   --Configure gate SMU.
   node[2].smua.source.func = node[2].smua.OUTPUT_DCAMPS
   node[2].smua.source.rangei = gateBiasCurr
   node[2].smua.source.leveli = 0
   node[2].smua.source.limitv = gateVoltLimit
   node[2].smua.source.delay = gateOnTime

   --Configure anode SMU source parameters.
   smua.source.func = smua.OUTPUT_DCAMPS
   smua.sense = smua.SENSE_LOCAL
   smua.source.rangei = anodeMaxCurr
   smua.source.leveli = 0
   smua.source.limitv = anodeVoltLimit

   --Configure anode SMU measurement parameters.
   smua.measure.delay = 0.005
   --Must have delay if you want reasonable anode measurements.

   smua.measure.rangev = anodeVoltLimit
   smua.measure.nplc = numNPLC
   smua.measure.autozero = smua.AUTOZERO_ONCE

   smua.nvbuffer1.clear()
   smua.nvbuffer2.clear()
   smua.nvbuffer1.appendmode = 1
   smua.nvbuffer2.appendmode = 1
   smua.nvbuffer1.collecttimestamps = 1
   smua.nvbuffer2.collecttimestamps = 1

   --Create custom sweeps for anode and gate SMUs.
   numSwpPts = 2*((anodeMaxCurr - anodeStartCurr)/(anodeStepCurr)) + 1

   gateSwp = {}
   anodeSwp = {}

   for i = 1, numSwpPts do
      if math.mod(i,2) == 0 then
         gateSwp[i] = 0
      else
         gateSwp[i] = gateBiasCurr
      end
   end

   for j = 1, numSwpPts, 2 do
      anodeSwp [j] = anodeStartCurr + ((j-1)/2)*anodeStepCurr
      anodeSwp[j+1] = anodeStartCurr + ((j-1)/2)*anodeStepCurr
   end
```

```
   smua.source.output = 1
   node[2].smua.source.output = 1

   for k = 1, numSwpPts, 2 do
      smua.source.leveli = anodeSwp[k]
      node[2].smua.source.leveli = gateSwp[k]
      node[2].smua.source.leveli = gateSwp[k+1]
      smua.measure.iv(smua.nvbuffer1,smua.nvbuffer2)
      checkCmpl = smua.source.compliance
   --[[
      If in compliance, then device is off. If out of compliance, then
      device is on and we have reached latching current.
   --]]

      if checkCmpl == false then
         finalIL = anodeSwp[k]
         break
      end
   end

   node[2].smua.source.output = 0
   smua.source.output = 0
   print("Latching current(amps): ", finalIL)
   print("Raw data follows:")
   printData()
end
```

Example usage

The function in this script allows updates to the test parameters without rewriting or re-running the script. To run the test, call the `latchingCurr()` function, passing in the appropriate values for test parameters.

latchingCurr() parameters		
Parameter	**Units**	**Description**
`gateBiasCurr`	amps	Gate current to apply with the Series 2600A SourceMeter instrument.
`gateVoltLimit`	volts	Voltage limit (compliance) for the Series 2600A connected to the gate terminal of the thyristor.
`gateOnTime`	seconds	Amount of time to apply the gate signal.
`anodeStartCurr`	amps	Initial anode current provided by the Model 2657A. Should be much less than the specified latching current.
`anodeMaxCurr`	amps	Maximum anode current to apply with the Model 2657A.
`anodeStepCurr`	amps	Step size of the current sweep occurring at the anode of the thyristor.
`anodeVoltLimit`	volts	Voltage limit (compliance) of the Model 2657A.
`numNPLC`	not applicable	Integration time, specified as the number of power line cycles.

An example of how to call this function is shown here:

```
latchingCurr(0.060, 10, 1e-3, 0.020, 0.120, 0.005, 600, 0.01)
```

This call uses the Series 2600A SourceMeter instrument to apply 60 mA for 1 ms to trigger the thyristor to turn on. Meanwhile, the Model 2657A applies 20 mA to the anode terminal. The gate signal is removed by programming the Model 2600A to force zero amps.

The program monitors the voltage limit on the Model 2657A. If the voltage limit is indicated, the device is off. The Model 2657A then increases the supplied anode current and the device is re-triggered. This process is repeated until the thyristor remains on. This final programmed current limit is the latching current.

The result of the function call for this example is:

```
Latching Current (Amps): 2.90000e-02)
```