Obtaining current-voltage characteristic with Keithley 2401 through GPIB on Python

M. R. Soubkovsky

Master's in Applied Physics and Physics Engineering
University of Lorraine and CentraleSupélec

Abstract

The goal of this study was to obtain a current-voltage characteristic with the SMU Keithley 2401 through GPIB and RS232 protocols using Python.

1 Device features and commands used

1.1 Keithley 2401

1.2 Technical Specifications

Keithley 2401 (Fig. 1) is a **Source Measure Unit**[1], in other words it is an equipment that allows the user to use it as a source and measure at the same time.

Figure 1: Keithley 2401. (Source:`https://www.distrelec.de`)

The equipment is composed by two input/output (called smu a and smu b) channels that can be simultaneously used. The user is able to communicate with the device through either USB, GPIB, RS232.

1.3 Coding Interface

1.3.1 General Commands

The general commands in the Keithley 2401 follow the IEEE Std 488.2. These common commands that are supported by the SMU are listed in Table 1. Although commands are shown in uppercase, common commands are not case sensitive and either uppercase or lowercase can be used. Note that although these commands are essentially the same as those defined by the IEEE Std 488.2 standard, the Series 2400 does not strictly conform to that standard.[2]

As an example, on Python the code for obtaining the device identification **IDN?* would look like:

```
print(keithley.query("*IDN?"))
```

1.3.2 Device Specific Commands

The specific commands that were used to operate the equipment are listed in Table 2.

Code	Name	Description
*IDN?	Identification query	Gives the identification tag of the device
*RST	Reset command	Returns the Series 2400B to default conditions
*TST?	Self-test query	Returns a 0
*CLS	Clear status	Clears all event registers and Error Queue
*TRG	Trigger command	Generates the trigger.EVENT_ID trigger event for use with the trigger model.
*OPC	Operation complete command	Set the Operation Complete bit in the Standard Event Register after all pending commands, including overlapped commands, have completed

Table 1: General Commands

2 First tests and communication through MAX

In order to assure that the connection with the device is well established, a series of primary tests is performed. For these tests we used the software Measurement & Automation Explorer (MAX) provided by Natural Instruments (Fig. 2).

MAX Visa Test Panel (Fig. 3) provides the user with a simple GUI to connect with the device and execute the first series of tests and check if the connection is well established. An extra step that needs to be taken on the Keithley 2401 that is not crucial on the 2602B is setting a Termination Character. On MAX the correct Termination Character can be tried on the VISA Test Panel (Fig. 4).

The identification number (Table 1) is usually the first test that was executed to make sure the connection was well made. As example is shown on Figure 5.

3 Python Program

In order to explain the whole program, first, a short description of the libraries and the functions will be given.

3.1 Libraries used

A list of libraries that were imported in Python and necessary to the execution of the code that follows is shown below and a brief description of the library is given. Libraries:

- **matplotlib.pyplot**: Provides a MATLAB-like plotting framework inside matplotlib
- **numpy**: adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays
- **visa**: enables you to control all kinds of measurement devices independently of the interface (e.g. GPIB, RS232, USB, Ethernet)
- **tkinter**: standard Python interface to the Tk GUI toolkit
- **time**: provides various time-related functions

Code	Description
:SYSTem:BEEPer:STATe 0	deactivate machine beep
:system:preset	reset the Keithley 2400
:output1:state 1	turn on the smu as an output
:output1:state 0	turn off the smu as an output
:SOURce:VOLTage [level]	adjust voltage of smu to the desired level
:MEASure?	return value of current I measured on smu X
:curr:protection 0.01	set smu current protection to 0.01 A
:source:function:mode VOLT	set smu mode to voltage source
:format:elements curr	tell the smu to read only values of current

Table 2: Keithley 2401 Specific Commands. Source:[2]

3.2 Functions

The following list of functions lists the functions that are part of the program with its parameters and specifications.

- **connexion_choice(connection)**: enables the user to choose the identifier of the protocol of connection and connects with the device.

  connection: *string*, contains the identifier of the connection

- **close_all()**: close any connection
- **reset()**: reset the smu to the default state
- **switchON([onoff=False])**: allows the user to turn on or off the smuX chosen

  onoff: *string*, [Optional] True to turn on and False to Turn off. Default: False

- **measurement(volts_min,volts_max,nb)**: sends the voltage to the device and measures the current

  volts_min: *float*, initial output voltage that is sent to the circuit

  volts_max: *float*, final output voltage that is sent to the circuit

  nb: *int*, number of measurements

  RETURNS: *float* measure of current in Amps

- **complete_measure(volts_min,volts_max,nb)**: calls other functions in order to automatically make all the measurements

  volts_min: *float*, initial output voltage that is sent to the circuit

  volts_max: *float*, final output voltage that is sent to the circuit

  nb: *int*, number of measurements

  RETURNS: *list* [input voltage(*float*), measure of current in Amps(*float*)]

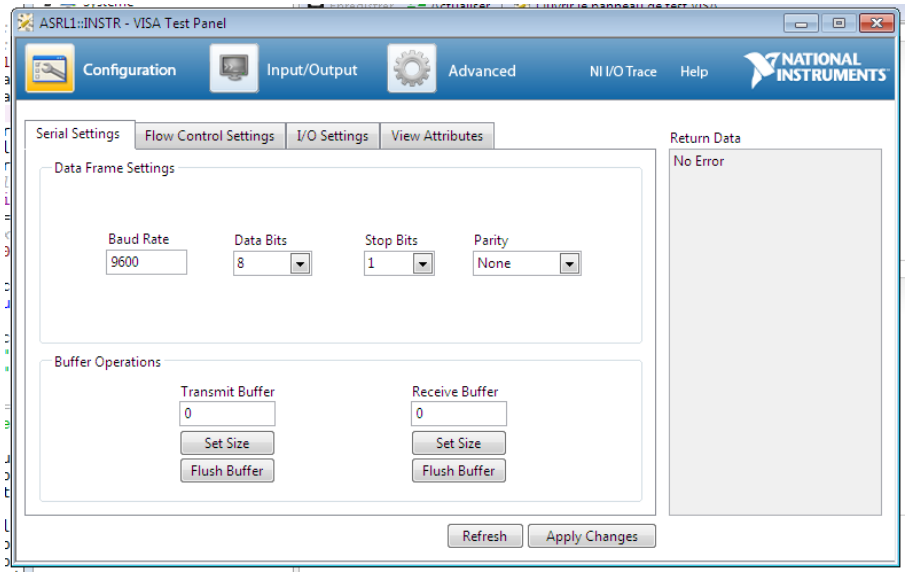

Figure 2: MAX Interface

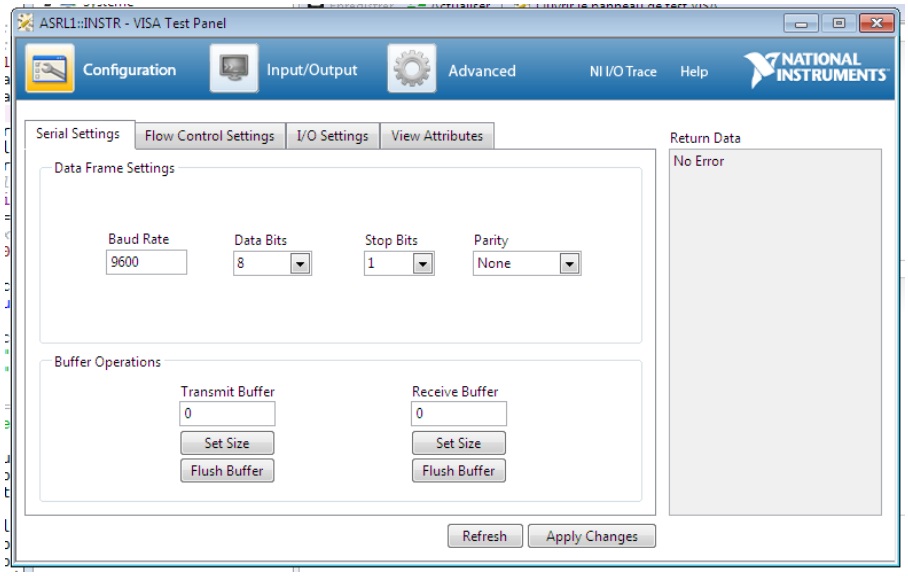

Figure 3: MAX - VISA Test Panel

3.3 Code

First of all, the libraries described beforehand need to be imported

```
1    import visa
2    import numpy as np
3    import matplotlib.pyplot as plt
4    from tkinter import Button, Tk, Frame, Entry, Label, Checkbutton, BooleanVar, StringVar
5    import time
```

Then we use a series of functions to perform individual tasks. These functions are thoroughly described in the Section 3.2.

```
6    def connexion_choice(connexion):
7        """Permet de choisir la connexion
```

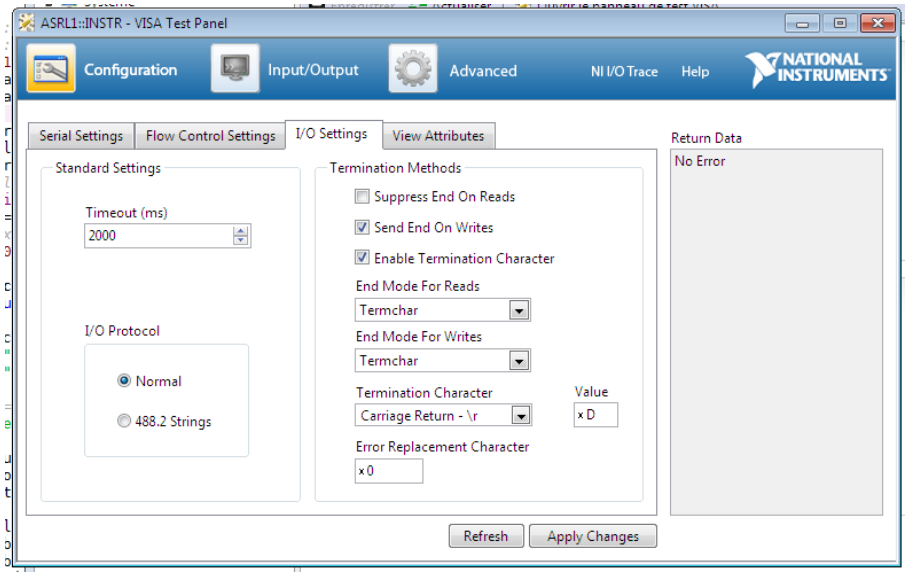

Figure 4: Setting a Termination Character

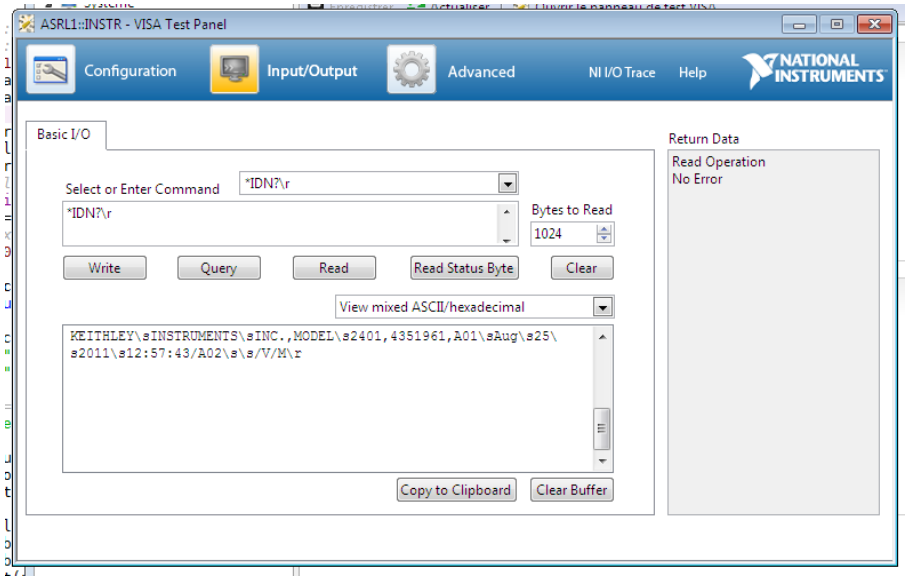

Figure 5: MAX - VISA Test Panel *IDN?

```
8           \nAllow to choose the connexion"""
9           global keithley
10          try:
11              rm = visa.ResourceManager() #import visa
12              rm.list_resources() #import visa
13              keithley = rm.open_resource(connexion) #creat device connexion
14          except:
15              print("Connexion error, check the connexion (GPIB,RS232,USB,Ethernet) and it's
                ↪  number")
16              raise StopIteration ("Erreur de connexion. Verifier la connexion
                ↪  (GPIB,RS232,USB,Ethernet) et son numeros \
17                                  \nConnexion error, check the connexion
    ↪  (GPIB,RS232,USB,Ethernet) and it's number")
18      #---------------------------------------------------------------------
19      def close_all():
```

```
        """Coupe la connexion
        \nClose the connexion"""
        try:
            reset() #reset smu
            keithley.write(":SYSTem:BEEPer:STATe 0") #desactive le beep/deactivate sound
            keithley.close() #ferme la connexion/close the connexion
            print("Connexion closed")
        except:
            print("Closing error")
            raise StopIteration ("Erreur de fermeture \
                                 \nClosing error")
    #------------------------------------------------------------------------
    def reset():
        """reset smu"""
        try:
            keithley.write(":system:preset")
        except:
            print("Reset error")
            raise StopIteration ("Erreur de reset \
                                 \nReset error")
    #------------------------------------------------------------------------
    def switchON(onoff=False):
        """Active/desactive le smu
        \nTurn on/off the smu"""
        try:
            keithley.write((":output1:state 1" if onoff else ":output1:state 0"))
            print("Source on" if onoff else "Source off")
        except:
            print("Source can't be turn on/off")
            raise StopIteration ("Erreur de changement d'etats \
                                 \nSource can't be turn on/off")
    #------------------------------------------------------------------------
    def measurement(volts_min,volts_max,nb,delay=0):
        """Envoi des tensions et mesure des courants. Enregistre les mesures dans la
        ↪ variable measure
        \nSend tensions and measure currents. Save measures in measure variable"""
        measure=[] #creat array for futur measures
        tension_input=np.linspace(volts_min,volts_max,nb) #creat an array with all tensions
        ↪ needed for measurement
        print('U(V)\t I(A)\t\t points\ttemps(s)') #display tension,current,points,time
        time_begin=time.time()
        i=0
        for x in tension_input: #loop on tensions values
            i+=1
            keithley.write(":SOURce:VOLTage %f" % x) #send the voltage x to the smu
            time.sleep(delay) #delay not needed

            y=float(keithley.query(":MEASure?")) #read current value I(A)
            time_end=time.time()
            print ("%3.3f\t%s\t%s/%s\t%3.3f" % (x,y,i,nb,time_end-time_begin)) #display
            ↪ tension,current,points,time
```

```
68              if y>=0.01: #use to stop if current too high
69                  close_all()
70                  print("current too high")
71                  raise ValueError ("current too high") #used as safety if the protection fail
                        ↪ (Redundancy)
72              measure.append(y) #add the current value to a array regrouping all measures
73
74          return tension_input,measure #return the voltage and current array
75      #-------------------------------------------------------------------------
76      def complete_measure(volts_min,volts_max,nb,smux="temp",delay=0):
77          """Fonction principale prennant les valeurs de tkinter en entrer. Trace les mesures
                ↪ et les sauvegardes.
78          Principal fonction taking tkinter value as input. Plot measures and save them."""
79          switchON(True) #activate smu
80
81          keithley.write(":curr:protection 0.01") #used to prevent too high currents running
                ↪ into the circuit
82          keithley.write(":source:function:mode VOLT") #smua devient source de tension (et
                ↪ donc ne peut être que mesure de courant)
83          keithley.write(":format:elements curr") #tell the smu to read only currents values
84
85          tension_input,measure=measurement(volts_min,volts_max,nb) #envoi les tensions
                ↪ choisis et mesure les courants associées
86          keithley.write(":SOURce:VOLTage 0") #Go back to 0 volt after measures dones
87
88          plt.figure(num='Diode '+smux+' Characteristic') #plot differents figure according to
                ↪ a specific name
89          plt.clf() #clear the graph to avoir superposing data from the same set (can be
                ↪ deactivated if need to superpose)
90          plt.title("Diode "+smux+" Characteristic")
91          plt.ylabel('I(A)')
92          plt.xlabel('U(V)')
93          plt.plot(tension_input,measure, '+', label='Diode '+smux) #display
                ↪ current(input_tension) with dots
94          plt.legend() #add legend to the graph (take label from plot)
95          plt.savefig('Diode %s Characteristic I(U).svg' %smux, format='svg', dpi=1000,
                ↪ bbox_inches='tight') #save the graph in a vector file
96          plt.show() #plot data
97
98          np.savetxt('Diode %s Characteristic I(U).csv' %
                ↪ smux,np.transpose((tension_input,measure)),delimiter="\t") #save data on a
                ↪ binary file
99          switchON(False) #deactivate smu
```

After defining the functions, we attribute values to the variables, raising an error if the voltage passes the chosen limit:

```
98      connexion='GPIB0::24::INSTR'
99      #connexion='COM1'
100     volts_min=0  #min voltage
```

```
    nb=101        #nb de mesures
    volts_max=1    # max voltage
    delay=0  #time between applying voltage and measuring current (not needed)
    smux='temp'

    connexion_choice(connexion) #try to connect the devide using GPIB or RS232
    reset() #reset smu
    keithley.write(":SYSTem:BEEPer:STATe 0") #deactivate beep
```

And finally we execute the TKinter code:

```
    def compute():
        """Fonction utiliser par tkinter pour commander l'instrument
        \nFonction use by tkinter to pilote the instrument"""
        message1["text"] = "" #reset messages
        message2["text"] = ""
        message3["text"] = ""
        message4["text"] = ""
        message5["text"] = ""
        message6["text"] = ""
        message7["text"] = ""
        smux=str(smux_entry.get()) #return the smux value in the tkinter entry

        try:
            volts_min=float(volt_min_entry.get())    # min voltage
            volts_max=float(volt_max_entry.get())    # max voltage
            if abs(volts_max)>10 or abs(volts_min)>10:
                texte5="abs(volt) <=10"
                message5["text"] = texte5
                print(texte5)
            else:
                nb=int(point_number_entry.get()) #measures nb
                if nb<1:
                    texte4="nb>0"
                    message4["text"] = texte4
                    print(texte4)
                else:
    #                delay=float(delay_entry.get()) #not needed
    #                print(smux,volts_min,volts_max,nb,delay) #used to debug
                    try:  #issues with try: hide internals errors
                        complete_measure(volts_min,volts_max,nb,smux=smux)
                        texte7="Measures done"
                        message7["text"] = texte7
                        print(texte7)
                    except:
                        texte6="Error from measurement detected"
                        message6["text"] = texte6
                        print(texte6)
                        reset() #cause False positive floats error if using without
                        ↪  instrument and previous commands disable
```

```
152             except:
153                 texte2="floats"
154                 message2["text"] = texte2
155                 print(texte2)
156 
157         root = Tk() #used to creat user interface
158         frame = Frame(root)
159         root.title("Keithley options")
160         frame.pack()
161 
162         L0 = Label(frame, text="diode's name:") #fixed text
163         L0.grid(row=0, column=0)
164         smux_entry = Entry(frame, textvariable=StringVar(frame, value=smux), bd =2, width=7)
            ↪  #stringvar is used to have default values
165         smux_entry.grid(row=0, column=1) #grid is used to position items on the interface
166 
167         message1 = Label(frame, text="")   #allow to display messages when activate
168         message1.grid(row=0, column=3)
169 
170         L1 = Label(frame, text="start volt(V)")
171         L1.grid(row=1, column=0)
172         volt_min_entry = Entry(frame, textvariable=StringVar(frame, value=volts_min), bd =2,
            ↪  width=7)
173         volt_min_entry.grid(row=1, column=1)
174 
175         message3 = Label(frame, text="")
176         message3.grid(row=1, column=3)
177 
178         L2 = Label(frame, text="end volt(V)")
179         L2.grid(row=2, column=0)
180         volt_max_entry = Entry(frame, textvariable=StringVar(frame, value=volts_max), bd =2,
            ↪  width=7)
181         volt_max_entry.grid(row=2, column=1)
182 
183         message5 = Label(frame, text="")
184         message5.grid(row=2, column=3)
185 
186         L3 = Label(frame, text="nbr pts")
187         L3.grid(row=3, column=0)
188         point_number_entry = Entry(frame, textvariable=StringVar(frame, value=nb), bd =2,
            ↪  width=7)
189         point_number_entry.grid(row=3, column=1)
190 
191         message4 = Label(frame, text="")
192         message4.grid(row=3, column=3)
193 
194         #L4 = Label(frame, text="delay(s)")
195         #L4.grid(row=4, column=0)
196         #delay_entry = Entry(frame, textvariable=StringVar(frame, value=delay), bd =2, width=7)
197         #delay_entry.grid(row=4, column=1)
198 
```

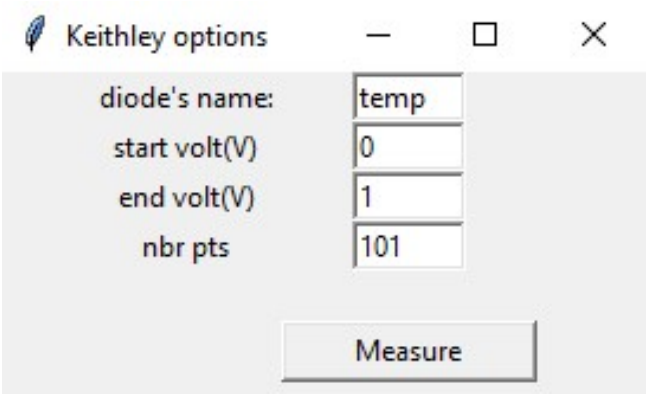

Figure 6: GUI TKinter with measurement options

```
199     message2 = Label(frame, text="")
200     message2.grid(row=4, column=3)
201
202     message6 = Label(frame, text="")
203     message6.grid(row=5, column=3)
204
205     compute_button = Button(frame, text="Measure", width=14, command=compute) #button used
        ↪  to get all values and start measures
206     compute_button.grid(row=5, column=1)
207
208     message7 = Label(frame, text="")
209     message7.grid(row=5, column=3)
210
211     root.mainloop() #instance looping until closed
212     close_all() #reset and close connexion with the instrument
```

The GUI can be seen on Fig. 6.
After the clicking on "Compute", a csv file with the data for the voltage and current, and a pdf chart are saved on the same folder as the python code; and the chart is shown with matplotlib.pyplot.

4 Practical Tests

A test was performed to obtain the current-voltage characteristic of a PN diode. The result can be seen on Figure 7.

5 GitHub Repository

This project is public under MIT License on GitHub. The repository is accessible through this link `https://github.com/marcelrsoub/keithley-visa-measurements`.

References

[1] Source measure unit, August 2018. Page Version ID: 854854432.

[2] Series 2600b System SourceMeter Manual.

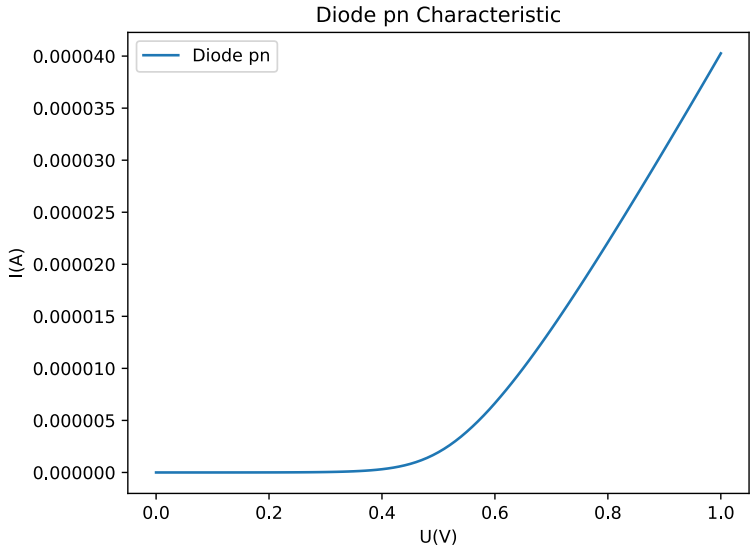

Figure 7: Practical test performed to obtain the Current-Voltage characteristic for a PN diode with the program