NAME

rotctl − control antenna rotators

SYNOPSIS

rotctl [*OPTION*]... [*COMMAND*]...

DESCRIPTION

Control antenna rotators. **rotctl** 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 s summary of the supported options:

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

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

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

−s, --serial-speed=baud
Set serial speed to *baud* rate. Uses maximum serial speed from rotator backend capabilites as default.

−t, --send-cmd-term=char
Change the termination *char* for text protocol when using the *send_cmd* command. The default value is <CR>. 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 rotctl to switch to binary protocol. See the *send_cmd* command for further explanation.

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

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

Use -L option for a list.

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

−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. ’rotctl -l | more’.

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

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

−V, −−version
Show version of **rotctl** 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 rotator to be controlled, or the rotator 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, 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 "\get_info" to see the rotor's info.

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

A summary of commands is included below (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.):

P, set_pos 'Azimuth' 'Elevation'
Set position: Azimuth and Elevation as double precision floating point values.

p, get_pos
Get position: 'Azimuth' and 'Elevation' as double precision floating point values.

M, move 'Direction' 'Speed'
Move the rotator in a specific direction at the given rate.

Values are integers where Direction is defined as 2 = Up, 4 = Down, 8 = Left, and 16 = Right. Speed is an integer between 1 and 100. Not all backends that implement the move command use the Speed value. At this time only the gs232a utilizes the Speed parameter.

S, stop Stop the rotator.

K, park
Park the antenna.

C, set_conf 'Token' 'Value'
Set a configuration parameter. It is safe to give "Token" a value of '0' (zero). "Value" may be a string up to 20 characters.
See -L output

R, reset 'Reset'
Reset the rotator.

Integer value of '1' for Reset All.

_, get_info
Get misc information on the rotator.

At the moment returns 'Model Name'.

w, send_cmd 'Cmd'

Send raw command string to the rotator.
<CR> (or send-cmd-term, see *-t* option) is appended automatically at the end of the command for text protocols. For binary protocols, enter values as \0xAA\0xBB

Locator Commands

These commands offer conversions of Degrees Minutes Seconds to other formats, Maidenhead square locator conversions and distance and azimuth conversions.

L, lonlat2loc 'Longitude' 'Latitude' 'Loc Len [2-12]'

Returns the Maidenhead locator for the given 'Longitude' and 'Latitude'.

Both are floating point values. The precision of the returned square is controlled by 'Loc Len' which should be an even numbered integer value between 2 and 12.

For example, "+L -170.000000 -85.000000 12\n" returns "Locator: AA55AA00AA00\n".

l, loc2lonlat 'Locator'

Returns 'Longitude' and 'Latitude' in decimal degrees at the approximate center of the requested grid square (despite the use of double precision variables internally, some rounding error occurs). West longitude is expressed as a negative value. South latitude is expressed as a negative value. Locator can be from 2 to 12 characters in length.

For example, "+l AA55AA00AA00\n" returns "Longitude: -169.999983\nLatitude: -84.999991\n".

D, dms2dec 'Degrees' 'Minutes' 'Seconds' 'S/W'

Returns 'Dec Degrees', a signed floating point value.

Degrees and Minutes are integer values and Seconds is a floating point value. S/W is a flag with '1' indicating South latitude or West longitude and '0' North or East (the flag is needed as computers don't recognize a signed zero even though only the Degrees value only is typically signed in DMS notation).

d, dec2dms 'Dec Degrees'

Returns 'Degrees' 'Minutes' 'Seconds' 'S/W'.

Values are as in dms2dec above.

E, dmmm2dec 'Degrees' 'Dec Minutes' 'S/W'

Returns 'Dec Degrees', a signed floating point value.

Degrees is an integer value and Minutes is a floating point value. S/W is a flag with '1' indicating South latitude or West longitude and '0' North or East (the flag is needed as computers don't recognize a signed zero even though only the Degrees value only is typically signed in DMS notation).

e, dec2dmmm 'Dec Deg'

Returns 'Degrees' 'Minutes' 'S/W'.

Values are as in dmmm2dec above.

B, qrb 'Lon 1' 'Lat 1' 'Lon 2' 'Lat 2'

Returns 'Distance' 'Azimuth' where Distance is in km and Azimuth is in degrees.

All Lon/Lat values are signed floating point numbers.

A, a_sp2a_lp 'Short Path Deg'
Returns 'Long Path Deg' or -RIG_EINVAL upon input error..

Both are floating point values within the range 0.00 to 360.00.

a, d_sp2d_lp 'Short Path km'
Returns 'Long Path km'.

Both are floating point values.

EXAMPLES

Start **rotctl** for RotorEZ using the first serial port on Linux:

$ rotctl -m 401 -r /dev/ttyS0

Start **rotctl** for RotorEZ using COM2 on Win32:

$ rotctl -m 401 -r COM2

Start **rotctl** using **rpc.rotd** and querying the position:

$ rotctl -m 101 -r localhost \get_pos

Connect to a running **rotctld** with rotor model 2 ("NET rotctl") on the local host and specifying the TCP port, and querying the position:

$ rotctl -m 2 -r localhost:4533 \get_pos

DIAGNOSTICS

The **-v**, **--version** 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

rotctl 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

This suspiciously empty section...

REPORTING BUGS

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

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

AUTHOR

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

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

COPYRIGHT

SEE ALSO

hamlib(3), **rotctld**(8)