NAME

rigctl − control radio transceivers and receivers

SYNOPSIS

rigctl [*OPTION*]... [*COMMAND*]...

DESCRIPTION

Control radio transceivers and receivers. **rigctl** accepts **commands** from the command line as well as in interactive mode if none are provided on the command line.

Keep in mind that **Hamlib** is BETA level software. While a lot of backend libraries lack complete rig support, the basic functions are usually well supported. The API may change without publicized notice, while an advancement of the minor version (e.g. 1.1.x to 1.2.x) indicates such a change.

Please report bugs and provide feedback at the e-mail address given in the REPORTING BUGS section. Patches and code enhancements are also welcome.

OPTIONS

This program follows the usual GNU command line syntax, with long options starting with two dashes (‘-’).

Here is a summary of the supported options:

−m, --model=id
Select radio model number. See model list (use ’rigctl -l’).

NB: **rigctl** (or third party software) will use rig model 2 for NET rigctl (rigctld).

−r, --rig-file=device
Use *device* as the file name of the port the radio is connected. Often a serial port, but could be a USB to serial adapter. Typically /dev/ttyS0, /dev/ttyS1, /dev/ttyUSB0, etc. on Linux or COM1, COM2, etc. on Win32.

−p, --ptt-file=device
Use *device* as the file name of the Push-To-Talk device using a device file as described above.

−d, --dcd-file=device
Use *device* as the file name of the Data Carrier Detect device using a device file as described above.

−P, --ptt-type=type
Use *type* of Push-To-Talk device. Supported types are RIG, DTR, RTS, PARALLEL, NONE, overriding PTT type defined in the rig’s backend.

Some side effects of this command are that when type is set to DTR, read PTT state comes from Hamlib frontend, not read from the radio. When set to NONE, PTT state cannot be read or set even if rig backend supports reading/setting PTT status from the rig.

−D, --dcd-type=type
Use *type* of Data Carrier Detect device. Supported types are RIG, DSR, CTS, CD, PARALLEL, NONE.

−s, --serial-speed=baud
Set serial speed to *baud* rate. Uses maximum serial speed from rig backend capabilities as the default.

−c, --civaddr=id
Use *id* as the CI-V address to communicate with the rig. Only useful for Icom rigs.

NB: the *id* is in decimal notation, unless prefixed by *0x*, in which case it is hexadecimal.

−t, --send-cmd-term=char
Change the termination *char* for text protocol when using the *send_cmd* command. The default value is <CR> (0x0d). Non ASCII printable characters can be specified as an ASCII number, in hexadecimal format, prepended with 0x. You may pass an empty string for no termination char. The string '-1' tells rigctl to switch to binary protocol. See the *send_cmd* command for further explanation.

For example, to specify a command terminator for Kenwood style text commands pass "-t ';'" to rigctl. See EXAMPLES below.

−L, --show-conf
List all config parameters for the radio defined with -m above.

−C, --set-conf=parm=val[,parm=val]*
Set config parameter. e.g. stop_bits=2

Use -L option for a list.

−l, --list
List all model numbers defined in **Hamlib** and exit. As of 1.2.15.1 the list is sorted by model number.

N.B. In Linux the list can be scrolled back using Shift-PageUp/ Shift-PageDown, or using the scrollbars of a virtual terminal in X or the cmd window in Windows. The output can be piped to 'more' or 'less', e.g. 'rigctl -l | more'.

−u, --dump-caps
Dump capabilities for the radio defined with -m above and exit.

−o, --vfo
Set vfo mode, requiring an extra VFO argument in front of each appropriate command. Otherwise, VFO_CURR is assumed when this option is not set.

−v, --verbose
Set verbose mode, cumulative (see DIAGNOSTICS below).

−h, --help
Show summary of these options and exit.

−V, --version
Show version of **rigctl** and exit.

N.B. Some options may not be implemented by a given backend and will return an error. This is most likely to occur with the *−−set-conf* and *−−show-conf* options.

Please note that the backend for the radio to be controlled, or the radio itself may not support some commands. In that case, the operation will fail with a **Hamlib** error code.

COMMANDS

Commands can be entered either as a single char, or as a long command name. Basically, the commands do not take a dash in front of them on the command line, as the options do. They may be typed in when in interactive mode or provided as argument(s) in command line interface mode.

Since most of the **Hamlib** operations have a *set* and a *get* method, an upper case letter will be used for *set* method whereas the corresponding lower case letter refers to the *get* method. Each operation also has a long name; in interactive mode, prepend a backslash to enter a long command name.

Example: Use "\dump_caps" to see what this radio can do.

Please note that the backend for the radio to be controlled, or the radio itself may not support some commands. In that case, the operation will fail with a **Hamlib** error message.

Here is a summary of the supported commands (In the case of "set" commands the quoted string is replaced

by the value in the description. In the case of "get" commands the quoted string is the key name of the value returned.):

F, set_freq ’Frequency’
Set ’Frequency’, in Hz.

f, get_freq
Get ’Frequency’, in Hz.

M, set_mode ’Mode’ ’Passband’
Set ’Mode’: USB, LSB, CW, CWR, RTTY, RTTYR, AM, FM, WFM, AMS, PKTLSB, PKTUSB, PKTFM, ECSSUSB, ECSSLSB, FAX, SAM, SAL, SAH, DSB.

Set ’Passband’ in Hz, or ’0’ for the Hamlib backend default.

m, get_mode
Get ’Mode’ ’Passband’.

Returns Mode as a string from *set_mode* above and Passband in Hz.

V, set_vfo ’VFO’
Set ’VFO’: VFOA, VFOB, VFOC, currVFO, VFO, MEM, Main, Sub, TX, RX.

In VFO mode only a single VFO parameter is required.

v, get_vfo
Get current ’VFO’.

Returns VFO as a string from *set_vfo* above.

J, set_rit ’RIT’
Set ’RIT’, in Hz, can be + or -.

A value of ’0’ resets RIT and *should* turn RIT off. If not, file a bug report against the Hamlib backend.

j, get_rit
Get ’RIT’, in Hz.

Z, set_xit ’XIT’
Set ’XIT’, in Hz can be + or -.

A value of ’0’ resets RIT and *should* turn RIT off. If not, file a bug report against the Hamlib backend.

z, get_xit
Get ’XIT’, in Hz.

T, set_ptt ’PTT’
Set ’PTT’, 0 (RX), 1 (TX), 2 (TX mic), 3 (TX data).

t, get_ptt
Get ’PTT’ status.

0x8b, get_dcd
Get ’DCD’ (squelch) status, 0 (Closed) or 1 (Open)

R, set_rptr_shift ’Rptr Shift’
Set ’Rptr Shift’: "+", "-" or something else for none.

r, get_rptr_shift
Get ’Rptr Shift’. Returns "+", "-" or "None".

O, set_rptr_offs ’Rptr Offset’
Set ’Rptr Offset’, in Hz.

o, get_rptr_offs
Get ’Rptr Offset’, in Hz.

C, set_ctcss_tone ’CTCSS Tone’
Set ’CTCSS Tone’, in tenths of Hz.

c, get_ctcss_tone
Get ’CTCSS Tone’, in tenths of Hz.

D, set_dcs_code ’DCS Code’
Set ’DCS Code’.

d, get_dcs_code
Get ’DCS Code’.

0x90, set_ctcss_sql ’CTCSS Sql’
Set ’CTCSS Sql’ tone, in tenths of Hz.

0x91, get_ctcss_sql
Get ’CTCSS Sql’ tone, in tenths of Hz.

0x92, set_dcs_sql ’DCS Sql’
Set ’DCS Sql’ code.

0x93, get_dcs_sql
Get ’DCS Sql’ code.

I, set_split_freq ’Tx Frequency’
Set ’TX Frequency’, in Hz.

i, get_split_freq
Get ’TX Frequency’, in Hz.

X, set_split_mode ’TX Mode’ ’TX Passband’
Set ’TX Mode’: AM, FM, CW, CWR, USB, LSB, RTTY, RTTYR, WFM, AMS, PKTLSB, PKTUSB, PKTFM, ECSSUSB, ECSSLSB, FAX, SAM, SAL, SAH, DSB.

The ’TX Passband’ is the exact passband in Hz, or ’0’ for the Hamlib backend default.

x, get_split_mode
Get ’TX Mode’ and ’TX Passband’.

Returns TX mode as a string from *set_split_mode* above and TX passband in Hz.

S, set_split_vfo ’Split’ ’TX VFO’
Set ’Split’ mode, ’0’ or ’1’, and ’TX VFO’ from *set_vfo* above.

s, get_split_vfo
Get ’Split’ mode, ’0’ or ’1’, and ’TX VFO’.

N, set_ts ’Tuning Step’
Set ’Tuning Step’, in Hz.

n, get_ts
Get ’Tuning Step’, in Hz.

U, set_func ’Func’ ’Func Status’
Set ’Func’ ’Func Status’.

Func is one of: FAGC, NB, COMP, VOX, TONE, TSQL, SBKIN, FBKIN, ANF, NR, AIP, APF, MON, MN, RF, ARO, LOCK, MUTE, VSC, REV, SQL, ABM, BC, MBC, AFC, SATMODE, SCOPE, RESUME, TBURST, TUNER.

Func Status argument is a non null value for "activate", "de-activate" otherwise, much as TRUE/FALSE definitions in C language.

u, get_func
Get ’Func’ ’Func Status’.

Returns Func as a string from *set_func* above and Func status as a non null value.

L, set_level ’Level’ ’Level Value’
Set ’Level’ and ’Level Value’.

Level is one of: PREAMP, ATT, VOX, AF, RF, SQL, IF, APF, NR, PBT_IN, PBT_OUT, CWPITCH, RFPOWER, MICGAIN, KEYSPD, NOTCHF, COMP, AGC (0:OFF, 1:SUPERFAST, 2:FAST, 3:SLOW, 4:USER, 5:MEDIUM, 6:AUTO), BKINDL, BAL, METER, VOXGAIN, ANTIVOX, SLOPE_LOW, SLOPE_HIGH, RAWSTR, SWR, ALC, STRENGTH.

The Level Value can be a float or an integer.

l, get_level
Get ’Level’ ’Level Value’.

Returns Level as a string from *set_level* above and Level value as a float or integer.

P, set_parm ’Parm’ ’Parm Value’
Set ’Parm’ ’Parm Value’

Parm is one of: ANN, APO, BACKLIGHT, BEEP, TIME, BAT, KEYLIGHT.

p, get_parm
Get ’Parm’ ’Parm Value’.

Returns Parm as a string from *set_parm* above and Parm Value as a float or integer.

B, set_bank ’Bank’
Set ’Bank’. Sets the current memory bank number.

E, set_mem ’Memory#’
Set ’Memory#’ channel number.

e, get_mem
Get ’Memory#’ channel number.

G, vfo_op ’Mem/VFO Op’
Perform ’Mem/VFO Op’.

Mem VFO operation is one of: CPY, XCHG, FROM_VFO, TO_VFO, MCL, UP, DOWN, BAND_UP, BAND_DOWN, LEFT, RIGHT, TUNE, TOGGLE.

g, scan ’Scan Fct’ ’Scan Channel’
Perform ’Scan Fct’ ’Scan Channel’.

Scan function/channel is one of: STOP, MEM, SLCT, PRIO, PROG, DELTA, VFO, PLT.

H, set_channel ’Channel’
Set memory ’Channel’ data. Not implemented yet.

h, get_channel
Get memory ’Channel’ data. Not implemented yet.

A, set_trn ’Transceive’
Set ’Transceive’ mode (reporting event): OFF, RIG, POLL.

a, get_trn
Get 'Transceive' mode (reporting event) as in *set_trn* above.

Y, set_ant 'Antenna'
Set 'Antenna' number (0, 1, 2, ..).

y, get_ant
Get 'Antenna' number (0, 1, 2, ..).

***, reset 'Reset'**
Perform rig 'Reset'.

0 = None, 1 = Software reset, 2 = VFO reset, 4 = Memory Clear reset, 8 = Master reset. Since these values are defined as a bitmask in rig.h, it should be possible to AND these values together to do multiple resets at once, if the backend supports it or supports a reset action via rig control at all.

b, send_morse 'Morse'
Send 'Morse' symbols.

0x87, set_powerstat 'Power Status'
Set power On/Off/Standby 'Power Status'.

0 = Power Off, 1 = Power On, 2 = Power Standby. Defined as a bitmask in rig.h.

0x88, get_powerstat
Get power On/Off/Standby 'Power Status' as in *set_powerstat* above.

0x89, send_dtmf 'Digits'
Set DTMF 'Digits'.

0x8a, recv_dtmf
Get DTMF 'Digits'.

_, get_info
Get misc information about the rig (no VFO in 'VFO mode' or value is passed).

1, dump_caps
Not a real rig remote command, it just dumps capabilities, i.e. what the backend knows about this model, and what it can do.

TODO: Ensure this is in a consistent format so it can be read into a hash, dictionary, etc. Bug reports requested.

N.B.: This command will produce many lines of output so be very careful if using a fixed length array! For example, running this command against the Dummy backend results in over 5kB of text output.

VFO parameter not used in 'VFO mode'.

2, power2mW 'Power [0.0..1.0]' 'Frequency' 'Mode'
Returns 'Power mW'

Converts a Power value in a range of *0.0* ... *1.0* to the real transmit power in milli-Watts (integer). The *frequency* and *mode* also need to be provided as output power may vary according to these values.

VFO parameter not used in 'VFO mode'.

4, mW2power 'Power mW' 'Frequency' 'Mode'
Returns 'Power [0.0..1.0]'

Converts the real transmit power in milli-Watts (integer) to a Power value in a range of *0.0 ... 1.0*. The *frequency* and *mode* also need to be provided as output power may vary according to these values.

VFO parameter not used in 'VFO mode'.

w, send_cmd 'Cmd'

Send raw command string to rig. This is useful for testing and troubleshooting rig commands and responses when developing a backend.

For binary protocols enter values as \0xAA\0xBB. Expect a 'Reply' from the rig which will likely be a binary block or an ASCII string depending on the rig's protocol (see your radio's computer control documentation).

The command terminator, set by the *send-cmd-term* option above, will terminate each command string sent to the radio. This character should not be a part of the input string.

EXAMPLES

Start **rigctl** for a Yaesu FT-920 using a USB to serial adapter on Linux in interactive mode:

$ rigctl -m 114 -r /dev/ttyUSB1

Start **rigctl** for a Yaesu FT-920 using COM1 on Win32 while generating TRACE output to **stderr**:

$ rigctl -m 114 -r COM1 -vvvvv

Start **rigctl** for a Yaesu FT-920 using a USB to serial adapter while setting baud rate and stop bits:

$ rigctl -m 114 -r /dev/ttyUSB1 -s 4800 -C stop_bits=2

Start **rigctl** for an Elecraft K3 using a USB to serial adapter while specifying a command terminator for the 'w' command:

$ rigctl -m 229 -r /dev/ttyUSB0 -t';'

Connect to a running **rigctld** with rig model 2 ("NET rigctl") on the local host and specifying the TCP port, setting frequency and mode:

$ rigctl -m 2 -r localhost:4532 F 7253500 M LSB 0

DIAGNOSTICS

The **-v**, **--verbose** option allows different levels of diagnostics to be output to **stderr** and correspond to -v for BUG, -vv for ERR, -vvv for WARN, -vvvv for VERBOSE, or -vvvvv for TRACE.

A given verbose level is useful for providing needed debugging information to the email address below. For example, TRACE output shows all of the values sent to and received from the radio which is very useful for radio backend library development and may be requested by the developers.

EXIT STATUS

rigctl exits with:
0 if all operations completed normally;
1 if there was an invalid command line option or argument;
2 if an error was returned by **Hamlib**.

BUGS

set_chan has no entry method as of yet, hence left unimplemented.

This almost empty section...

REPORTING BUGS

Report bugs to <hamlib-developer@lists.sourceforge.net>.

We are already aware of the bugs in the previous section :-)

AUTHORS

Written by Stephane Fillod, Nate Bargmann, and the Hamlib Group

<http://www.hamlib.org>.

COPYRIGHT

SEE ALSO

hamlib(3), **rigctld(8)**