DC Module Reference

This document is the reference for the DC sub module of the Python SLab system.

Index

Introduction

The “SLab System” is a software solution that provides a low cost access to circuit measurements by the use of low cost development boards.
The complete system has a natural tendency to feature creeping. In order to make the dimension of the project easy to manage, it has been divided in several modules.
The main module, associated to the file slab.py, contains all the code that directly access the Hardware Board and the most immediate low complexity commands.
The functionality of the main module is extended by the use of additional modules. This document describes the DC module, associated to the file slab_dc.py that includes a set of functions to generate DC curves of different circuit configurations.

If you want to use the DC module you will also need to use the main slab module as it is the one that includes the ***connect*** command that starts the communication with the Hardware Board. That means that you will need to use two imports.

```
>>> import slab
>>> import slab_dc as dc
```

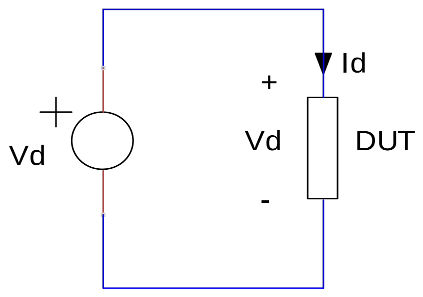
If you need to use other additional modules you can add more lines for other sub modules. Observe that we have used an alias name for the module namespace, so, a command with the name “command” will be accessed as “dc.command”.

All commands in this module generate a plot from a sequence of DC measurements.

Plot commands use the Matplotlib library to draw the figures. Figures drawn are blocking. So, in order to end the command you must close the figure. Figures can be panned, zoomed and saved on disk if needed.

Two terminal device I-V plots

Commands on this section obtain the I-V characteristics of a two terminal component.



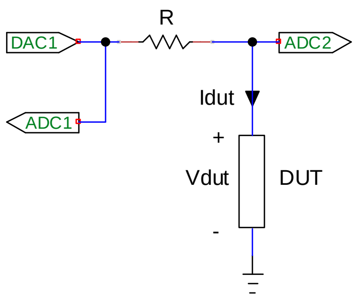
A two terminal device can be usually characterized in DC by its I-V characteristic. That is the measurement of the device current for each possible voltage at its terminals.

The three commands ***curveVI***, ***curveVIref*** and ***curveVIbridge*** are useful to obtain the I(V) DC curve of a two terminal device under test (DUT) in different scenarios.

curveVI (v1, v2, vi, r, wt, returnData)

Arguments	v1	Start voltage	V1
	v2	End voltage	
	vi	Voltage step (Defaults to 0.1V)	
	r	Resistor value in kOhm (Defaults to 1k)	
	wt	Wait time (Defaults to 0.1s)	
	returnData	Return plot data (Defaults to False)	
Returns		See text. Draws a plot.	

This command measures the current Idut against voltage Vdut curve of a two terminal device under test (DUT) for positive voltages. In order to perform the measurement, the negative terminal of the device shall be connected to GND and the positive terminal to ADC 2. A resistor shall also be connected between the DAC 1 output and ADC 1 input and the positive terminal of the DUT as shown in the following figure:



The command will set DAC 1 voltage between **v1** and **v2** with **vi** step increments. If **vi** is omitted, it defaults to 0.1 V. A wait of **wt** seconds is included between each new DAC 1 value and the measurement on ADC 1. If **wt** is omitted it defaults to 0.1s.
When the measurement ends, DAC 1 voltage is set to zero to minimize power consumption.

From the measurements, current at the DUT is computed and the Idut(Vdut) curve is drawn.

$$V_{dut} = V_{ADC2} \qquad I_{dut} = \frac{V_{ADC1} - V_{ADC2}}{R}$$

Note that we use ADC 1 connected to DAC 1 output. In theory we could obtain Idut from V_{DAC1} and V_{ADC2} not needing the ADC 1 connection. In practice it is a good idea to obtain voltage values for the same kind of device (ADC in this case) if there is the possibility to substract them when they are nearly equal. In our case, using two ADCs give a better current measurement around zero than using only one ADC and the set DAC value.
Moreover, you are not guaranteed to really have the set value at the DAC output so, specially at the end ranges of the DAC. Using an ADC guarantees that you know the voltage at the node.

Note that we don't set the DUT input voltage range with the **v1** and **v2** parameters. Those parameters set the voltage range at the resistor, the more current the device consumes, the less device voltage range we get. It is tempting to use a low value resistor to increase the voltage range, but there are two problems using this approach:

- Setting a low value resistor gives a small V_{ADC1} - V_{ADC2} drop on the resistor. That limits the accuracy of the current measurement.

- Resistor **r** limits the current at the device. In order to guarantee that the device is not damaged and that the buffering circuit doesn't reach its compliance limits, it is recommended to limit the maximum current. We can calculate the limit as:

$$I_{DUT\ max} = \frac{V_{DAC\ 1\ max} - V_{DUT\ min}}{r} = \frac{v2}{r} \qquad r = \frac{v2}{I_{DUT\ max}}$$

You need to ensure that DAC 1 buffer circuit can provide the $I_{DUT\ max}$ current.

The command only works for positive Vdut voltages. In order to obtain I(V) curves that include negative voltages use the ***curveVIref*** or ***curveVIbridge*** instead.

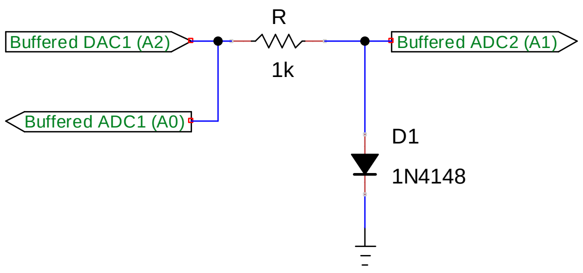
If the optional parameter ***returnData*** is True or ***setPlotReturnData(True)*** was called previously, the command return a two element tuple with numpy arrays of:

- Vdut values
- Idut values

Example 03:

Measure the forward curve of a 1N4148 diode using the STM32 Nucleo F303RE board.

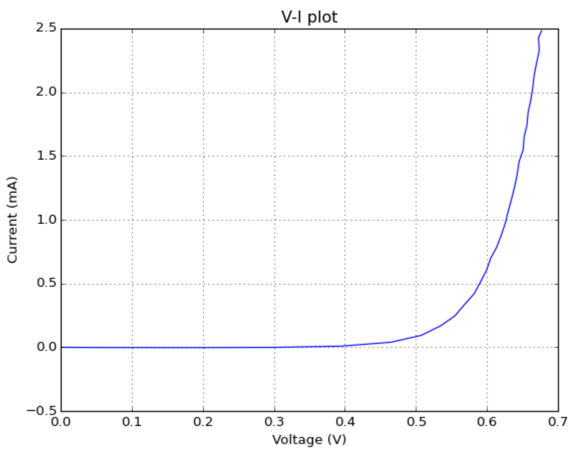
We will connect the diode as DUT and we will buffer inputs and outputs to prevent DAC and ADC loading errors.



The following command will generate the I(V) curve. We sweep DAC 1 voltage between 0V and 3.2V and use the default values for **vi**, **r** and **wt**.

```
>>> dc.curveVI(0.0,3.2)
```

After the command run we will obtain a curve like the following one:

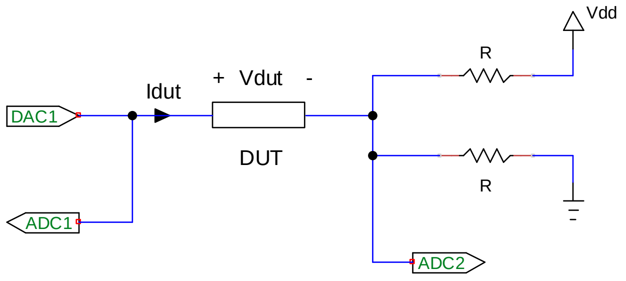


By using a 1kΩ resistor we have limited the current on the diode to a maximum of 3.2mA. In practice, as voltage on the diode increases with current, the maximum current is 2.5mA. Observe also, that the drop on r limits the measured V_{DUT} voltage to less than 0.7V.

curveVIref (v1, v2, vi, r, wt, returnData)

Arguments	v1	Start voltage	V1
	v2	End voltage	
	vi	Voltage step (Defaults to 0.1V)	
	r	Resistor value in kOhm (Defaults to 1k)	
	vr	Reference voltage (Defaults to Vdd/2)	
	wt	Wait time (Defaults to 0.1s)	
	returnData	Return plot data (Defaults to False)	
Returns		See text. Draws a plot.	

This command measures the current Idut against voltage Vdut curve of a two terminal device under test (DUT).
In order to perform the measurement, the negative terminal of the device shall be connected to the central point of two equal **r** value resistors connected to Vdd and GND and to ADC 2.
DAC 1 and ADC 1 shall be connected to the positive terminal of the DUT and t.
The following figure shows the circuit connections:



By connecting the negative terminal of the DUT as shown, negative voltages and currents can be measured. As the total voltage range has not changed compared to the CurveVI command, the measurement of negative voltages reduces the positive voltage range.

The command will set DAC 1 voltage between **v1** and **v2** with **vi** step increments. If **vi** is omitted, it defaults to 0.1 V. A wait of **wt** seconds is included between each new DAC 1 value and the measurement on ADC 1 and ADC 2. If **wt** is omitted it defaults to 0.1s.
When the measurement ends, DAC 1 voltage is set to Vdd/2 to minimize power consumption.

From the measurements, voltage and current at the DUT is computed and the Idut(Vdut) curve is drawn.

$$V_{dut} = V_{ADC1} - V_{ADC2} \qquad I_{dut} = 2\frac{V_{ADC2} - V_{dd}/2}{R}$$

Results obtained by using this command depend on both resistors having the same **r** value. If this condition is not met, an error in the current value will be shown on the curve.

As in the ***CurveVI*** command, **r** needs to be chosen value both to obtain good current accuracy and to limit the current in the DUT. In this case, also, the circuit always consumes current as the two resistors are always connected between Vdd and GND. Default current consumption of the circuit, when I_{DUT} is zero is:

$$I_{Base} = \frac{Vdd}{2r}$$

Current trough the DUT is bounded by:

$$|I_{DUT}|_{Max} = \frac{Vdd}{r}$$

You need to ensure that DAC 1 buffer circuit can provide this current.

The command requires that both resistors are equal and that the vdd value is exactly known. If not, current won't show as zero for zero voltage. In order to increment the precision of the measurement you can provide the reference voltage using the optional ***vr*** parameter. In that case, the current is calculated as:

$$I_{dut} = 2\frac{V_{ADC2} - V_r}{R}$$

In order to obtain this Vr voltage you just need tho measure the voltage at ADC2 after disconnecting DAC1 from the circuit.

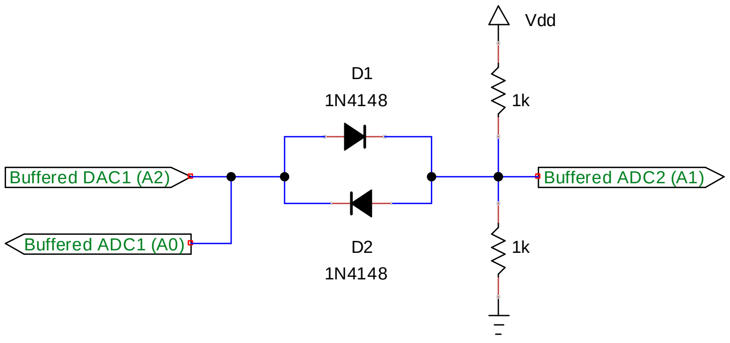
If parameter ***returnData*** is True or ***setPlotReturnData(True)*** was called previously, the command return a two element tuple with numpy arrays of:

- Vdut values
- Idut values

Example 04:

Measure the I(V) curve of two antiparallel 1N4148 diodes using the STM32 Nucleo F303RE board.

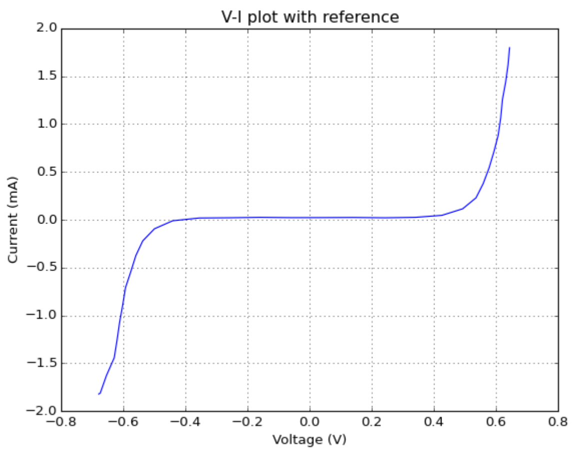
We will set the pair of diodes as DUT and buffer inputs and outputs to prevent DAC and ADC loading errors.



The following command will generate the I(V) curve. We sweep DAC 1 voltage between 0V and 3.2V and use the default values for **vi**, **r** and **wt**.

```
>>> dc.curveVIref(0.0,3.2)
```

When the program runs we will obtain a curve like the following one:



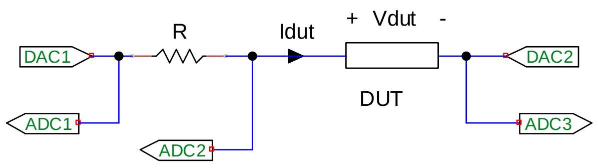
Note that current is no exactly 0 at 0 voltage. That is probably due to using two not exactly equal 1kΩ resistors. In order to increase the current measurement precision, the real reference voltage can be measured and be indicated as the optional ***vr*** parameter.

Current is limited to 3.2mA due to the use of 1kΩ resistors. The voltage drop on the diodes limits the maximum current below 2mA.

curveVIbridge (v1max, v2max, vi, r, wt, returnData)

Arguments	v1max	Maximum DAC 1 voltage	V1
	v2max	Maximum DAC 2 voltage	
	vi	Voltage step (Defaults to 0.1V)	
	r	Resistor value in kOhm (Defaults to 1k)	
	wt	Wait time (Defaults to 0.1s)	
	returnData	Return plot data (Defaults to False)	
Returns		See text. Draws a plot.	

The previous ***curveVIref*** enables us to obtain the I(V) curve of a two terminal device including negative voltages at expense of the positive voltage range. By using a bridge configuration, this command enables us to measure the negative voltage range of a device without compromising the positive range. In order to perform the measurement, the negative terminal of the device shall be connected to DAC 2 and ADC 3 and the positive terminal to ADC 2. A resistor shall also be connected between the DAC 1 output and ADC 1 input and the positive terminal of the DUT as shown in the following figure:



The command will obtain the I(V) curve in two stages. In the first stage, DAC 2 voltage is set to zero and DAC 1 voltage changes between 0 V and **v1max** with **vi** step increments. In the second stage, DAC 1 voltage is set to zero and DAC 2 voltage changes between 0 V and **v2max** with **vi** step increments. If **vi** is omitted, it defaults to 0.1 V.
A wait of **wt** seconds is included between each new DAC value and the measurement on ADC 1. If **wt** is omitted it defaults to 0.1s.
When the measurement ends, both DACs are set to zero to minimize power consumption.

From the measurements, voltage and current at the DUT is computed and the Idut(Vdut) curve is drawn.

$$V_{dut} = V_{ADC2} - V_{DAC3} \qquad I_{dut} = \frac{V_{ADC1} - V_{ADC2}}{R}$$

As in the two previous command, **r** value needs to be chosen both to obtain good current accuracy and to limit the current in the DUT. Current trough the DUT is limited to:

$$|I_{DUT}|_{Max} = \frac{Vdd}{r}$$

You need to ensure that DAC 1 buffer circuit can provide this current.

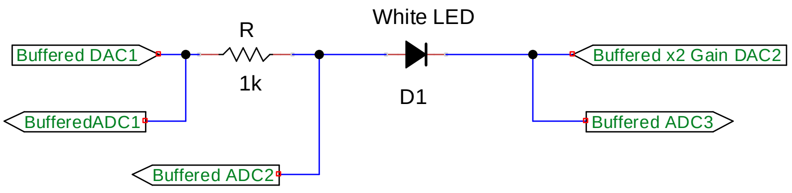
If parameter ***returnData*** is True or ***setPlotReturnData(True)*** was called previously, the command return a two element tuple with numpy arrays of:

- Vdut values
- Idut values

Example 05:

Obtain the I(V) response of a white led both for positive and negative voltages using the STM32 Nucleo F303RE board.

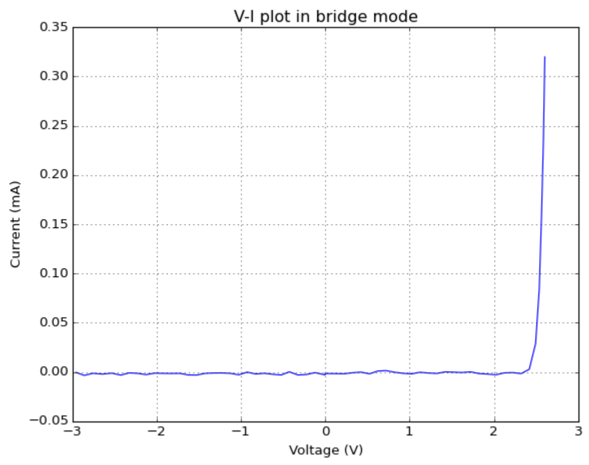
The threshold of a white LED is too high to use the ***CurveVIref*** command, so we will use the bridge configuration. All inputs and outputs will be buffered to prevent loading errors. As the DAC2 output on the F303RE board can only generate voltages up to half Vdd, we will use a gain 2 configuration that will, after calibration, enable us to set voltages in the buffered DAC 2 output up to Vdd.



The following program will generate the I(V) curve:

```
>>> slab.setDCreadings(400)
>>> dc.curveVIbridge(3.2,3.2,vmin=0.2)
```

First line increments the number of averaged readings for each measurement to reduce measurement noise. Second line generates the curve. We have set vmin to 0.2 V so that the lower range of the DACs is not used.
After the command, the following curve will be shown.

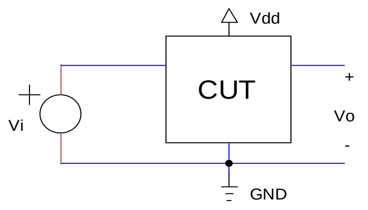


Maximum current is limited to 3.2mA due to the use of a 1kΩ resistor. As voltage drop in a blue LED is high (about 2.5V) current is limited to about 0.5mA. That limit can be calculated:

$$I_{max} = \frac{V_{DAC\ 1\ max} - V_{DAC\ 2\ min} - V_{D\ @\ Imax}}{r}$$

Input-Output voltage plots

Commands in this section obtain the voltage DC response of a circuit under test (CUT). That is, they obtain the output voltage for a sequence of input voltage values.



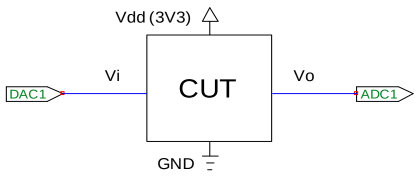
The following three commands ***CurveVV***, ***CurveVVref*** and ***HystVVcurve*** are useful in order to obtain the DC output voltage to input voltage curve of a circuit under test (CUT).

curveVV(v1, v2, vi, wt, adc2, returnData)

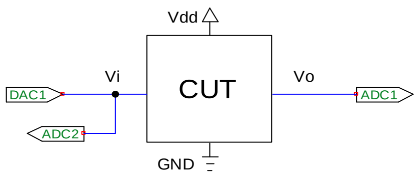
Arguments	v1	Start voltage	V1
	v2	End voltage	
	vi	Voltage step (Defaults to 0.1V)	
	wt	Wait time (Defaults to 0.1s)	
	adc2	Use ADC2 to measure Vi (Defaults to False)	
	returnData	Return plot data (Defaults to False)	
Returns		See text. Draws a plot.	

Draw voltage set with DAC 1 against voltage measured with ADC 1.
Measurements start with DAC 1 at **v1** voltage and end at **v2** voltage. Voltage is increased in a **vi** step between one measurement and the next one. If **vi** is omitted, it defaults to 0.1 V.
Between the change of the DAC voltage and the measurement with the ADC the module waits **wt** seconds. If **wt** is omitted it defaults to 0.1 s.

This command is useful for obtaining the DC voltage response of a circuit. The circuit under test (CUT), as shown in the following figure, is driven by an input voltage generated by DAC 1 and its output is read at ADC 1.



If optional parameter ***adc2*** is True, ADC2 will be used to measure the Vi voltage. It is useful if you want to be sure of the Vi voltage really set by DAC 1. The following figure shows the proper connections in this case.



If optional parameter ***returnData*** is True or ***setPlotReturnData(True)*** was called previously, the command return a two element tuple with numpy arrays of:

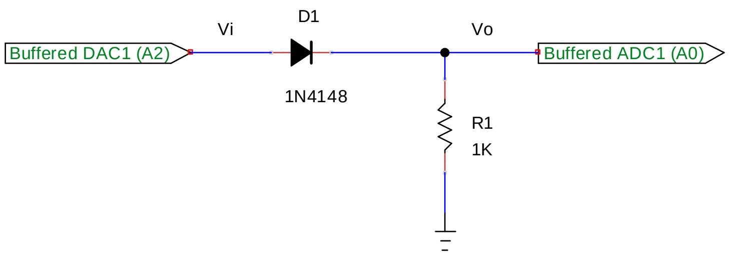
- Vi values
- Vo values

The same functionality of this command can be obtained with the ***dcSweepPlot*** command contained in the slab.py module. This command, however, is more simple and having it on the DC module gives this module a more complete set of commands.

Example 06:

Measure a voltage drop circuit using the STM32 Nucleo F303RE board

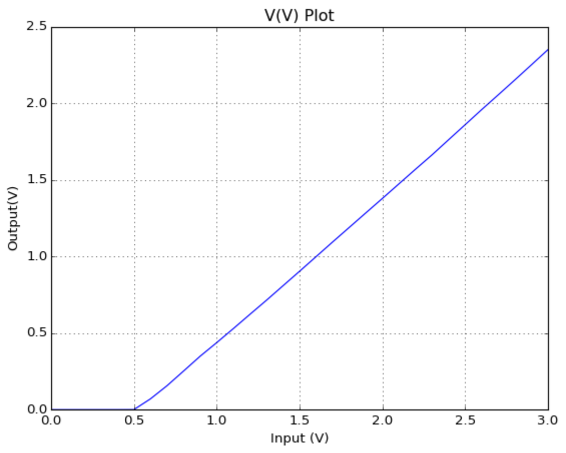
The voltage drop circuit is shown in the following figure. For Vi input voltages below 0.5V, D1 does not operate and output voltage is zero. For voltages over 0.5V, the output voltage is decreased respect to the input voltage due to the D1 voltage drop.
We will use a buffered setup, not shown in the figure, on the F303RE board. Two follower opamps connect to DAC 1 at pin A2 and ADC 1 at pin A0.



In order to test the Vo(Vi) curve of the circuit we can execute the following set of commands:

```
>>> dc.curveVV(0.0,3.0)
```

That will get a figure as the one shown below:



Observe that the diode drop is 0.5V only at the start, and that it increase until 0.65V at the end.

curveVVref(v1, v2, vi, wt, adc3, returnData)

Arguments	v1	Start voltage	V1
	v2	End voltage	
	vi	Voltage step (Defaults to 0.1V)	
	wt	Wait time (Defaults to 0.1s)	
	adc3	Use ADC3 to measure Vi (Defaults to False)	
	returnData	Return plot data (Defaults to False)	
Returns		See text. Draws a plot.	

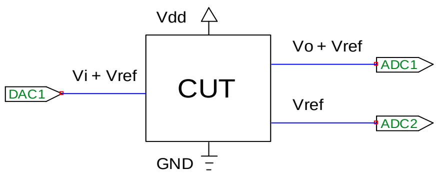
Obtain a circuit under test (CUT) Vo(Vi) voltage taking into account a not grounded reference for the circuit.

Set circuit input voltage with DAC 1 and measures output voltage with ADC 1.
Reference voltage of the circuit is measured with ADC 2.
Input Vi and output Vo voltages are calculated as:

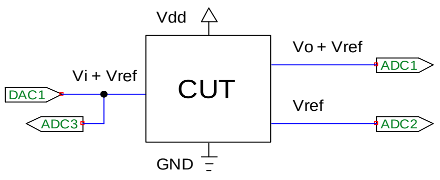
$$V_{ref} = V_{ADC\ 2} \qquad V_i = V_{DAC\ 1} - V_{ref} \qquad V_o = V_{ADC\ 1} - V_{ref}$$

Measurements start with DAC 1 at **v1** voltage and end at **v2** voltage. Voltage is increased in a **vi** value between one measurement and the next one. If **vi** is omitted, it defaults to 0.1 V.
Between the change of the DAC voltage and the measurement with the ADC the module waits **wt** seconds. If **wt** is omitted it defaults to 0.1 s.
From the measurements, Vi and Vo are calculated and plotted.

This command is useful for obtaining the DC voltage response of a circuit that uses a not grounded reference. The circuit under test (CUT), as shown in the following figure, is driven by an input voltage generated by DAC 1, its output is read at ADC 1 and the reference is read at ADC2.



If optional parameter ***adc3*** is True, ADC3 will be used to measure the Vi voltage. It is useful if you want to be sure of the Vi voltage really set by DAC 1. The following figure shows the proper connections in this case.

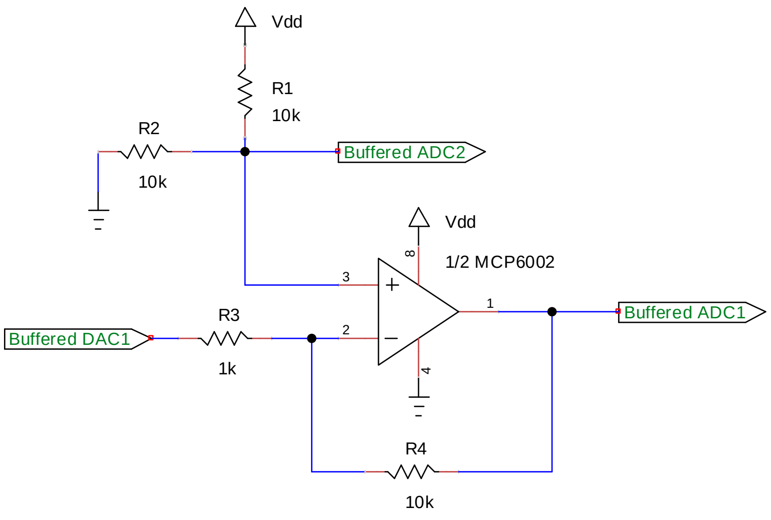


If parameter ***returnData*** is True or ***setPlotReturnData(True)*** was called previously, the command return a two element tuple with numpy arrays of:

- Vi values
- Vo values

Example 07:
Obtain the Vo(Vi) curve of a 10 gain operational amplifier inverter circuit with reference.

The following figure shows the circuit we want to test:

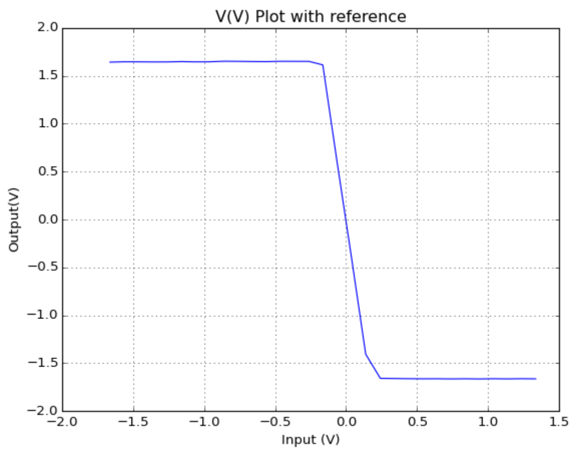


The circuit uses one of the two full-rail operational amplifiers included in the MCP6002 integrated circuit. Resistors R3 and R4 set the gain between DAC1 and ADC 1 to -10. Resistors R1 and R2 set the reference voltage for the circuit to halfway between Vdd and GND.

The following command performs the measurement:

```
>>> dc.curveVVref(0.0,3.0)
```

That will give a figure like the following one:

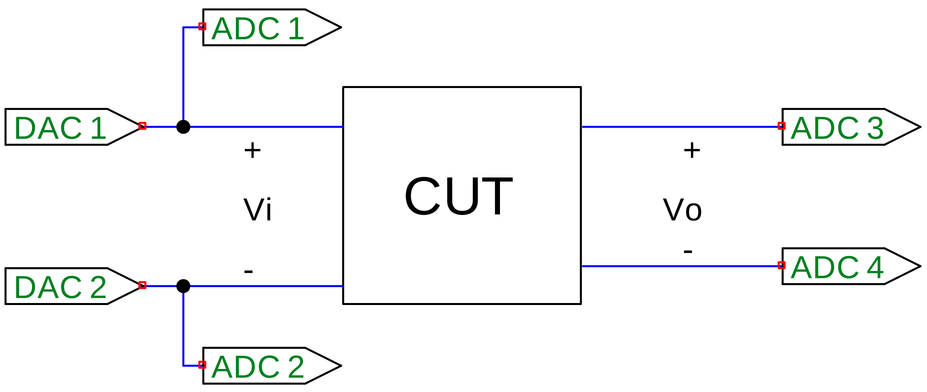


We see that output is zero when input is zero (respect to the Vdd/2 reference). We can also check that the gain is really -10.

curveVVbridge(vp, vn, vi, vmin, wt, returnData)

Arguments	vp	Positive maximum voltage	V1
	vn	Negative maximum voltage	
	vi	Voltage step (Defaults to 0.1V)	
	vmin	Minimum DAC voltage (Defaults to 0 V)	
	wt	Wait time (Defaults to 0.1s)	
	returnData	Return plot data (Defaults to False)	
Returns		See text. Draws a plot.	

Obtain a circuit under test (CUT) Vo(Vi) using a bridge configuration.



The command generates two sweeps:

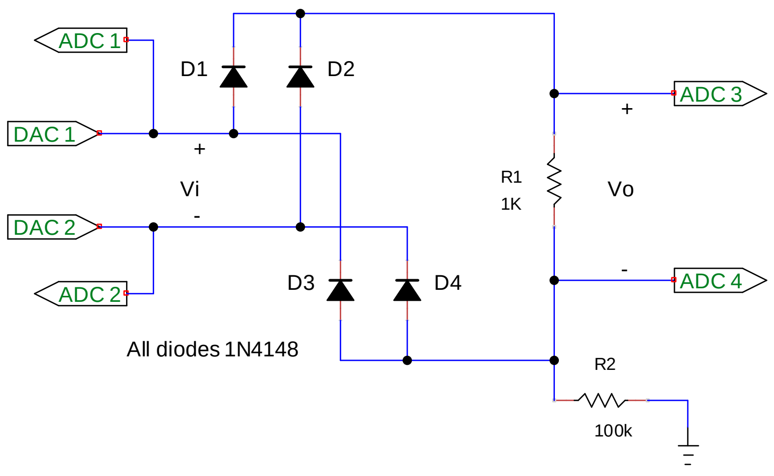
In the first sweep, DAC 2 is set to zero and DAC 1sweeps from ***vmin*** to ***vp*** in ***vi*** increments.
In the second sweep, DAC 1 is set to zero and DAC 1 sweeps from ***vmin*** to **vn** in **vi** increments.

Input and output voltages are defined as:

$$Vi = V_{ADC1} - V_{ADC2} \qquad Vo = V_{ADC3} - V_{ADC4}$$

Example 26:
Obtain the Vo(Vi) curve of a diode bridge.

The following figure shows the circuit we want to test:

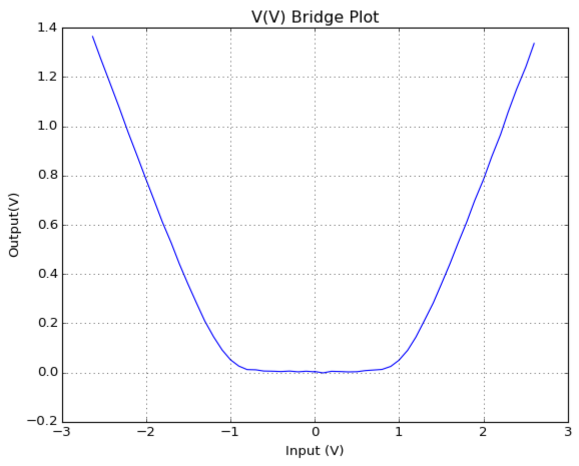


We have added a 100 kΩ R2 resistor from the output negative terminal to ground because, when no diode conducts, the output floats and that will provide bad measurements.

We increase the number of readings to reduce measurement noise and perform a Vo(Vi) bridge curve. We also set the minimum DAC voltage to 0.2 V to prevent working on the lower end of the DAC range.

```
>>> slab.setDCreadings(400)
>>> dc.curveVVbridge(3.2,3.2,vmin=0.2)
```

The obtained curve is:



hystVVcurve(v1, v2, vi, wt, returnData)

Arguments	v1	Start voltage	V1
	v2	End voltage	
	vi	Voltage step (Defaults to 0.1V)	
	wt	Wait time (Defaults to 0.1s)	
	returnData	Return plot data (Defaults to False)	
Returns		See text. Draws a plot.	

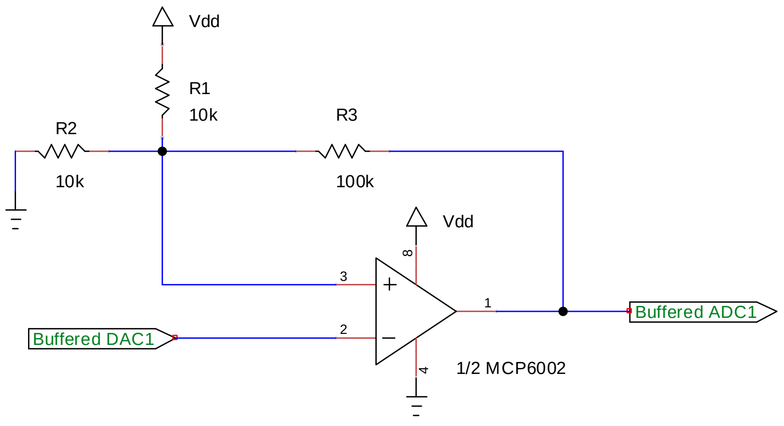
Some circuits have a DC response that changes if the input voltage is rising or falling. The hysteresis curve command is similar to the ***curveVV*** command, but measures the response of the circuit both in the forward direction with DAC 1 voltage going from **v1** to **v2** and in the back direction with DAC 1 voltage going from **v2** to **v1**. In both cases voltage is measured at ADC 1.

If ***returnData*** is True or ***setPlotReturnData(True)*** was called previously, the command return a four element tuple with numpy arrays of:

- Vi forward values
- Vo forward values
- Vi reverse values
- Vo reverse values

Example 08:
Measure a hysteresis inverting comparator curve using the STM32 Nucleo F303RE board

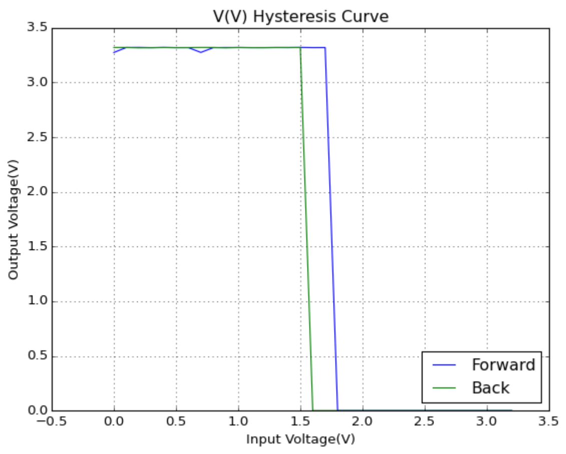
The circuit to test is shown in the following figure. R1 and R2 set the center of the comparator change at Vdd/2 that is about 1.65V as the F303RE board is powered at 3.3V. R3 sets the hysteresis of the comparator.



In order to see the IO curve we can issue the command that sweeps DAC 1 between 0V and 3V both rising (Forward) and falling (Back).

```
>>> slab.hystVVcurve(0.0,3.0)
```

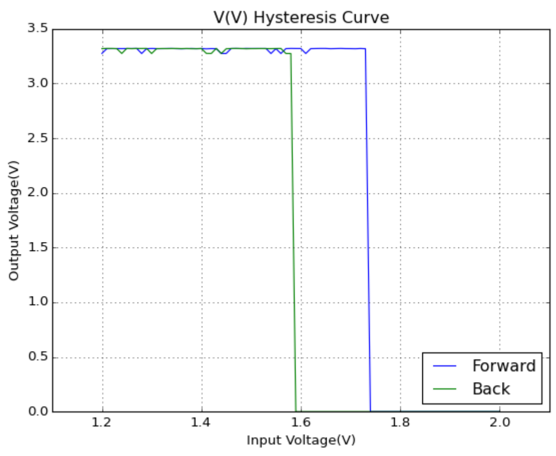
We will obtain a curve like the one below:



It seems that forward (blue) and back (blue) curves are not vertical. This is due to the default 0.1V step of the previous command. That means that we have, for instance on the input in the back direction, one point at 1.6V and another at 1.5V.
If we want more precise curve around the change zone of the comparator we can perform a finer 10mV step between 1.2V and 2V:

```
>>> slab.hystVVcurve(1.2,2.0,0.01)
```

That will give the following curve:



From the curve we can see that the limits are 1.58V in the back green curve and 1.73V in the forward blue curve. The mean value between those limits is 1.655V that is quite near to the 1.65V expected value.

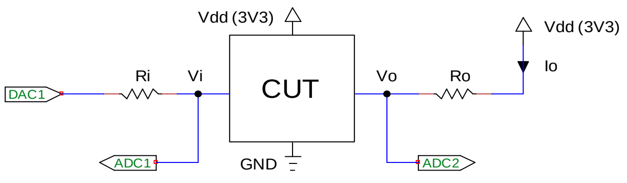
Current output transfer curves

The following two commands ***transferCurveVI*** and ***transferCurveII*** are designed to measure circuits whose output is defined on current instead of voltage.

transferCurveVI(v1, v2, vi, wt, ro, returnData)

Arguments	v1	Start voltage	V1
	v2	End voltage	
	vi	Voltage step (Defaults to 0.1V)	
	wt	Wait time (Defaults to 0.1s)	
	ro	Output resistor value in kOhm (Defaults to 1k)	
	returnData	Return plot data (Defaults to False)	
Returns		See text. Draws a plot.	

This command measures the output current of a circuit under test (CUT) against its input voltage. The following figure shows the setup of connections to perform the measurement. Again, buffers are not shown to simplify the drawing.



Depending on the circuit Ri can be omitted. Sometimes, however, it will interest to limit the input current on the circuit using this resistor. Either way, it is not used in the calculations.

The command will set DAC 1 voltage between **v1** and **v2** with **vi** step increments. If **vi** is omitted, it defaults to 0.1 V. A wait of **wt** seconds is included between each new DAC 1 value and the measurement on ADC 1 and ADC 2. If **wt** is omitted it defaults to 0.1s.
When the measurement ends, DAC 1 voltage is set to zero to minimize power consumption.

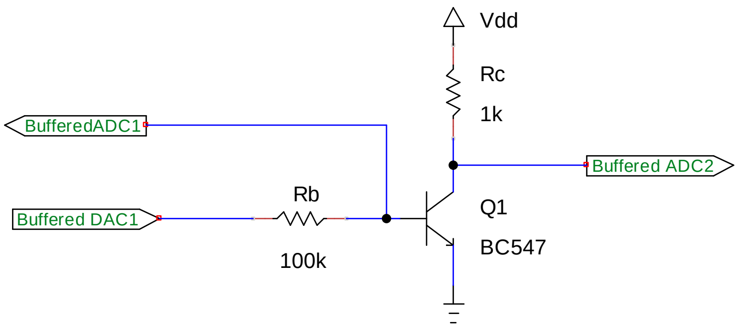
From the measurements, a Io(Vi) curve will be shown.

If parameter ***returnData*** is True or ***setPlotReturnData(True)*** was called previously, the command return a two element tuple with numpy arrays of:

- Vi values
- Io values

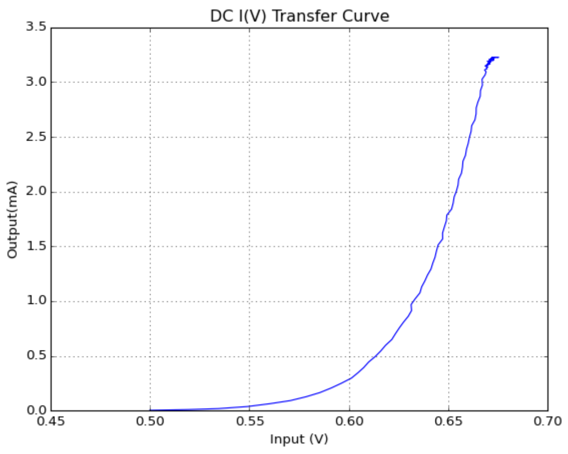
Example 09:
Measure the collector current against the base voltage on a BC547 bipolar junction transistor (BJT) using a STM32 Nucleo F303RE board

We will setup the following circuit:



In order to obtain the Ic(Vbe) curve we issue the following commands. First we increase the number of measurements to average to increase the accuracy and then we draw the curve. We have used a fine 20mV step to provide a better representation at the lower range of the curve.

```
>>> slab.setDCreadings(100)
>>> dc.transferCurveVI(0.5,3.0,0.02)
```



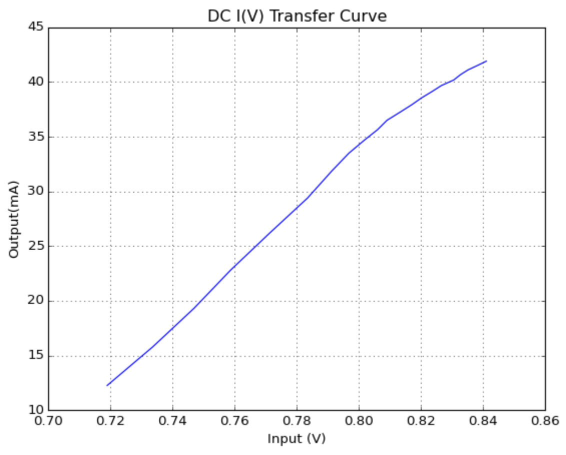
As ro is 1kΩ and the transistor saturates at V_{CE} of about 0.2V, maximum current before saturation is about 3.1mA. So, points above that current are not guaranteed to be in the linear active region of the BJT.

If we want to operate at higher currents we can reduce ri and ro. We want to limit the collector current to 50mA because this transistor cannot operate above 100mA and because the nucleo board cannot provide more than 300mA to all connected elements. Using a 68Ω ro value we will enter saturation at about 45mA. We will also decrease ri to 8.2kΩ so that there is enough base current to have a high collector current.

The following command performs the measurements. As the **vi**, **wt** and **ro** are not in the proper order, the name of the parameter shall be included. That way you don't need to remember the order of the optional parameters. Note also that ro is given in kΩ. This time we don't need to use a 20mV step because we start on DAC 1 at 1V instead of 0.5V.

```
>>> dc.transferCurveVI(1.0,3.0,vi=0.1,wt=0.1,ro=0.068)
```

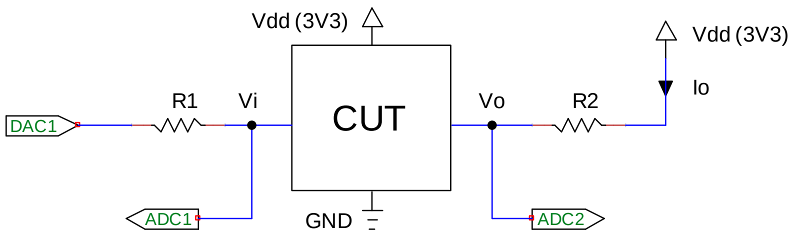
The obtained curve will be:



transferCurveII(v1, v2,vi ,r1, r2, wt, returnData)

Arguments	v1	Start voltage	V1
	v2	End voltage	
	vi	Voltage step (Defaults to 0.1V)	
	r1	Input resistor value in kOhm (Defaults to 1k)	
	r2	Output resistor value in kOhm (Defaults to 1k)	
	wt	Wait time (Defaults to 0.1s)	
	returnData	Return plot data (Defaults to False)	
Returns		See text. Draws a plot.	

This command is similar to the previous one and measures the output current of a circuit under test (CUT) against its input current. The connections are the same than in the previous case.



As we need Ri to compute the input current, this time, this resistor cannot be omitted.

The command will set DAC 1 voltage between **v1** and **v2** with **vi** step increments. If **vi** is omitted, it defaults to 0.1 V. A wait of **wt** seconds is included between each new DAC 1 value and the measurement on ADC 1 and ADC 2. If **wt** is omitted it defaults to 0.1s.
When the measurement ends, DAC 1 voltage is set to zero to minimize power consumption.

From the measurements, a Io(Ii) curve will be shown.

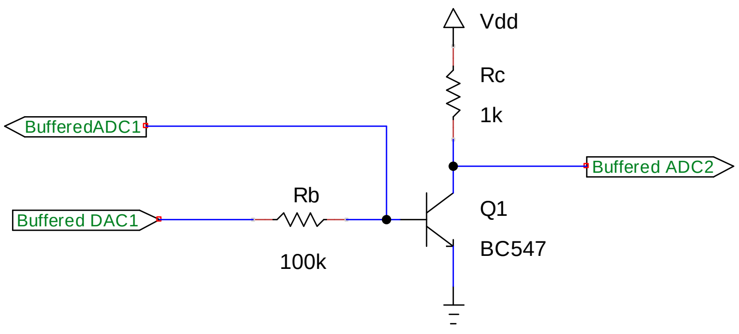
If ***returnData*** is True or ***setPlotReturnData(True)*** was called previously, the command return a two element tuple with numpy arrays of:

- Ii values
- Io values

Example 10:

Measure the collector current against the base current on a BC547 transistor using a STM32 Nucleo F303RE board.

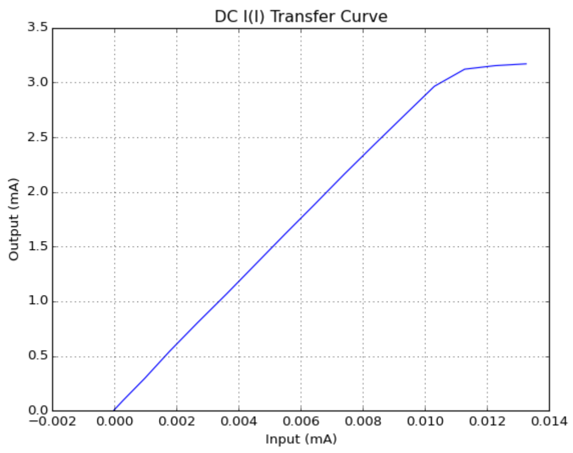
We will setup the same as in the example 09:



In order to obtain the Ic(Vbe) curve we issue the following command. Parameteres **ro** and **wt** use their default values of 1kΩ and 0.1s.

```
>>> dc.transferCurveII(0.5,2.0,0.1,100.0)
```

We obtain the following graph where we observe a constant β ratio between Ic and Ib up to the point where the transistor enters in the saturation region.

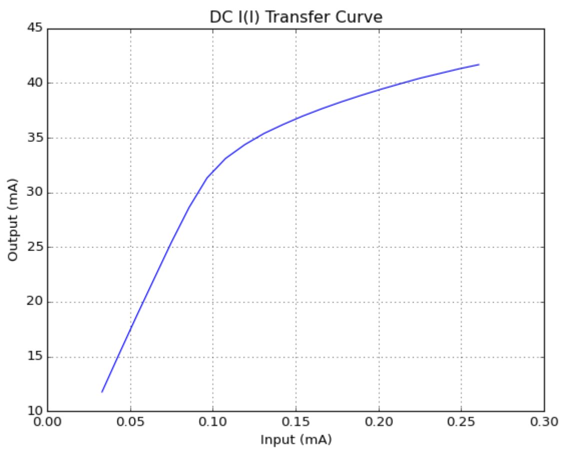


At low currents, β of the transistor is reasonably constant. This parameter, however, degrades at high currents. If we want to operate at higher currents we can reduce ri and ro. Using a 68Ω **ro** value we will enter saturation at about 45mA. We will also reduce **ri** 8.2kΩ to provide enough base current.

Before issuing the ***transferCurveII*** command we will also increase the number of readings to reduce the measurement errors.

```
>>> slab.setDCreadings(50)
>>> dc.transferCurveII(1.0,3.0,0.1,8.2,0.068)
```

That will get the following figure:



We can see that β degrades at currents over 30mA. at currents over 30mA.

So far, there are commands to obtain the output to input curves in several situations:

Input	Output	Commands
Voltage	Voltage	CurveVV CurveVVref HystVVcurve
Voltage	Current	TransferCurveVI
Current	Current	TransferCurveII

Those commands don't include all possibilities. For instance, there is no command for circuits with current input and voltage output. Moreover, hysteresis and negative ranges are only measured in voltage to voltage circuits. As all this commands rely on the Basic DC commands, it is possible to define any new commands if the stock defined ones are not suited for a particular measurement.

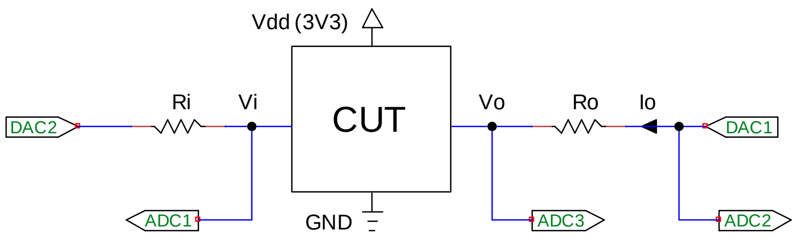
Three terminal device curves

The following two commands ***vDeviceCurve*** and ***iDeviceCurve*** are designed to obtain the characteristic curves of three terminal devices like transistors. The first one is for devices with input defined by a voltage and the second one is for devices with input defined by a current.

vDeviceCurve(vi1, vi2, vii, vo1, vo2, voi, ro, wt)

Arguments	vi1	Input start voltage	
	vi2	Input end voltage	
	vii	Input voltage step	
	vo1	Output start voltage	
	vo2	Output end voltage	V1
	voi	Output voltage step (Defaults to 0.1V)	
	ro	Output resistor value in kOhm (Defaults to 1k)	
	wt	Wait time (Defaults to 0.1s)	
Returns		Nothing. Draws a plot.	

The two previous commands show the output current of a CUT against a variable input but they don't control the output voltage. This command shows the output curves Io(Vo) at several input voltages. The circuit is similar to the one in the previous two commands but this time we also use DAC 2.



Depending on the circuit Ri can be omitted. Sometimes, however, it will interest to limit the input current on the circuit using this resistor. Either way, it is not used in the calculations.

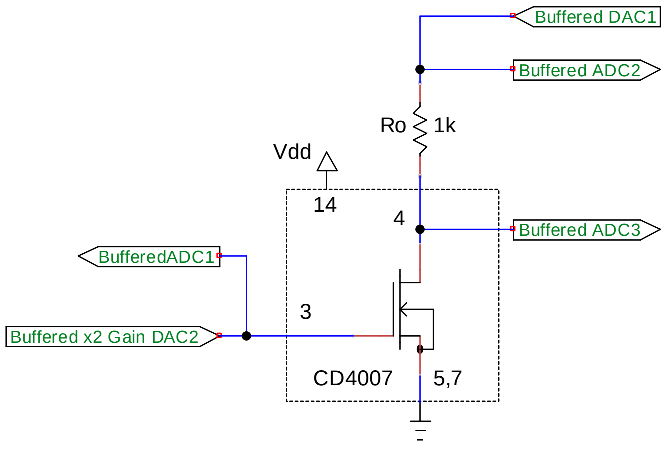
The command will set DAC 2 voltage between **vi1** and **vi2** with **vii** step increments and measure the input voltage **Vi**. Then it measures the Io(Vo) curve using DAC 1 values between **vo1** and **vo2** with **voi** increments. If **voi** is omitted, it defaults to 0.1 V. A wait of **wt** seconds is included between each new DAC 1 value and the measurement on ADC 2. If **wt** is omitted it defaults to 0.1s.

From the measurements, a family of Io(Vi) curves, one for each Vi value, will be shown.
This command only gives meaningful results if the input voltage Vi does not depend on the output voltage Vo or output current Io.

Example 11:

Measure the curves of a NMOS transistor from a CD4007 package using a STM32 Nucleo F303RE board

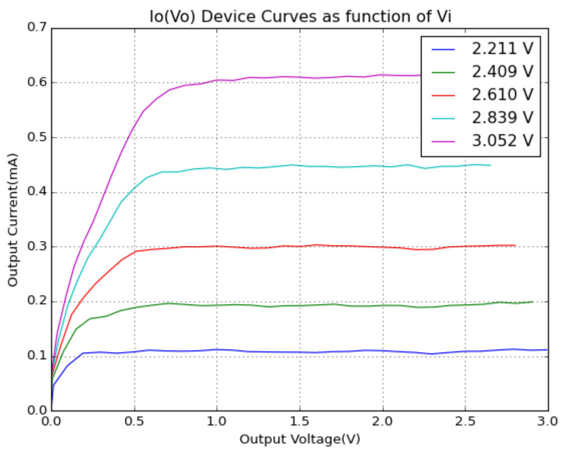
The CD4007 is a 14 pin integrated circuit that includes six MOS transistors. Two NMOS and two PMOS. The following circuit enables us to measure one of the included NMOS transistors. Numbers inside de CF4007 box relate to pin numbers.



The following command shows the transistor curves for gate voltages from 2.2V to 3.2V in 0.2V steps. At each gate voltage an Id(Vds) curve is shown.

```
>>> dc.vDeviceCurve(2.2,3.2,0.2,0.0,3.2,0.1)
```

That will give the family of curves:

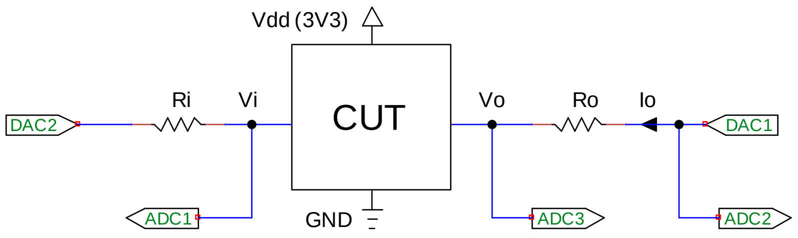


We note that the output voltage range is reduced as the current increases due to the voltage drop at resistor **ro**.

iDeviceCurve(vi1, vi2, vii, vo1, vo2, voi, ri, ro, wt)

Arguments	vi1	Input start voltage	V1
	vi2	Input end voltage	
	vii	Input voltage step	
	vo1	Output start voltage	
	vo2	Output end voltage	
	voi	Output voltage step (Defaults to 0.1V)	
	ri	Input resistor value in kOhm (Defaults to 1k)	
	ro	Output resistor value in kOhm (Defaults to 1k)	
	wt	Wait time (Defaults to 0.1s)	
Returns		Nothing. Draws a plot.	

This command is similar to the previous one. It shows the output curves Io(Vo) at several input currents. The circuit is the same of the previous command.



As we need Ri to compute the input current, this time, this resistor cannot be omitted.

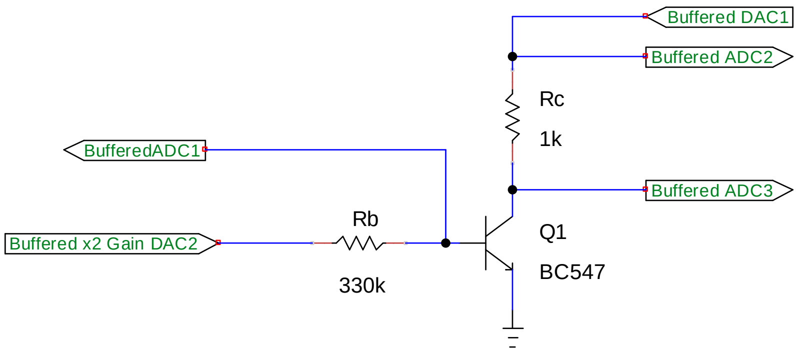
The command will set DAC 2 voltage between **vi1** and **vi2** with **vii** step increments and measure the input current **Ii**. Then it measures the Io(Vo) curve using DAC 1 values between **vo1** and **vo2** with **voi** increments. If **voi** is omitted, it defaults to 0.1 V. A wait of **wt** seconds is included between each new DAC 1 value and the measurement on ADC 2 and ADC 3. If **wt** is omitted it defaults to 0.1s.

From the measurements, a family of Io(Vi) curves, one for each Ii value, will be shown.
This command only gives meaningful results if the input current Ii does not depend on the output voltage Vo or output current Io.

Example 12:

Measure the collector current curves against the base current on a BC547 transistor using a STM32 Nucleo F303RE board

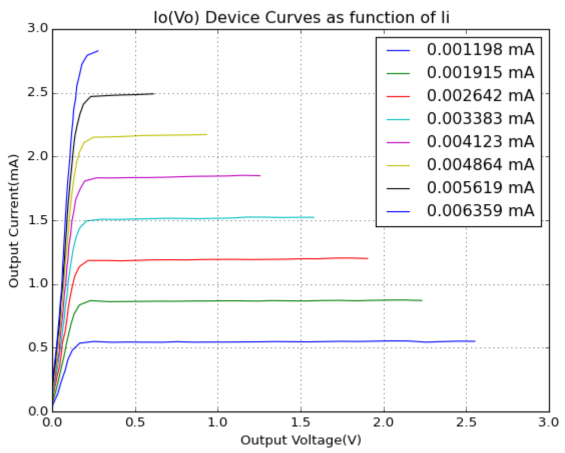
The following circuit enables us to obtain the curves for the transistor:



Using the following command we can show curves for DAC 2 values between 1V and 3V in 0.25V steps. We use the default ro value of 1kΩ. Note that, as base current is not forced but measured, we cannot generate curves for exact values of the base current.

```
>>> dc.iDeviceCurve(1.0,3.0,0.25,0.0,3.2,0.1,ri=330.0)
```

The command will yield the following family of currents.



Observe that output voltage range reduces as current increases due to the voltage drop on **ro**.

References

STM32 Nucleo Page

http://www.st.com/en/evaluation-tools/stm32-mcu-nucleo.html

MBED Developer Page

https://developer.mbed.org/

Python page:

https://www.python.org/

Anaconda page:

https://www.continuum.io/downloads

TinyCad

https://sourceforge.net/projects/tinycad/

Circuit images on this document have been drawn using the free software TinyCad