Setting up rigctld for multiple programs

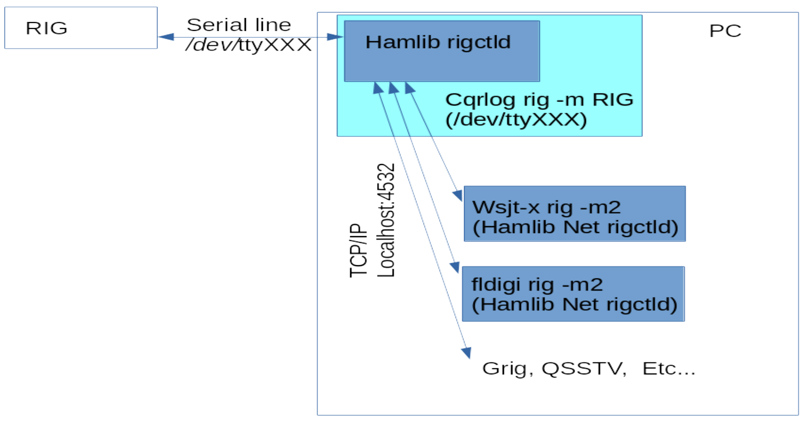
revised 2018-12-06

If you have several programs that want to access your rig's CAT via same serial port, at same time, that will end up to problems. Linux allows several programs access to same device, unlike Windows, that will tell user that serial port is already in use. So it is users duty to handle situation.

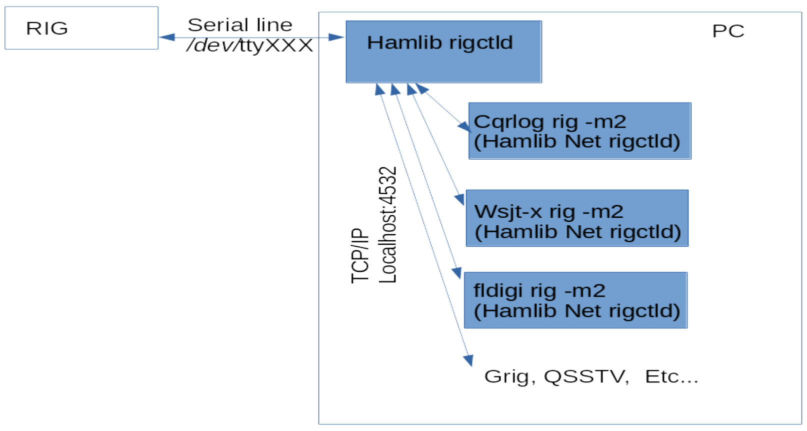
Receiving data from serial device with multiple programs is not defective. But as soon as programs try to write to device it leads to situation where two or more simultaneous transmits roll over each other

For getting freedom to use many programs at same time we have to start a program that can, alone, communicate to rig via serial port and on the other hand can communicate other way to many programs.Rigctld does just that. It can communicate with several programs via TCP and one rig via serial device. You just need to set up it right.

Rigctld started via cqrlog. Cqrlog must be running before, and while, other programs are running



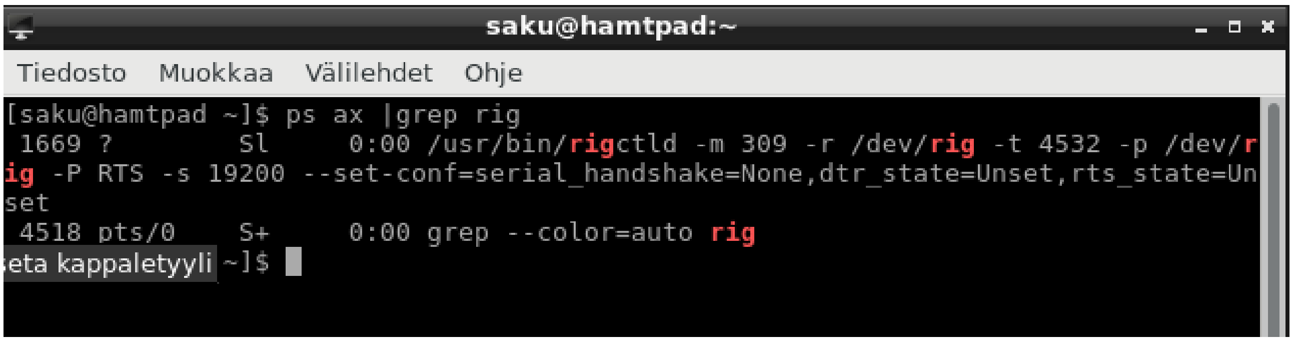
Rigctld started from linux start or user login (via script)



If you have working cqrlog setup open cqrlog and check that you get right frequency from your rig. At that point open a command console and give:

ps ax | grep rig

Followed with return. You should see something like this:



```
[saku@hamtpad ~]$ ps ax |grep rig
 1669 ?        Sl     0:00 /usr/bin/rigctld -m 309 -r /dev/rig -t 4532 -p /dev/r
ig -P RTS -s 19200 --set-conf=serial_handshake=None,dtr_state=Unset,rts_state=Un
set
 4518 pts/0    S+     0:00 grep --color=auto rig
eta kappaletyyli ~]$ 
```

The first line (here starting with 1669) is the line that has complete rigctld start command with parameters issued by cqrlog.
Pain that line staring from /usr and ending to last word (here Unset) and copy it to clipboard for further usage.

Create a new file to your user folder with your favorite text editor. I use here command line editor called nano.

nano ~/rigctld_start.sh

Add following lines to editor:

```
#!/bin/bash
#
sleep 2
#
# Nothing to do if it is already running.
#
a=$(/usr/bin/pidof rigctld)
if [ "$a" != "" ]; then
  /usr/bin/date >> /tmp/start.log
  echo "Already running." >> /tmp/start.log
else
  /usr/bin/date >> /tmp/start.log
  /usr/bin/rigctld  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx &
  echo "Started." >> /tmp/start.log
fi
echo "----------------------" >> /tmp/start.log
```

NOTE! Some Linux versions have date and pidof in different folders than /usr/bin. Check with command: whereis pidof, and whereis date and fix paths in script to follow your linux folders. Tnx to: John, W4JKL

NOTE: the line having ”/usr/bin/rigctld xxxxxxxxxx”. Replace it with the one you did copy to clipboard on previous stage to get rigctld started with your rig and your parameters.
The last characters after your pasted rigctld line should be ” &” (space and &). Add them manually. Be careful that you have the complete line with all parameters. Space and & at the end of line will cause rigctld to run on background.

If you do not need/like log file /tmp/start.log you can leave all lines with ”>>” away.

After saving and ending editor change permissions for run this file with command:

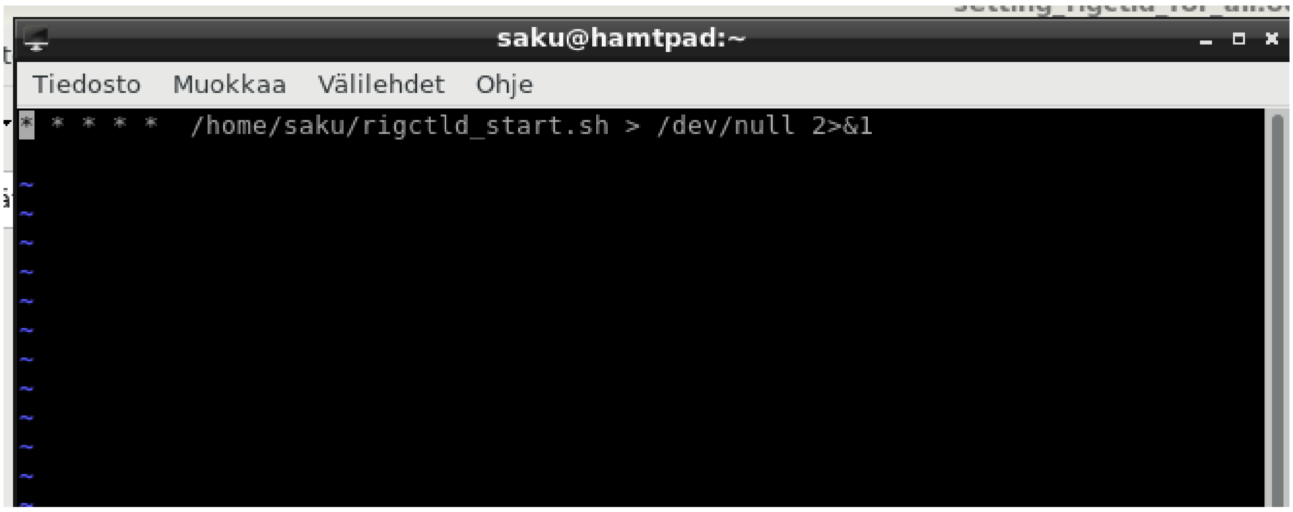
chmod a+x ~/rigctld_start.sh

Next we set up user crontab that will start and keep rigctld running. Command is:

crontab -e

If you already have something timed via crontab there might be some lines. We add one line more:
*** * * * * /home/saku/rigctld_start.sh > /dev/null 2>&1**

Note: “saku” should be replaced with the username you are using!



```
* * * * *  /home/saku/rigctld_start.sh > /dev/null 2>&1
~
~
~
~
~
~
~
~
~
~
~
```

Depending your linux, editing crontab may be done with **nano** (as we used before) or **vi** or some other text editor. Few words about **vi** that looks like this:
To start adding line press **I ,** put cursot to right place, then write your line. You will see text ”-- insert --” at bottom left after **i** is pressed.
To save this file press first **Esc** key. Then **:** key. **:** will appear to bottom left of the console.
Then write **wq** and press return. You should see **crontab: installing new crontab.**

Now your **rigctld_start.sh** is run every minute and if rigctld is not running it will be started.

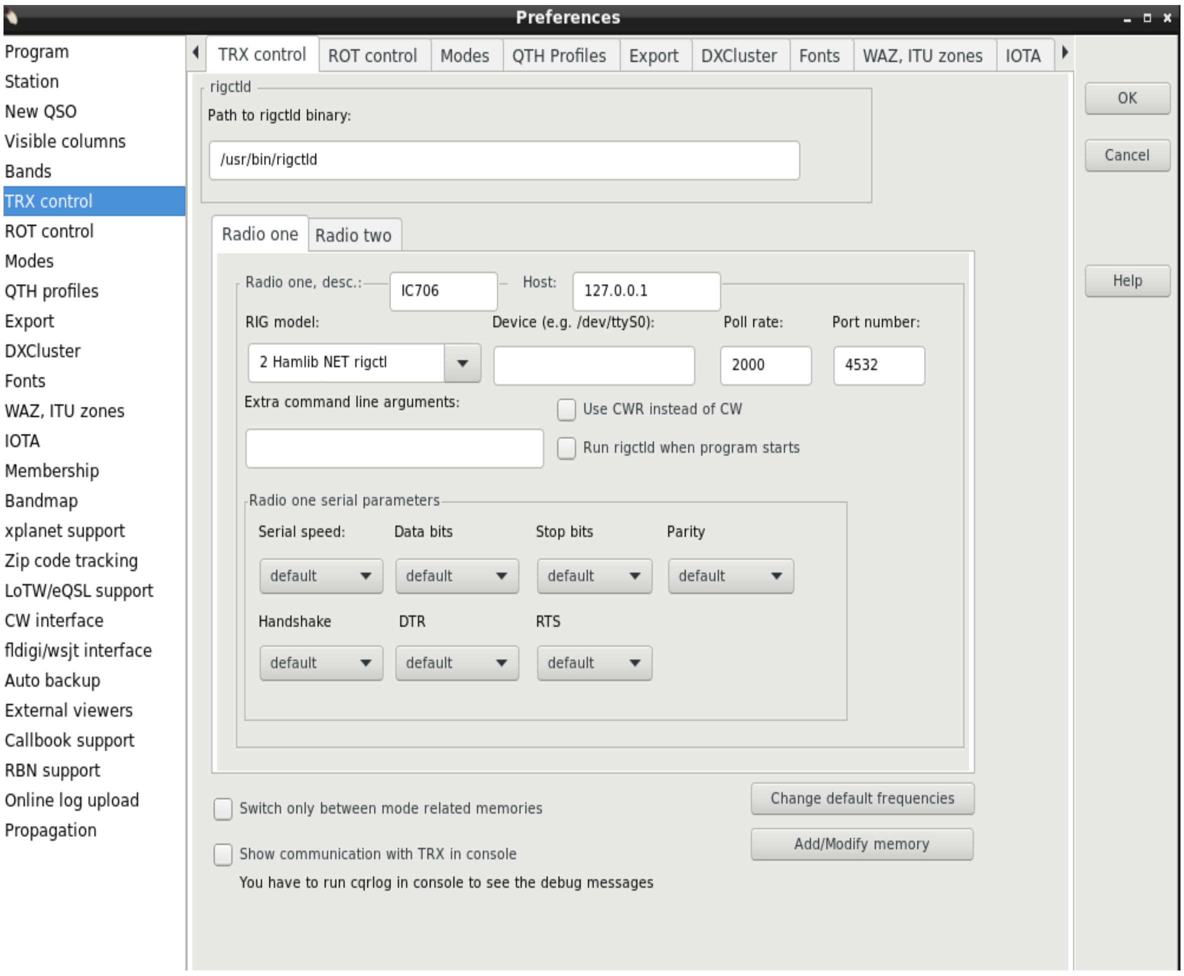
You may test this by closing all programs and computer. Then powering it up again and when your desktop is there again, open command console and give:

ps ax | grep rig

With in a minute the response should be again that same rigctld -line we copied at first place.
You can repeat same command easily by pressing ”arrow up”-key and then return as you may need to do it several times to see when rigctld has started.

Then it is time to set up all other programs. Good point there is that every program for now on has same setup, no matter what your rig model and serial device is! So you can copy these direct and they work as rig + port specific setup is only in the rigctld parameters.

cqrlog:



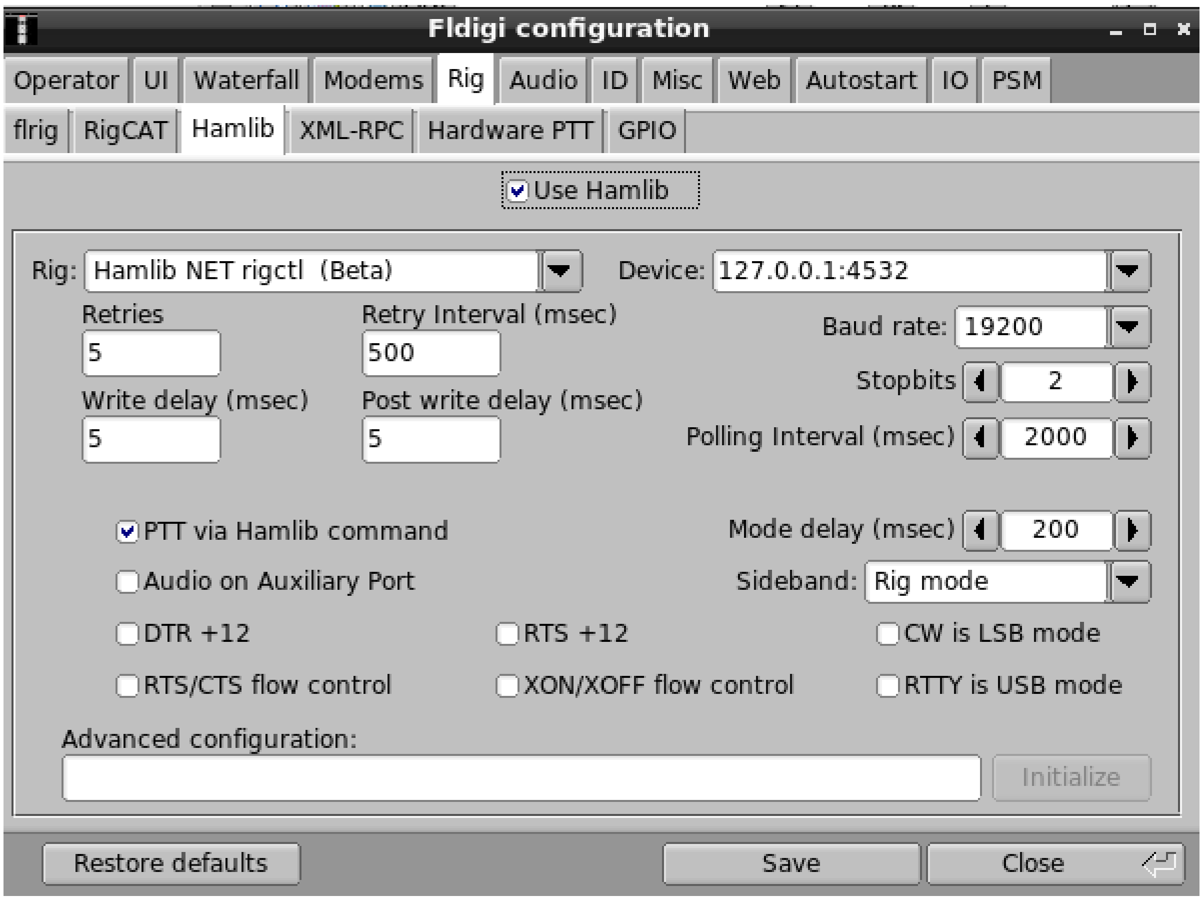
rigctld
Path to rigctld binary: /usr/bin/rigctld

Radio one, desc.: IC706 Host: 127.0.0.1
RIG model: 2 Hamlib NET rigctl
Device (e.g. /dev/ttyS0):
Poll rate: 2000
Port number: 4532
Extra command line arguments:
☐ Use CWR instead of CW
☐ Run rigctld when program starts

Radio one serial parameters
Serial speed: default
Data bits: default
Stop bits: default
Parity: default
Handshake: default
DTR: default
RTS: default

☐ Switch only between mode related memories
☐ Show communication with TRX in console
You have to run cqrlog in console to see the debug messages

fldigi:



Fldigi configuration

Operator | UI | Waterfall | Modems | Rig | Audio | ID | Misc | Web | Autostart | IO | PSM

flrig | RigCAT | Hamlib | XML-RPC | Hardware PTT | GPIO

[✓] Use Hamlib

Rig: Hamlib NET rigctl (Beta)

Device: 127.0.0.1:4532

Retries: 5

Retry Interval (msec): 500

Baud rate: 19200

Stopbits: 2

Write delay (msec): 5

Post write delay (msec): 5

Polling Interval (msec): 2000

[✓] PTT via Hamlib command

Mode delay (msec): 200

[ ] Audio on Auxiliary Port

Sideband: Rig mode

[ ] DTR +12 [ ] RTS +12 [ ] CW is LSB mode

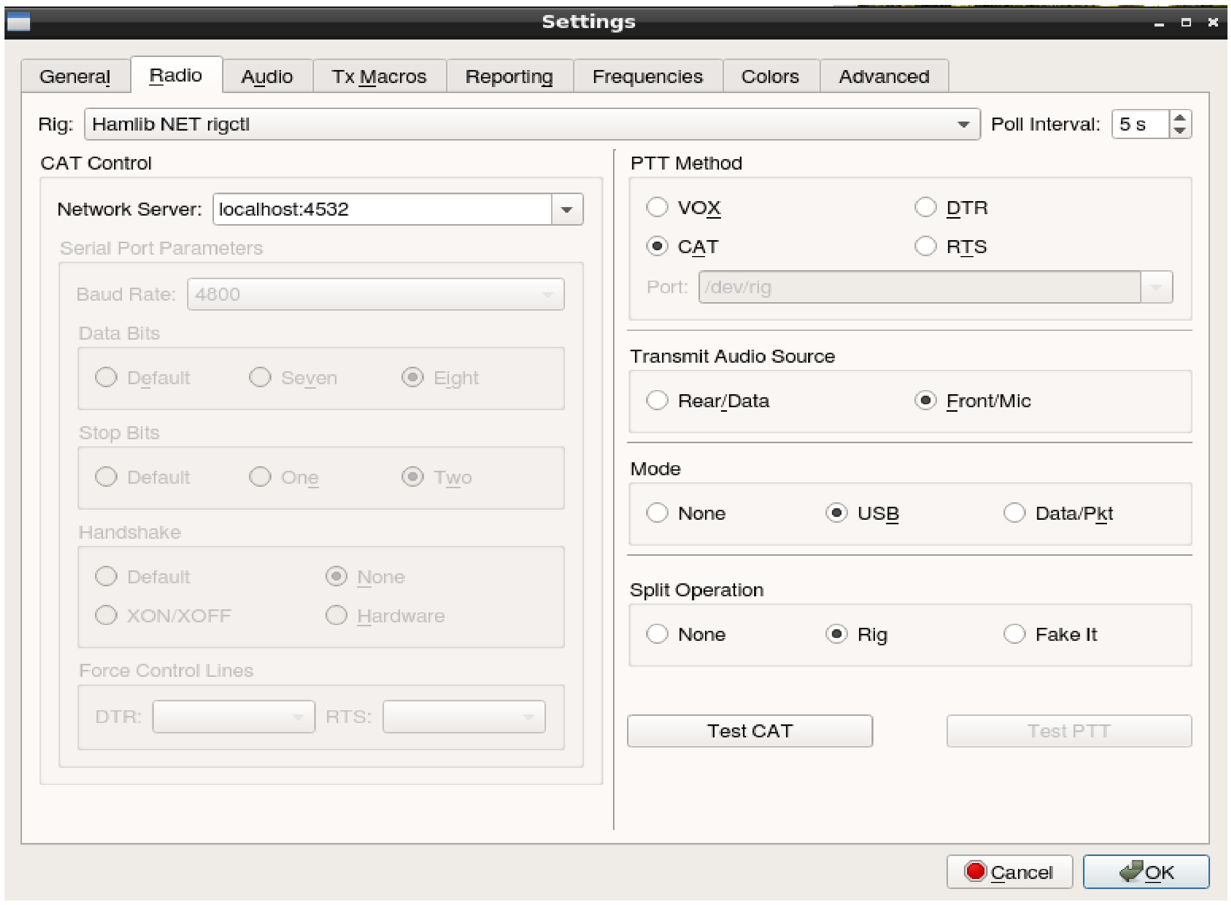
[ ] RTS/CTS flow control [ ] XON/XOFF flow control [ ] RTTY is USB mode

Advanced configuration:

Initialize

Restore defaults | Save | Close

wsjtx:



Settings

General | Radio | Audio | Tx Macros | Reporting | Frequencies | Colors | Advanced

Rig: Hamlib NET rigctl

Poll Interval: 5 s

CAT Control

Network Server: localhost:4532

Serial Port Parameters

Baud Rate: 4800

Data Bits: Default Seven [✓ Eight]

Stop Bits: Default One [✓ Two]

Handshake: Default [✓ None] XON/XOFF Hardware

Force Control Lines: DTR: RTS:

PTT Method: VOX DTR [✓ CAT] RTS

Port: /dev/rig

Transmit Audio Source: Rear/Data [✓ Front/Mic]

Mode: None [✓ USB] Data/Pkt

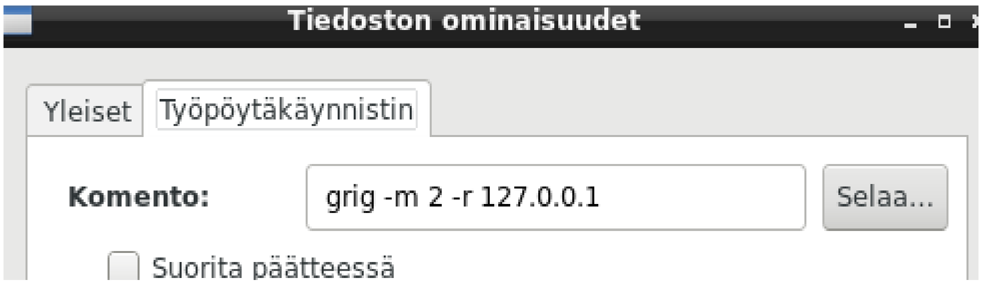
Split Operation: None [✓ Rig] Fake It

Test CAT | Test PTT

Cancel | OK

grig:

(grig has settings as parameters in start line)

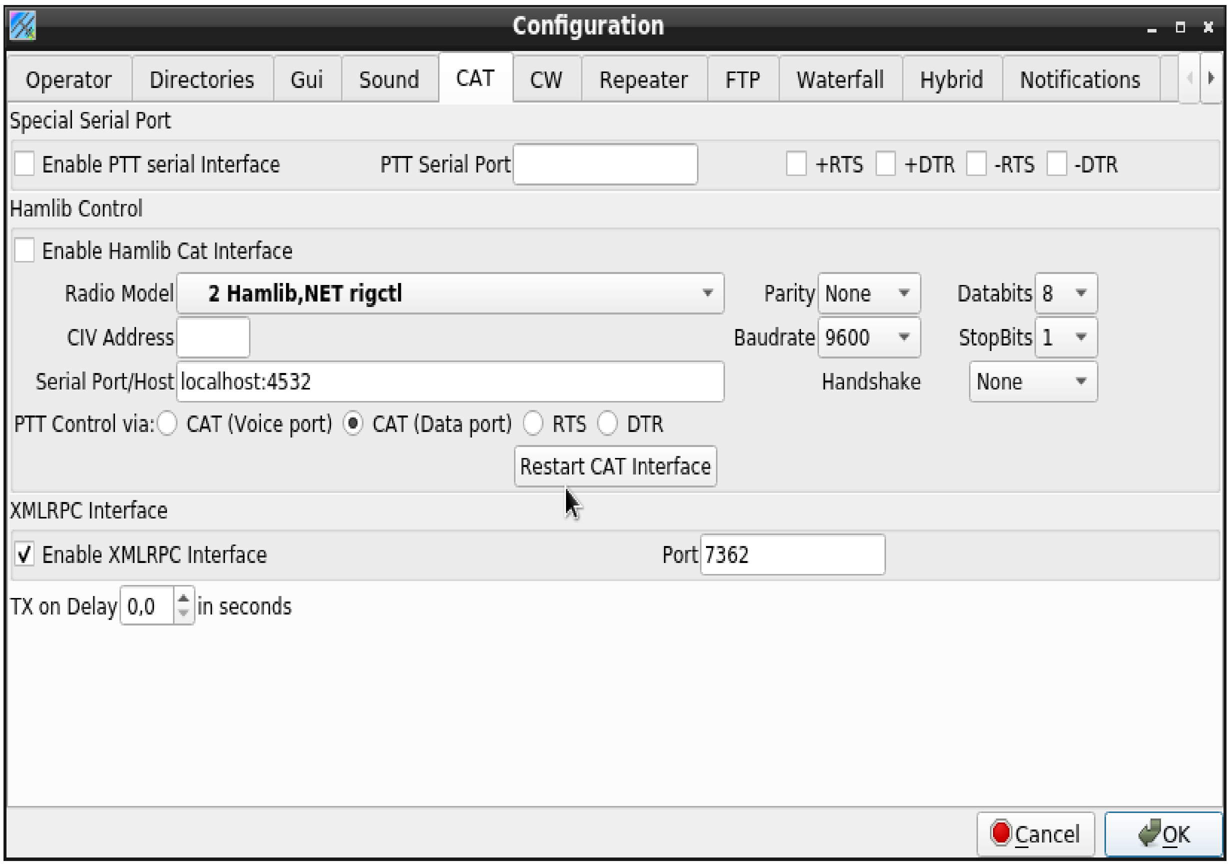

Tiedoston ominaisuudet

Yleiset | Työpöytäkäynnistin

Komento: grig -m 2 -r 127.0.0.1 [Selaa...]

☐ Suorita päätteessä

qsstv:


Configuration

Operator | Directories | Gui | Sound | CAT | CW | Repeater | FTP | Waterfall | Hybrid | Notifications

Special Serial Port

☐ Enable PTT serial Interface   PTT Serial Port: (empty)   ☐ +RTS ☐ +DTR ☐ -RTS ☐ -DTR

Hamlib Control

☐ Enable Hamlib Cat Interface

Radio Model: 2 Hamlib,NET rigctl   Parity: None   Databits: 8

CIV Address: (empty)   Baudrate: 9600   StopBits: 1

Serial Port/Host: localhost:4532   Handshake: None

PTT Control via: ○ CAT (Voice port) ● CAT (Data port) ○ RTS ○ DTR

[Restart CAT Interface]

XMLRPC Interface

✓ Enable XMLRPC Interface   Port: 7362

TX on Delay: 0,0 in seconds

[Cancel] [OK]

And so on. Adding new programs needs that program supports hamlib/rigctld and if so, select only rig model number 2 (Net hamlib rigctld) As simple as that.

Now you can run any individual program, or all programs at same time, without any conflicts with rig cat control.