John's Geekblog - Notebook of Time and Frequency Experiments (John Ackermann N8UR)

June 10, 2017

$20 P25 Trunking Scanner (Part 1)

A couple of weeks ago at the Dayton Hamvention I bought a fancy Whistler TRX2 scanner that can receive the new P25 digital trunking system that's been rolled-out statewide here in Ohio. Unfortunately, after a lot of attempts, I still can't get it to reliably receive the talkgroups I'm most interested in.

Out of frustration (and boredom) I decided to research the couple of projects I'd heard about to implement P25 decoding, and after a few hours work (and some great help from a mailing list) I now have it running very nicely on my laptop. The coolest part is that the only hardware needed for this is a USB dongle SDR (software defined radio). I'm using one from RTL-SDR.com that cost $20. (You can order them from TAPR.)

Here's a Youtube video showing about 5 minutes of traffic I recorded on a Friday evening.

In part 2, I'll walk through the software installation and configuration process.

$20 P25 Trunking Scanner (Part 2)

Here's how to install the OP25 scanning software. If you don't already have Gnuradio installed on a Linux system, do those two things first.

Any current Linux distribution works. I use Linux Mint which is very similar to Ubuntu.

The easiest way to install Gnuradio is to use a script called SBRAC build-gnuradio. Click on the link, then save the contents of the browser screen (easiest way is to select everything on the web page, then copy that into an empty file and save as "build-gnuradio" in your home directory.) Open a terminal window, make sure you're in your home directory, and enter "sh build-gnuradio" (without the quote marks) then hit return. You'll get a couple of prompts, and then the computer will start downloading, building, and installing the gnuradio system. This will take time — maybe a few hours — so be patient.

Once Gnuradio is installed, you can download and install the OP25 scanner software. You might want to look at the OP25 Project page first. Then:

1. Install a couple of program packages that op25 needs. On an Ubuntu or similar system do:"sudo apt-get install libitpp-dev libpcap-dev git" (without the quotes.

2. Now download and build the software. From your home directory, run the following commands in a terminal window:

```
cd ~
git clone git://op25.osmocom.org/op25.git
cd op25
```

```
mkdir build
cd build
cmake ../
make
sudo make install
sudo ldconfig
```

3. When finished, you'll have a directory called op25 that contains the software.

In part 3, I'll describe how to configure and run the op25 software.

$20 P25 Trunking Scanner (Part 3)

The info in this and in part 4 was aided tremendously by several folks on the "op25-dev" mailing list. Without their help, I would not have even known where to start in getting my system running. I hope that what I'm documenting here will be helpful for others going down this path.

If you've successfully installed Linux, Gnuradio, and op25 on your computer, you are ready to test the system, and then configure it for trunking.

The scanner application is buried several directories down, and you need to change to that directory and run the program there. So do this:

```
cd ~/op25/op25/gr-op25_repeater/apps
```

In that directory, you'll find the main program, `scope.py`, and several configuration files with `.tsv` suffixes — most are examples for specific systems.

Assuming you're using a RTL dongle, plug that puppy in to your USB port and then issue this command:

```
./scope.py --args 'rtl' -f 853.600e6 -g 65 -o 17e3 -N 'LNA:35' -V -v 0 -S 250000 -q '-1'
```

Change the value following the `-f` to the control channel frequency of the P25 site. (The `e6` is scientific notiation and indicates to multiply the value by one million.)

If all is well, you'll see a box open on your screen showing a spectrum display, with tabs to several other displays as well.

What does the gibberish in the command line mean?

`--args 'rtl'` indicates that you're using a RTL dongle receiver.

`-f 853.600e6` indicates the frequency to monitor. 853.600 MHz is the primary control panel for the Mongtomery County, Ohio P25 site.

`-g 65` sets the audio gain; I'm not sure just what effect it has.

`-o 17e3` sets a tuning offset — it's a little like a local oscillator, and it avoids system-generated noise that appears at the center of the RF tuning. Here, I am moving the channel frequency 17 kHz from the tuned frequency.

`-N 'LNA:35'` sets the RTL gain.

`-V` turns on the audio vocoder so you can hear the traffic.

`-v 0` sets the verbosity of the console logging. Set to 255 to see everything.

`-S 250000` sets the sample rate. We only need to receive one NBFM channel at a time, so setting this to a small number reduces unnecessary CPU load. 250000 samples/second is the slowest rate the RTL-SDR.com dongle supports.

`-q -1` adjusts for any frequency error in the dongle. Mine is pretty close, so I only need to adjust down by 1 PPM.

When you run this command, you should see the program open a window showing a spectrum display tab with the control frequency just about centered. If you click on the "traffic" tab you should see site information at the top, and then individual commands coming from the system controller.

If you've gotten this far, the system works. All you need to do now is set up the scanning/trunking capability. That's in part 4.

$20 P25 Trunking Scanner (Part 4)

With op25 successfully decoding the control channel, now you just need to set up the trunking configuration. To do this, you'll probably need to set up three files:

1. `trunk.tsv` defines the sites to be monitored.

2. `.tsv` provides a table of talkgroup numbers and names for that site.

3. Because you probably only want to listen to a subset of the talkgroups, you can optionally create a file called `.tsv` with a list, one talkgroup per line, of the talkgroups you want to hear.

Here's what my `trunk.csv` file looks like for the Montgomery County, Ohio MARCS-IP system:

```
"Sysname" "Control Channel List" "Offset" "NAC" "Modulation" "TGID Tags File" "Whitelist"
"Blacklist" "Center Frequency"
"Montgomery" "853.600" "0" "0x343" "CQPSK" "montgomery.tsv" "montgomery_wl.txt" ""
```

Note: (a) the first line needs to be included, and (b) each value is surrounded by quotes and separated by a tab character — NOT SPACES.

The `Sysname` value is an arbitrary one that you assign. The `Control Channel List` needs to include at least the primary controller (note no need for "e6″ here) and can also include alternate control channels. `Offset` is the same value you would enter with the command line `-q` parameter.

Now, the `NAC` field really threw me. When I went to the Radio Reference Montgomery County Site Information page, it listed a `System ID` of 348, and a `Site NAC` value of 25A. Neither of those are correct for this field. I can't find a web reference that ties `0x343` to the Montgomery county system. I got this number from my initial monitoring of the control channel before adding the trunking configuration. I strongly advise you to do the same, as this is what kept me from decoding until the light finally dawned.

The next field is `Modulation`. There are two choices: C4FM, widely used by non-Motorola systems, and CQPSK, which is derived from a Motorola standard called "LSM" — Linear Simulcast Modulation. This was another thing that threw me; the Radio Reference page indicated that the Montgomery County site uses C4FM. I only got things to work using CQPSK, though.

Next is the `TGID Tags` field. This optionally specifies a field that has human-friendly names to decode the talkgroup numbers. Here are a few lines of my tag file, which I call `montgomery.tsv`:

```
58232 Kettering Fire Dispatch
58238 Kettering Police Dispatch
58239 Kettering Police Ch B
58240 Kettering Police
```

Note again that the separator between talkgroup number and name must be a space.

Next are two optional fields, where talkgroups may be whitelisted or blacklisted. It only makes sense to use one of these two. I created a whitelist file called `montgomery_wl.tsv` that limits the talkgroups to a few I'm really interested in:

```
58014
58023
58024
58293
58294
58333
53334
```

I haven't tested yet whether the whitelist file can include tab-separated talkgroup descriptions as well, or has to be just a list of numeric entries.

Now, use the same scope.py command line as before, but add "-T trunk.tsv" at the end. You should now be scanning and will hear audio traffic from the monitored talkgroups.

One final thing — the display in my recorded demo at Youtube looks slightly different as I changed the placement of the system, talkgroup, and frequency information on the display.