Obtaining current-voltage characteristic with Keithley 2701 through RS232 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 measure the current, resistance or voltage with the DAQ System Keithley 2701 through RS232 protocol using Python.

1 Device features and commands used

1.1 Keithley 2701

1.2 Technical Specifications

Keithley 2701 (Fig. 1) is a DAQ System (Data Acquisition System), in other words it is an equipment that allows the user to make specific measurements. The device is a $6^1/_2$-digit high-performance multimeter/data acquisition system. It can measure voltage (DC and AC), current (DC and AC), resistance (2- and 4-wire), temperature (thermocouple, thermistor, and 4-wire RTD), frequency and period, and test continuity.[1]

Figure 1: Keithley 2701. (Source:`https://www.distrelec.de`)

The user is able to communicate with the device through either Ethernet and RS232.

1.3 Coding Interface

1.3.1 General Commands

The general commands in the Keithley 2701 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 2700 does not strictly conform to that standard.[1]

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 2700B 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 2701 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
:format:elements READ	give only one value (specified on :FUNction)
:FUNCtion 'VOLTage'	specify Volt measurement
:FUNCtion 'CURRent'	specify Current measurement
:FUNCtion 'RESistance'	specify Resistance measurement
READ?	return value measured

Table 2: Keithley 2701 Specific Commands. Source:[1]

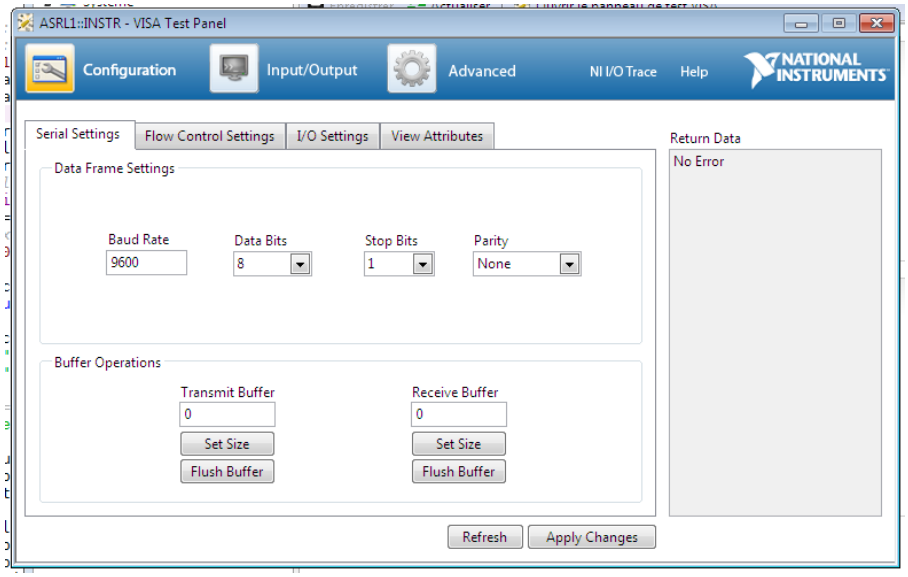

Figure 2: MAX Interface

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 Keithley 2701 to the default state
- **measurement(nb,caract,delay=0)**: sends the voltage to the device and measures the current

  nb: *int*, number of measurements

  caract: *string*, string that contains the type of measurement (used to display on GUI)

  delay: *float*, delay bewteen measurements(in ms)

  RETURNS: *nd.array* voltage and measure array

- **complete_measure(nb,caract[,delay=0])**: calls other functions in order to automatically make all the measurements

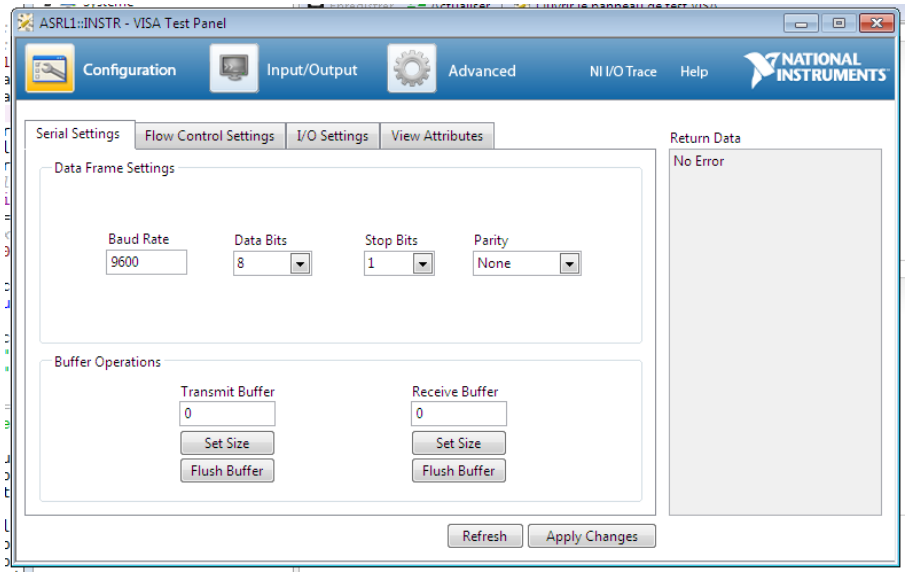

Figure 3: MAX - VISA Test Panel

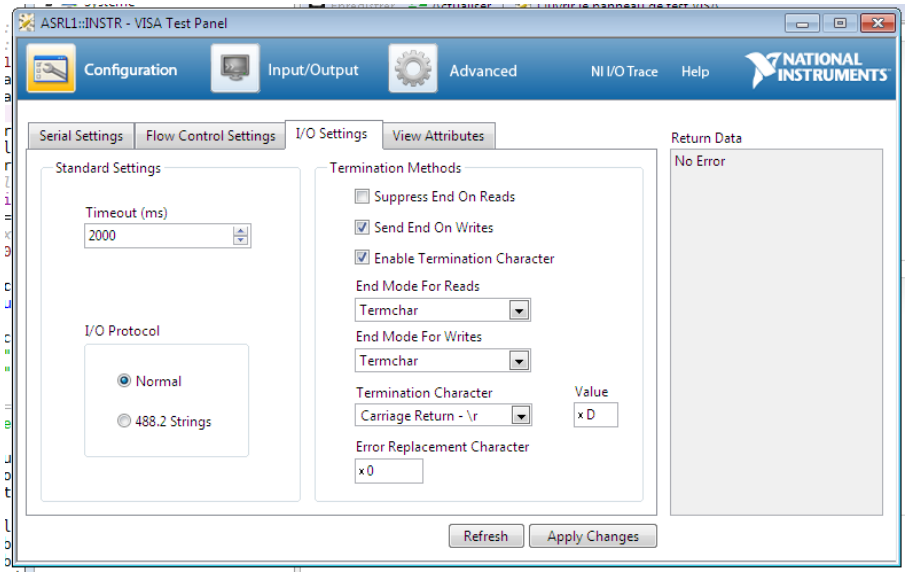

Figure 4: Setting a Termination Character

nb:	*int*,	number of measurements
caract:	*string*,	string that contains the type of measurement (used to display on GUI)
delay:	*float*,	delay bewteen measurements(in ms)
RETURNS:	*list*	[input voltage(*float*), measure of current in Amps(*float*)]

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, StringVar, Checkbutton, BooleanVar,
     ↪  IntVar, Radiobutton
```

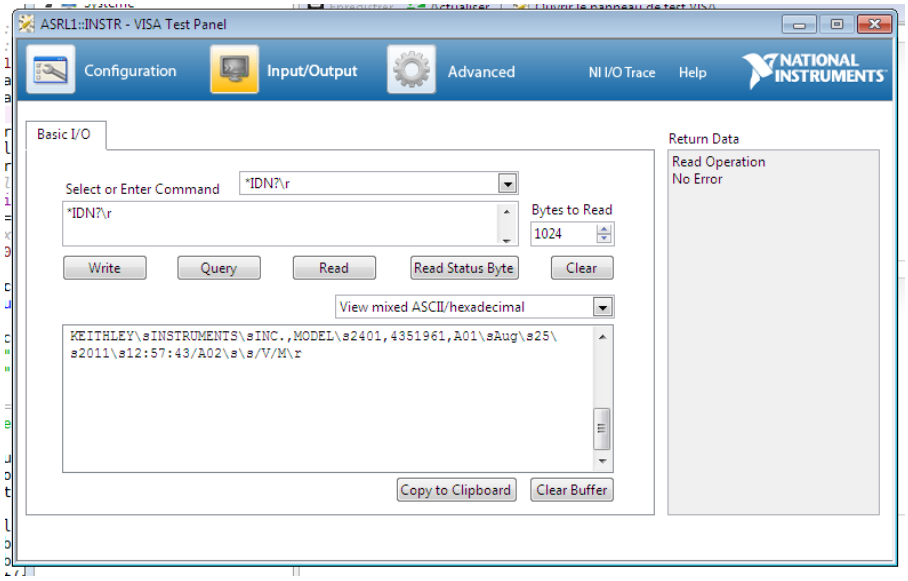

Figure 5: MAX - VISA Test Panel *IDN? for Keithley 2400

```
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
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, send_end=True, read_termination='\r')
                  #creat device connexion
14            print("connexion succeed")
15        except:
16            print("Connexion error, check the connexion (GPIB,RS232,USB,Ethernet) and it's
                  number")
17            raise StopIteration ("Erreur de connexion. Verifier la connexion
                  (GPIB,RS232,USB,Ethernet) et son numeros \
18                                \nConnexion error, check the connexion
  (GPIB,RS232,USB,Ethernet) and it's number")
19    #-------------------------------------------------------------------------
20    def close_all():
21        """Coupe la connexion
22        \nClose the connexion"""
23        try:
24            reset() #reset smu
25            keithley.write(":SYSTem:BEEPer:STATe 0") #desactive le beep/deactivate sound
26            keithley.close() #ferme la connexion/close the connexion
27            print("Connexion closed")
28        except:
```

```
29              print("Closing error")
30              raise StopIteration ("Erreur de fermeture \
31                                    \nClosing error")
32      #-------------------------------------------------------------------------
33      def reset():
34          """reset smu"""
35          try:
36              keithley.write("*RST\r")
37              print("instrument reset")
38          except:
39              print("Reset error")
40              raise StopIteration ("Erreur de reset \
41                                    \nReset error")
42
43      #-------------------------------------------------------------------------
44      def measurement(nb,caract,delay=0):
45          """Envoi des tensions et mesure des courants. Enregistre les mesures dans la
            ↪ variable measure
46          \nSend tensions and measure currents. Save measures in measure variable"""
47          global tension_input,measure
48          measure=[] #creat array for futur measures
49          tension_input=[]
50          i=0
51          keithley.write(":format:elements READ") #give only value
52          if caract=='Um(V)':
53              keithley.write(":FUNCtion 'VOLTage'") #change to volt measurement
54
55          elif caract=='I(V)':
56              keithley.write(":FUNCtion 'CURRent'")
57
58          elif caract=='Resistance(Ohm)':
59              keithley.write(":FUNCtion 'RESistance'")
60
61          print('\nWrite the real voltage or write any string to stop the measurement.\nWrite
            ↪ nothing in order to keep the theoretical value')
62          for x in tension_input_temp: #loop on tensions values
63              i+=1
64              print("\nSet voltage to %s(V)" % x)
65              try:
66                  real_value=float(raw_input('Input voltage: ') or x)
67              except:
68                  print("Measure stopped")
69                  break
70              tension_input.append(real_value)
71
72              time.sleep(delay) #delay not needed
73      #         y=x
74              y=float(keithley.query("READ?")) #read current value I(A)
75      #
76              print('Us(V)\t%s\tpoints' % caract) #display tension,current,points,time
```

```
77              print ("%s\t%s\t%s/%s" % (real_value,y,i,nb)) #display
                ↪  tension,current,points,time
78
79              measure.append(y) #add the current value to a array regrouping all measures
80          return tension_input,measure #return the voltage and current array
81      #-------------------------------------------------------------------------
82      def complete_measure(nb,caract,delay=0):
83          """Fonction principale prennant les valeurs de tkinter en entrer. Trace les mesures
            ↪  et les sauvegardes.
84          Principal fonction taking tkinter value as input. Plot measures and save them."""
85          global tension_input,measure
86
87          tension_input,measure=measurement(nb,caract) #envoi les tensions choisis et mesure
            ↪  les courants associées
88
89      #obsolete ?
90          if len(tension_input) != len(measure):
91              tension_input=tension_input[0:-(len(tension_input)-len(measure))] #reduce input
                ↪  to measure size
92          if measure:
93              plt.figure(num='Measure ') #plot differents figure according to a specific name
94          #    plt.clf() #clear the graph to avoir superposing data from the same set (can be
            ↪  deactivated if need to superpose)
95              plt.title("Measure ")
96
97              if caract=='Resistance(Ohm)':
98                  plt.ylabel(r'Resistance$(\Omega)$')
99              elif caract=='Um(V)':
100                 plt.ylabel(r'$U_m(V)$')
101             else:
102                 plt.ylabel(caract)
103             plt.xlabel(r'$U_s$(V)')
104             plt.plot(tension_input,measure, '+', label='Measure ') #display
                ↪  current(input_tension) with dots
105             plt.legend() #add legend to the graph (take label from plot)
106             plt.savefig('Measure %s.svg' % (caract), format='svg', dpi=1000,
                ↪  bbox_inches='tight') #save the graph in a vector file
107             plt.show() #plot data
108
109             np.savetxt('Measure %s.csv' %
                ↪  (caract),np.transpose((tension_input,measure)),delimiter="\t") #save data on
                ↪  a binary file
```

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

```
98      connexion='COM1'
99      nb=11       #nb de mesures
100     delay=0  #time between applying voltage and measuring current (not needed)
101     smux='temp'
```

```
    caract='Um(V)'

    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 utilisée par tkinter pour commander l'instrument
        \nFonction use by tkinter to pilote the instrument"""
        msg_floats["text"] = ""
        msg_nb["text"] = ""
        msg_high_volt["text"] = ""
        msg_measure["text"] = ""
        msg_nb["text"] = ""

        smux=str(smux_entry.get()) #return the smux value in the tkinter entry
        if measure_choice.get()==0: #see if can measure resistance and volt or resistance
        ↪  and current                    #if so, change into "if volt and current true"
        ↪  then change loop measure and caract
            caract='Um(V)'
        elif measure_choice.get()==1:
            caract='I(V)'
        elif measure_choice.get()==2:
            caract='Resistance(Ohm)'
        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:
                txt_high_volt="abs(volt) <=10"
                msg_high_volt["text"] = txt_high_volt
                print(txt_high_volt)
            else:
                nb=int(point_number_entry.get()) #measures nb
                if nb<1:
                    txt_nb="nb>0"
                    msg_nb["text"] = txt_nb
                    print(txt_nb)
                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,caract,smux=smux)
                        txt_measure="Measures done"
                        msg_measure["text"] = txt_measure
                        print(txt_measure)
                    except:
                        txt_measure="Error from measurement detected"
                        msg_measure["text"] = txt_measure
```

```
154                         print(txt_measure)
155                         reset() #cause False positive floats error if using without
                            ↪ instrument and previous commands disable
156             except:
157                 txt_floats="floats"
158                 msg_floats["text"] = txt_floats
159                 print(txt_floats)
160
161         root = Tk() #used to creat user interface
162         frame = Frame(root)
163         root.title("KMT - Keithley Measurement Tool")  #[Marcel]: I changed the labels just a
            ↪ bit
164         frame.pack()
165
166         L0 = Label(frame, text="Measure's name:") #fixed text
167         L0.grid(row=0, column=0)
168         smux_entry = Entry(frame, textvariable=StringVar(frame, value=smux), bd =2, width=7)
            ↪ #stringvar is used to have default values
169         smux_entry.grid(row=0, column=1) #grid is used to position items on the interface
170
171
172
173         msg_floats = Label(frame, text="")
174         msg_floats.grid(row=2, column=2)
175
176         L3 = Label(frame, text="Number of points")
177         L3.grid(row=3, column=0)
178         point_number_entry = Entry(frame, textvariable=StringVar(frame, value=nb), bd =2,
            ↪ width=7)
179         point_number_entry.grid(row=3, column=1)
180
181         msg_nb = Label(frame, text="")
182         msg_nb.grid(row=3, column=2)
183
184         measure_choice= IntVar()
185         measure_choice.set(1)
186         options = [
187             ("Voltage(V)"),
188             ("Current(A)"),
189             ("Resistance()")
190         ]
191         Check_button=Label(frame, text="""Choose measurement:""")
192         Check_button.grid(row=4, column=0)
193
194         for val, language in enumerate(options):
195             option_button=Radiobutton(frame,
196                                       text=language,
197                                       variable=measure_choice,
198                                       value=val)
199             option_button.grid(row=val+4, column=1)
200
```

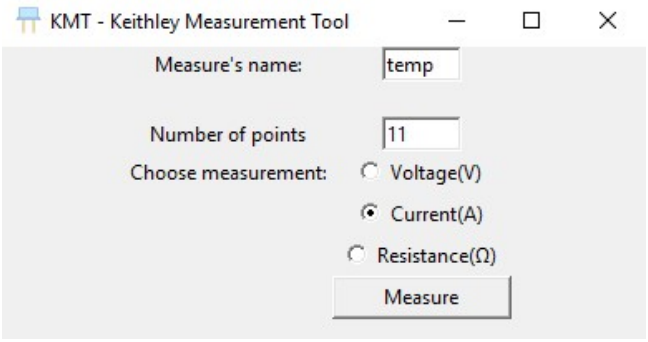

Figure 6: GUI TKinter with measurement options

```
201     compute_button = Button(frame, text="Measure", width=14, command=compute) #button used
        ↪  to get all values and start measures
202     compute_button.grid(row=8, column=1)
203
204     msg_measure = Label(frame, text="")
205     msg_measure.grid(row=8, column=2)
206
207     root.iconbitmap(default='crystal_oscillator1600.ico')
208
209     root.mainloop() #instance looping until closed
210     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.

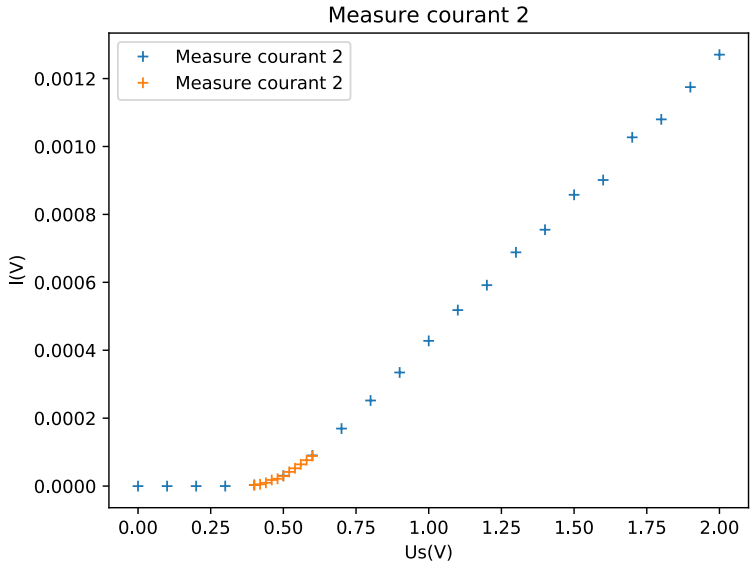

Figure 7: Practical test performed to obtain the Current-Voltage characteristic for a PN diode with the program

5 GitHub Depository

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

References

[1] Series 2700 System SourceMeter Manual.