Instrument control using SCPI and Bash scripts

Posted on **April 1, 2014**

You don't need a bloated LabView installation for controllilng your lab instruments. If they have a LAN connector, you can probably talk to them using simple bash (or cmd or pyton or …) scripts.

Instrument Requirements

- LAN-port
- SCPI support (see here for the shortest introduction possible)
- Working network connection
  - Try to find out the instrument's IP-address and on what ports it listens on. Standard port is 5025.
  - Careful: Often there is no username/password, so anyone in your network can control your instrument! I suggest to set up a dedicated lab subnet.

High level overview

- Hook up instrument to LAN
- Connect to port 5025
- Send SCPI commands
- Receive data and redirect it to a file

Hooking up instrument to LAN

Plug a LAN connector into your lab instrument and power it on. Many lab instruments support DHCP and will automatically get an IP address. Find the IP-adress by going through all instrument menus that are related to "SYSTEM" or "REMOTE".

Connecting to port 5025

You can use telnet for a test connection (see Windows section below for an example). It is available on most Windows and Linux installations. But on Linux (including Ubuntu), the prettier alternative is netcat.

Ubuntu

Open a terminal and install netcat (should be installed already):

```
sudo apt-get install netcat
```

Open a netcat session to the lab instrument:

```
netcat 149.100.100.5 5025
```

Here, 149.100.100.5 is the instrument's IP adress, and 5025 ist the port that we want to connect to on the instrument. Port 5025 is the standard port for SCPI commands.

Query the instrument for its ID number by typing

```
*IDN?
```

Press enter and the device will return its ID. Press `CTRL+C` to exit the netcat session.

You can also run netcat non-interactively by sending input commands through a "pipe" | :

```
echo "*IDN?" | netcat -q 1 149.100.100.5 502
```

The option "-q 1" makes netcat wait for 1 second for the answer from the instrument. You only need this option for queries. (You don't strictly need it – but I had a few queries fail before using that wait option.)

Redirect netcat's output to a file:

```
echo "*IDN?" | netcat -q 1 149.100.100.5 5025 > ans.txt
```

Then type

```
cat ans.txt
```

and you will see the contents of the file.

To set something, you can use

```
echo "FREQUENCY:CENTER 100MHz" | netcat 149.100.100.5 5025
```

if your device understands that command, its center frequency will now be set to 100MHz. This command has no output, so there is no point in redirecting it to a file.

Windows

On Windows, this is a bit harder. PuTTY comes with a command line utility called plink – but I can't get it to open raw connections. Installing netcat is tricky because antivirus software classifies it as risky (it's often included in malware). Telnet always works (also in corporate environments with strict antivirus settings), but telnet does not support scripting. We'll script it anyways.

Note: On Windows 7, you have to activate telnet first.

In a cmd prompt, type

```
telnet
```

It will greet you by saying something like “Welcome – the escape combination is CTRL++. (This is different for each Locale.) You will need this escape sequence later.

Type

```
o 149.100.100.5 5025
```

The bottom of the screen will now keep saying “Connecting”. Don’t be fooled by this – most likely you are already connected. Type

```
*IDN?
```

and press enter. The device should answer with its ID number, but it may take two or three tries, because backspace doesn’t work in Windows telnet.

Quitting: Press your escape key combination (in my case pressing CTRL and + simultaneously) to get to the telnet prompt. Type q, then enter, to quit.

Let’s script this. Telnet, by default, does not accept input pipes and cannot be scripted. We’ll be using a visual basic script that simulates key presses. It’s easier than it sounds.

Create a file called “telnet_commands.vbs”. Paste the following lines into it:

```
set OBJECT=WScript.CreateObject("WScript.Shell")
WScript.sleep 50
OBJECT.SendKeys "*IDN?{ENTER}"
WScript.sleep 500
OBJECT.SendKeys "^({+})"
WScript.sleep 50
OBJECT.SendKeys "q{ENTER}"
```

- `set OJBECT=...` Create a “shell” object that can do system stuff. This can be used to simulate key presses.
- `WScript.sleep 50` Wait 50ms.
- `OBJECT.SendKeys ...` Simulate button presses. These go to whatever program that has “focus” (is active) right now.
- `OBJECT.SendKeys "^({+})"` Press control key `^`, keep it pressed `()` while pressing plus key `{+}`. The plus sign can have special meaning, which is why it’s escaped by curly braces.

Create another file called query_idn.bat. Paste the following lines into it:

```
@echo OFF
 start telnet 149.100.100.5 5025 -f output.txt
 cscript telnet_commands.vbs
```

- `@echo OFF` Don't echo (repeat) this command in the output (the @ at the beginning), and switch echoing off for all subsequent commands.
- `start telnet...` Start a telnet session in its own window, write the output to a file
- `cscript ...` Execute the vbs file

Double-click the bat file and rejoice. The first line of the output file output.txt is bogus junk, but the second line has the interesting stuff.

This entry was posted in **Lab** and tagged **bash**, **SCPI** by **hoeckerson**. Bookmark the **permalink [http://hoeckerson.de/notes/2014/04/instrument-control-using-scpi-and-bash-scripts/]** .