SET : set DPS output voltage or current to specified new value
1st parameter: V or C (for voltage or current)
2nd parameter: new value (in volts, or amps)

Example:

```
SET       V   10.0   # set voltage to 10.0V

SET       C   2.3    # set current to 2.3A
```

Note:

No check if the SET values are actually possible with that DPS, its input voltage and current, or the device under test !

INC : increments or decrements DPS output voltage or current by specified value

1st parameter: V or C

2nd parameter: increment (can be negative)

Example:

```
    SET        V  1.0    # set voltage to 1.0V
    INC        V  4.5    # increase by 4.5, voltage is now 5.5V
    INC        V -0.5    # decrease by 0.5, voltage is now 5.0V
```

Note:
If the result of INC would be negative, the value is set to 0

No check if the INC values are actually possible with that DPS, its input voltage and current, or the device under test !

OUTPUT : turns DPS output on or off
1st parameter: ON or OFF

Example:

```
OUTPUT    OFF       # turn output off

OUTPUT    ON        # turn output back on
```

MAX :   set DPS protection values to specified new value
1st parameter: V or C or P
2nd parameter: new value

Example:

```
    MAX      V   5.5    # set over-voltage protection to 5.5V

    MAX      C   0.8    # set over-current protection to 800mA

    MAX      P   5      # set over-power protection to 5W
```

Note:

1. No check if the MAX values are actually possible with that DPS
2. If any of the protection values trigger the program prints a warning and stops execution. You must clear the fault on the DPS front panel.

IF : set a condition for the next WAIT or GOTO instruction

- 1st parameter: V or C or P
- 2nd parameter: condition (`<  <=  ==  >=  >`)
- 3rd parameter: value to compare against

Example:

```
    IF        V   >= 4.9    # is voltage >= 4.9V ?
```

Note:

1. No check if the values are actually possible with that DPS
2. Only one active condition at any time
3. A WAIT or GOTO instruction always deletes the condition

WAIT : waits a specified number of seconds or until a condition is TRUE
1st parameter: time in seconds

Example:

```
SET        V  14.5
SET        C  2.0
OUTPUT     ON

IF         C  >= 0.01      # wait until battery is connected
WAIT       0               # conditional wait
WAIT       5               # timed wait

IF         V  >= 14.48V    # max voltage = charge complete
WAIT       36000           # or 10 hours have elapsed
OUTPUT     OFF
```

Note:

1. A WAIT or GOTO instruction always deletes the condition
2. Depending on battery, the condition may not be reachable, hence the timeout
3. If you try this, add a beefy diode between DPS and the battery to prevent any back-feed from the battery into the DPS.

GOTO : jumps to the specified target when no active condition or condition is TRUE
continues with next instruction if condition is FALSE
1st parameter: label to jump to (without colon)

Example:

```
        SET       V   0

UP:     INC       V   0.5
        WAIT      5
        IF        V   <    4.98     # still less than 5V ?
        GOTO      UP                # Yes, do more incrementing

        WAIT      60                # No, stay 5V peak for a bit

DOWN:   INC       V   -1
        WAIT      1
        IF        V >= 1            # still above 0V ?
        GOTO      DOWN              # Yes, do more decrementing

        OUTPUT    OFF               # No, all done
```

RECORD : turns recording on or off and sets the recording level

1st parameter: level

0 = off,
1 = instruction based,
2 = time based,
3 = change based

2nd parameter: time in seconds (only used for level 2)

Example:

```
      RECORD    1  0         # record only after SET, INC ..
      # some instructions
      RECORD    2  10        # record every 10 seconds
      # some instructions
      RECORD    3  0         # record detected changes
      # some instructions
      RECORD    0  0         # turn recording off
```

Note:

1. the 2nd parameter is needed in all cases, but the value does not matter except for level 2

CALL : calls an external (operating system) command

1st parameter: command

2nd parameter: empty or parameters for command

3rd parameter: empty or more parameters for command

Example:

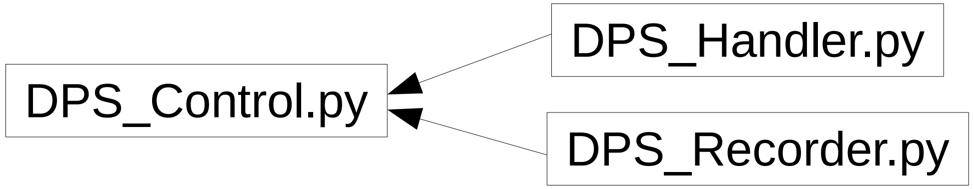
```
    # calling an external batch/script file to obtain a measurement value (e.g. from a multimeter)
    CALL ‘MEASURE.BAT’   ‘>$F’ # Windows version
    CALL ‘./measure.sh’  ‘>$F’ # Linux version

    # calling an external program to take a snapshot photo from a webcam
    CALL 'ffmpeg -f dshow -i video="Webcam C920" -vframes 1 -loglevel quiet -y ' '${R}_$N.jpg  # windows
    CALL 'ffmpeg -i /dev/video0 -vframes 1 -loglevel quiet -y ' '${R}_$N.jpeg"  # linux
```

Note:

1. If recording is not turned on, CALL instructions are ignored
2. Every CALL instruction executed is automatically sequentially numbered (1, 2..) and that number is included in the recorded data.

Call key values	Substituted to:	Example:		Notes
		Before	After	
`$D`	date+time string	`File$D.TXT`	`File20190522200142.TXT`	Changes with every call and every run of program
`$N`	Call number	`File$N.JPG`	`File0001.JPG`	Changes with every call, same for multiple program runs
`$R`	Same as recording file name	`File$R.JPG`	`File20190522195848.JPG`	Changes with every run, stays the same during the run.
`$F`	Same as recording file name with prefix _ and extension .tmp	`$F`	`_20190522195848.tmp`	Changes with every run, stays the same during the run. After the call the program will read the 1st line in the file and insert the content into the recording. **The file is then automatically deleted**
`$$`	$	`text$$sample`	`text$sample`	Use if you need the $ character in your parameter string

`${D}`
`${N}`
`${R}`
`${F}`
	(same as $D)
	(same as $N)
	(same as $R)
	(same as $F)
		`Text${D}_$N.TXT`
		`TEST${N}FILE`
		`PIC${R}_$N.JPG`
			`Text20190522200142_0001.TXT`
			`TEST0001FILE`
			`PIC20190522195848_0001.JPG`
				Use for example if $D is followed directly by character(s) with no space (or dot or $) character(s) in between



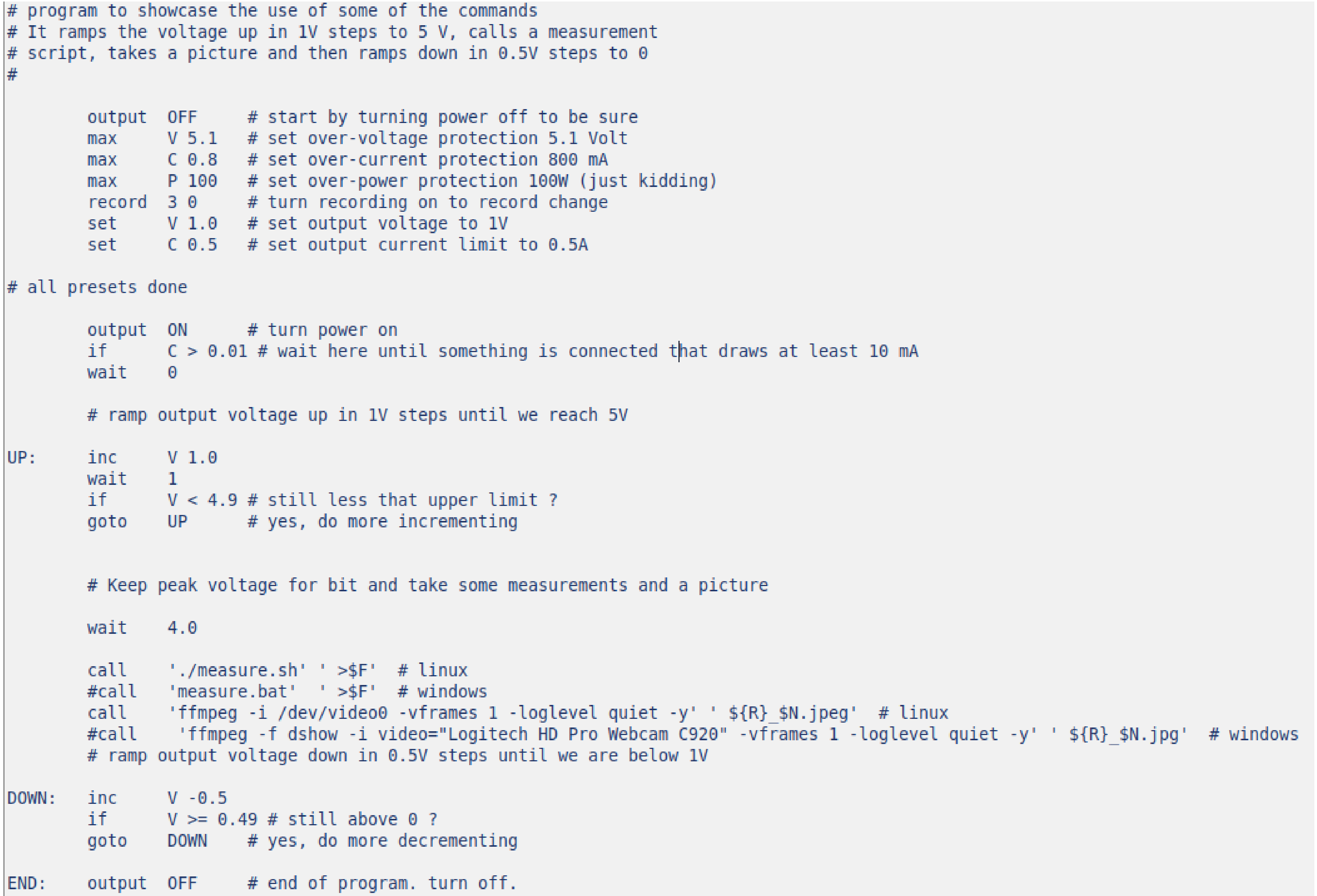
DPS_Control.py program-file --port <port> –speed <speed> -d <debug level>

program-file: text file that contains the instructions to be executed

<port>: serial port number (default: port that has a HL-340 USB serial adapter)
<speed>: default is 19200
<debug level>: default is 1 (trace execution),
other values: 0 = off (silent execution) 2= trace and parser

Example:

```
Windows:
            C:\Users\Test\Test\py DPS_Control.py program.txt
            C:\Users\Test\Test\py DPS_Control.py program.txt --port COM7
Linux:
            user:/test$ python3 DPS_Control.py program.txt
            user:/test$ python3 DPS_Control.py program.txt --port /dev/rfcomm0
```



```
# program to showcase the use of some of the commands
# It ramps the voltage up in 1V steps to 5 V, calls a measurement
# script, takes a picture and then ramps down in 0.5V steps to 0
#

        output  OFF     # start by turning power off to be sure
        max     V 5.1   # set over-voltage protection 5.1 Volt
        max     C 0.8   # set over-current protection 800 mA
        max     P 100   # set over-power protection 100W (just kidding)
        record  3 0     # turn recording on to record change
        set     V 1.0   # set output voltage to 1V
        set     C 0.5   # set output current limit to 0.5A

# all presets done

        output  ON      # turn power on
        if      C > 0.01 # wait here until something is connected that draws at least 10 mA
        wait    0

        # ramp output voltage up in 1V steps until we reach 5V

UP:     inc     V 1.0
        wait    1
        if      V < 4.9 # still less that upper limit ?
        goto    UP      # yes, do more incrementing


        # Keep peak voltage for bit and take some measurements and a picture

        wait    4.0

        call    './measure.sh' ' >$F'  # linux
        #call   'measure.bat'  ' >$F'  # windows
        call    'ffmpeg -i /dev/video0 -vframes 1 -loglevel quiet -y' ' ${R}_$N.jpeg'  # linux
        #call    'ffmpeg -f dshow -i video="Logitech HD Pro Webcam C920" -vframes 1 -loglevel quiet -y' ' ${R}_$N.jpg'  # windows
        # ramp output voltage down in 0.5V steps until we are below 1V

DOWN:   inc     V -0.5
        if      V >= 0.49 # still above 0 ?
        goto    DOWN    # yes, do more decrementing

END:    output  OFF     # end of program. turn off.
```