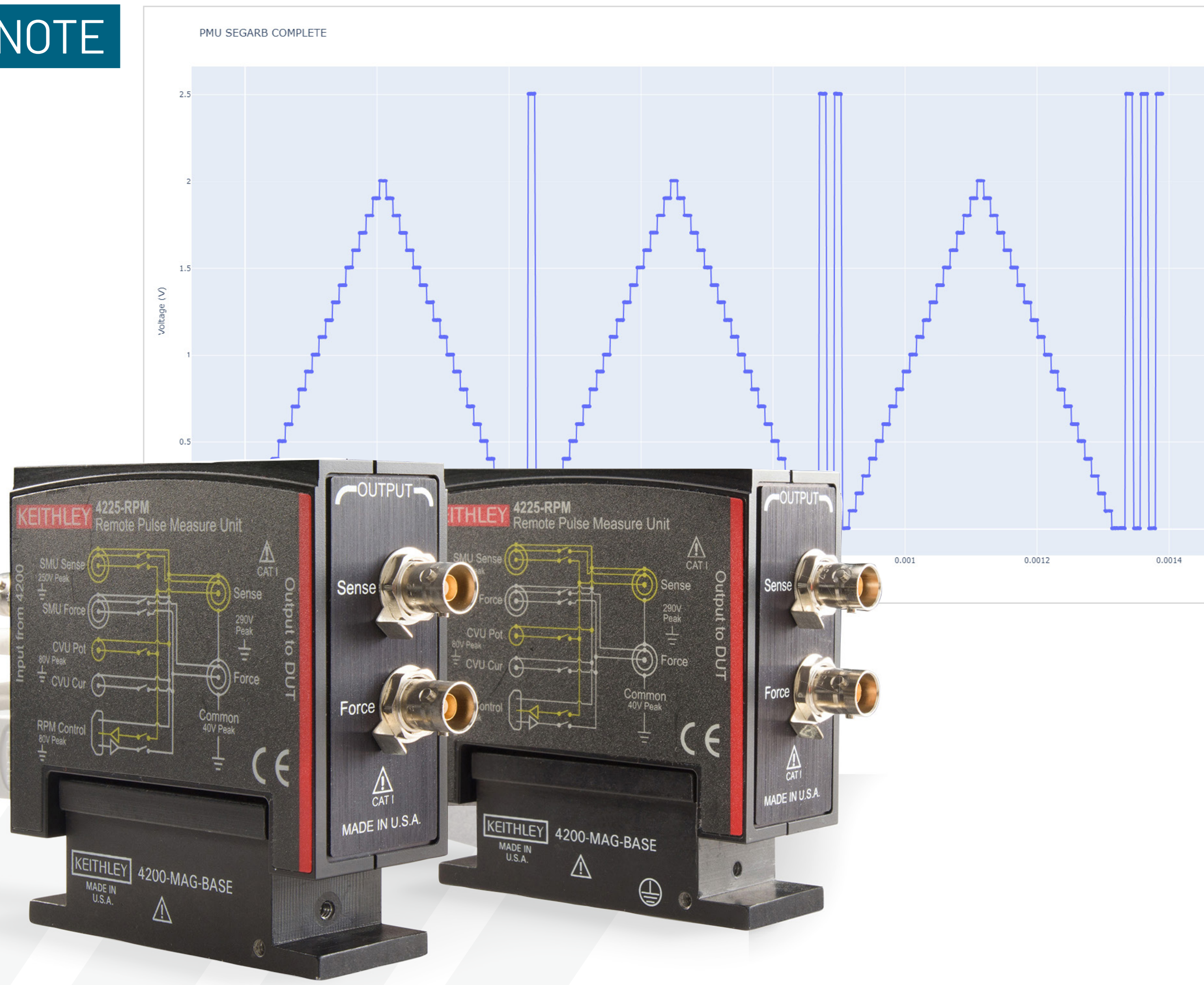
Tektronix®

Pulse IV Test Automation with the Keithley 4225-PMU Pulse Measure Unit

Using Remote Control in Keithley External Control Interface (KXCI) for High-Speed Measurements

APPLICATION NOTE



KEITHLEY
A Tektronix Company

Introduction

Ultra-fast IV sourcing and measuring is important to many semiconductor applications, including nonvolatile memory, power device characterization, CMOS, reliability and MEMS devices. These semiconductor tests are made using the Keithley 4225-PMU Pulse Measure Unit (PMU), a 2-channel high-speed voltage source and time-based current measurement module for the 4200A-SCS Parameter Analyzer. The PMU has three modes of ultra-fast IV source and measure modes: Pulse IV, Waveform Capture and Segment ARB™. These three modes are illustrated in **Figure 1**.

Using pulse IV signals to characterize devices instead of DC signals makes it possible to reduce the effects of self-heating or to minimize current drift. Waveform capture mode, or transient IV, outputs high-speed voltage pulses and measures the current and voltage response in the time domain. Pulsed sourcing, or Segment ARB, can be used to stress a device using an AC signal during reliability cycling or in a multi-level waveform mode to program and erase memory devices.

The easy-to-use interactive Clarius software comes with the 4200A-SCS and has a test library that includes many PMU applications. However, some PMU measurements may need to be part of an automated test controlled by an external computer. In these cases, remote control of the PMU is necessary.

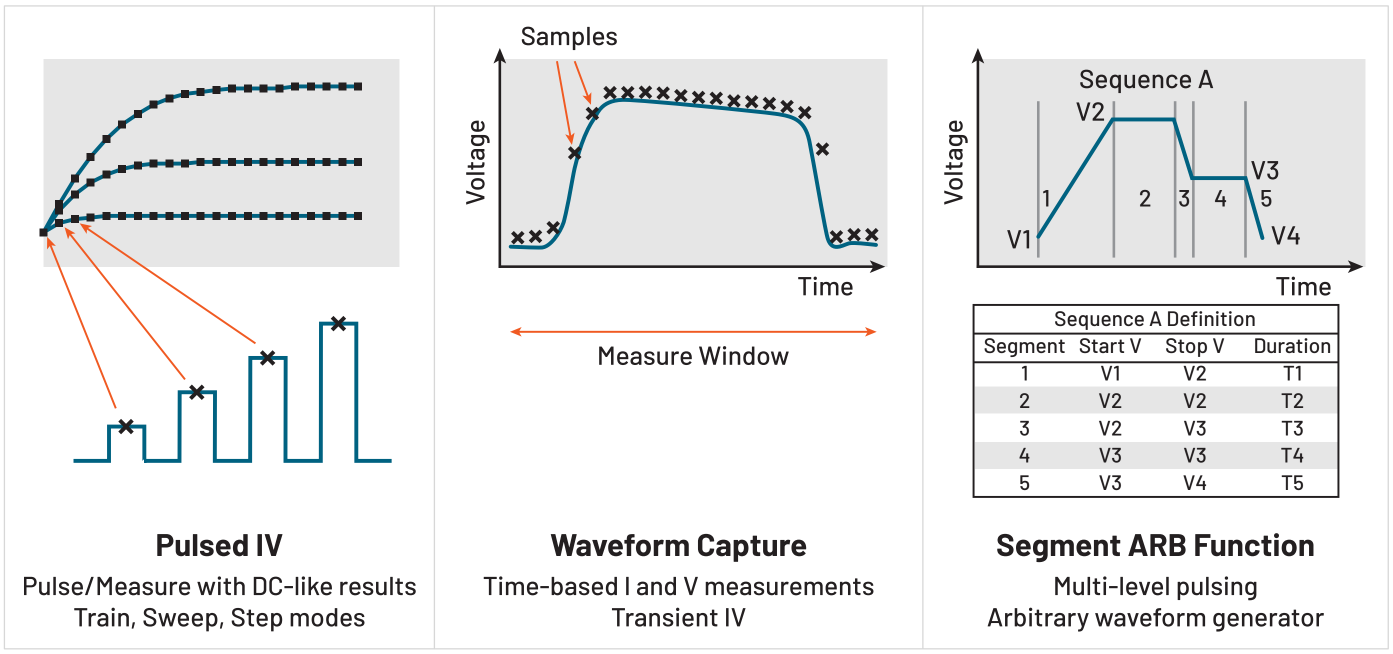
The Keithley External Control Interface (KXCI) enables remote control of the instrument modules in the 4200A-SCS by sending commands from a computer. The controlling computer can be connected to the 4200A-SCS through GPIB or ethernet to remotely send the KXCI commands using a coding environment.

Starting with Clarius V1.13, KXCI commands for high-speed IV sourcing and measuring have been added to the Clarius software for remote controlling the PMU and the 4220-PGU Pulse Generator Unit (PGU). This enables test automation outside of the Clarius interface software. These commands, along with programming examples for each mode, are explained in this application note. Tables of PMU KXCI commands are listed in Appendix A and B at the end of this application note.

Getting Started Using KXCI

With the Keithley External Control Interface (KXCI), an external computer is used to directly control the SMUs, CVUs, PMUs and PGUs in the 4200A-SCS Parameter Analyzer. Each of the modules has its own command set and can be used to configure a variety of different tests.

The first step to using the KXCI software tool is to close the Clarius software and open the Keithley Configuration Tool (KCon), located on the desktop, and configure settings for either GPIB or ethernet communications. After these settings are configured, close KCon and open the KXCI



Sequence A Definition

Segment	Start V	Stop V	Duration
1	V1	V2	T1
2	V2	V2	T2
3	V2	V3	T3
4	V3	V3	T4
5	V3	V4	T5

Figure 1. Ultra-fast IV modes of the 4225-PMU Pulse Measure Unit.

application. Once you have KXCI opened, you can start sending commands to the modules in the 4200A-SCS. More detailed configuration information about using KXCI and the command sets for all the instruments are in the manual *Model 4200A-SCS KXCI Remote Control Programming*.

Basic information on getting started using KXCI with Python is in the application note *Controlling the 4200A-SCS Parameter Analyzer Using KXCI and Python 3*. This application note describes using Visual Studio code with Python 3 and NI VISA to control the 4200A-SCS using KXCI commands.

Examples of Ultra-Fast IV: Pulse IV, Waveform Capture and Segment ARB

This section includes KXCI programming examples of the three modes of ultra-fast IV: Pulse IV, Waveform Capture and Segment ARB.

Pulse IV

Pulse IV refers to any test with a pulsed voltage source and a corresponding high-speed, time-based current measurement that provides DC-like results. The voltage and current measurements are an average, or spot mean, of readings taken in a predefined measurement window on the pulse. The user defines the parameters of the pulse, including the pulse width, period, rise/fall times and amplitude.

The following pulse IV programming example generates an IV sweep on a 1 kohm resistor. As illustrated in **Figure 2**, one end of the resistor is connected to the center conductor of the coax cable (HI) of PMU CH1 and the other side of the resistor is connected to PMU common (LO), or the outside shield of the coax cable.

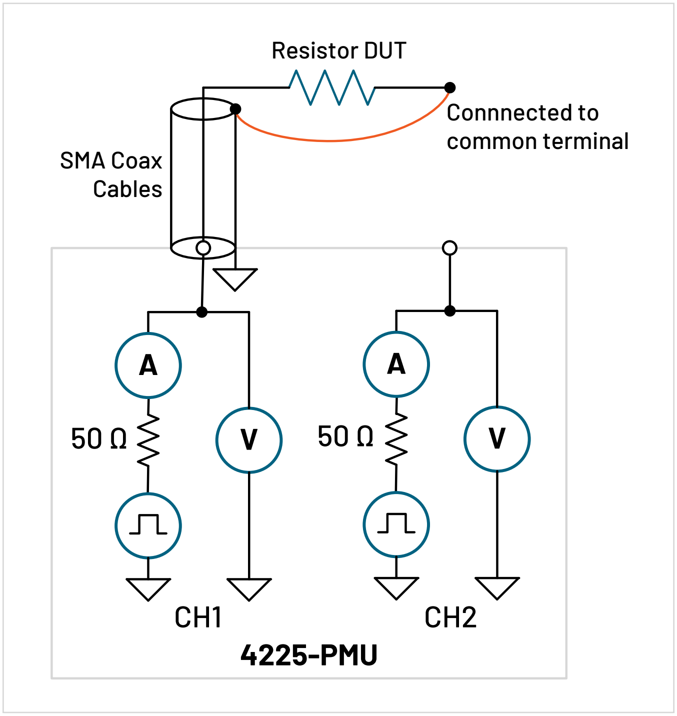
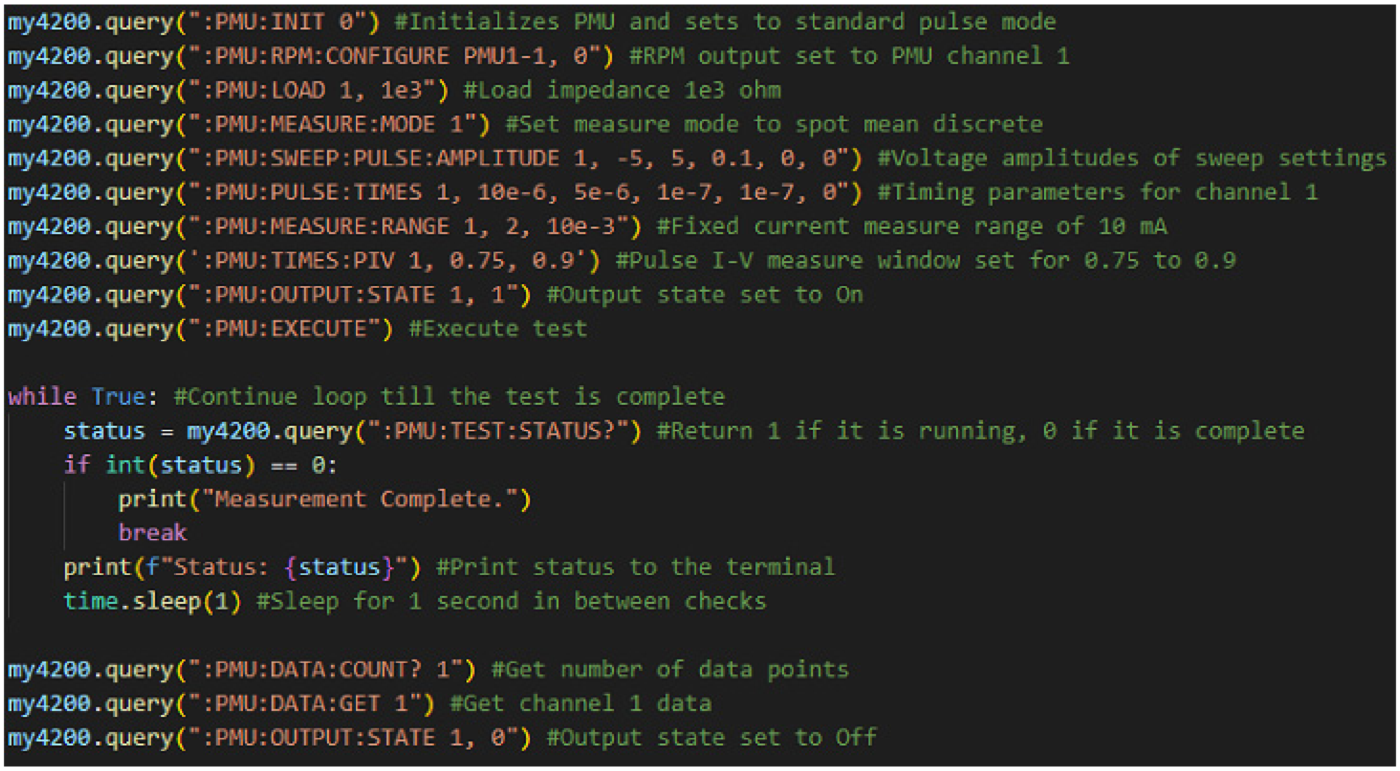


Figure 2. Connecting PMU CH1 to a resistor.

Part of a Python script to create a pulse IV sweep is shown in **Figure 3**. The code includes the amplitude sweep parameters (including start V = –5 V, stop V = 5 V, step size = 0.1 V and base V = 0 V) and the pulse timing parameters (period = 10e–6 s, pulse width = 5e –6 s, rise and fall times = 1e –7 s).

Other defined parameters include the measure window and the measure range. The measure window command (:PMU:TIMES:PIV) is the percent range on the pulse top where the average, or spot mean, is derived. In this example, the measure window is between 0.75 and 0.9 of the top of the pulse. For each pulse, one reading is derived. The current measure range (:PMU:MEASURE:RANGE) is fixed at 10 mA, but autorange or limited autorange can also be used. Using autorange enables the PMU to find the best current range and is useful for devices that have a large change in current during a voltage sweep, such as a diode.



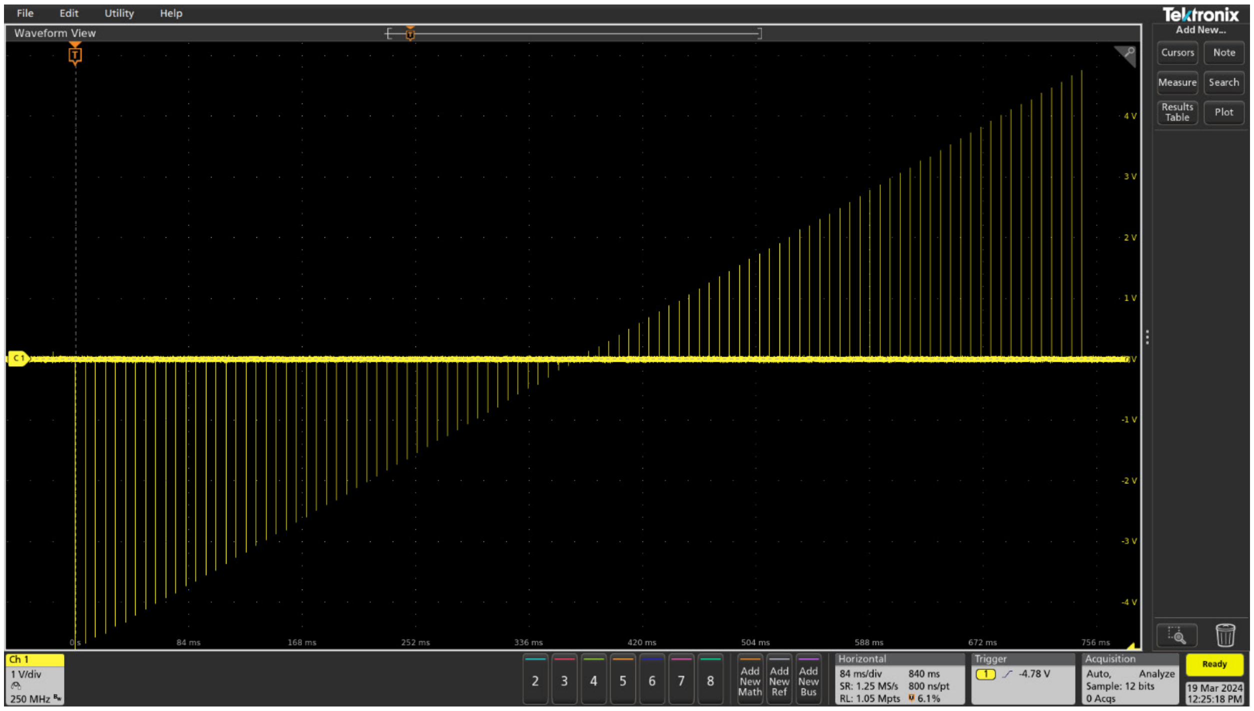
```
my4200.query(":PMU:INIT 0") #Initializes PMU and sets to standard pulse mode
my4200.query(":PMU:RPM:CONFIGURE PMU1-1, 0") #RPM output set to PMU channel 1
my4200.query(":PMU:LOAD 1, 1e3") #Load impedance 1e3 ohm
my4200.query(":PMU:MEASURE:MODE 1") #Set measure mode to spot mean discrete
my4200.query(":PMU:SWEEP:PULSE:AMPLITUDE 1, -5, 5, 0.1, 0, 0") #Voltage amplitudes of sweep settings
my4200.query(":PMU:PULSE:TIMES 1, 10e-6, 5e-6, 1e-7, 1e-7, 0") #Timing parameters for channel 1
my4200.query(":PMU:MEASURE:RANGE 1, 2, 10e-3") #Fixed current measure range of 10 mA
my4200.query(':PMU:TIMES:PIV 1, 0.75, 0.9') #Pulse I-V measure window set for 0.75 to 0.9
my4200.query(":PMU:OUTPUT:STATE 1, 1") #Output state set to On
my4200.query(":PMU:EXECUTE") #Execute test

while True: #Continue loop till the test is complete
    status = my4200.query(":PMU:TEST:STATUS?") #Return 1 if it is running, 0 if it is complete
    if int(status) == 0:
        print("Measurement Complete.")
        break
    print(f"Status: {status}") #Print status to the terminal
    time.sleep(1) #Sleep for 1 second in between checks

my4200.query(":PMU:DATA:COUNT? 1") #Get number of data points
my4200.query(":PMU:DATA:GET 1") #Get channel 1 data
my4200.query(":PMU:OUTPUT:STATE 1, 0") #Output state set to Off
```

Figure 3. Python code to generate a pulse IV sweep.

Once the code is executed, the PMU outputs a pulse IV sweep from –5 V to 5 V in 0.1 mV steps. **Figure 4** shows the scope capture from the Tektronix MSO5 Series Oscilloscope of the 101 pulses in the sweep. Only the spot mean of each of these pulses is derived and used in the IV measurement of the resistor.



Ch 1
1 V/div
250 MHz

Horizontal
84 ms/div 840 ms
SR: 1.25 MS/s 800 ns/pt
RL: 1.05 Mpts 6.1%

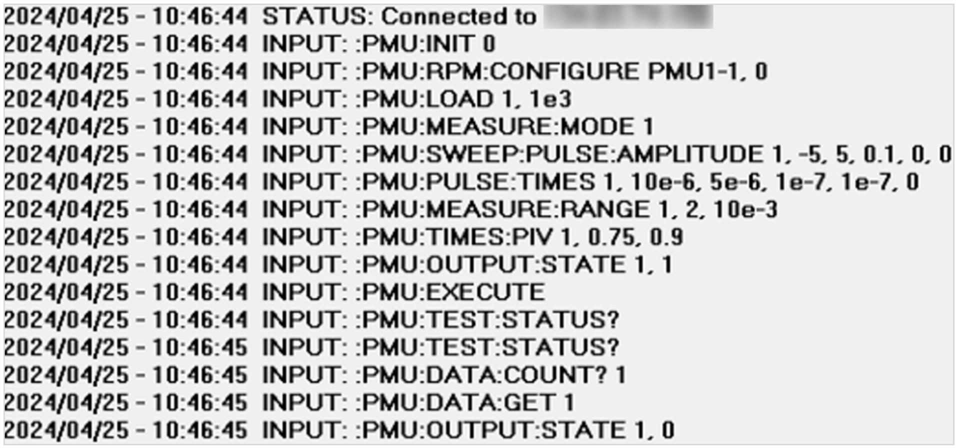
Trigger
1 -4.78 V

Acquisition
Auto, Analyze
Sample: 12 bits
0 Acqs

19 Mar 2024
12:25:18 PM

Figure 4. Scope screen capture of pulse IV sweep.

Once the code is executed using the :PMU:EXECUTE command, all the commands sent to the PMU are logged in the KXCI console, as shown in **Figure 5**, along with any messages or errors. Also listed in the KXCI console are the commands sent that are used to retrieve the data. The :PMU:TEST:STATUS? command determines if the sweep is finished executing. The :PMU:DATA:COUNT? command is used to determine how many readings are stored in the data buffer. Finally, the :PMU:DATA:GET command retrieves the data from the buffer.



```
2024/04/25 - 10:46:44  STATUS: Connected to [redacted]
2024/04/25 - 10:46:44  INPUT: :PMU:INIT 0
2024/04/25 - 10:46:44  INPUT: :PMU:RPM:CONFIGURE PMU1-1, 0
2024/04/25 - 10:46:44  INPUT: :PMU:LOAD 1, 1e3
2024/04/25 - 10:46:44  INPUT: :PMU:MEASURE:MODE 1
2024/04/25 - 10:46:44  INPUT: :PMU:SWEEP:PULSE:AMPLITUDE 1, -5, 5, 0.1, 0, 0
2024/04/25 - 10:46:44  INPUT: :PMU:PULSE:TIMES 1, 10e-6, 5e-6, 1e-7, 1e-7, 0
2024/04/25 - 10:46:44  INPUT: :PMU:MEASURE:RANGE 1, 2, 10e-3
2024/04/25 - 10:46:44  INPUT: :PMU:TIMES:PIV 1, 0.75, 0.9
2024/04/25 - 10:46:44  INPUT: :PMU:OUTPUT:STATE 1, 1
2024/04/25 - 10:46:44  INPUT: :PMU:EXECUTE
2024/04/25 - 10:46:44  INPUT: :PMU:TEST:STATUS?
2024/04/25 - 10:46:45  INPUT: :PMU:TEST:STATUS?
2024/04/25 - 10:46:45  INPUT: :PMU:DATA:COUNT? 1
2024/04/25 - 10:46:45  INPUT: :PMU:DATA:GET 1
2024/04/25 - 10:46:45  INPUT: :PMU:OUTPUT:STATE 1, 0
```

Figure 5. Executed PMU KXCI commands of pulse IV sweep as shown in the KXCI console.

Once the data is retrieved, the current can be plotted as a function of voltage using any plotting tool, as shown in **Figure 6**. This plot was created by an additional Python tool that allows for visualization of the data return. Notice there is one point plotted for each point in the sweep.

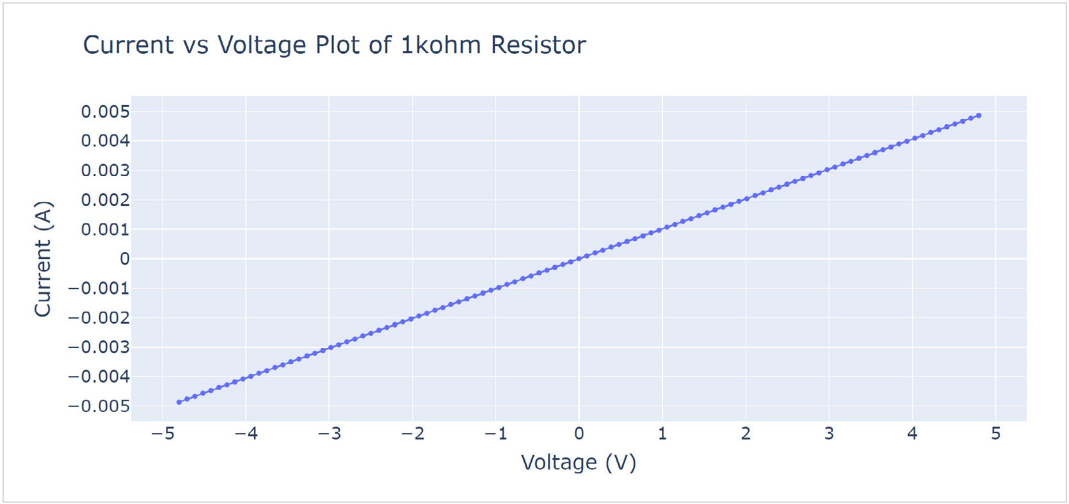


Figure 6. Results of pulse IV sweep on a 1 kohm resistor.

Waveform Capture

The waveform capture, or transient IV, mode outputs high speed voltage pulses and measures the resulting current and voltage transients in the time domain.

The next example uses the waveform capture mode of the PMU to show the time-based response of the drain current and drain voltage of a MOSFET. **Figure 7** shows the connections between the two channels of the PMU to the three terminals of the MOSFET. CH1 outputs a single 2 V pulse to the gate. CH2 outputs a 1 V pulse to the drain and captures the transient response of the drain current and voltage. The source terminal of the MOSFET is connected to Common or the outside shell of the coax cable.

Python code that outputs voltage pulses on PMU CH1 (gate) and PMU CH2 (drain) and measures the resulting drain current and voltage on PMU CH2 is shown in **Figure 8**. In this example, both channels are connected to the device, so they both need to be configured. However, commands to configure measurements are only sent to PMU CH2, since only PMU CH2 will return data.

The :PMU:PULSE:TRAIN command configures the pulse base and amplitude voltage for each channel. In this case, CH1 outputs a single 2 V pulse and CH2 outputs a 1 V pulse. The :PMU:PULSE:TIMES command sets the timing parameters on each channel (period = 1e-6 s, pulse width = 5e-7 s, rise and fall times = 1e-7 s and delay time = 1e-7 s).

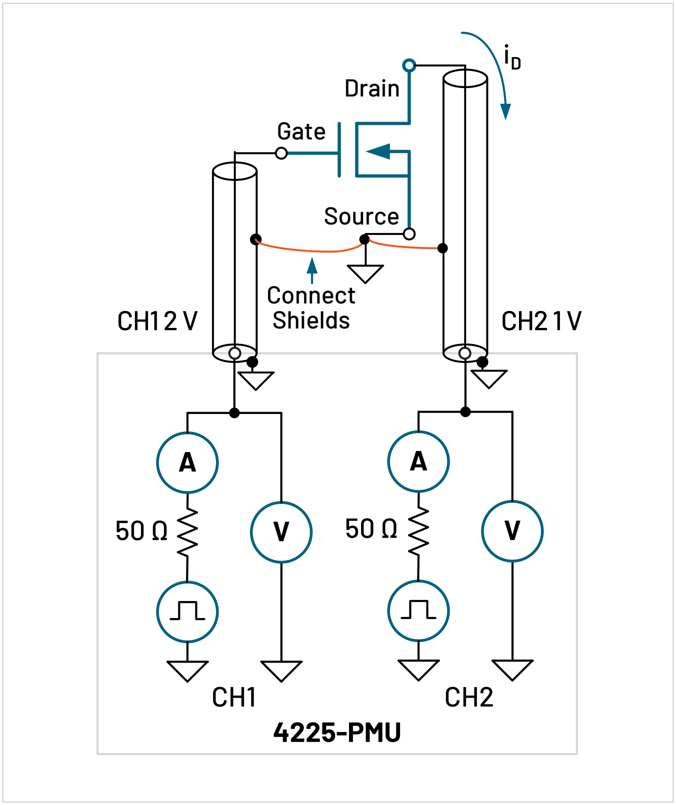
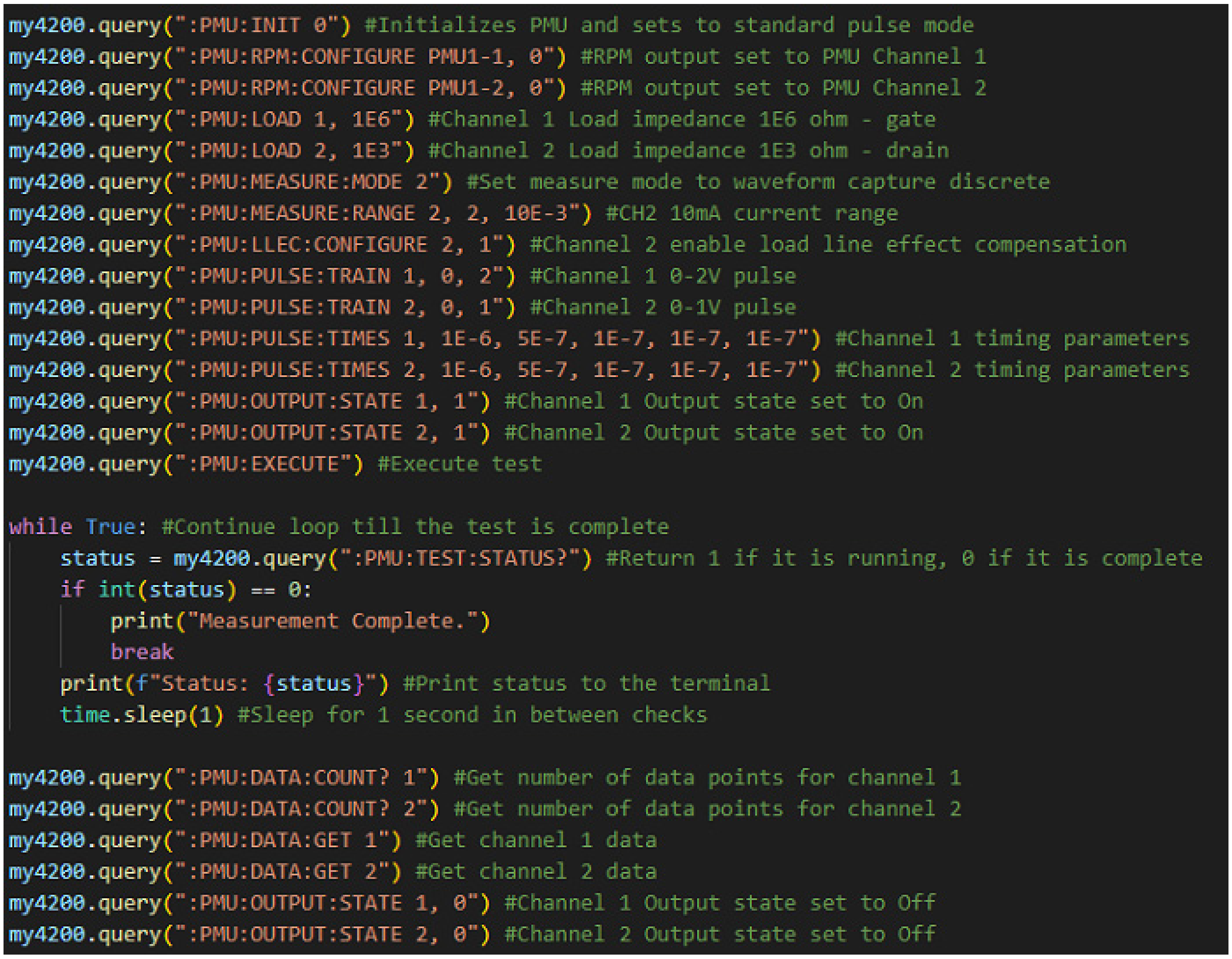


Figure 7. Connections from the two channels of the PMU to a three-terminal MOSFET.

The measurements are made with the Load Line Effect Compensation (LLEC) feature enabled (:PMU:LLEC:CONFIGURE 2, 1) on CH2. LLEC employs a mathematical algorithm that compensates for the voltage drop across the 50 ohm output impedance of the PMU and the voltage drop across the lead resistance and connections to the DUT.



```
my4200.query(":PMU:INIT 0") #Initializes PMU and sets to standard pulse mode
my4200.query(":PMU:RPM:CONFIGURE PMU1-1, 0") #RPM output set to PMU Channel 1
my4200.query(":PMU:RPM:CONFIGURE PMU1-2, 0") #RPM output set to PMU Channel 2
my4200.query(":PMU:LOAD 1, 1E6") #Channel 1 Load impedance 1E6 ohm - gate
my4200.query(":PMU:LOAD 2, 1E3") #Channel 2 Load impedance 1E3 ohm - drain
my4200.query(":PMU:MEASURE:MODE 2") #Set measure mode to waveform capture discrete
my4200.query(":PMU:MEASURE:RANGE 2, 2, 10E-3") #CH2 10mA current range
my4200.query(":PMU:LLEC:CONFIGURE 2, 1") #Channel 2 enable load line effect compensation
my4200.query(":PMU:PULSE:TRAIN 1, 0, 2") #Channel 1 0-2V pulse
my4200.query(":PMU:PULSE:TRAIN 2, 0, 1") #Channel 2 0-1V pulse
my4200.query(":PMU:PULSE:TIMES 1, 1E-6, 5E-7, 1E-7, 1E-7, 1E-7") #Channel 1 timing parameters
my4200.query(":PMU:PULSE:TIMES 2, 1E-6, 5E-7, 1E-7, 1E-7, 1E-7") #Channel 2 timing parameters
my4200.query(":PMU:OUTPUT:STATE 1, 1") #Channel 1 Output state set to On
my4200.query(":PMU:OUTPUT:STATE 2, 1") #Channel 2 Output state set to On
my4200.query(":PMU:EXECUTE") #Execute test

while True: #Continue loop till the test is complete
    status = my4200.query(":PMU:TEST:STATUS?") #Return 1 if it is running, 0 if it is complete
    if int(status) == 0:
        print("Measurement Complete.")
        break
    print(f"Status: {status}") #Print status to the terminal
    time.sleep(1) #Sleep for 1 second in between checks

my4200.query(":PMU:DATA:COUNT? 1") #Get number of data points for channel 1
my4200.query(":PMU:DATA:COUNT? 2") #Get number of data points for channel 2
my4200.query(":PMU:DATA:GET 1") #Get channel 1 data
my4200.query(":PMU:DATA:GET 2") #Get channel 2 data
my4200.query(":PMU:OUTPUT:STATE 1, 0") #Channel 1 Output state set to Off
my4200.query(":PMU:OUTPUT:STATE 2, 0") #Channel 2 Output state set to Off
```

Figure 8. Python code to output 500 ns gate and drain pulses and measure drain I and V.

Once the :PMU:EXECUTE command is used to start the test, you can use the :PMU:TEST:STATUS? command to check if the test finished. In waveform capture mode, the test will return the voltage, current, time and status from each channel that was configured to make measurements, in this case, CH2. **Figure 9** shows the transient drain voltage and current of the MOSFET.

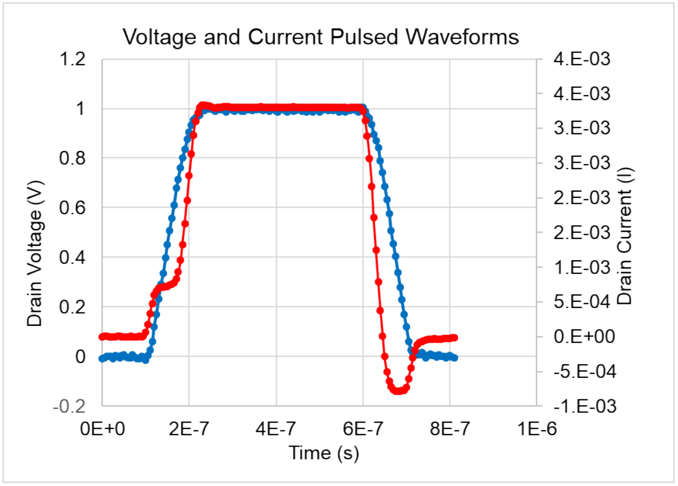


Figure 9. Transient drain current and voltage responses of a MOSFET.

Segment ARB Waveform

Each channel of the PMU can be configured to output its own Segment ARB waveform composed of user-defined line segments, up to 2048. There are separate commands for time interval, start and stop voltage values, start and stop measure window values, output trigger and output relay state (open or closed). Each command defines that parameter for all the segments in the waveform. Both spot mean and sample mode measurements are supported for each segment.

Segment ARB sequences are constructed using the :PMU:SARB commands, which are listed in Appendix B. The :PMU:SARB commands define all parts of each segment, such as the start voltage, stop voltage, time and measure type. For example, the time of every segment is defined sequentially using the :PMU:SARB:SEQ:TIME command and the starting voltages of each segment are defined by the PMU:SARB:SEQ:STARTV command. The use of these commands is demonstrated in the following example.

This example will output a Segment ARB sequence that sources a 35 V, 1e-3 s pulse and then a –35 V, 1e –3 s pulse on PMU CH1. PMU CH2 forces 0 V and measures the resulting current and voltage. The circuit diagram is shown in **Figure 10**.

Forcing voltage on one side of the resistor and measuring current on the other side is called the low-side measurement technique and is used for ultra-fast high impedance measurements. This technique avoids errors due to leakage current and longer settling time. Further information on this method can be found in the Keithley application note, *Making Low Current Pulse I-V Measurements with the 4225-PMU Pulse Measure Unit and 4225-RPM Remote/Preamplifier Switch Modules*.

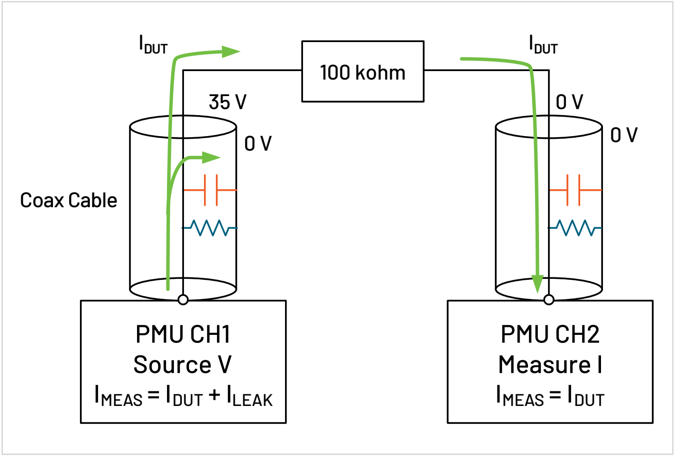


Figure 10. Circuit diagram showing the Low-side Measurement Technique.

PMU CH1 is configured to output the Segment ARB sequence shown in **Figure 11**. (NOTE: The timing axis is not to scale.) This sequence has nine segments that generate a +35 V pulse for 1e-3 s and then a –35 V pulse for 1e–3 s. There are 1e–3 s segments for rise and fall times. Each of the nine segments has a unique time, start voltage and stop voltage, as configured by the following commands:

Parameter	Command (CH1, Sequence 1)
Time	:PMU:SARB:TIME: 1, 1, 10e-6, 10e-6, 1e-3, 10e-6, 1e-3, 10e-6, 1e-3, 10e-6, 10e-6
Start Voltage	:PMU:SARB:SEQ:STARTV 1, 1, 0, 0, 35, 35, 0, 0, -35, -35, 0
Stop Voltage	:PMU:SARB:SEQ:STOPV 1, 1, 0, 35, 35, 0, 0, -35, -35, 0, 0

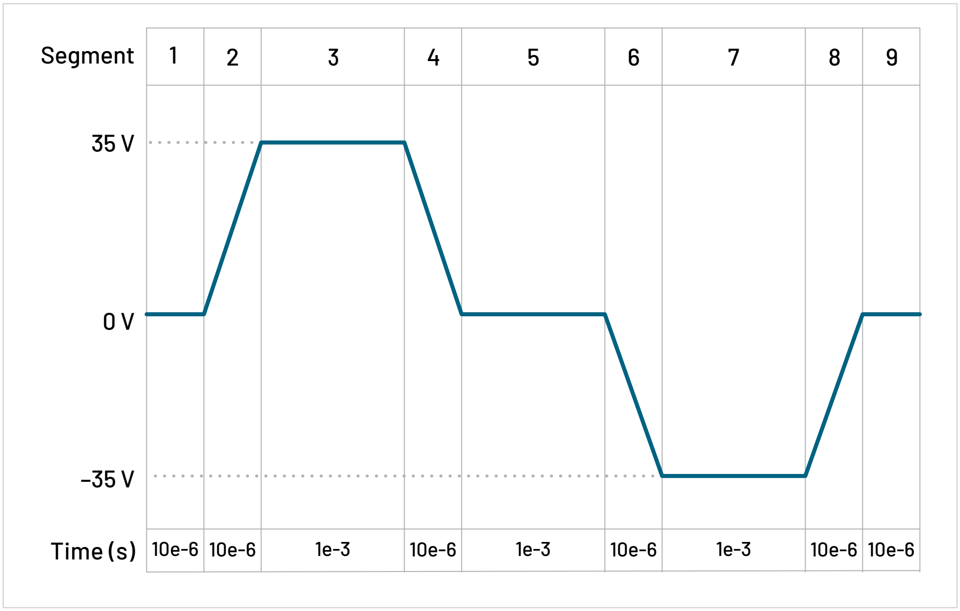
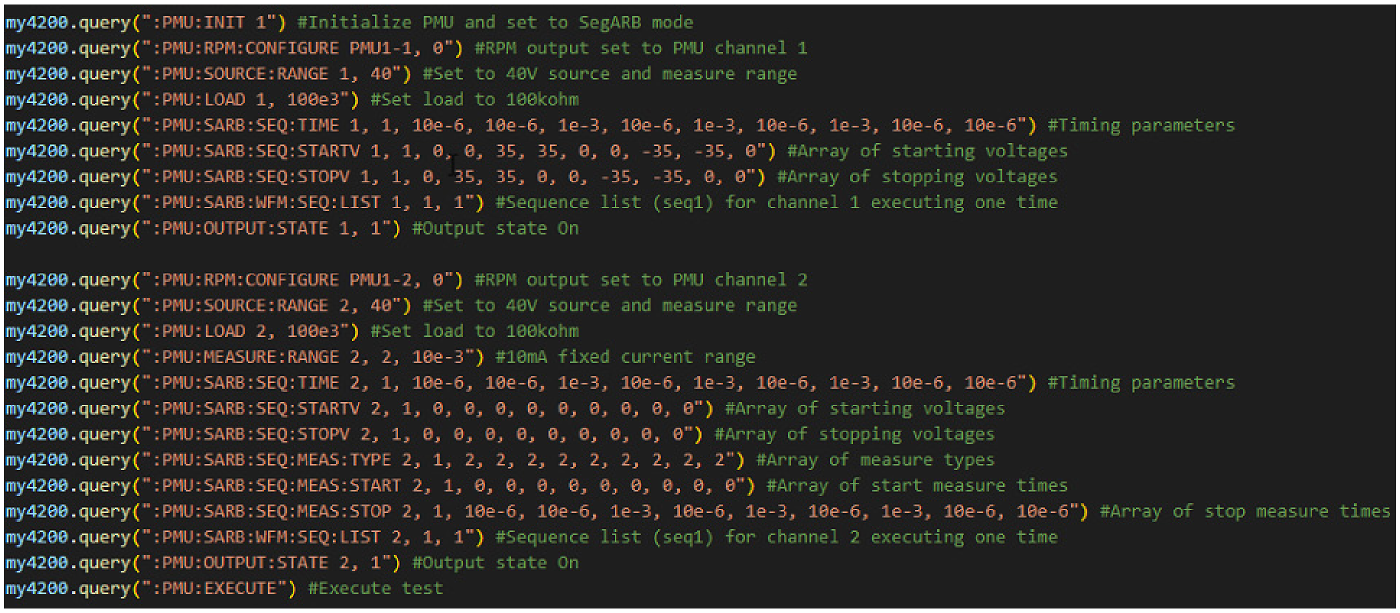


Figure 11. Segment ARB sequence for CH1. (NOTE: Time not to scale.)

PMU CH1 is configured to output only. PMU CH2 is configured to force 0 V and measure the waveform capture current and voltage on each segment. The start and stop measure times are also configured on CH2. The Python code used to control CH1 and CH2 is listed in **Figure 12**.

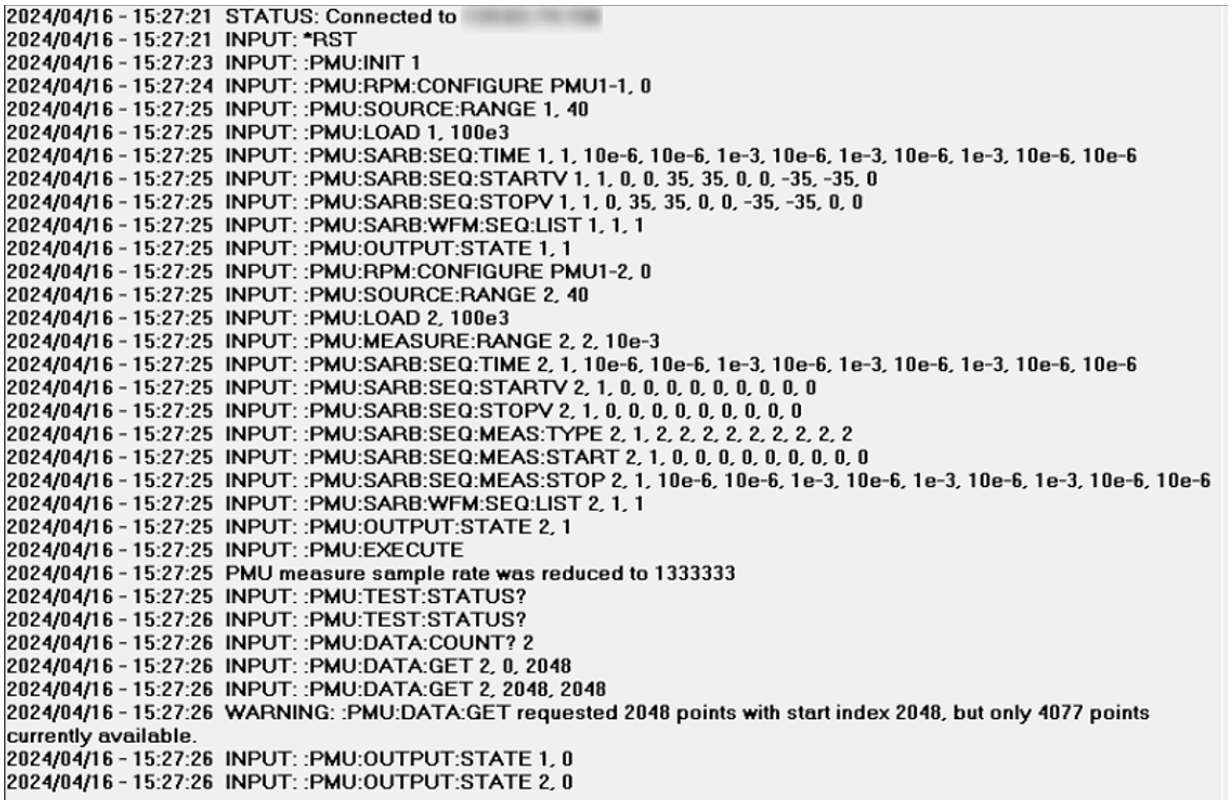


```
my4200.query(":PMU:INIT 1") #Initialize PMU and set to SegARB mode
my4200.query(":PMU:RPM:CONFIGURE PMU1-1, 0") #RPM output set to PMU channel 1
my4200.query(":PMU:SOURCE:RANGE 1, 40") #Set to 40V source and measure range
my4200.query(":PMU:LOAD 1, 100e3") #Set load to 100kohm
my4200.query(":PMU:SARB:SEQ:TIME 1, 1, 10e-6, 10e-6, 1e-3, 10e-6, 1e-3, 10e-6, 1e-3, 10e-6, 10e-6") #Timing parameters
my4200.query(":PMU:SARB:SEQ:STARTV 1, 1, 0, 0, 35, 35, 0, 0, -35, -35, 0") #Array of starting voltages
my4200.query(":PMU:SARB:SEQ:STOPV 1, 1, 0, 35, 35, 0, 0, -35, -35, 0, 0") #Array of stopping voltages
my4200.query(":PMU:SARB:WFM:SEQ:LIST 1, 1, 1") #Sequence list (seq1) for channel 1 executing one time
my4200.query(":PMU:OUTPUT:STATE 1, 1") #Output state On

my4200.query(":PMU:RPM:CONFIGURE PMU1-2, 0") #RPM output set to PMU channel 2
my4200.query(":PMU:SOURCE:RANGE 2, 40") #Set to 40V source and measure range
my4200.query(":PMU:LOAD 2, 100e3") #Set load to 100kohm
my4200.query(":PMU:MEASURE:RANGE 2, 2, 10e-3") #10mA fixed current range
my4200.query(":PMU:SARB:SEQ:TIME 2, 1, 10e-6, 10e-6, 1e-3, 10e-6, 1e-3, 10e-6, 1e-3, 10e-6, 10e-6") #Timing parameters
my4200.query(":PMU:SARB:SEQ:STARTV 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0") #Array of starting voltages
my4200.query(":PMU:SARB:SEQ:STOPV 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0") #Array of stopping voltages
my4200.query(":PMU:SARB:SEQ:MEAS:TYPE 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2") #Array of measure types
my4200.query(":PMU:SARB:SEQ:MEAS:START 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0") #Array of start measure times
my4200.query(":PMU:SARB:SEQ:MEAS:STOP 2, 1, 10e-6, 10e-6, 1e-3, 10e-6, 1e-3, 10e-6, 1e-3, 10e-6, 10e-6") #Array of stop measure times
my4200.query(":PMU:SARB:WFM:SEQ:LIST 2, 1, 1") #Sequence list (seq1) for channel 2 executing one time
my4200.query(":PMU:OUTPUT:STATE 2, 1") #Output state On
my4200.query(":PMU:EXECUTE") #Execute test
```

Figure 12. PMU KXCI code to generate +35 V and –35 V pulses.

Once the code is executed, the commands are logged in the KXCI console, as shown in **Figure 13**.



```
2024/04/16 - 15:27:21  STATUS: Connected to [redacted]
2024/04/16 - 15:27:21  INPUT: *RST
2024/04/16 - 15:27:23  INPUT: :PMU:INIT 1
2024/04/16 - 15:27:24  INPUT: :PMU:RPM:CONFIGURE PMU1-1, 0
2024/04/16 - 15:27:25  INPUT: :PMU:SOURCE:RANGE 1, 40
2024/04/16 - 15:27:25  INPUT: :PMU:LOAD 1, 100e3
2024/04/16 - 15:27:25  INPUT: :PMU:SARB:SEQ:TIME 1, 1, 10e-6, 10e-6, 1e-3, 10e-6, 1e-3, 10e-6, 1e-3, 10e-6, 10e-6
2024/04/16 - 15:27:25  INPUT: :PMU:SARB:SEQ:STARTV 1, 1, 0, 0, 35, 35, 0, 0, -35, -35, 0
2024/04/16 - 15:27:25  INPUT: :PMU:SARB:SEQ:STOPV 1, 1, 0, 35, 35, 0, 0, -35, -35, 0, 0
2024/04/16 - 15:27:25  INPUT: :PMU:SARB:WFM:SEQ:LIST 1, 1, 1
2024/04/16 - 15:27:25  INPUT: :PMU:OUTPUT:STATE 1, 1
2024/04/16 - 15:27:25  INPUT: :PMU:RPM:CONFIGURE PMU1-2, 0
2024/04/16 - 15:27:25  INPUT: :PMU:SOURCE:RANGE 2, 40
2024/04/16 - 15:27:25  INPUT: :PMU:LOAD 2, 100e3
2024/04/16 - 15:27:25  INPUT: :PMU:MEASURE:RANGE 2, 2, 10e-3
2024/04/16 - 15:27:25  INPUT: :PMU:SARB:SEQ:TIME 2, 1, 10e-6, 10e-6, 1e-3, 10e-6, 1e-3, 10e-6, 1e-3, 10e-6, 10e-6
2024/04/16 - 15:27:25  INPUT: :PMU:SARB:SEQ:STARTV 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0
2024/04/16 - 15:27:25  INPUT: :PMU:SARB:SEQ:STOPV 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0
2024/04/16 - 15:27:25  INPUT: :PMU:SARB:SEQ:MEAS:TYPE 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2
2024/04/16 - 15:27:25  INPUT: :PMU:SARB:SEQ:MEAS:START 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0
2024/04/16 - 15:27:25  INPUT: :PMU:SARB:SEQ:MEAS:STOP 2, 1, 10e-6, 10e-6, 1e-3, 10e-6, 1e-3, 10e-6, 1e-3, 10e-6, 10e-6
2024/04/16 - 15:27:25  INPUT: :PMU:SARB:WFM:SEQ:LIST 2, 1, 1
2024/04/16 - 15:27:25  INPUT: :PMU:OUTPUT:STATE 2, 1
2024/04/16 - 15:27:25  INPUT: :PMU:EXECUTE
2024/04/16 - 15:27:25  PMU measure sample rate was reduced to 1333333
2024/04/16 - 15:27:25  INPUT: :PMU:TEST:STATUS?
2024/04/16 - 15:27:26  INPUT: :PMU:TEST:STATUS?
2024/04/16 - 15:27:26  INPUT: :PMU:DATA:COUNT? 2
2024/04/16 - 15:27:26  INPUT: :PMU:DATA:GET 2, 0, 2048
2024/04/16 - 15:27:26  INPUT: :PMU:DATA:GET 2, 2048, 2048
2024/04/16 - 15:27:26  WARNING: :PMU:DATA:GET requested 2048 points with start index 2048, but only 4077 points
currently available.
2024/04/16 - 15:27:26  INPUT: :PMU:OUTPUT:STATE 1, 0
2024/04/16 - 15:27:26  INPUT: :PMU:OUTPUT:STATE 2, 0
```

Figure 13: KXCI console with Segment ARB commands.

The resulting current measurements of the 100 kohm resistor measured by CH2 are shown in **Figure 14**.

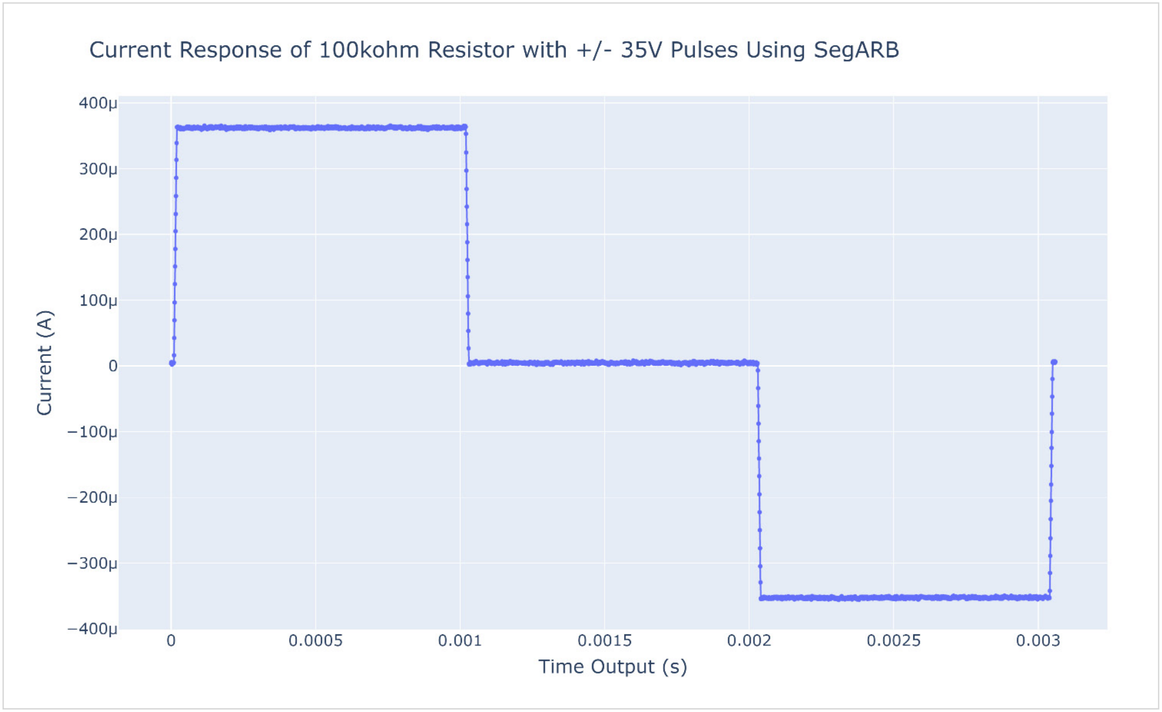


Figure 14: Current response results from Segment ARB waveform sequence with 100 kohm resistor.

Conclusion

PMU KXCI commands enable automation of ultra-fast IV measurements for pulse IV, waveform capture and Segment ARB modes of operation. The PMU is controlled either through ethernet or GPIB connections using an external computer. Several example Python programs using the PMU KXCI commands are available on the Tektronix GitHub site.

Appendix A. Pulse IV and Waveform Capture Commands

Command	Description
`:PMU:ABORT`	Aborts the present process running on a PGU and PMU.
`:PMU:CONNECTION:COMP`	Sets the type of PMU compensation and enables the compensation values.
`:PMU:DATA:COUNT?`	Determines how many readings are stored in the data buffer.
`:PMU:DATA:GET`	Gets real-time data, block by block (up to 2048 pts per block).
`:PMU:EXECUTE`	Runs a final verification, clears the data buffers and executes the configured pulse test.
`:PMU:INIT`	Resets the pulse card to default conditions and to either pulse mode or Segment ARB mode.
`:PMU:LLEC:CONFIGURE`	Enables or disables load line effect compensation.
`:PMU:LOAD`	Sets the output impedance.
`:PMU:MEASURE:CONNECTION:COMP`	Acquires short or current offset compensation data.
`:PMU:MEASURE:MODE`	Sets the measurement to spot mean or waveform capture and to either discrete or average modes.
`:PMU:MEASURE:PIV`	Chooses the readings to be returned in pulse IV mode.
`:PMU:MEASURE:RANGE`	Sets the current measure range type (autorange, limited autorange, or fixed range) and sets the range for limited and fixed range.
`:PMU:OUTPUT:STATE`	Sets the output state for the specified channel: Off or on.
`:PMU:PULSE:BURST:COUNT`	Defines the total number of pulses used in average or discrete modes with either spot mean or waveform measurements.
`:PMU:PULSE:TIMES`	Sets the pulse timing parameters: Period, width, rise time, fall time and delay.
`:PMU:PULSE:TRAIN`	Configures the base and amplitude voltage levels of the pulse.
`:PMU:RETAIN:CONFIG`	Retains the configured pulse settings without executing the test. This allows the :PMU:EXECUTE command to be repeatedly sent without sending all the PMU setting commands each time.
`:PMU:RPM:CONFIGURE`	Switches the RPM output between the PMU, CVU and SMU.
`:PMU:SAMPLE:RATE`	Selects the sample rate from 1E3 to 200E6 samples/s.
`:PMU:SOURCE:RANGE`	Sets both the force and measure voltage range to either 10 V or 40 V.
`:PMU:STEP:DC`	Configures the start, stop and step size of a DC voltage step. A DC voltage sweep must be configured on another channel.
`:PMU:STEP:PULSE:AMPLITUDE`	Configures the start, stop and step size of a pulse amplitude step with a fixed base. A DC voltage sweep must be configured on another channel.
`:PMU:STEP:PULSE:BASE`	Configures the start, stop and step size of a pulse base step with a fixed amplitude. A DC voltage sweep must be configured on another channel.
`:PMU:SWEEP:DC`	Configures the start, stop and step size of a DC voltage sweep.
`:PMU:SWEEP:PULSE:AMPLITUDE`	Configures the start, stop and step size of a pulsed voltage amplitude sweep with a fixed base.
`:PMU:SWEEP:PULSE:BASE`	Configures the start, stop and step size of a pulsed base sweep with a fixed amplitude.
`:PMU:TEST:STATUS?`	Determines if test is running or idle.
`:PMU:TIMES:PIV`	Defines the spot mean measurement window for pulse IV mode.
`:PMU:TIMES:WAVEFORM`	Defines measurement windows for pre and post pulse measurements in waveform capture mode.

Appendix B. Segment ARB Commands

Command	Description
`:PMU:SARB:SEQ:MEAS:START`	Defines an array of start measurement times for a Segment ARB sequence.
`:PMU:SARB:SEQ:MEAS:START:ADD`	Adds an additional array of start measurement times for a Segment ARB sequence.
`:PMU:SARB:SEQ:MEAS:STOP`	Defines an array of stop measurement times for a Segment ARB sequence.
`:PMU:SARB:SEQ:MEAS:STOP:ADD`	Adds an additional array of stop measurement times for a Segment ARB sequence.
`:PMU:SARB:SEQ:MEAS:TYPE`	Defines an array of measure types for a Segment ARB sequence.
`:PMU:SARB:SEQ:MEAS:TYPE:ADD`	Adds an additional array of measure types for a Segment ARB sequence.
`:PMU:SARB:SEQ:SSR`	Defines an array of values to control the high endurance output relay (HEOR).
`:PMU:SARB:SEQ:SSR:ADD`	Adds an additional array of values to control the high endurance output relay (HEOR).
`:PMU:SARB:SEQ:STARTV`	Defines an array of starting voltage levels for a Segment ARB sequence.
`:PMU:SARB:SEQ:STARTV:ADD`	Adds an additional array of starting voltage levels for a Segment ARB sequence.
`:PMU:SARB:SEQ:STOPV`	Defines an array of stopping voltage levels for a Segment ARB sequence.
`:PMU:SARB:SEQ:STOPV:ADD`	Adds an additional array of stopping voltage levels for a Segment ARB sequence.
`:PMU:SARB:SEQ:TIME`	Defines an array of segment times for a Segment ARB sequence.
`:PMU:SARB:SEQ:TIME:ADD`	Adds an additional array of segment times for a Segment ARB sequence.
`:PMU:SARB:SEQ:TRIG`	Defines an array of trigger values for the trigger output of a Segment ARB sequence.
`:PMU:SARB:SEQ:TRIG:ADD`	Adds an additional array of trigger values for the trigger output of a Segment ARB sequence.
`:PMU:SARB:WFM:SEQ:LIST`	Defines the Segment ARB waveform pulsemeasure sequence list.
`:PMU:SARB:WFM:SEQ:LIST:ADD`	Adds additional values for the Segment ARB waveform pulsemeasure sequence list.

Contact Information:

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

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

Rev. 02.2022

KEITHLEY
A Tektronix Company

Find more valuable resources at TEK.COM