Quick Start Guide for pyKeyer

A VERY ROUGH DRAFT!!!!!!!!!!

J. B. Attili, AA2IL
April 2025

1. Overview

Over the years, a number of very good logging programs have been developed. The most popular of these today is the N1MM logger. Unfortunately, this is a Windoz-only application and is very difficult to get working under linux. My experience with the other options available for linux are either not being actively maintained and/or are too bloated and/or lacking to be useful for contesting. Hence, the development of yet another keying/logging program.

This program is part of a suite of software that I use in my day-to-day amateur radio activities. This program provides the logging function and is fully integrated with the other components of this suite – see Figure 1.1. Although primarily developed for contesting, provisions have been included for non-contest operations such as rag-chewing, POTA/SOTA, etc.. Inasmuch as regular practice is requisite for a proficient operator, this program includes provisions for both receiving and sending practice as well.

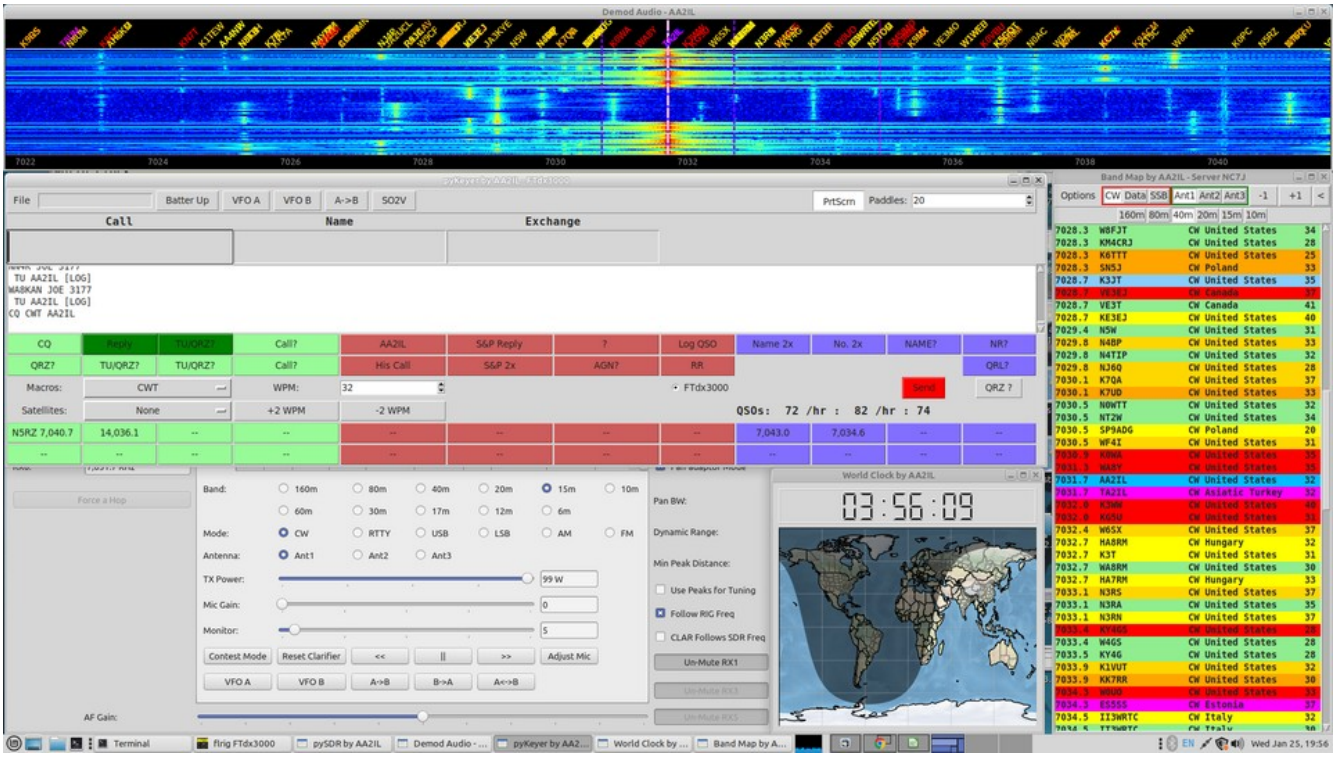

Figure 1.1. Screenshot of AA2IL suite of ham radio software during a typical CWT contest.

2. Background

I was originally licensed as a teenager in the late 1970s and always had an affinity for CW and chasing DX. Like many in our great hobby, life (i.e. job, family, etc.) got in the way and I was dormant for over 25-years. When I finally returned to the hobby in 2013, it was evident that things had changed dramatically. Not only was the equipment so much better but the computer had become an integral part of ham radio. As a professional electrical engineer, computers were always a tool for my work and I actually enjoy "hacking away."

Much of the software in common use in ham shacks runs under Windows. This was a bit of a surprise to me as, IMHO, development and interfacing with hardware is much easier linux. Although there are environments that allow Windows software to run under linux, configuring these environments can sometimes be tricky and all to often, the results are unsatisfactory. As I was unable to find a suitable logger for linux, I decided to set about developing my own

Another aspect of the hobby that had changed is that the CW speeds had increased dramatically, especially in radio sport. In the past, when morse code was a required element for obtaining a ham license, there were always slower ops around to converse and practice with to help build up proficiency. This is much more difficult now but, fortunately, there are computer programs to fill this void as well as training classes offered by clubs such as CWops and LICW. Accordingly, one of the features I wanted to include in my logger program was the ability to provide off-air practice and feedback for both sending and receiving CW. (Yes, I am well aware of and regularly Morse Runner and RufzXP but I also wanted to be practicing using the same interface I'd be using in an actual contest.)

3. Installation Instructions

Detailed installation instructions are provided in the README.md file in the github repository containing the source code.

3.1 Linux Installation – PC

Recent releases on linux distributions have imposed severe restrictions on the ability to install additional python libraries. This is meant to protect the integrity of the system libraries since python code has become a prevalent part of the OS. The recommended installation method is to use conda/mini-conda as described in README.md.

3.2 Linux Installation – Rasberry PI

The current Rasberry Pi OS is still very open so you are free to modify the system-wide libraries or use conda/min-conda as described above. I still use the former approach on my RPi 4 running Bookworm but I expect somewhere down the line, a "container" approach will be required.

3.3 Windoz Installation

Python is reasonably well supported on Windoz so it is preferable to install the python interpreter under windows and run the source from the command line, similar to linux. The advantage of this approach is that the latest code can be pulled from github and used. The downside is that there is considerably more "fiddling" with this approach.

Binary installers are available for Windoz 11 which provide a more-typical point-and-click experience. These binaries are rather large (>100MB) and are difficult to host on Github. If you'd rather use this approach, please e-mail me and I will send you a link to the latest binary.

3.4 Mac OS Installation

The Mac OS is linux-based and there should be no reason why these codes cannot be easily made to run on a Mac. Although my main shack computer is an old iMac (cira 2012), I have long since blown away the Mac OS and have no means of testing this theory. Any volunteers?

3.5 Support Files

TBD

4. Getting Started

Which ever OS and installation method, there are three codes available in this package:

pyKeyer.py	– The main logger app
paddling.py	– A program for sending practice
qrz.py	- A program to quickly look-up known information about an op

The latter two are standalone versions of features that are part of pyKeyer but are very useful in their own right. In what follows, we'll begin with these two codes as they have fewer options to become confused by.

All three applications in this package require some minimal amount of configuration information to work properly. This information is stored in a file called .keyerrc in the user's home directory. Inasmuch as this file doesn't exist, the first time you start any of these apps, you will be presented with a settings dialog where you can fill in this information – see Figure 4.1.

At a bare minimum, you will need to fill in the following:

My Call - your call sign, self explanatory;
My Keyer Device - the type of keyer you are using, usually WINKEYER;
My Keyer Device ID - an identifier for the USB port that the keyer is connected to; and
My Data Dir - the location of the various support files.

On windows, the keyer device is usually the com port used by device (e.g. COM4) and can be found using the control panel. Under linux, a suitable descriptor can be found via the command

```
python3 -m serial.tools.list_ports -v
```

This command also works on Windoz if you've installed the python interpreter. (You can also run paddling.py and it will issue the command for you.) On my shack computer, this results in

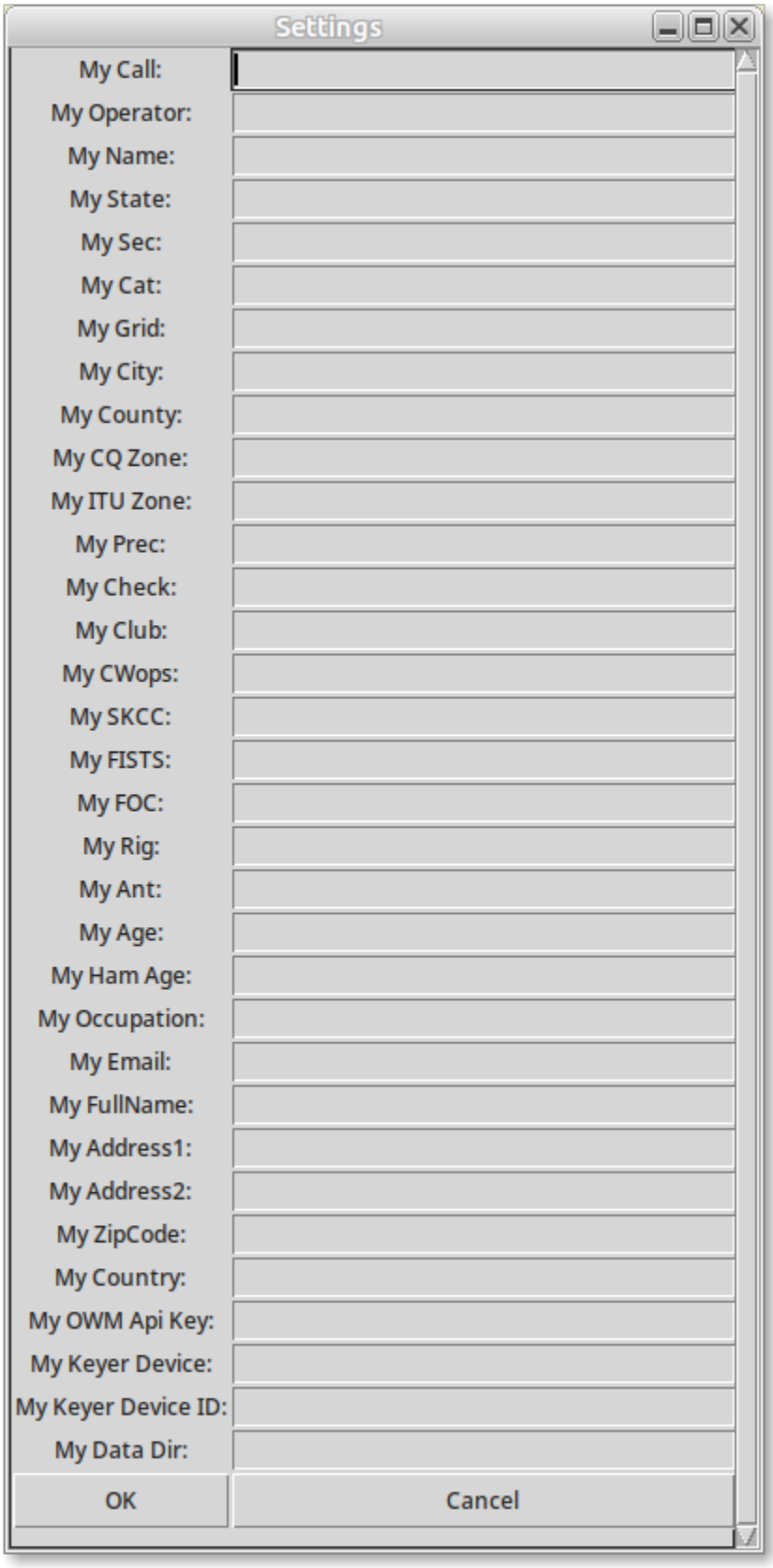

Figure 4.1. Empty Settings Dialog.

```
:
:
/dev/ttyUSB0
    desc: USB2.0-Ser!
    hwid: USB VID:PID=1A86:7523 LOCATION=3-4.3
/dev/ttyUSB1
    desc: CP2105 Dual USB to UART Bridge Controller - Enhanced Com Port
    hwid: USB VID:PID=10C4:EA70 SER=AH046H3M120067 LOCATION=3-2.1:1.0
/dev/ttyUSB2
    desc: CP2105 Dual USB to UART Bridge Controller - Standard Com Port
    hwid: USB VID:PID=10C4:EA70 SER=AH046H3M120067 LOCATION=3-2.1:1.1
```

This first device, USB0, is my homebrew K3NG Winkeyer clone so I use USB2.0-Ser to identify this device. The other two devices are the dual USB ports on my Yaesu FTdx3000 radio.

Finally, location of the various support images and data files (including master.csv) depends on how you installed the software. If you followed the installation instructions in README.md, verbatim, these files will be in ~/Python/data. If you used a windows binary installer, these files will be in C;\ Program Files (x86)\AA2IL. In either case, you have the option to leave the My Data Dir field empty and the software will find one of these defaults. However, if you choose to install the software elsewhere, you will need to fill in this field with the location of these files.

Figure 4.2 shows the settings field with this minimal configuration information filled in. However, to avoid having to revisit this issue as you make use of more of the program features, I recommend that you fill in as much of this basic information as you can. The data you entered is written to ~/.keyerrc once you press the “OK” button; The “Cancel” button provides an escape mechanism if you change your mind and don’t want to save the data. Figure 4.3 shows my completed settings dialog. If you need to change this information in the future, you can either 1) start any of these three app using the -settings command line parameter, 2) select Settings … under the File menu in pyKeyer, or 3) directly edit ~/.keyerrc using a text editor.

4.1 qrz.py

Most competitive contesters make use of a *call history file* (CHF) to fill in exchange information that is unlikely to change often, e.g. the name and state of an op. The team behind N1MM maintains these files and most ops will use the CFH from the most-recent running of a particular contest. The approach I’ve adopted is to combine the call histories from the various contests that I regularly participate in into a “master call history file,” master.csv. Also included in this master CHF is information scraped from the published membership rosters of the CWops, SKCC, FISTS and FOC clubs. The result is data from over ? Ops and includes the overwhelming majority of ops you are likely to run into in a contest. Qrz.py is a utility to investigate what information is stored in the master CSF about one or more ops.

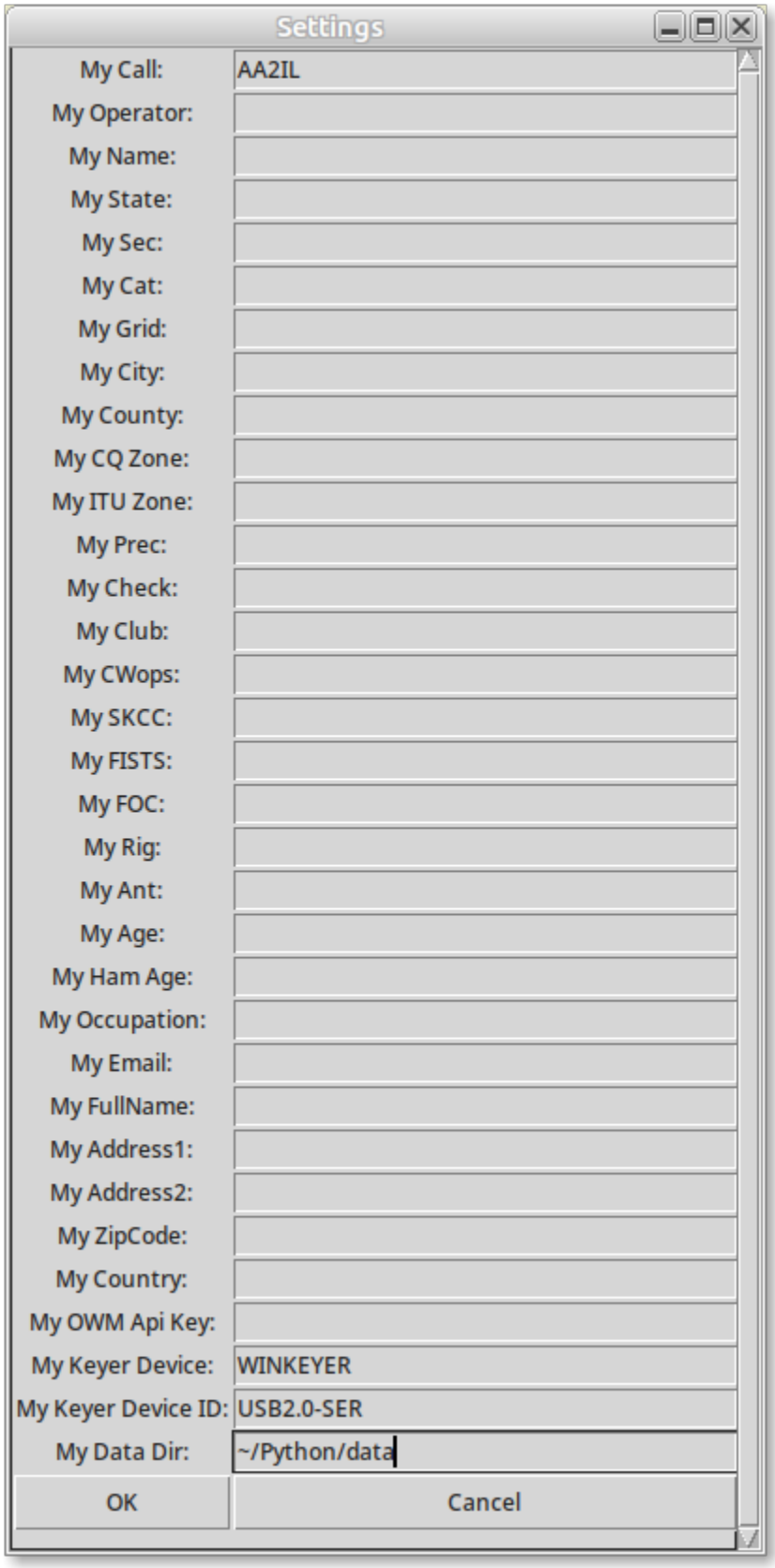
Settings
My Call:
AA2IL
My Operator:
My Name:
My State:
My Sec:
My Cat:
My Grid:
My City:
My County:
My CQ Zone:
My ITU Zone:
My Prec:
My Check:
My Club:
My CWops:
My SKCC:
My FISTS:
My FOC:
My Rig:
My Ant:
My Age:
My Ham Age:
My Occupation:
My Email:
My FullName:
My Address1:
My Address2:
My ZipCode:
My Country:
My OWM Api Key:
My Keyer Device:
WINKEYER
My Keyer Device ID:
USB2.0-SER
My Data Dir:
~/Python/data
OK
Cancel

Figure 4.2. Settings Dialog with minimal configuration information.

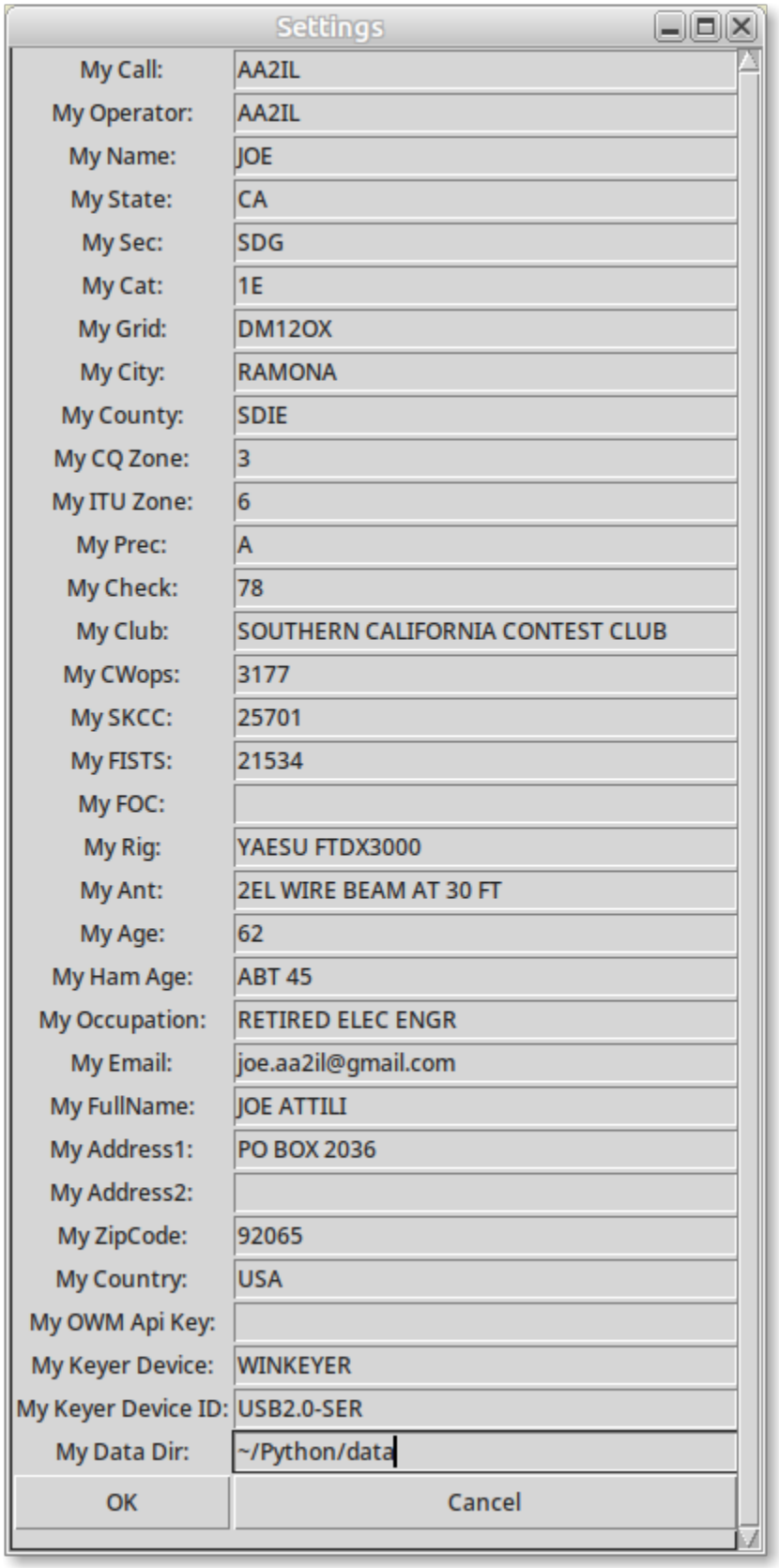

Figure 4.3. Settings Dialog with complete configuration information.

Qrz.py, as well as the other two app in this package, are meant to be “old style” command line utilities. As such, there are various options that can be specified. To see a list of this options, use the –help flag:

```
$ qrz.py --help
usage: qrz.py [-h] [-cwops] [-cw] [-settings] [call ...]

QRZ???

positional arguments:
  call        Call(s) worked

options:
  -h, --help  show this help message and exit
  -cwops      CWops Reverse Lookup
  -cw         Look only for CW QSOs
  -settings   Open setting window
```

Usually, we are interested in looking up a particular call sign, e.g.:

$ qrz.py aa2il

This results in the dialog shown in Figure 4.4. If you prefer to use a graphical user interface (GUI), don’t supply a call sign and you will be prompted to enter one or more call signs as in Figure 4.5. Multiple call signs can be separated by spaces of commas.

4.2 paddling.py

One mark of a good cw operator is a “fist” that is easy to copy and who make few mistakes. Achieving this takes dedicated practice and time on the air. I fully admit that I am very biased toward contesting and don’t do very much in the way of rag-chewing. Except for the SST and MST practice sessions, I am very reluctant to send using my paddles during a contest as I don’t want to slow down the other ops who are gracious enough to call me and/or return my call. Hence, I generally rely on my computer and keyer to do the sending for me and my fist has suffered accordingly.

There is only one program available dedicated to sending practice that I am aware of, Iambic Master. However, it program only uses the sprint format for practice messages and I wanted something that covers variety of message formats including panagrams and rag-chews. As such, I added a component to pyKeyer that generates test messages and uses feedback from the keying device to asses and tune my fist. Over time, I broke this component out as a stand-along app call paddling.py. Figure 4.6 shows graphical interface for this app.

The upper text window displays a practice message to be sent while the lower text window echos the characters from the keyer as they are sent. The green box marked “Current” indicates the distance between the message and what is actually sent. If the two match, this distance will be zero and the app advances to the next test message. The user can restart/repeat a message as often as he’d like and

visually compare the contents of the two text boxes to gain feedback on his fist. Pressing the spacebar will generate a new test message.

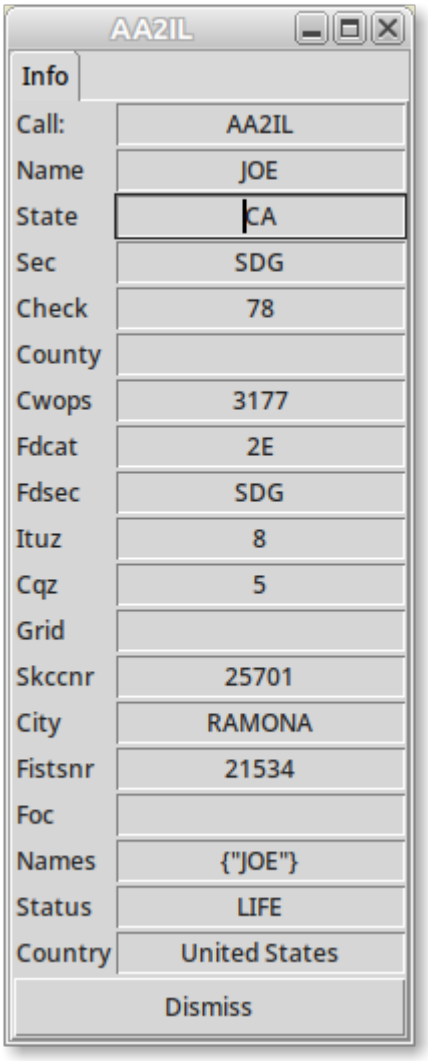

Figure 4.4. All information available in master call history file for my call.

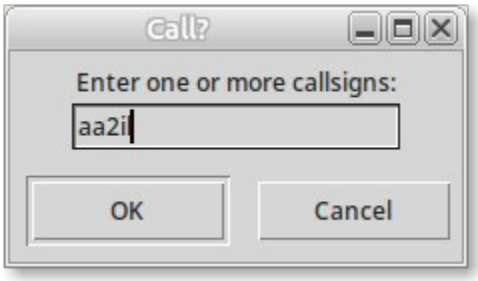

Figure 4.5. GUI call sign prompt.

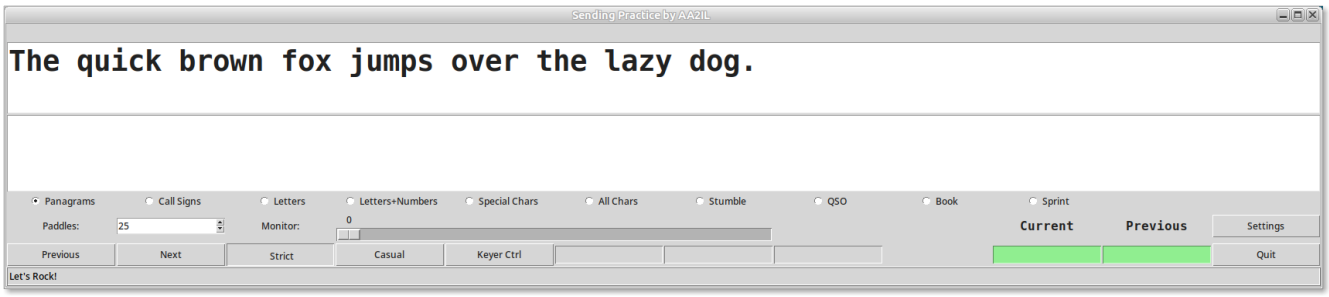

Figure 4.6. Sending Practice GUI.

The group of radial buttons below the two text boxes select the type of practice message. The "Paddles" spin-box control the sending speed (wpm). If a rig sidetone is being used (see below), the volume of the sidetone is controlled via the "Monitor" slider. The "Previous" and "Next buttons allow the user to go back to the previous or advance to the next test message respectively. The "Strict" and "Casual" buttons control how tightly the timing is critiqued while the "Keyer Ctrl" button brings up a window for controlling the keying device at a low-level. Finally, the "Settings" button brings up the settings dialog described earlier and the "Quit" end the practice session.

Here is a list of the command line arguments that are available:

```
$ paddling.py --help

usage: paddling.py [-h] [-wpm WPM]
                   [-rig
{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,KC505}
[{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,KC505} ..
.]]
                   [-keyer {WINKEY,NANO,K3NG,ANY}] [-kport KPORT] [-settings]

options:
  -h, --help            show this help message and exit
  -wpm WPM              Keyer Speed
  -rig
{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,KC505}
[{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,KC505} ..
.]
                        Connection Type
  -keyer {WINKEY,NANO,K3NG,ANY}
                        Keyer Type
  -kport KPORT          Connection Port for Keyer
  -settings             Open settings window
```

The -wpm flag sets the starting keyer speed (default is 25 wpm) and the -settings flag opens the settings dialog described earlier. -keyer and -kport can be used to specify the type of external keyer being used and the USB port it is attached to. It is preferable, however, to configure these in the setting dialog. Most users will likely be using a winkeyer.

Similarly, the -rig flag is used to configure the rig. For this particular app, the break-in function is turned off and the rig is only used for sidetone audio. There are two acceptable formats for this flag but it is best to supply two arguments. The first argument is the type of connection to establish with the rig and the second argument is the type of rig. The connection type can be one of HAMLIB, FLRIG, FLDIGI or DIRECT and specifies the "middle-ware" the manages the rig communications. The preferred protocol is HAMLIB but a DIRECT connection to the rig also works well for send practice. The KCAT option is developmental and should not be used for now.

At present, only a handful of rigs are supported – those that I either own or have owned in the past. The FTdx3000, FT991a, and IC9700 are known to currently work with this package. Adding support additional rigs should not be difficult but probably take a bit of effort to make the necessary modifications and test.

There are a few keyboard strokes that this app understands:

Space	– advance to next test message
+	- Increase keyer speed (wpm)
-	- Decrease keyer speed (wpm)

4.3 pyKeyer.py

pyKeyer.py is a full-featured contest logging program. Figure 4.7 shows the graphical interface for this app.

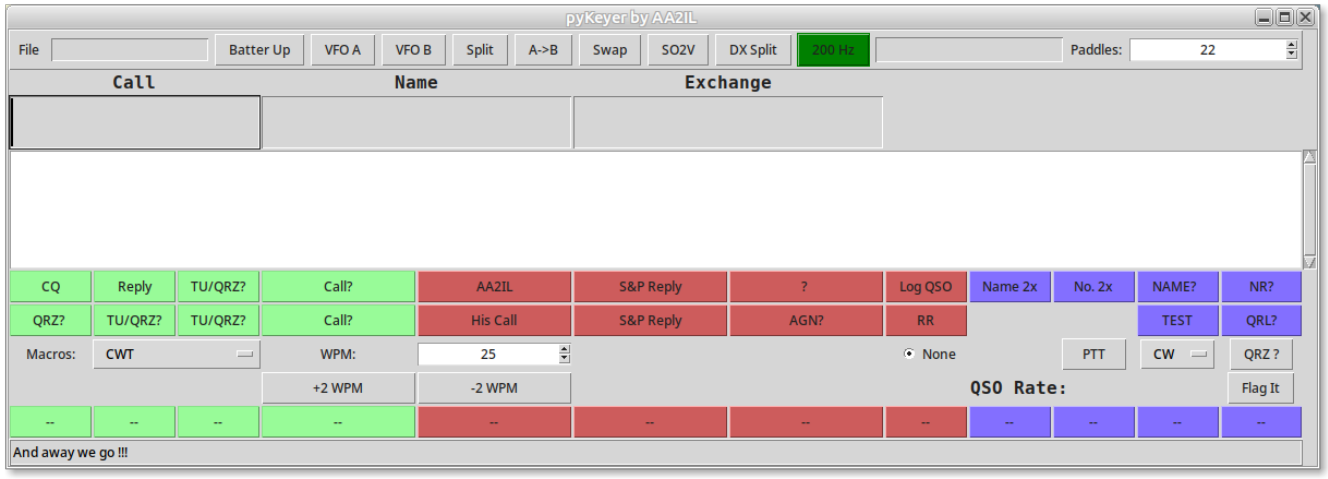

Figure 4.7. pyKeyer GUI.

4.3.1 The Top Row

In the upper left is the only pull-down menu – File. This menu provides access to various configuration and run-time options:

Settings …	- Brings up settings dialog described in Section ?
Rig Control …	- Brings up dialog for directly controlling rig options
Keyer Control …	- Brings up dialog for directly controlling keyer options
Paddling …	- Brings up sending practice window described in Section ?
Clear Stores …	- Clears stored state data from prior run (deprecated)
Set Rig CW	- Puts rig into CW mode
Capture Audio	- Toggles capture of receiver audio
Farnsworth	- Toggles Farnsworth CW timing
Tune	- Antenna tuning

Practice Mode	- Toggles Practice Mode
Auto-fill	- Toggle automatic filling of exchange fields from call history
Show Hints	- Show relevant info from call history
Show SCP	- Show Super Check Partial list of likely call signs
Auto-complete	- Toggles automatic completion of call sign if only one call in SCP list
Sidetone	- Toggles generation of sidetone through computer audio
Adjust Speed	- Toggles keyer speed adjustment in practice mode
Lock Speed	- Lock speeds of computer keying and paddles
Nano Echo	- Toggles echoing of characters from keyer
Split TX Window	- Toggles separate window for sending text
Immediate TX	- Toggles immediately transmission of typed in text
Clear All Spots	- Clears stored memory data

The next several buttons along the top row relate to advance operations involving VFO manipulations. There is some rudimentary support for Single Operator Two VFO (SO2V) and Single Operator Two Radio (SO2R) operations. The two "split" buttons are useful for chasing DX operations running split. The difference between these two is whether the split is effected using dual VFOs or via the rig's XIT/RIT functions.

The green button mark 200Hz provides rapid access to the audio bandwidth. This is particularly useful if you are attempting to pull out a weak signal. The text box next to this button is used for displaying miscellaneous information during an operation. Finally, the "Paddles" spinbox in the upper right allow the user to specify a different keyer speed for text sent via the paddles vs via the computer. This is useful if you cannot reliably control the paddles at the higher speeds typical of a contest. Unfortunately, this option is only available on my homebrew keyer.

4.3.2 Exchange Entry

The next row of entry boxes is where call sign and exchange information is entered. The boxes displayed vary depending on the particular contest. Navigation through these boxes is either via a mouse click (the hard way) or the forward and backward tab keys (preferred). If SCP or Hints are enabled, these boxes will appear here as well; data from these boxes can be selected via double clicking with the mouse.

4.3.3 Text Message Box

The text from the various message is echoed in the large box in the middle. You can also type in text to be sent by clicking on this box. There is also an option to split this box horizontally so that the sent text is kept separate. With this, it is still possible to send a message if no keyer is attached (or if you are using pyKeyer for RTTY operations).

4.3.4 Marcos

The two rows of buttons below the large text box are the keyboard macros. These are, of course, contest dependent and are broken up into three groups: Running (green), Search and Pounce (Red) and Repeats/Fills (Blue). The twelve buttons in the first row correspond to the twelve function keys and those in the second row correspond to the "shifted" function keys (e.g. CQ is send if F1 is pressed while QRZ? is sent if Shift-F1 is pressed). The buttons may also be selected via a mouse click.

4.3.5 Run-time Options

The group of GUI widgets provide easy access to various run-time options. The macros for a particular contest are selected via the pull-down “Macros” menu (CWT in the figure). The speed for the computer-generated text is controlled via the group of “WPM” widgets. In additon to the +2 WPM and -2 WPM buttons, keyer speed can be changed on the fly by pressing Page Up and Page Down on the keyboard.

Pykeyer can control up to three radios at a time. (I only use this in VHF contests which tend to be rather slow-paced in SoCal.) The group of radio buttons in the middle is used to select the active radio. “None” indicates no radio is connected. The PTT button can be used in a voice contest to effect a transmission or in a CW or RTTY contest to send a carrier (e.g. for antenna tuning). The mode pull-down allows the operator to change modes. The QRZ? button triggers a query to qrz.com for the current call in the call sign box. The area marked QSO Rate provides rate data for the last ten-minutes and hour. Finally, as there is currently no convenient way to correct a logging mistake, the “Flag It” button is provide to insert a note in the ADIF log file that the last qso needs to be reviewed. I use this in conjunction with written notes made during a contest to correct a log prior to submission.

4.3.6 Rapid Memory Store and Recalls

Near the bottom of the pyKeyer window is one or more rows of memory buttons. The idea behind these is to replicate the STORE and RECALL features available on most modern rigs. This is particular useful when Searching and Pouncing in contest when you come across a station that is busy. Rather than spending a lot of time trying to break through his pile-up, it is usually more efficient to mark his location and come back later. Left-clicking with the mouse on one of these buttons will not only store the current rig frequency but also any call sign and exchange information that has been entered. Right-clicking on a button will immediately restore that spot.

4.3.7 Status Bar

The final line of the pyKeyer GUI is a status bar where various messages are displayed. During may contests, a summary of your current score is displayed here.

4.3.8 Command Line Options

PyKeyer has many configuration options. Although most of these options are available from the GUI, this program is designed to be started from the command line. Below is a list of the available options:

```
$ pyKeyer.py  --help

usage: pyKeyer.py [-h] [-ss] [-naqp] [-cq160m] [-sst] [-mst] [-awt] [-skcc] [-cwt] [-cwopen]
[-pota] [-wpx] [-arrl_dx] [-arrl_10m] [-arrl_160m] [-cqp]
                  [-state [STATE ...]] [-aa] [-wfd] [-fd] [-vhf] [-mak] [-stew] [-jidx] [-
yuri] [-spdx] [-ocdx] [-marconi] [-solar] [-cqmm] [-holy] [-iota] [-marac]
                  [-sat] [-sprint] [-wrt] [-cqww] [-cqvhf] [-iaru] [-ten] [-foc] [-wag] [-sac]
[-beru] [-rac] [-ragchew] [-dx] [-calls] [-default] [-sending]
                  [-autofill] [-autocomplete] [-test] [-practice] [-immediate] [-sidetone] [-
split] [-hints] [-capture] [-aggressive] [-force] [-ca_only] [-wpm WPM]
                  [-farnsworth FARNSWORTH] [-paddles PADDLES] [-adjust] [-scp]
```

```
                [-rig
{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,K
C505}
[{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,
KC505} ...]]
                [-port PORT]
                [-rig2
{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,K
C505}
[{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,
KC505} ...]]
                [-port2 PORT2]
                [-rig3
{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,K
C505}
[{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,
KC505} ...]]
                [-port3 PORT3] [-max_age MAX_AGE] [-nrows NROWS] [-keyer
{NONE,NANO,K3NG,WINKEY,ANY}] [-kport KPORT] [-nano] [-k3ng] [-winkeyer] [-noecho] [-cwio]
                [-lock] [-mode {None,CW,SSB,RTTY}] [-log LOG] [-rotor {HAMLIB,NONE}] [-port9
PORT9] [-server] [-udp] [-shadow] [-gps] [-use_log_hist]
                [-use_adif_hist] [-digi] [-wfonly] [-geo GEO] [-desktop DESKTOP] [-settings]
[-special]

options:
  -h, --help            show this help message and exit
  -ss                   ARRL Sweepstakes
  -naqp                 NAQP
  -cq160m               CQ 160m CW
  -sst                  K1USN SST
  -mst                  ICWC MST
  -awt                  Japan AW Test
  -skcc                 SKCC
  -cwt                  CWops Mini Tests
  -cwopen               CWops CW Open
  -pota                 POTA
  -wpx                  CQ WPX (CW or RTTY)
  -arrl_dx              ARRL Intl DX
  -arrl_10m             ARRL 10m
  -arrl_160m            ARRL 160m
  -cqp                  California QP
  -state [STATE ...]    State QP
  -aa                   All Asia DX
  -wfd                  Winter Field Day
  -fd                   ARRL Field Day
  -vhf                  ARRL VHF
  -mak                  Makrothen RTTY
  -stew                 Stew Parry
  -jidx                 JIDX CW
  -yuri                 Yuri Gagarin DX
  -spdx                 SP DX CW
```

```
  -ocdx                 OC DX CW
  -marconi              Marconi Memorial
  -solar                Solar Eclipse
  -cqmm                 CQMM DX
  -holy                 Holyland DX
  -iota                 RSGB IOTA
  -marac                MARAC US Counties
  -sat                  Satellites
  -sprint               NCCC CW Sprint
  -wrt                  Weekly RTTY Test
  -cqww                 CQ Worldwide
  -cqvhf                CQ VHF
  -iaru                 IARU HF Championship
  -ten                  10-10 CW
  -foc                  FOC BW
  -wag                  Work All Germany
  -sac                  Scandinavia Activity
  -beru                 RSGB Commonwealth
  -rac                  RAC Summer or Winter QPs
  -ragchew              Regular Ragchew QSO
  -dx                   DX QSO
  -calls                Random Calls
  -default              Default (quick) QSO
  -sending              Sending Practice
  -autofill             Auto fill known information
  -autocomplete         Auto Complete Partial Callsigns
  -test                 Disable TX
  -practice             Practice mode
  -immediate            Send text immediately
  -sidetone             Sidetone Osc
  -split                Split Text Window
  -hints                Show hints
  -capture              Record Rig Audio
  -aggressive           Aggressively keep main window on top
  -force                Force rig connection (debugging)
  -ca_only              Only use California Stations for Practice
  -wpm WPM              Keyer Speed
  -farnsworth FARNSWORTH
                        Farnsworth Speed
  -paddles PADDLES      Paddle Speed
  -adjust               Adjust speed based on correct copy
  -scp                  Enable Super Check partial
  -rig
{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,K
C505}
[{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,
KC505} ...]
                        Connection Type - 1st rig
  -port PORT            Connection Port - 1st rig
  -rig2
{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,K
```

```
C505}
[{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,
KC505} ...]
                        Connection Type - 2nd rig
  -port2 PORT2          Connection Port - 2nd rig
  -rig3
{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,K
C505}
[{FLDIGI,FLRIG,DIRECT,HAMLIB,KCAT,ANY,NONE,FTdx3000,FT991a,IC9700,IC7300,TS850,IC706,TYT9000d,
KC505} ...]
                        Connection Type - 3rd rig
  -port3 PORT3          Connection Port - 3rd rig
  -max_age MAX_AGE      Max age in hours
  -nrows NROWS          No. STO/RCL rows
  -keyer {NONE,NANO,K3NG,WINKEY,ANY}
                        Keyer Type
  -kport KPORT          Connection Port for Keyer
  -nano                 Use Nano IO Interface
  -k3ng                 Use K3NG IO Interface
  -winkeyer             Use Winkeyer IO Interface
  -noecho               Don't Echo response from Nano IO to text box
  -cwio                 Use FLRIG or HAMLIB for CW IO
  -lock                 Lock Paddle Speed to Computer Speed
  -mode {None,CW,SSB,RTTY}
                        Rig Mode
  -log LOG              Log file name
  -rotor {HAMLIB,NONE}  Rotor connection Type
  -port9 PORT9          Rotor connection Port
  -server               Start hamlib server
  -udp                  Start UDP server
  -shadow               Shadow helper
  -gps                  Read GPS info from .gpsrc file
  -use_log_hist         Use history from log
  -use_adif_hist        Use history from adif log
  -digi                 RTTY or other Digi Contest
  -wfonly               FLDIGI is in Waterfall Only mode
  -geo GEO              Geometry
  -desktop DESKTOP      Desk Top Work Space No.
  -settings             Open setting window
  -special              Special settings for VHF work
```

At first glance, this seems like an over-whelming list. However, the majority of the options select a single individual contest, e.g. -ss selects the ARRL sweepstakes (November each year), -cwt selects the weekly CWops mini-tests (four times every Wednesday), etc.. In addition to competitive contests, you can also select -pota for POTA/SOTA ops, -dx for chasing hit-and-run DX, and -ragchew for conversational QSOs.

Configuring your rig and keyer is similar to what was described in Section ? regarding paddling.py.

Add to this!!!!!!!!!!

4.3.9 Practice Mode

PyKeyer was originally developed for practicing contesting and I still often use it as a warm-up prior to a major contest. It recommended that a new user try this feature as a first step in becoming familiar with this program.

If the -rig flag is not supplied, the program automatically enters “practice mode.” In addition, if none of the keyer-related flags is supplied, sidetone audio is automatically generated through the computer. Hence, to start a practice session, the user simply needs to select a contest, e.g.

pyKeyer.py -cwt

(This was the command used to create figure ?.)

Once the GUI appears, the practice session is initiated by pressing F1 (Call CQ). The computer will respond with a call sign randomly selected from the master CHF. The op the types in the call and presses F2 to send his exchange. If the call was correct, the computer will respond with his exchange. If there was a mistake in the call sign, the call will be repeated prior to sending the exchange. The op then records the exchange and responds by pressing F3 to indicate the end of the QSO. If any repeats/fill are required, you can send a “?” or press the appropriate Blue Function button. If there was an error in the recorded exchange, this will be noted in the large text box. An short CWT practice session illustrating this is shown in Figure 4.8

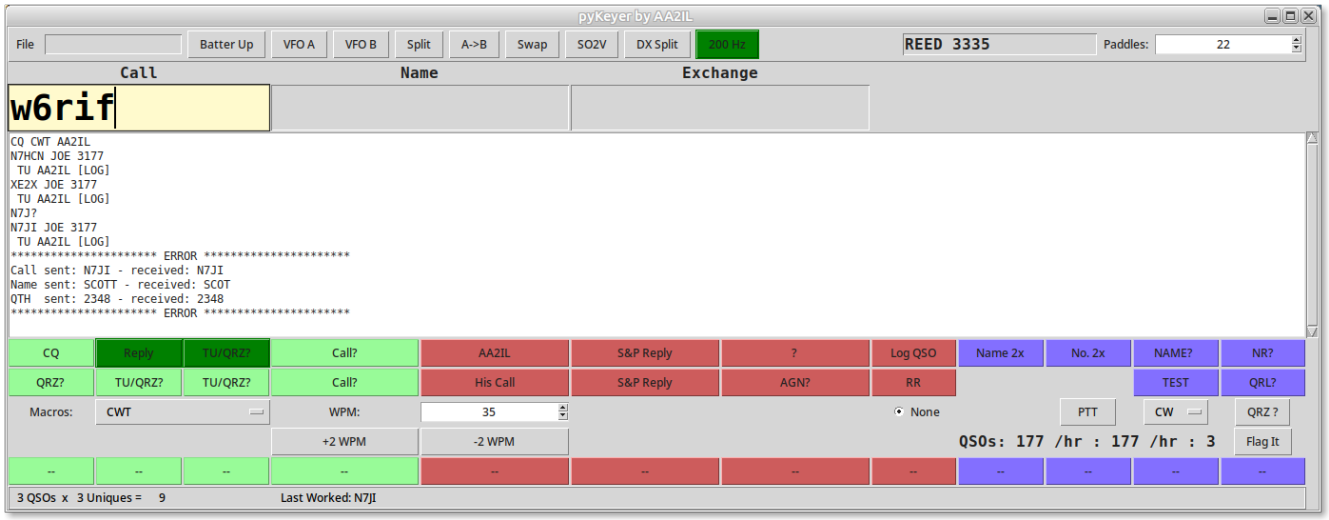

Figure 4.8. A short CWT practice session.

4.3.10 RTYY and Voice Modes

PyKeyer can be used as the logger for RTTY and voice contests as well.

Discuss pairing with fldigi decoder.