Op25 documentation

April, 2017
==========

This file contains notes on the new version OP25 receiver (rx.py) which
replaces the prior version scope.py. The primary differences are:
 * The dependency on WX is completely removed. By default OP25 runs in
   a console window in text-only mode.
 * An optional real-time plot can be selected when launching rx.py:
   contellation, datascope (eye pattern), or symbol trace.
 * cqpsk versus fsk4 mode is now selected as a command line parameter.
 * most rx.py command-line parameters are compatible with scope.py.
 * reduced CPU consumption, as the frame assembler block now runs as a
   sink-only GR block.
 * dependency on 14.04 is completely removed. Should now run in later
   ubuntu and fedora versions with only minor changes (not yet tested).

ADDITIONAL REQUIRED PACKAGES
============================
sudo apt-get install gnuplot-x11

EXAMPLE COMMAND LINE
====================
./rx.py --args 'rtl' --gains 'lna:49' -f 456.7e6 -T tsys.tsv -q -1 -S 1000000 -P symbol -o 50000 -w
2> stderr.2

Running stderr to a file (e.g., "2> stderr.2") is recommended to avoid
screen misprinting.

NOTE: For phase1 voice the "-V" option is not used. Instead the
"-w" option is used (see AUDIO SERVER section, below). For P25 phase 2/TDMA,
the "-2" option is required in addition to the "-w" option.

TERMINAL OPERATION
==================
After starting rx.py if plotting is in use a separate gnuplot window
should open. You must click on the terminal window to restore it to
focus, otherwise all keystrokes are consumed by gnuplot. Once in the
terminal window there are several keyboard commands:
 h - hold
 l - lockout

s - skip
q - quit program
, - decrease fine tune by 100Hz
. - increase fine tune by 100Hz
< - decrease fine tune by 1200Hz
> - increase fine tune by 1200Hz
1 - toggle fft plot
2 - toggle constellation plot (cqpsk)
3 - toggle symbol plot
4 - toggle datascope plot (fsk4)
5 - toggle mixer plot
There are also two experimental commands (should not be used in -T mode)
f - manually change frequencies
t - if currently tuned to a CC, autostart scanning talkgroups

The "t" command allows trunk tracking without setting up a trunking TSV file. However running with the -T <filename> command line option is preferred as that allows use of white/black lists and talkgroup tags files.

If the terminal window freezes there may have been a crash. Press Ctrl-Z to suspend the program and examine stderr.2 for error messages. If there is a traceback please report the full traceback (and the command line) to the mail list.

REMOTE TERMINAL
==========
Adding (for example) "-l 56111" to the rx.py command starts rx.py but does not attach a curses terminal. Instead the program runs as normal in the foreground (hit CTRL-C to end rx.py as desired). To connect to a running instance of rx.py, (in this example)

```
./terminal.py 127.0.0.1 56111
```

NOTE: rx.py and terminal.py need not run on the same machine. The machine where terminal.py is running need not have an SDR device directly attached; but GNU Radio (and OP25) must be available.

WARNING: there is no security or encryption on the UDP port.

EXTERNAL UDP AUDIO SERVER
==========
Starting rx.py with the "-w -W host" options directs udp audio data to be sent over the network to the specified remote host. It can then be received and played back with either of the following methods:
1. Execute ./audio.sh on a remote machine equipped with python2.7,

libasound.so.2 and the sockaudio.py file.
-or-
2. Execute the command:
nc -kluvw 1 127.0.0.1 23456 | aplay -c1 -f S16_LE -r 8000

NOTE: audio underruns are to be expected when using nc | aplay as the pcm stream is interrupted every time a radio transmission ends. The sockaudio player is designed to handle this more gracefully, and generally only underruns due to high cpu utilization or reception/decoding errors.

INTERNAL AUDIO SERVER

Starting rx.py with the "-U" command line option enables an internal udp audio server which will play received audio through the default ALSA device. Optionally you may specify which ALSA device to use by setting the "-O audio_out" option along with "-U".

As of this writing (Aug 2017) it is still necessary to specify the "-w" (wireshark) option if using either the internal or external audio server.

PLOT MODES

Four types of plotting are currently implemented, via the -P parameter:
* fft
* constellation
* datascope
* symbol

The symbol mode is allowed both in fsk4 and cqpsk modes. The datascope mode works only with fsk4 demod mode (-D fsk4). The constellation mode only works when the cqpsk demod mode is selected (or defaulted).

A couple of notes specific to plot mode:

1. At program startup time the gnuplot window is given the focus after it opens. Before you can enter terminal commands you need to click on the terminal window once to make it the active window.

2. In some cases the gnuplot window is displayed on top of the terminal window used by OP25. If so it may be necessary to move one or the other of the two windows.

COMMAND LINE OPTIONS

```
Usage: rx.py [options]

Options:
  -h, --help            show this help message and exit
  --args=ARGS           device args
  --antenna=ANTENNA     select antenna
  -a, --audio           use direct audio input
  -A, --audio-if        soundcard IF mode (use --calibration to set IF freq)
  -I AUDIO_INPUT, --audio-input=AUDIO_INPUT
                        pcm input device name.  E.g., hw:0,0 or /dev/dsp
  -i INPUT, --input=INPUT
                        input file name
  -b Hz, --excess-bw=Hz
                        for RRC filter
  -c Hz, --calibration=Hz
                        USRP offset or audio IF frequency
  -C Hz, --costas-alpha=Hz
                        value of alpha for Costas loop
  -D DEMOD_TYPE, --demod-type=DEMOD_TYPE
                        cqpsk | fsk4
  -P PLOT_MODE, --plot-mode=PLOT_MODE
                        constellation | fft | symbol | datascope | mixer
  -f Hz, --frequency=Hz
                        USRP center frequency
  -F IFILE, --ifile=IFILE
                        read input from complex capture file
  -H HAMLIB_MODEL, --hamlib-model=HAMLIB_MODEL
                        specify model for hamlib
  -s SEEK, --seek=SEEK  ifile seek in K
  -L LOGFILE_WORKERS, --logfile-workers=LOGFILE_WORKERS
                        number of demodulators to instantiate
  -S SAMPLE_RATE, --sample-rate=SAMPLE_RATE
                        source samp rate
  -t, --tone-detect     use experimental tone detect algorithm
  -T TRUNK_CONF_FILE, --trunk-conf-file=TRUNK_CONF_FILE
                        trunking config file name
  -v VERBOSITY, --verbosity=VERBOSITY
                        message debug level
  -V, --vocoder         voice codec
  -n, --nocrypt         silence encrypted traffic
  -o Hz, --offset=Hz    tuning offset frequency [to circumvent DC offset]
  -p, --pause           block on startup
  -w, --wireshark       output data to Wireshark
```

```
  -W WIRESHARK_HOST, --wireshark-host=WIRESHARK_HOST
                        Wireshark host
  -u WIRESHARK_PORT, --wireshark-port=WIRESHARK_PORT
                        Wireshark udp port
  -r RAW_SYMBOLS, --raw-symbols=RAW_SYMBOLS
                        dump decoded symbols to file
  -R RX_SUBDEV_SPEC, --rx-subdev-spec=RX_SUBDEV_SPEC
                        select USRP Rx side A or B (default=A)
  -g GAIN, --gain=GAIN  set USRP gain in dB (default is midpoint) or set audio
                        gain
  -G GAIN_MU, --gain-mu=GAIN_MU
                        gardner gain
  -N GAINS, --gains=GAINS
                        gain settings
  -O AUDIO_OUTPUT, --audio-output=AUDIO_OUTPUT
                        audio output device name
  -U, --udp-player      enable built-in udp audio player
  -q FREQ_CORR, --freq-corr=FREQ_CORR
                        frequency correction
  -d FINE_TUNE, --fine-tune=FINE_TUNE
                        fine tuning
  -2, --phase2-tdma     enable phase2 tdma decode
  -Z DECIM_AMT, --decim-amt=DECIM_AMT
                        spectrum decimation
```

MULTI-RECEIVER
==============

The multi_rx.py app allows an arbitrary number of SDR devices and channels to be defined. Each channel may have one or more plot windows attached.

Configuration is achieved via a json file (see cfg.json for an example).
In this version, channels are automatically assigned to the first device found whose frequency span includes the selected frequency.

As of this writing (winter, 2017), neither trunking nor P25 P2/TDMA are supported in multi_rx.py. The rx.py app should be used for P25 trunking, for either P1/FDMA or P2/TDMA.

Below is a summary of the major config file keys:

```
demod_type:     'cqpsk' for qpsk p25 only; 'fsk4' for ysf/dstar/dmr/fsk4 p25
filter_type:    'rc' for p25; 'rrc' for dmr and ysf; 'gmsk' for d-star
plot:           'fft', 'constellation', 'datascope', 'symbol'
                [if more than one plot desired, provide a comma-separated list]
```

```
destination:    'udp://host:port' or 'file://<filename>'
name:           arbitrary string used to identify channels and devices

bug: 'fft' and 'constellation' currently mutually exclusive
bug: 'gmsk' needs work

Note: DMR audio for the second time slot is sent on the specified port number
plus two.  In the example 'udp://127.0.0.1:56122', audio for the first slot
would use 56122; and 56124 for the second.

The command line options for multi_rx:
Usage: multi_rx.py [options]

Options:
  -h, --help            show this help message and exit
  -c CONFIG_FILE, --config-file=CONFIG_FILE
                        specify config file name
  -v VERBOSITY, --verbosity=VERBOSITY
                        message debug level
  -p, --pause           block on startup
```

OP25 Multiprotocol Digital Voice TX

The OP25 project in 2009 released an encoder and software TX for P25 (conventional phase I C4FM/FDMA) based on the fullrate IMBE codec contribution from Pavel Yazev.
With the addition of a software AMBE halfrate encoder the OP25 Group now releases an updated TX with voice encoders for-

- P25
- DMR
- YSF
- DSTAR

This release was tested using a PC soundcard connected to the direct modulator input (i.e., 9,600 data port) of an Icom IC-820H configured for high speed data operation [AMOD/PACT switch set to PACT]. Not all PCs tested successfully so perhaps the choice of sound card isn't irrelevant, or the interface circuit needs more work... The software TX drives the audio output at optimum (close to full) signal levels; no output level adjust is provided within the app. Soundcard gain/loss must be carefully adjusted using a Linux sound mixer app so that the proper FM deviation is achieved (see below).

This release has also been tested with the HackRF SDR; other SDR devices supported by gr-osmosdr should work as well.

INSTALLATION

After cloning the repo change to the top level directory:
git clone https://git.osmocom.org/op25
cd op25

Next install OP25:
./install.sh

You may be asked for your password for permission to install the OP25 files in the system libraries.

STARTING THE PROGRAM

After installing OP25 cd to the op25/gr-op25_repeater/apps/tx subdirectory. Here's an example command line:
python dv_tx.py -r -f ~/uncompressed.wav -p dstar -c dstar-cfg.dat

Hackrf Example:
python dv_tx.py -p dmr --args hackrf -q -21 -Q 925187500 -c dmr-cfg.dat -f ~/file1.wav -F ~/file2.wav -r

In this example we've selected the dstar protocol with input from a wav file (must be standard rate 8000/S16_LE/mono). Below is a full list of options.
Multiprotocol Digital Voice TX (C) Copyright 2017 Max H. Parke KA1RBI
Usage: dv_tx.py [options]

Options:
```
  -h, --help            show this help message and exit
  -a ARGS, --args=ARGS  device args
  -b BT, --bt=BT        specify bt value
  -c CONFIG_FILE, --config-file=CONFIG_FILE
                        specify the config file name
  -f FILE1, --file1=FILE1
                        specify the input file slot 1
  -F FILE2, --file2=FILE2
                        specify the input file slot 2 (DMR)
  -g GAIN, --gain=GAIN  input gain
  -i IF_RATE, --if-rate=IF_RATE
                        output rate to sdr
```

-I AUDIO_INPUT, --audio-input=AUDIO_INPUT
pcm input device name. E.g., hw:0,0 or /dev/dsp
-N GAINS, --gains=GAINS
gain settings
-O AUDIO_OUTPUT, --audio-output=AUDIO_OUTPUT
pcm output device name. E.g., hw:0,0 or /dev/dsp
-o OUTPUT_FILE, --output-file=OUTPUT_FILE
specify the output file
-p PROTOCOL, --protocol=PROTOCOL
specify protocol
-q FREQUENCY_CORRECTION, --frequency-correction=FREQUENCY_CORRECTION
ppm
-Q FREQUENCY, --frequency=FREQUENCY
Hz
-r, --repeat input file repeat
-R, --fullrate-mode ysf fullrate
-s SAMPLE_RATE, --sample-rate=SAMPLE_RATE
output sample rate
-t TEST, --test=TEST test pattern symbol file
-v VERBOSE, --verbose=VERBOSE
additional output

MULTI-CHANNEL TRANSMITTER

The multi_tx.py demonstration app transmits four RF channels, one for each of the supported protocols. It requires SDR hardware (such as the HackRF); any device supported by the gr-osmosdr driver and libraries should work. Example command line:
python multi_tx.py --args hackrf --gains rf:0,if:0 -q -19.5 -Q 442187500 -f ~/rand3.wav -R
The "-Q" (frequency) option gives the frequency of the first of the four channels; the remaining channel frequencies appear at regular intervals (default spacing is 100 KHz). The channels are, starting at the lowest frequency: dmr, p25, dstar, and ysf. The ysf channel format may be set to "wide", using "-R" (default=narrow). The "-q" option is needed for PPM correction (unless a time base such as GPSDO is used). The input file ("-f" option) requires WAV format, 8000 samples/sec, S16_LE. Audio from the WAV file is transmitted on all frequencies. However if the "-A" option is given, program content for the second DMR time slot is instead taken live from the sound card audio input port ("-I" option).

GENERAL

Input audio may come either from WAV files (8000/S16_LE/mono) or from the PC microphone / line input connector. If -f *filename* (or -F *filename* in DMR mode) is selected the input is read from file(s); otherwise the GR audio source is selected. Likewise output is sent to the soundcard unless -o is given. When using files the -r option can be specified to replay the input file(s) in a

continuous loop. Without this option the program exits at EOF. The -p *protocol* option is used to select which of the four DV standards (dmr, dstar, p25 or ysf) is to be used. The dmr and ysf modes also utilize a configuration file, specified using the -c *filename* parameter. See the specific section (below) for more details on the configuration file.
There currently appears to be an audio bug that causes the audio to sound "overdriven" or "clipped". The dstar and to a lesser extent the ysf-halfrate coder is most affected by this; dstar suffers from an audio level pumping or AVC malady. **UPDATE** (Mar. 2017) A hack was added that greatly reduces this problem in ysf. At this point only dstar seems in need of further work.
Note: in general the program currently doesn't currently send "header" or "terminator" packets which typically appear once at the start/end of voice transmissions. Instead we rely on the ability inherent in all of these protocols to start receiving even for transmissions already in progress. For two of the four protocols (dstar and ysf in fullrate mode) this means we forfeit an opportunity to transmit the station callsign. This needs to be fixed in a future release...

OPERATION

Proper FM deviation is set by adjusting the output levels using the 'alsamixer' app while monitoring with an FM deviation meter. The "Datascope" option in the scope.py OP25 gui app (running on a separate PC) can also be used to view the deviation. If this method is used scope.py must first be calibrated using a known accurate station. The test station is then adjusted to have the same deviation as the known station (leaving all receiver parameters unchanged other than perhaps the frequency). Note: if the sound interface has multiple audio bands (e.g., bass, treble, etc.) all such settings should be set to midrange; a flat response is required.

DMR

DMR operates in Base Station (BS) mode and is 2-slot TDMA with both slots operating in Group Voice mode. [DMR Mobile Station mode utilizes time-slotted channel access and is not directly compatible with the usual FM transmitter hardware without serious modification]. The configuration file parameters that you are most likely to change are the Station Address (sa) and Group Address (ga) parameters [the latter is also known as the talkgroup ID]. See the comments in the sample DMR config file dmr-cfg.dat (supplied). It is also possible to disable one or both channels if desired.

DSTAR

As of this writing DSTAR mode has an audio clipping bug causing the speech to have a distored, overdriven character. Given sufficient time this should be quite fixable. Also the "slow speed data" feature needs to be implemented. Until this is done the config file (if present) is ignored. As mentioned, the -g *gain* parameter can be used to adjust the input gain, although this doesn't fully cure all dstar quality issues.

P25

The P25 coder was the earliest modulation format supported in OP25. It operates with a fixed NAC (Network Access Code) of 0x293 and transmits fullrate voice.

YSF

The ysf mode supports both halfrate (default) and fullrate voice, with some clipping distortion noted in the halfrate mode. Fullrate mode is enabled using the -R option. The halfrate mode also supports the interspersed callsign and repeater ID data, but this is currently untested. See the sample ysf config file ysf-cfg.dat (supplied).

CREDITS

Too many to list; thanks to Matt Ames, Pavel Yazev, Jonathan Naylor, unknown author(s) of DSD (DSDAuthor), Mathias Weyland, Sylvain Munaut; undoubtedly others.
