

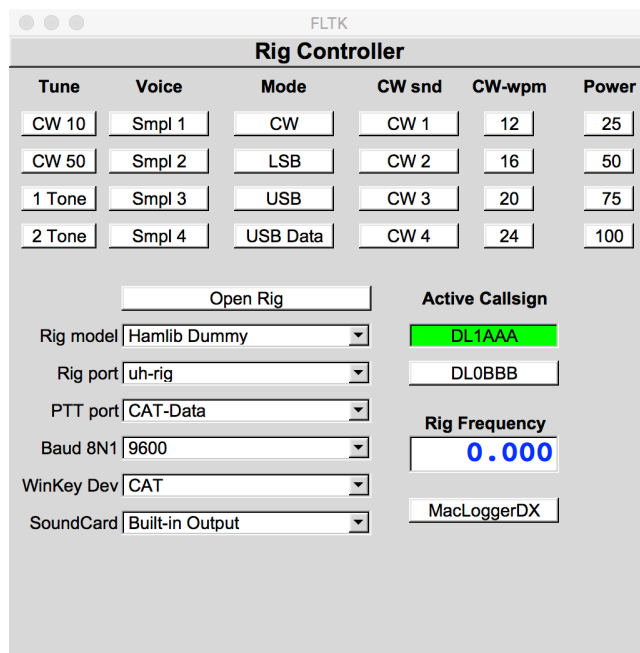
Manual for the Rig Controller

– DL1YCF –

This program is essentially a Hamlib rigctld daemon with a graphical user interface, which in addition has some additional "buttons" to control the rig (setting RF power level, CW keying speed, mode etc.). I made this program for two purposes:

- a) having a central piece of software that actually communicates with the rig, and acting as a server for digimode and logging programs such that they can simultaneously "talk" to the rig via the RigController program. This is especially helpful if using a microHam device under MacOS, because many digimode and logging programs support Hamlib but not microHam devices.
- b) having a software to adjust output power etc. because one of my rig has a defective knob to do so manually (Kenwood TS570 owners know what I am talking about).
- c) Because it was easy to integrate, the RigController also can send CW (using either the built-in CW feature of the rig or a WinKeyer) and voice samples (because I do not have a voice keyer), it also can generate a one-tone and a two-tone audio signal for TX testing.
- d) Because I found no other way to connect my logging application (MacLoggerDX) to the radio, this software also generates, if told to do so, openURL events that tell MacLoggerDX about the actual frequency and mode.

If the Rig application is started for the first time, it opens its only window that looks like this:



The screenshot shows a window titled "Rig Controller" with a table of settings and configuration options below it.

Tune	Voice	Mode	CW snd	CW-wpm	Power
CW 10	Smpl 1	CW	CW 1	12	25
CW 50	Smpl 2	LSB	CW 2	16	50
1 Tone	Smpl 3	USB	CW 3	20	75
2 Tone	Smpl 4	USB Data	CW 4	24	100

Below the table, there is an "Open Rig" button. To the right, there is an "Active Callsign" section with a green button labeled "DL1AAA" and a text field labeled "DL0BBB". Below that, there is a "Rig Frequency" section with a blue button labeled "0.000" and a text field labeled "MacLoggerDX".

Configuration options on the left include:

- Rig model: Hamlib Dummy
- Rig port: uh-rig
- PTT port: CAT-Data
- Baud 8N1: 9600
- WinKey Dev: CAT
- SoundCard: Built-in Output

The entries in the menus **Rig model**, **Rig port**, ..., **SoundCard** are those last used by the program (this is stored in `$HOME/.rigcontrol/Prefs`). The values shown in the above panel are the built-in default values. For testing purposes, you can click **Open Rig**, this should work.

Opening a connection to the rig

The first thing you have to do is to open the connection to the rig. Here you have to choose the right settings as described below and click the "Open Rig" button. It becomes green, and if it stays green for a while, then the connection to the rig is established. At this time, the blue number should change to the current frequency of the rig (in MHz). The choices you can make are explained now:

Rig Model. Choose the model of your rig (if it is supported by Hamlib).

Rig Port. Choose the serial port used for connection to the rig, or **uh-rig**. This special name indicates that communication with the rig is over a microHam interface (in my version of hamlib).

PTT Port. Choose a serial port if it is used for PTT switching via DTR, or **CAT-Data / CAT-Mic** if PTT is switched via CAT commands, or **uh-ptt** if PTT switching is done via a microHam interface. If your rig supports two different CAT commands for switching in TX mode (for using audio from the front (microphone) or the rear (data) jacket, you have to select either CAT-Mic and CAT-Data accordingly. If your rig only has a single CAT command for activating the TX, either choice will do.

Baud 8N1. Choose the baud rate for the CAT connection to the rig (this applies both to a serial port and a microHam interface). Note that we always use 8 data bits, no parity, and hardware handshake (although it would be easy to extend this).

WinKey Dev. Choose the serial port connecting a WinKey device, or **uh-wkey** if the Winkeyer inside a microham interface is to be used, or **CAT** if CW is done via hamlib commands (if the rig supports this!).

The screen-shot on the previous page shows the default values used the first time you start the rig controller application. Note that for the virtual "dummy" rig, the rig port will not be used.

After the connection to the rig is established, the **Open Rig** should be highlighted (green) and one of the buttons in each of the **Mode**, **CW-wpm** and **Power** rows should turn pink. The rig (and the Winkey device) is switched to 20 wpm, its RF power is changed to the nearest value from the menu (25, 50, 75 or 100 watts) and what is currently selected for the rig, and the current mode is highlighted. In my example it looks like this:

FLTK

Rig Controller

Tune	Voice	Mode	CW snd	CW-wpm	Power
CW 10	Smpl 1	CW	CW 1	12	25
CW 50	Smpl 2	LSB	CW 2	16	50
1 Tone	Smpl 3	USB	CW 3	20	75
2 Tone	Smpl 4	USB Data	CW 4	24	100

Open Rig

Rig model Kenwood TS-570D

Rig port uh-rig

PTT port uh-ptt

Baud 8N1 9600

WinKey Dev uh-wkey

SoundCard microHAM CODEC

Active Callsign

DL1YCF

DL0XK

Rig Frequency

14.029

MacLoggerDX

which means that I control a Kenwood TS-570D via a microHam interface (in my case, my DigiKeyer-II), and that I use this device also for PTT switching and for CW keying. As long as the **Open Rig** button is highlighted, a server daemon is active that is functionally equivalent to the hamlib **rigctld** daemon. This means you can connect from other applications to the rig through our rig controller.

For example, if you now fire up an application such as FLDIGI or WSJT-X, you can connect to the "real" transceiver if you use the "**Hamlib NET rigctl**" virtual rig. If the digimode program wants to get/set the VFO frequency or set/unset PTT, it connects to the rig controller and this one then "talks" with the real rig (Kenwood TS570d, in this example). You can watch how the rig frequency changes when you switch to another band within WSJT-X, or if you change the mode to LSB or CW within FLDIGI.

Note that in my case, the primary callsign is set to DL1YCF, and the alternate callsign is set to DL0XK (a club station call). Currently, DL1YCF is active (highlighted green). The only effect of the active callsign is that it can be inserted

into CW messages. The call signs and the CW messages are stored in the file `$HOME/.rigcontrol/Settings`, which in my case reads

```
DL1YCF
DL0XK
#
cq cq cq de # # cq de # # + k
test #
cfm ur 599
```

The first two lines contain the primary and secondary call sign, and the following four lines contain the text to be sent in CW by clicking the buttons **CW 1** through **CW 4**. Note that the "#" in the CW texts will be expanded to the active call sign. Switching back and forth between the primary and secondary call just involves clicking the respective button. So with this setup, clicking **CW 1** just sends DL1YCF in morse code.

The **Tune** menu

Clicking the Buttons **CW 1** or **CW 50** switches the rig to FM mode and sets 10 or 50 watt RF output power. Thus a carrier of 10 or 50 watts is produced, which may be used for tuning. Clicking the Buttons **1 Tone** or **2 Tone** will activate PTT and produce a one-tone or two-tone audio signal on the selected sound card. This can be used for TX/PA measurements.

The **Voice** menu

Clicking the buttons **Smpl1** through **Smpl4** will play a pre-recorded audio file (WAV format) whose file name must be (with `x=1,2,3,4`) `$HOME/.rigcontrol/voicex.wav`. This implements a voice keyer for CQ calls etc. The samples are read into memory when the program starts.

Support for **MacLogger DX**

This is only interesting for those using this logbook program on the Macintosh. Since our rig controller now occupies the port to the rig, we have to tell MacLoggerDX about the current frequency and mode. We do this periodically by

using openURL commands. For example, to set the frequency in MacLoggerDX to 7024 kHz and the mode to CW, the program just exec's the MacOS command (via fork/execl/waitpid, for the UNIX gurus)

```
/usr/bin/open -g mldx://tune?freq=7.024&mode=CW
```

This way, you always have the correct Frequency in your logbook. This command is only executed if the Frequency differs more than 3 kHz from the frequency sent last. This support for MacLoggerDX is only activated if the **MacLoggerDX** button is clicked and active as long as this button is highlighted (green colour).