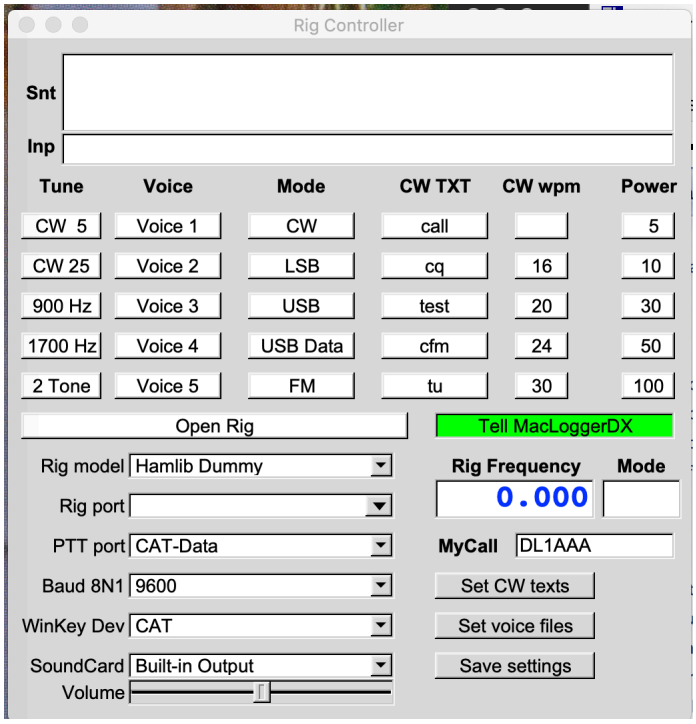
Manual for the Rig Controller

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 (on MacOS computers).



On non-Apple computers the field "Tell MacLoggerDX" does not appear. Note that when starting the rig controller later, the entries in the menus **`Rig model`**, **`Rig port`**, ..., **`SoundCard`** may be different, since they are then read in from preferences files stored in the `$HOME/.rigcontrol` folder. For testing purposes, you can click "Open Rig", this works since the hamlib "Dummy" rig needs no port. If the connection is established, the "Open Rig" bar get high-lighted (green), and soon thereafter the rig settings are shown by high-lighting (in pink) the entries in the Mode and Power column (default mode of the "Dummy" rig: FM, default output power: 5 W). In the "CW wpm" column, the default CW speed (20 wpm) should be highlighted. Moreover, the "Rig Frequency" field should soon be filled with the Rig's dial frequency, which is upon startup 145 MHz for the "Dummy" rig.

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 (a successful connection means that the "Open Rig" button becomes permanently high-lighted in green).
The following choices can (have to) be made:

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. The serial ports found in the system when starting the program are already filled, as well as ":19090" which is the default port to connect to some SDR programs offering CAT over TCP, and "uh-rig" which is the correct port if Hamlib auto-detects a microHam interface such as the DigiKeyer or the MicroKeyer. You can also type in any other TCP port or serial device (e.g. "/dev/ttyS1", or "192.168.1.100:19090"). Note that if you save the preferences, this choice will be the default upon next startup.

PTT Port. Choose a serial port for using PTT switching via DTR. Besides the serial ports found in the system, the values **CAT-Data** and **CAT-Mic** are pre-filled in the list. These are used if CAT commands are used to switch PTT. If the rig supports this, **CAT-Mic** is used to switch PTT if using the microphone jack for audio input, while **CAT-Data** is used if a jack on the rear or the rig is used for audio input. Furthermore, "uh-ptt" is pre-set which is used for hamlib talking to microHam devices (see above).

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). This field is not used if a TCP connection to the rig is used.

WinKey Dev. Choose the serial port connecting a WinKey (K1EL) 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!). Note that if CAT is used, then the default CW speed is 20 wpm, while if a WinKey device is used, the default speed is given by the position of the speed pot.

SoundCard. This is only needed for the buttons in the "Voice" column, and for the buttons "900 Hz", "1700 Hz", and "2 Tone". These buttons send "sound" to the rig (either pre-recorded wav files for the voice keyer, or some test tones), and a sound card connected to the microphone input of the rig must be selected here. Modern rigs have such sound cards built into the rig. The **Volume** slider can be used to adjust the AF level.

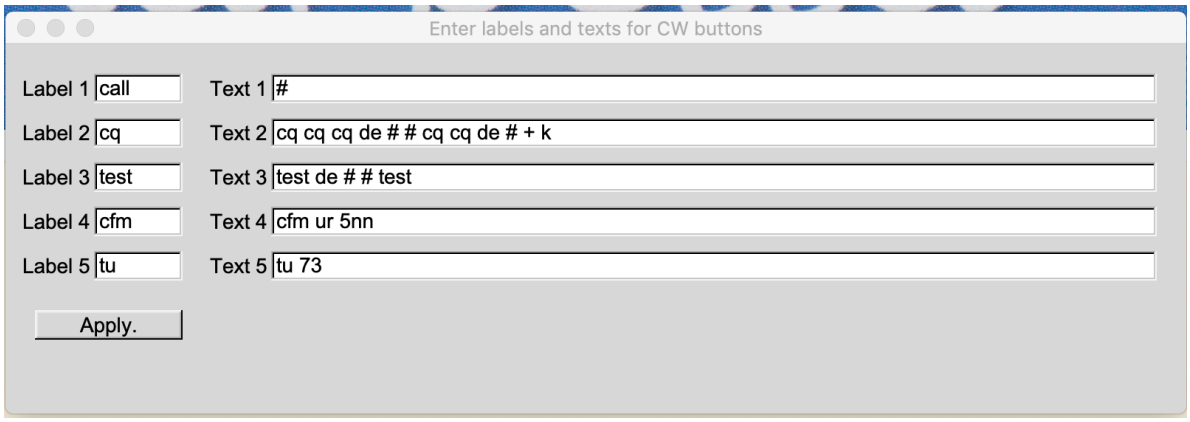
Changing the default settings

There are several settings (beside the rig connection parameters) which should be adjusted before actually using the rig controller. This first one is the own call-sign in the field "MyCall": just click in the text field, type in your call sign and hit "enter". The call-sign will then be converted to upper case. This call sign needs to be set correctly for the fixed CW texts.

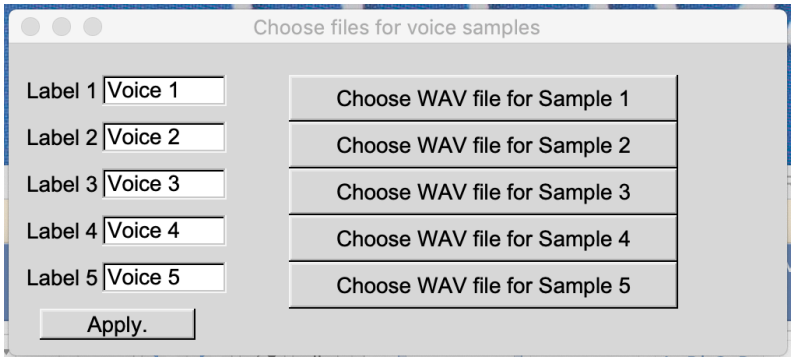
What comes next is are the button labels in the "CW TXT" column, as well as the CW texts to be transmitted when hitting one of the buttons. The default setting for the buttons is ("#" expands to your call sign in the MyCall text field)

Button label	CW text
call	#
cq	cq cq cq de # # cq cq de # + k
test	test de # # test
cfm	cfm ur 5nn
tu	tu 73

You can change both the button labels and the CW texts by hitting the "Set CW texts" button. The following window pop up, and you can edit both the labels and the CW texts. If you are finished editing, click "Apply".



This program also contains a "voice keyer", that is, digital voice (stored in WAV files) can be sent to the rig through a sound card (the sound card may well be inside the rig). Up to five different samples can be chosen by hitting one of the buttons in the "Voice" column. These files can be selected by hitting the "Set voice files" button. The following window pops up:



In the text fields, you can edit the button label, and in the right part of the window, clicking one of the "Choose ...." buttons opens a file dialog. The wav files need to have a sample rate of 48000 Hz. Hitting "Apply" stores the settings. Note that only the names of the wav files are stored, not the files themselves. So if you move the files after selecting them, they will not be loaded automatically when the rig controller is started the next time.

The settings (Rig settings, CW texts, voice files) can be saved by clicking the "Save settings" button. In addition, all settings are saved each time a connection to a rig has been successfully established.

Rig controller functions: the menus

The TUNE column

Button	Function
CW 5	Temporarily set the rig to FM mode and 5 Watts, and activate PTT. This should emit a 5 watts carrier that can be used for tuning.
CW 25	Temporarily set the rig to FM mode and 25 Watts, and activate PTT. This should emit a 25 watts carrier that can be used for tuning.
900 Hz	Activate PTT and send a 900 Hz sine tone to the selected sound card.
1700 Hz	Activate PTT and send a 1700 Hz sine tone to the selected sound card.
2 Tone	Activate PTT and send a two-tone (900 and 1700 Hz) audio signal to the selected sound card.

Note: while active, the buttons are high-lighted (green). Hitting the button a second time removes PTT and stops the action. When leaving the "CW 5" and "CW 25" modes, the original mode and power is restored.

The VOICE column

Hitting one of the buttons sends the audio stored in the corresponding wav file to the selected sound card.

The MODE column

Hitting one of the buttons changes the mode of the rig. "USB DATA" is sometimes called "PKT USB". On rigs that support it, this mode is primarily used for digital operation.

The CW TXT column

Hitting one of the buttons sends the corresponding CW text. CW text sent is documented in the "Snt" text field (top-most field of the main window).

The CW wpm column

Hitting one of the buttons sets the CW speed (in wpm). If using a WinKey (K1EL) device, the topmost button has the label "Pot" and means that the speed is taken from the speed pot of the WinKey device. For CAT CW, the topmost button choose 12 wpm.

The Power column

Hitting one of the buttons sets the RF output power of the rig.

Typing in free CW text

In the text field labeled "Inp", free text can be entered that will be sent as CW. The hamlib API does not allow to send single characters (only entire words). So you can type and also correct within a word, but as soon a blank space is entered or you hit the enter key, the word is sent.

Connecting the rig from WSJT-X or Fldigi

If the rig controller is connected to a rig via a serial line, other programs such as digimode programs can no longer use this serial line. To circumvent this problem, the digimode program (WSJT-X or Fldigi, for example) need to connect to the Rig controller via TCP, and then the requests are forwarded to the rig.
This can be done if the digimode program supports the Hamlib API. WSJT-X does so by default, and in Fldigi you have to check "Use Hamlib" in the "Hamlib" tab. As rig model, simply choose "Hamlib Net rigctl". The default port is :4532 and usually need not be specified. This way, you can have WSJT-X and the rig controller both running on your computer and talking to the rig.

Support for MacLoggerDX (MacOS only)

This is only interesting for those using this logbook program on the Macintosh. If the rig controler controls the rig via a serial line, this line is "occupied" and MacLoggerDX (the logbook program that I use on my Mac) cannot communicate directly with the rig. To inform MacLoggerDX about the currently used frequency and mode, activate this mode by hitting "Tell MacLoggerDX" if it is not high-lighted in green.

The rig controller periodically obtains the current frequency from the rig, and sends it to MacLoggerDX if it has changed by more than 3 kHz from the value sent last. The mode that is sent to MacLoggerDX must be entered manuall in the "Mode" text field, this way you can enter the correct mode (FT8, PSK, RTTY) if you work digital, since the rig mode is most likely USB or USB DATA in either case.

Note on compiling the Rig controller

Libraries needed for the rig controller include "Portaudio19" (for sending sound to the rig) and "fltk-1.3" (the GUI). "hamlib" is also needed but we have to compile hamlib alongside with the rig controller because the rig controller uses parts of hamlib's source code. You have to use the most recent version of hamlib (4.0) since some of the internal processing changed since 3.3. To compile hamlib, the packages "autoconf" and "libtool" (and possibly more) are needed.

Sample commands are typed in red, and output grabbed from the screen is given in blue:

It is easiest to create an empty directory <dir> and clone the source code of hamlib and the rig controller there via the commands

```
cd <dir>
git clone http://github.com/dl1ycf/hamlib.git
git clone http://github.com/dl1ycf/RigCtldGUI
```

For hamlib, you may as well take the source code from the official repository:

```
git clone http://github.com/hamlib/hamlib
```

Most of the time, both repositories should be identical, but I try to keep a version of hamlib from which I know it works with the rig controller on my own github account.

Start with compiling hamlib. Go to the hamlib directory and configure hamlib:

```
cd <dir>
autoreconf -i
./configure
```

The output may differ from system to system, it should end similar to what is shown here in blue:

```
----------------------------------------------------------------------

 Hamlib Version 4.0~git configuration:

 Prefix          /usr/local
 Preprocessor    gcc -E
 C Compiler      gcc -g -O2
 C++ Compiler    g++ -g -O2

 Package features:
    With C++ binding                yes
    With Perl binding               no
    With Python binding             no
    With TCL binding                no
    With Lua binding                no
    With rigmem XML support         no
    With Readline support           yes

    Enable HTML rig feature matrix  no
    Enable WinRadio                 yes
    Enable USRP                     no
    Enable USB backends             yes
    Enable shared libs              yes
    Enable static libs              yes

----------------------------------------------------------------------
```

Then, you can compile and install hamlib with the commands

```
make -j 4
sudo make install
```

The "make install" usually requires administrator privileges, hamlib is installed in /usr/local.

Compilation of the rig controller is easy:

```
cd <dir>
make
```

On MacOS systems, you may want to create a "click-able" app file. This can be done with

```
make app
```