Setting up GPIB-Linux

Daniel Jiménez
División de Fisica Aplicada, CICESE

June 10, 2014

In modern Linux systems, those with kernel versions 2.6 or newer, one can use a National Instruments USB-GPIB-HS interface with very little trouble by using the Linux-GPIB package and following these instructions. For maximum compatibility, you should make sure you are running a Debian-based system like Ubuntu or Linux Mint (only Linux Mint 15 has been tested). Please make sure you have a working internet connection and that your user account has permissions to run the `sudo` command; if there is only one user account, then this will most likely be the case.

1 Installation

If you are working on a newly installed system, it might be needed to install the compilation tools. In order to do so, please type the following at the terminal prompt:

```
sudo apt-get install build-essential
```

You will be asked to input your administrative password. This will install the compiler infra-structure and should only take a couple of minutes.

Navigate with a web browser to

```
sourceforge.net/projects/linux-gpib
```

and proceed to download the latest version of the package (As of writing, 3.2.19). Decompress the contents in an accessible location on the filesystem (most easily done by double clicking on the compressed file on most graphical systems). Consider reading the README file that is included in the files.

Compile the source drivers by passing the following commands to the terminal prompt:

```
./configure
```

```
make
```

```
sudo make install
```

This last command will probably request your password once again, please provide it and wait while the installation takes place. After this has been completed, the device drivers for GPIB devices have been installed and are almost ready to be used; you can either restart the computer to load the new kernel modules or else type the following command to do it manually:

```
sudo modprobe ni_usb_gpib
```

This completes the installation of the device drivers.

2 Configuration

There are many different GPIB boards which can be used with the installed drivers. The system requires specifying which device driver must be loaded by modification of a plain-text configuration file located at `/etc/gpib.conf`. This file is write-protected for regular users so it must be opened with a text editor with administrator privileges. A simple graphical editor which is usually available is `gedit`, you may run a super user instance of this program as

```
gksudo gedit /etc/gpib.conf &
```

Notice this is `gksudo` and not `sudo`! This is the correct form of the command for graphical programs. The ampersand `&` is used to prevent the terminal from being locked while the graphical program runs (and is not strictly necessary). If `gksudo` is not installed, then you can install it simply by typing

```
sudo apt-get install gksudo
```

In case `gedit` is not available, an old school terminal editor which is always* available is `nano`, you may run this as a super user with

```
sudo nano /etc/gpib.conf
```

Once within the editor you will see the contents of the file, which was placed there as a placeholder by the installation procedure, you must proceed to modify it by changing the `board type` variable to read `"ni_usb_b"` and, if you are going to work with the Ortec 974 counter. You may also choose to set up a device to read `pad = 4` (look at the end of the file), though this is to provide a facility which is not used in the code provided by the author (as you can tell by the fact that the Ortec 994 is not listed, yet it works well in the example code). You may also change the names of the board (from the `"violet"` default) and the device (perhaps to read `"counter"`). The rest of the parameters need not be changed as the defaults are good for our board.

This next part requires knowing which kernel version you are running. Either run the command `uname -a` or look around in the folder `/lib/modules/` for the largest version number. Write this number down. Now check that there is a folder with address `/etc/ld.so.conf.d/`. In case there is, continue with the following. Create a new text file as follows

```
gksudo gedit /etc/ld.so.conf.d/gpib.conf
```

and add the following line to it

```
/lib/modules/3.8.0-25-generic/gpib
```

Where the version number (here 3.8.0-25) corresponds to that which you wrote down. In case there isn't, then just add this last line to `/etc/ld.so.conf`.

This completes the configuration process.

3 Runtime

The contents of the configuration file which we have modified are put into effect by running the command

```
sudo gpib_config
```

from anywhere on the fiesystem. This requires the NI USB-GPIB-HS interface to be plugged into the usb socket and (possibly) into the GPIB device which is to be controlled. **This command needs to be run every time the computer is rebooted before you can access the GPIB bus!**.

After the configuration file has been properly loaded, we can launch any of our GPIB-enabled programs by prefixing the command `sudo` to it. If you are to use a particular program (such as my own `ortec_2ch` or any other), then you must navigate to the containing folder from the terminal environment (use the `cd` command to change the working directory) and finally run the program as

```
sudo ./ortec_2ch
```

Where the `./` part of the command indicates the program is located in the current directory (otherwise, the terminal will look for a program in the location(s) where the system-installed programs reside).

4 Ortec 974 test procedure

Once the configuration file is loaded, you can test the bus by using the provided utility `ibtest`. Since this is an installed program, you can run it from any location in the filesystem by issuing the following command

```
sudo ibtest
```

This requires the NIM bin to be turned on, as well as the NI USB-GPIB-HS board to be plugged into the computer.

While running this program, we can choose to open the Ortec counter by choosing to open a device (type `d`, then press `enter`) and then choosing the primary address as 4 (type `4`, then press `enter`). Now we may choose to read a data string from the device (choose `r` and the default number of bytes, the startup string `%001000070` should appear on the screen).

Press the `N` and/or `M` button a few times on the face of the Ortec counter so the bottom panel now reads a number different from 00 in the right hand side. Choose to write a string (option `w`) and type `INIT`, then press `enter`. The panel on the Ortec counter should now read 00 as it does after startup.

If you read another data string from the counter, you should get one of the OK signals. Specifically, I received `%000000069`.

This completes the read-write Ortec 974 test procedure.

5 Compiling

Once you've successfully done the previous steps, you're likely to want to compile some of your own code. In order to do this, I'll assume you have a

C source file named `ortec.c` in the current working directory. Issuing the command

```
gcc ortec.c -o ortec -lgpib
```

takes care of compiling your code and linking to the GPIB library which is now installed on your system.

Consider placing your compilation line in an executable script which has termination *.sh (to make it executable write `chmod +x nameofscript.sh`) and **remember to run your newly compiled code as a super-user by prefixing `sudo`!**

All of the code I've developed, which includes the interesting `ortec.c`, as well as this instruction manual can be downloaded from my github repository (most conveniently as a zip file) which is located at

```
https://github.com/danieljg/ortec-gpib
```

Please mind the `LICENSE` file, as it concerns my copyright.

6 Hack to provide a keyboard killswitch

The killswitch functions are provided by the `ncurses` library. This, however, includes a symbol within its compiled object code which collides with a name defined by gpib-linux. The solution to using both involves changing the name of one the colliding variables and is described below. It should probably be copy-pasted into the terminal, unless you know what you're doing.