Dire Wolf
User Guide

Decoded
Information from
Radio
Emissions for

Windows
Or
Linux
Fans

Version 1.0 – May 2014

Contents

1 Introduction

Dire Wolf is a software modem and APRS* encoder/decoder. It can be used stand-alone to receive APRS messages, as a digital repeater ("digipeater"), and Internet Gateway (IGate). It can also be used as a virtual TNC for other applications such as APRSIS32, UI-View32, Xastir, APRS-TW, YAAC, SARTrack, and many others. Both KISS and AGWPE network protocols are supported for use by applications.

Software and documentation can be downloaded from http://home.comcast.net/~wb2osz/site/

First time users might want to begin with the **Quick Start Guide** and come back here later to find more details.

2 Features

- Software replacement for hardware based Packet TNC.
- 300, 1200, and 9600 baud data rates.
- Compatible with Software defined radios such as gqrx and rtl_fm.
- Operation with one or two radios.
- APRStt gateway using latitude/longitude or UTM coordinates.
- Internet Gateway (IGate) with IPv6 support.
- Multiple decoders per channel to tolerate HF SSB mistuning.
- Interface with many popular applications by
    - AGW network protocol
    - KISS serial port.
    - KISS network protocol
- Decoding of received information for troubleshooting.
- Beaconing of periodic Position and Object reports.
- Very flexible Digipeating including selective routing between channels.
- Separate raw packet decoder: decode_aprs
- Support for UTF-8 character set.
- Runs in two different environments:
    - Microsoft Windows XP or later. Pentium 3 or equivalent or later.
    - Linux, regular PC or embedded systems such as Raspberry Pi

3 Connection to Radio

For receiving all you need to do is connect your receiver speaker to the “Line In” or microphone jack on your computer.

If you are using a laptop, with a built-in microphone, you could probably just set it near your radio’s speaker in a quiet setting.

If you want to transmit, you will need to get audio from the computer to the microphone input of your transceiver. If you have a serial port (either builtin or a USB to RS232 adapter cable), the RTS or DTR line can be used to activate the transmitter. GPIO pins can be used on suitable Linux systems. Otherwise you will need a VOX circuit.

Others have documented this extensively so I won’t duplicate the effort. Many homebrew plans and commercial products are available. A few random examples:

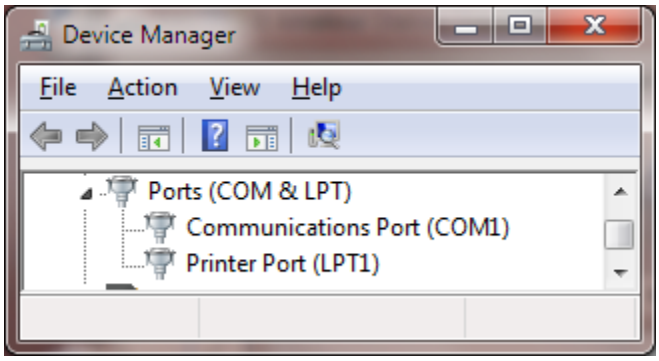
- http://www.qsl.net/wm2u/interface.html
- http://zs1i.blogspot.com/2010/02/zs1i-soundcard-interface-ii-project.html
- http://www.kb3kai.com/tnc/soft-tnc.pdf

Google for something like ham radio sound card interface or sound card packet to find others.

3.1 Don’t have a serial port?

Maybe you do but don’t know about it.

My new computer didn’t have a serial port on the back. This was a disappointment because I still have some useful gadgets that use a good old fashioned RS-232 port. I was surprised to see a serial port and parallel printer port displayed in the Device Manager:



The connectors exist on the motherboard. It was only necessary to add appropriate cables to bring them out to the rear panel. You can also buy PCI cards with serial ports or use an adapter cable with USB on one end and RS-232 on the other end.

4 Installation & Operation – Microsoft Windows XP or later

If using Linux, skip section 4 and proceed to section 5.

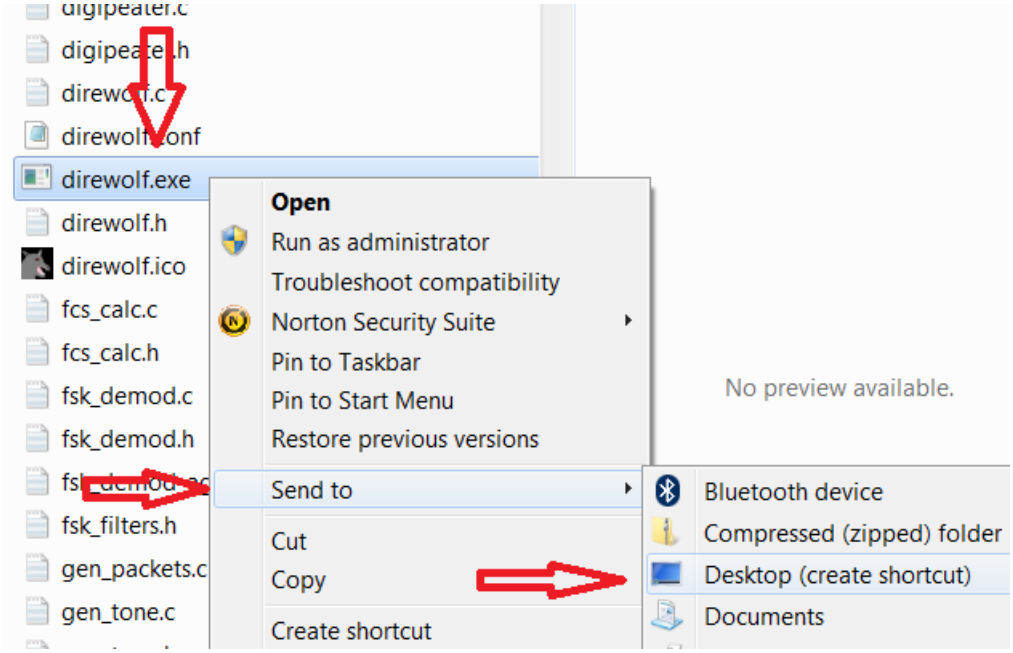
Obtain the Windows version from the Downloads section of http://home.comcast.net/~wb2osz/site/

A Pentium 3 processor or equivalent or later is required for the prebuilt version. If you want to use a computer from the previous century, see instructions in Makefile.win.

Put the Dire Wolf distribution file, direwolf-0.9-win.zip (or similar name depending on version), in some convenient location such as your user directory. In this example, we will use C:\Users\John
In Windows Explorer, right click on this file and pick "Extract All..." Click on the Extract button.
You should end up with a new folder containing:

- direwolf.exe -- The application.
- decode_aprs.exe -- APRS raw data decoder.
- Quick-Start-Guide-Windows.pdf -- Quick start guide for new users.
- User-Guide.pdf -- This document.
- and a few others ...

In Windows Explorer, right click on **direwolf.exe** and pick **Send To > Desktop** (create shortcut).

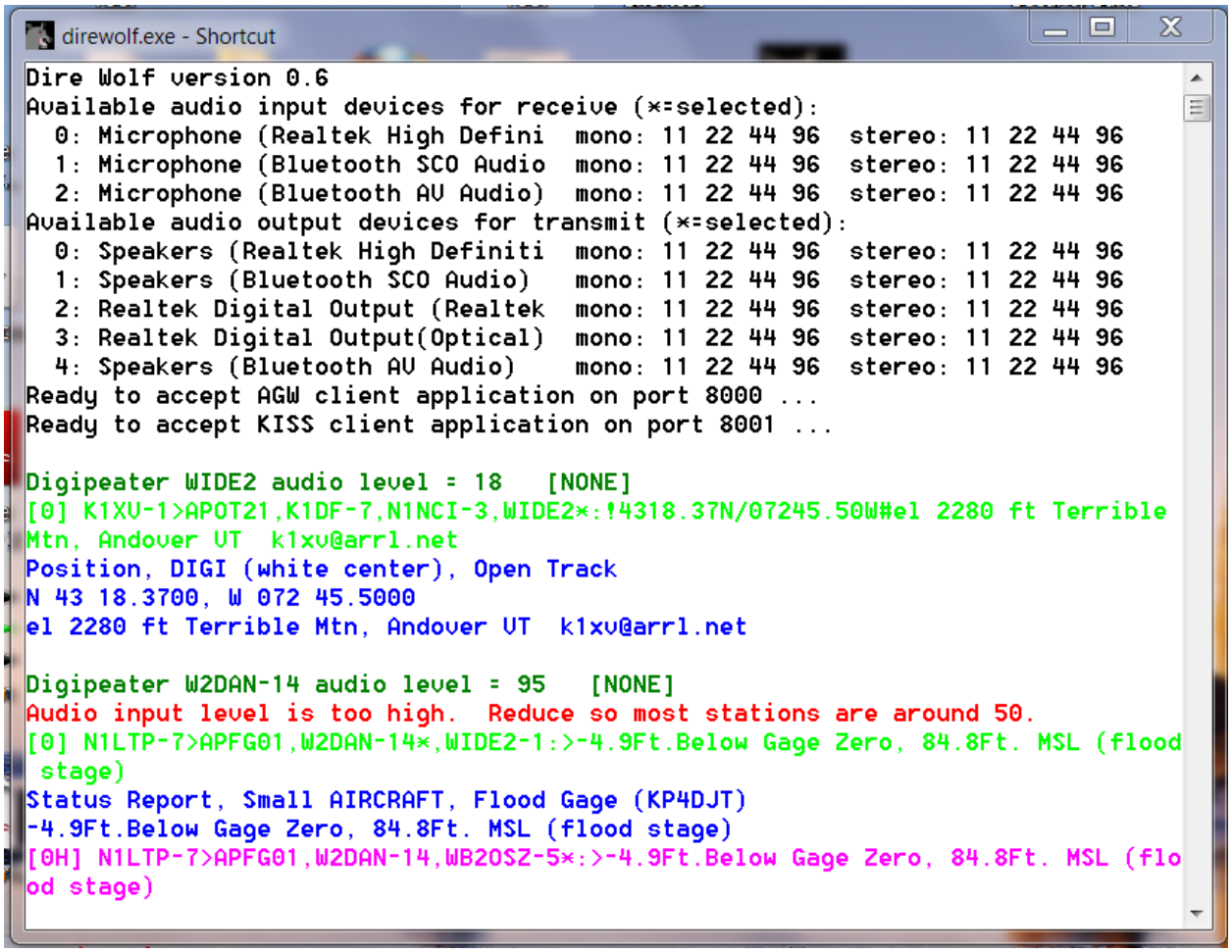


Look for the new direwolf.exe icon on your desktop.

4.1 Run Dire Wolf

Double click on the desktop icon: 

and you should get a new window similar to this:



It starts with some informational messages in black.

A group of several lines is displayed for each packet received.

The first line of each group, in dark green, contains the audio level of the station heard.

The raw data is displayed in green and deciphered information is in blue.

Transmitted packets are in magenta. In the example above, we see that Dire Wolf is being used as a digipeater.

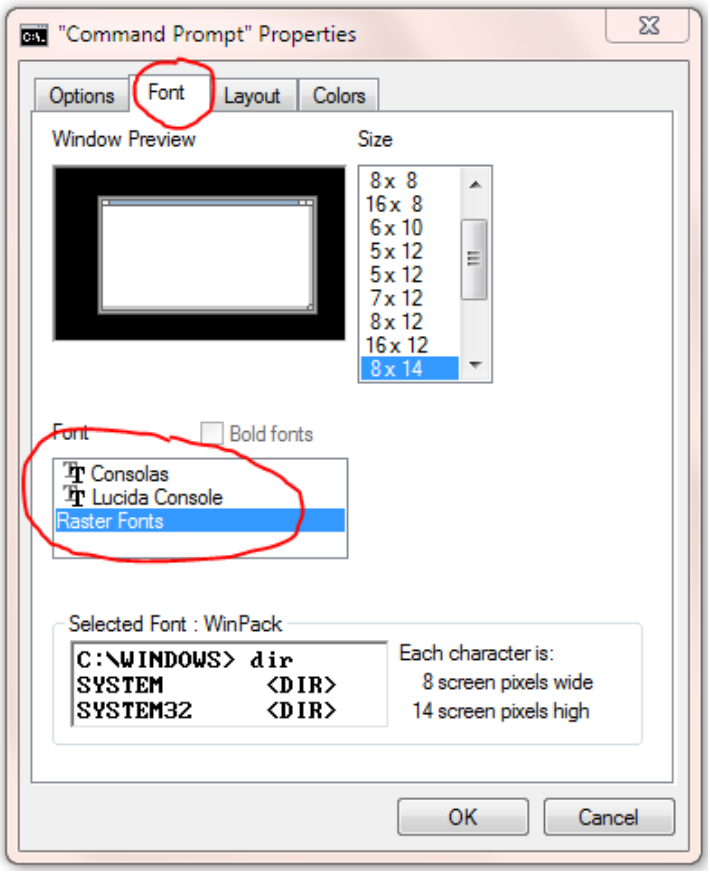
Sometimes you will see error messages in red when invalid data is received or other problems are noticed.

The rest of section 4 describes how to use Dire Wolf with other packet radio applications such as APRSIS32 and UI-View. If you are not interested using them this time, skip ahead to section 7, Basic Operation.

When using the network interfaces, Dire Wolf and the client application can be running on different computers. You could have a Linux computer in the "shack" running Dire Wolf as a digipeater. You could connect to it from a Windows Laptop, running APRSIS 32, in another part of the house. In this case you would specify the name or address of the first computer instead of using "localhost."

4.2 Select better font

You might need to change the font for best results. Right-click on the title bar and pick Properties from the pop-up menu. Select the Font tab. Notice the list of fonts available. The one called "Raster Fonts" has a very limited set of characters. **Choose one of the others.** For more details, see section called **UTF-8 Characters**.



4.3 AGW TCPIP socket interface

Dire Wolf provides a server function with the "AGW TCPIP Socket Interface" on default port 8000.

4.3.1 APRSIS32

1. First, start up Dire Wolf.
2. Run APRSIS32.
3. From the "Configure" menu, pick "ports" then "new port…"
4. Select type "AGW" from the list. Enter "Dire Wolf" as the name. Click "Create" button.
5. When it asks, "Configure as TCP/IP Port?" answer Yes.
6. Enter "localhost" for the address and port 8000.
7. Finally click on "Accept.".

4.3.2 Ui-View

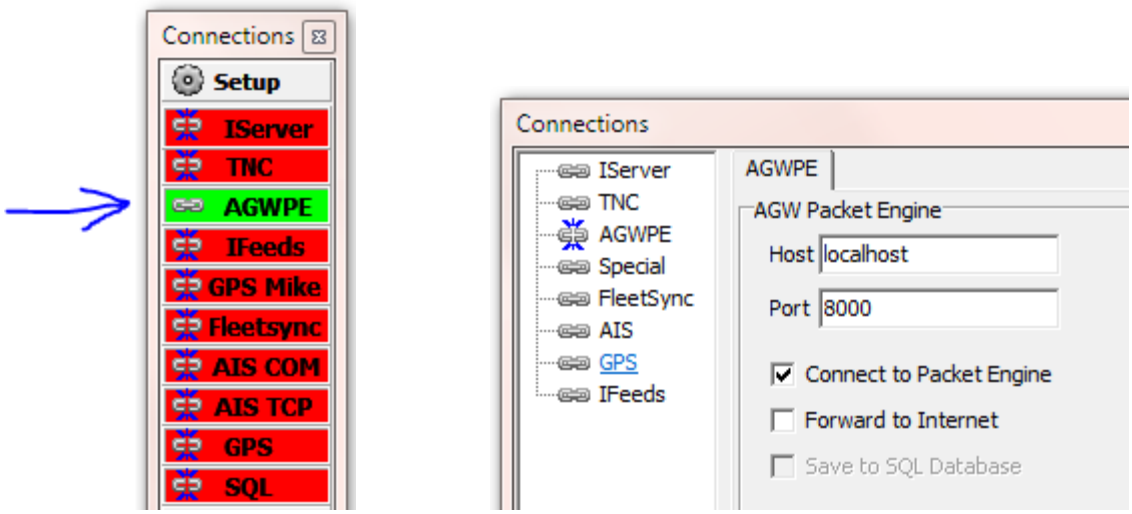
1. First, start up Dire Wolf.
2. Run UI-View32
3. From the Setup menu, pick Comms Setup.
4. Select Host mode: AGWPE from the list and click the "Setup" button.
5. Take defaults of localhost and 8000. Click on OK.
6. Click on OK for Comms Setup.

4.3.3 YAAC (Yet Another APRS Client)

1. First, start up Dire Wolf.
2. Run YAAC
3. From the Setup menu, pick Configure → by Expert Mode.
4. Select the "Ports" tab.
5. Click the "Add" button.
6. From the Port type list, choose AGWPE.
7. For Server Host name specify where Dire Wolf is running. Use "localhost" if both are running on the same computer.
8. For the port name list, you should see one or two items depending how Dire Wolf was configured.

4.3.4 SARTrack

1. First, start up Dire Wolf.
2. Run SARTrack.
3. Select AGWPE under Connections.
4. If SARTrack and Dire Wolf are running on different computers, enter the address of the host where Dire Wolf is running.



4.4 Kiss TNC emulation – serial port

Dire Wolf can act like a packet radio TNC using the KISS protocol by serial port.

To use this feature, you must install com0com as explained later in the Advanced Topics section. If you followed the instructions, other applications will think they are talking with a TNC on the COM4 serial port.

Here are detailed configuration steps for a couple popular applications.

4.4.1 APRSIS32

1. First start up Dire Wolf.
2. Run APRSIS32.
3. From the "Configure" menu, pick "ports" then "new port…"
4. Select type "KISS" from the list. Enter "Dire Wolf" as the name. Click "Create" button.
5. When it asks, "Configure as TCP/IP Port?" answer No.
6. For port configuration, pick "COM4" from the list. If you don't see COM4, com0com has not been installed properly. Go back and fix it.
7. The baud rate shouldn't matter because there is not a physical serial port. Leaving it black seems to be fine. Keep defaults of Party:None, Data:8, and Stop:1
8. Finally click on "Accept.".

4.4.2 UI-View32

1. First, start up Dire Wolf.
2. Run UI-View32
3. From the Setup menu, pick Comms Setup.
4. Select Host mode: KISS from the list, then COM port 4, and click the "Setup" button.

5. Clear all of the “Into KISS” and “Exit KISS” fields then click the OK button.
6. Click on OK for Comms Setup.

4.4.3 YAAC (Yet Another APRS Client)

1. First, start up Dire Wolf.
2. Run YAAC
3. From the Setup menu, pick Configure → by Expert Mode.
4. Select the “Ports” tab.
5. Click the “Add” button.
6. From the Port type list, choose Serial_TNC
7. For device name pick COM4.
8. Baud Rate doesn’t apply in this case because there is no physical serial port.
9. For Command to enter KISS mode, pick KISS-only.

4.5 Kiss TNC emulation – network

Dire Wolf can also use the KISS protocol over a network connection with default port 8001.

Here are detailed configuration steps for a popular application.

4.5.1 APRSIS32

1. First start up Dire Wolf.
2. Run APRSIS32.
3. From the “Configure” menu, pick “ports” then “new port…”
4. Select type “KISS” from the list. Enter “Dire Wolf” as the name. Click “Create” button.
5. When it asks, “Configure as TCP/IP Port?” answer Yes.
6. Enter “localhost” for the address and port 8001.
7. Finally click on “Accept.”.

Skip section 5 (Linux) and proceed to section 6.

5 Installation & Operation – Linux

This is distributed as open source so you can see how it works and make your own modifications. You will need the usual development tools such as **gcc** and **make**. Both OSS and ALSA sound systems are supported. ALSA is the default. If your computer has /dev/dsp, and you want to use OSS instead, look inside Makefile.linux and make the minor change described in the comments.

Special considerations for the Raspberry Pi are covered in a separate document.

I when using Ubuntu 10.10 & 11.04, I found that /usr/include/alsa was not present with the default configuration. It was necessary to install an additional package with this command:

```
sudo apt-get install libasound2-dev
```

Failure to install the libasound2-dev package will result in the compile error, "audio.c…: fatal error: **alsa/asoundlib.h: No such file or directory**."

Download the distribution file to your home directory. Build with the following commands in a bash shell.

```
unzip direwolf-1.0-src.zip
cd direwolf-1.0
make -f Makefile.linux
make -f Makefile.linux install
```

The final command is optional. It copies files to the following locations:

/usr/local/bin/direwolf	The application.
/usr/local/bin/decode_aprs	Utility to interpret "raw" data you might find on http://aprs.fi or http://findu.com
/usr/local/bin/text2tt tt2text ll2utm utm2ll aclients	Various other utility applications.
/usr/share/applications/direwolf.desktop	Application definition with icon, command to execute, etc.
/home/pi/Desktop/direwolf.desktop	Symbolic link to above. This causes an icon to be displayed on the desktop.
/usr/share/direwolf/tocalls.txt	Mapping from destination address to system type. Search order for tocalls.txt is first the current working directory and then /usr/share/direwolf. You might want to get a newer copy from: http://www.aprs.org/aprs11/tocalls.txt
/usr/local/share/direwolf/symbolsX.txt /usr/local/share/direwolf/symbols-new.txt	Descriptions and codes for APRS symbols.
/usr/share/direwolf/dw-icon.png	Icon for the desktop.
/usr/local/share/doc/direwolf/*	Various documentation.
/home/pi/direwolf.conf	Configuration file.

	Search order is current working directory then the user's home directory.
/home/pi/dw-start.sh	Script to start Dire Wolf if it is not running already.

Some of these files might not be suitable for your system depending on the type of desktop environment.

5.1 Select UTF-8 character set

For best results, you will want to be using the UTF-8 character set. Verify this by examining the LANG environment variable.

```
$ echo $LANG
```

Make sure that it ends with ".utf8" like these examples:

af_ZA.utf8
en_GB.utf8
fr_CH.utf8

See section called **UTF-8 Characters** for more details.

5.2 Run Dire Wolf

Run "direwolf" from the command line.

The rest of this section describes how to use Dire Wolf with other Linux packet radio applications such as Xastir. If you are not interested in setting it up at this time, skip ahead to section 7, Basic Operation.

5.3 AGW TCPIP socket interface

Dire Wolf provides a server function with the "AGW TCPIP Socket Interface" on default port 8000.

5.3.1 Xastir

1. Run "direwolf" from a bash shell window.
2. Run Xastir from another window.
3. From the "Interface" menu, pick "Interface Control."
4. Click the "Add" button.
5. From the "Choose Interface Type" list, pick "Networked AGWPE" and click "Add" button.
6. Take all the default values and click on "OK" button.

7. You should now be back to the “Interface Control” dialog box. Select the device mentioning “Networked AGWPE” and click the “Start” button. The device status should now be “UP.”
8. Click the “Close” button.
9. Watch all the stations appear on the map.

You might notice that the “Configure AGWPE” option for “Digipeat?” is grayed out. This is because the protocol does not have the ability to set the “has been repeated” bits in the “via” fields of the AX.25 protocol. You can overcome this restriction by using the KISS TNC interface.

5.4 Kiss TNC emulation – serial port

Dire Wolf can act like a packet radio TNC speaking the KISS protocol over a pseudo terminal.

5.4.1 Xastir

1. Run “direwolf -p” from a bash shell window.
2. Run Xastir from another window.
3. From the “Interface” menu, pick “Interface Control.”
4. Click the “Add” button.
5. From the “Choose Interface Type” list, pick “Serial KISS TNC” and click “Add” button.
6. For TNC Port, enter “/tmp/kisstnc”. Take all the other default values and click on “OK” button.
7. You should now be back to the “Interface Control” dialog box. Select the device mentioning “Serial KISS TNC” and click the “Start” button. The device status should now be “UP.”
8. Click the “Close” button.
9. Watch stations appear on the map.

You are probably wondering: What is /tmp/kisstnc? A pseudo terminal is used to simulate a TNC connected to a physical serial port. Unfortunately, you can’t specify the device name and it can be different each time such as /dev/pts/2 one time, /dev/pts/3 another time. It would be annoying to reconfigure the Xastir port name each time. Dire Wolf creates a symbolic link to the actual device name so the Xastir configuration can be set once.

5.4.2 Linux AX25

Dire Wolf can be used with Linux AX25 instead of a physical TNC. First install ax25-tools. On Ubuntu or Raspbian, it might be as simple as:

```
sudo apt-get update
sudo apt-get install ax25-tools
```

Add a port description to /etc/ax25/axports, as described in the AS25 HOWTO documentation. For example,

```
radio WB2OSZ-15 1200 255 2 comment
```

Start up Dire Wolf with the "-p" option to make the KISS pseudo terminal interface available.

```
direwolf -p
```

You should see a message something like this:

```
Virtual KISS TNC is available on /dev/pts/5
WARNING - Dire Wolf will hang eventually if nothing is reading from it.
Created symlink /tmp/kisstnc -> /dev/pts/5
```

Leave that command window alone and open a new one. These are some sample commands for a quick test. Your situation will vary. kissattach command needs to be run as root:

```
sudo /usr/sbin/kissattach /dev/pts/5 radio 44.56.4.118
sudo route add -net 44.0.0.0/8 ax0
ping 44.56.4.120
```

You should see it transmitting something.

If difficulties are encountered, try using the "-d k" option to display the KISS protocol messages. You might see something like this for a ping command to one of the 44.x.x.x addresses:

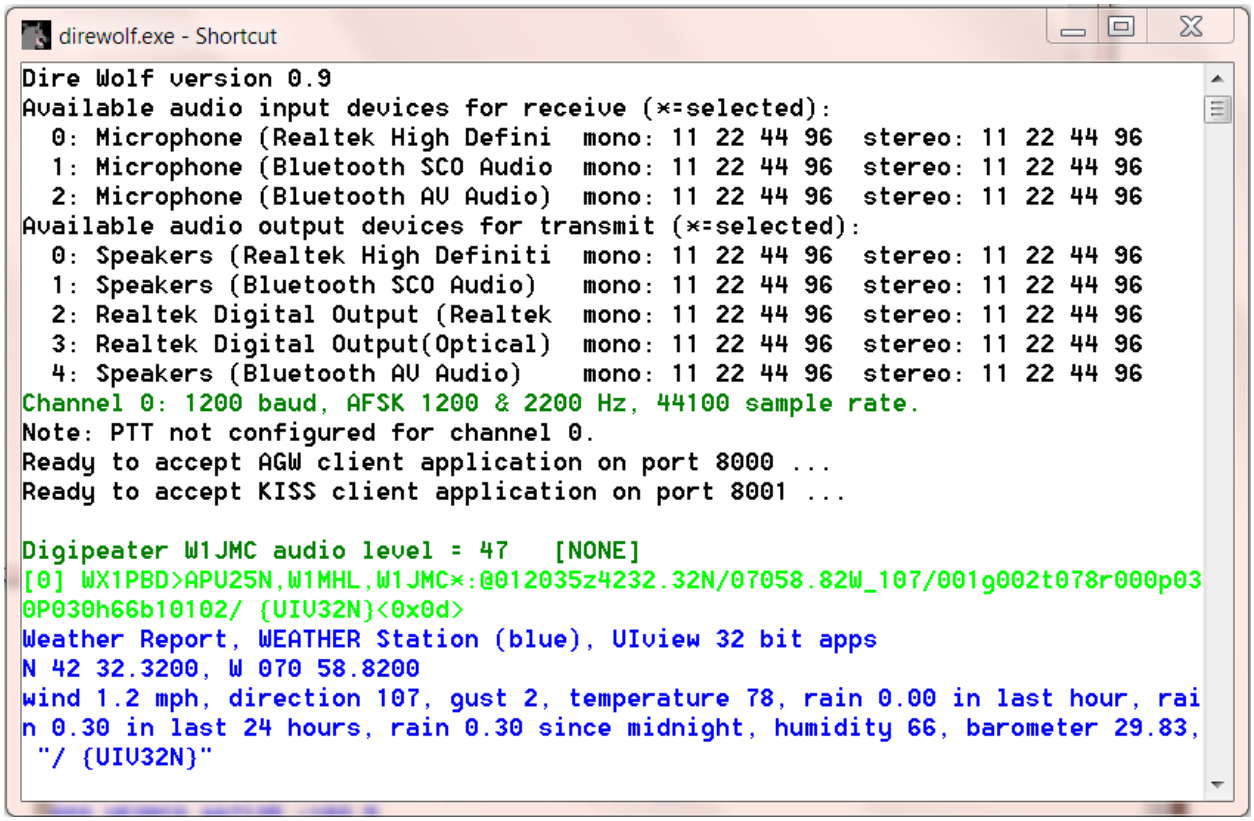
```
<<< Data frame from KISS client application, port 0, total length = 47
  000:  00 a2 a6 a8 40 40 40 60 ae 84 64 9e a6 b4 7f 03  ....@@@`..d.....
  010:  cd 00 03 00 cc 07 04 00 01 ae 84 64 9e a6 b4 1e  ...........d....
  020:  2c 38 04 76 00 00 00 00 00 00 00 2c 38 04 78     ,8.v.......,8.x
```

6 Basic Operation

Dire Wolf is not an interactive application. It has no graphical user interface. It is meant to be a replacement for a physical TNC used by other applications. It has a dumb terminal output so you can watch what is going on for troubleshooting.

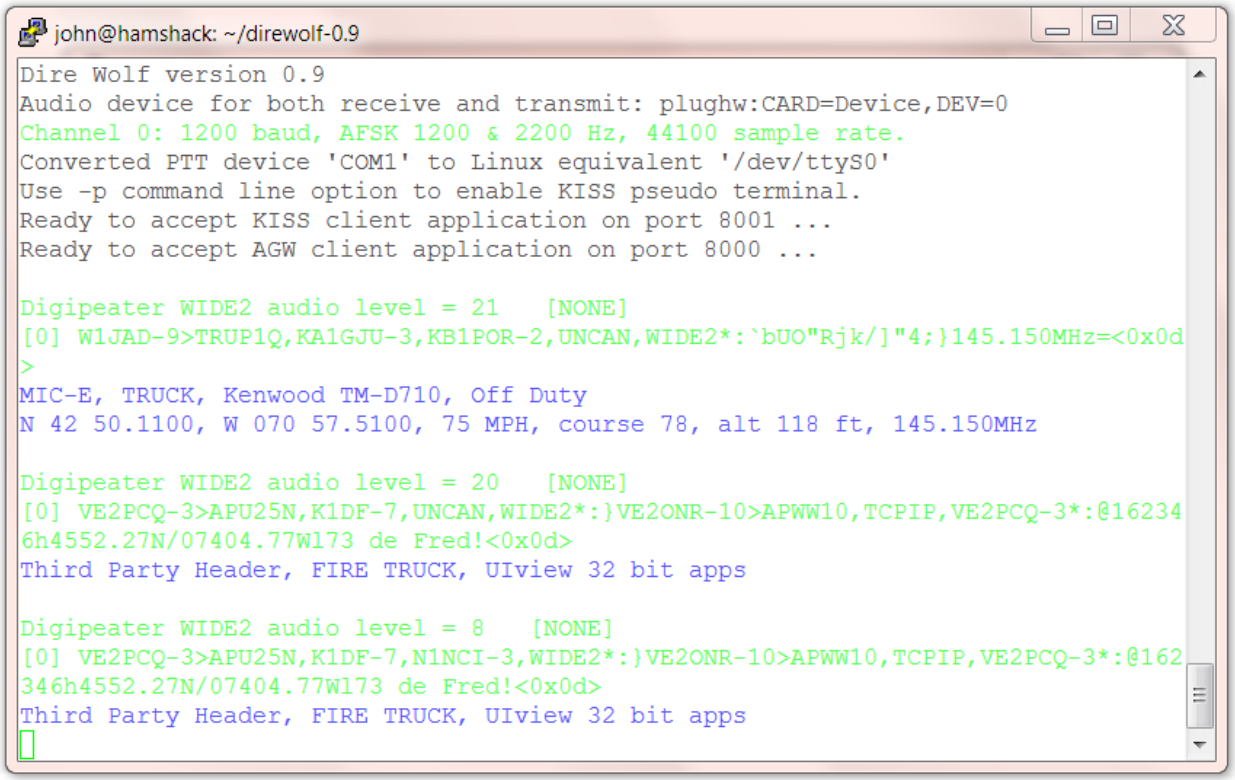
The exact appearance will vary depending on the version you are using.

You should see something like this for the Windows version:



It starts off listing the available audio devices. In this case, they are all part of the motherboard. A device, other than the default, can be specified in the configuration file. Details are in a later section.

You should see something like this for the Linux version:



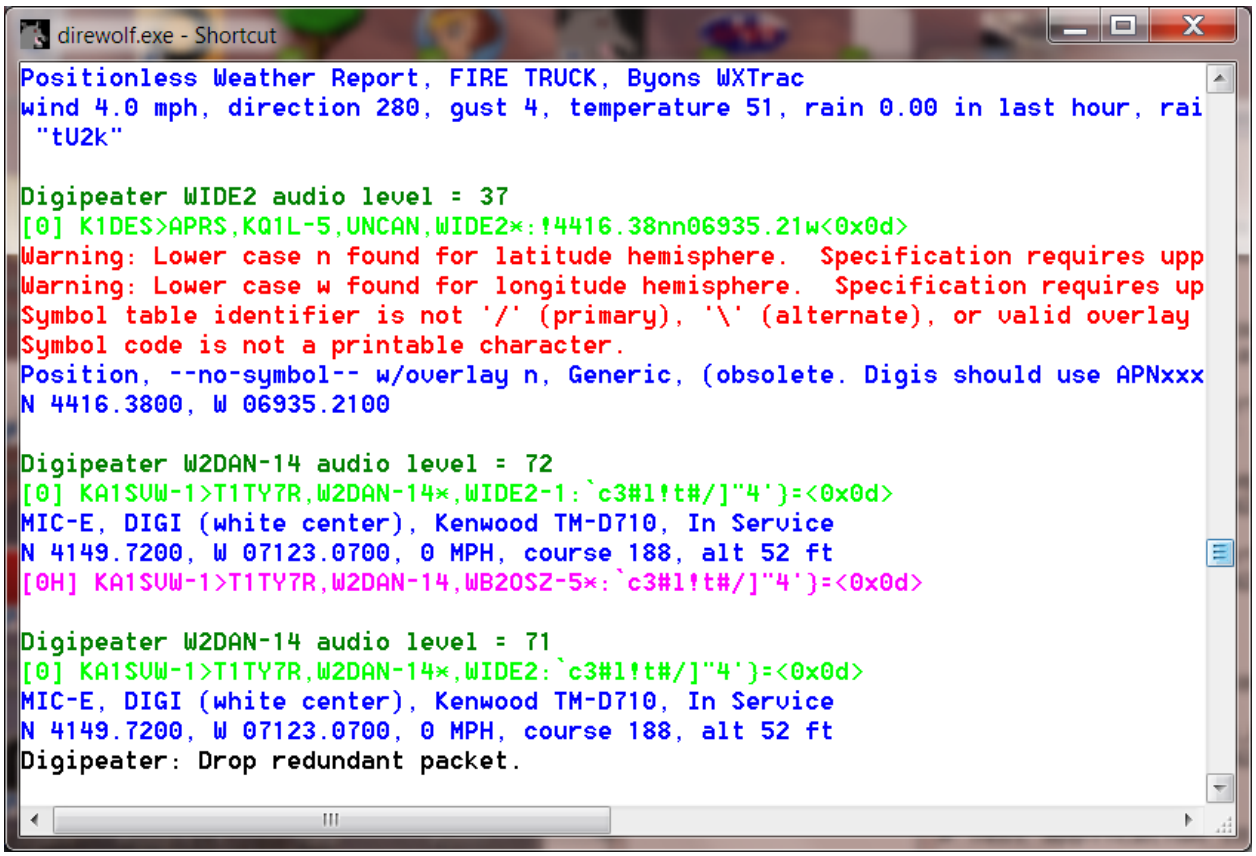
It starts with:

- The version number.
- Audio device being used.
- Modem configuration.
- A reminder that serial port KISS is off by default.
- Port numbers for use by client applications.

Different types of information are color coded:

- **Black** for information.
- **Dark Green** for the audio level. More about this below.
- **Green** for received data.
- **Blue** for a decoded version of the raw data.
    - The first line contains:
        - the message type (e.g. MIC-E, Position, or Weather)
        - symbol to be displayed (e.g. Truck, House)
        - equipment model or software application
        - MIC-E status (In Service, En Route, Off Duty, …)
        - transmitter power, antenna height, gain, and direction.
    - The second line contains:
        - Latitude & longitude, speed, course (direction in degrees), altitude
    - The optional third line contains a comment or weather information.

- **Magenta** for transmitted data. In this case, each line is preceded by the radio channel and priority. 0 for the first channel, 1 for the second if used. "H" means high priority for digipeated packets. "L" is for lower priority packets originating at this station.
- **Red** for errors. If a newcomer is wondering why his transmissions are not showing up in other applications, these error messages might provide a clue about the problem.



Other common errors are pointed out to help troubleshoot why signals are not interpreted as the sender probably expected.

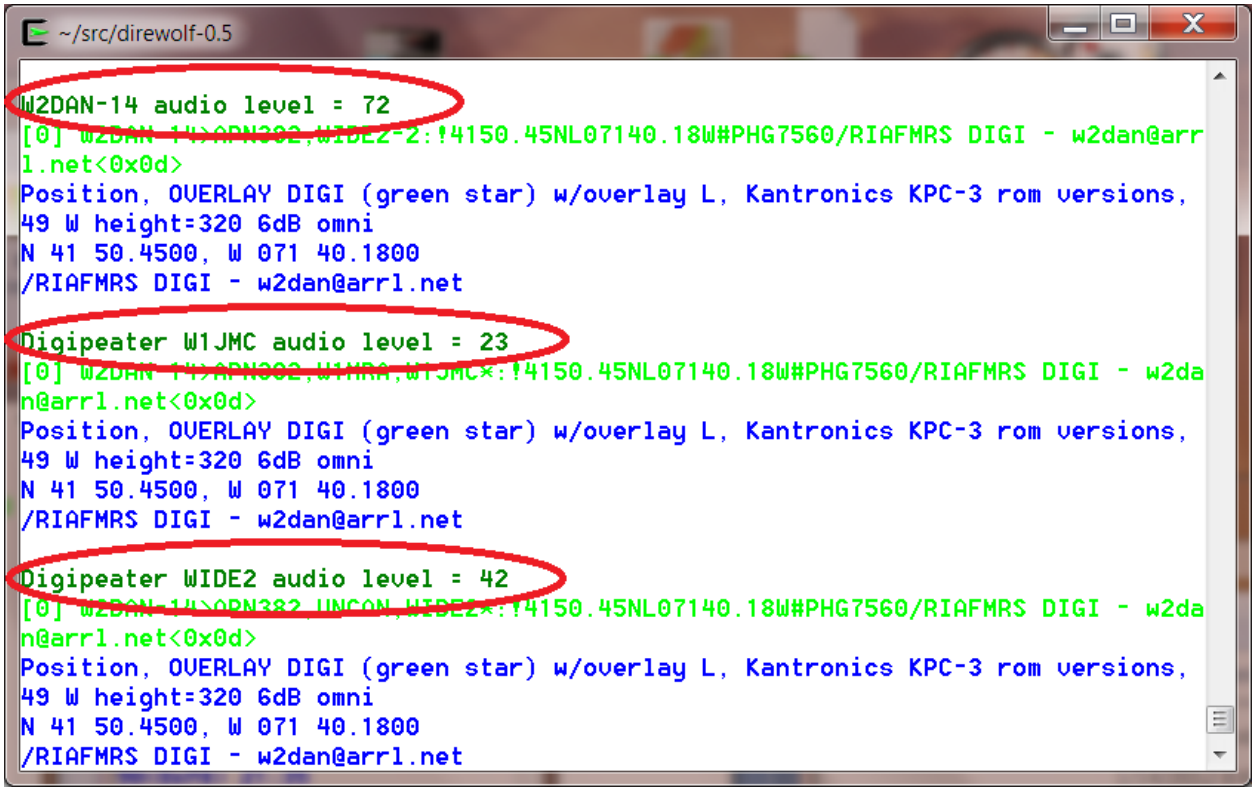
```
Digipeater W1XM audio level = 27    [NONE]
[0] N8VIM>BEACON,W1XM*,WIDE2-1:!4240.85N/07133.99W_PHG72604/ Pepperell, MA. WX. 442.9+ PL100<0
Didn't find wind direction in form c999.
Didn't find wind speed in form s999.
Didn't find wind gust in form g999.
Didn't find temperature in form t999.
Weather Report, WEATHER Station (blue)
N 42 40.8500, W 071 33.9900
, "PHG72604/ Pepperell, MA. WX. 442.9+ PL100"
```

```
Digipeater WIDE2 (probably UNCAN) audio level = 19   [NONE]
[0] W1TG-1>APU25N,KQ1L-8,UNCAN,WIDE2*:>210144zDX: W1XM-15 42.21.62N 71.05.36W 42.0 miles 199□ 21:
0x0d>
Status Report, Ambulance, UIview 32 bit apps
DX: W1XM-15 42.21.62N 71.05.36W 42.0 miles 199□ 21:30
Character code 0xf8 is probably an attempt at a degree symbol.
The correct encoding is 0xc2 0xb0 in UTF-8.


Digipeater WIDE2 (probably UNCAN) audio level = 19   [NONE]
[0] KC1AWV-15>APWW10,KQ1L-8,KB1POR-2,UNCAN,WIDE2*:>FN43mbI&DX: KB1DOK-2 20.2mi 346□ 02:17
Warning: Lower case letter in Maidenhead locator.  Specification requires upper case.
Error: Found 'D' instead of space required after symbol code.
Status Report, Igate Generic (please use mor, APRSISCE win32 version
Grid square = FN43mb, N 43 03.7500, W 070 57.5000
X: KB1DOK-2 20.2mi 346□ 02:17 4320.91N 07103.81W
Character code 0xb0 is probably an attempt at a degree symbol.
The correct encoding is 0xc2 0xb0 in UTF-8.
```

That's it. You can't interact with it directly. Use one of the many APRS / packet radio applications designed to interface with a physical TNC.

There is quite a bit of information packed in there.

The first line of each group contains the audio level of the station heard. This number depends on the volume level of your receiver and the gain setting of the computer audio input. The absolute numbers have no meaning but the relative values are revealing.



Consider the items circled above.

- In the first case, we are hearing the original transmission directly.
- In the other two cases, we are hearing the same thing from two different digipeaters.

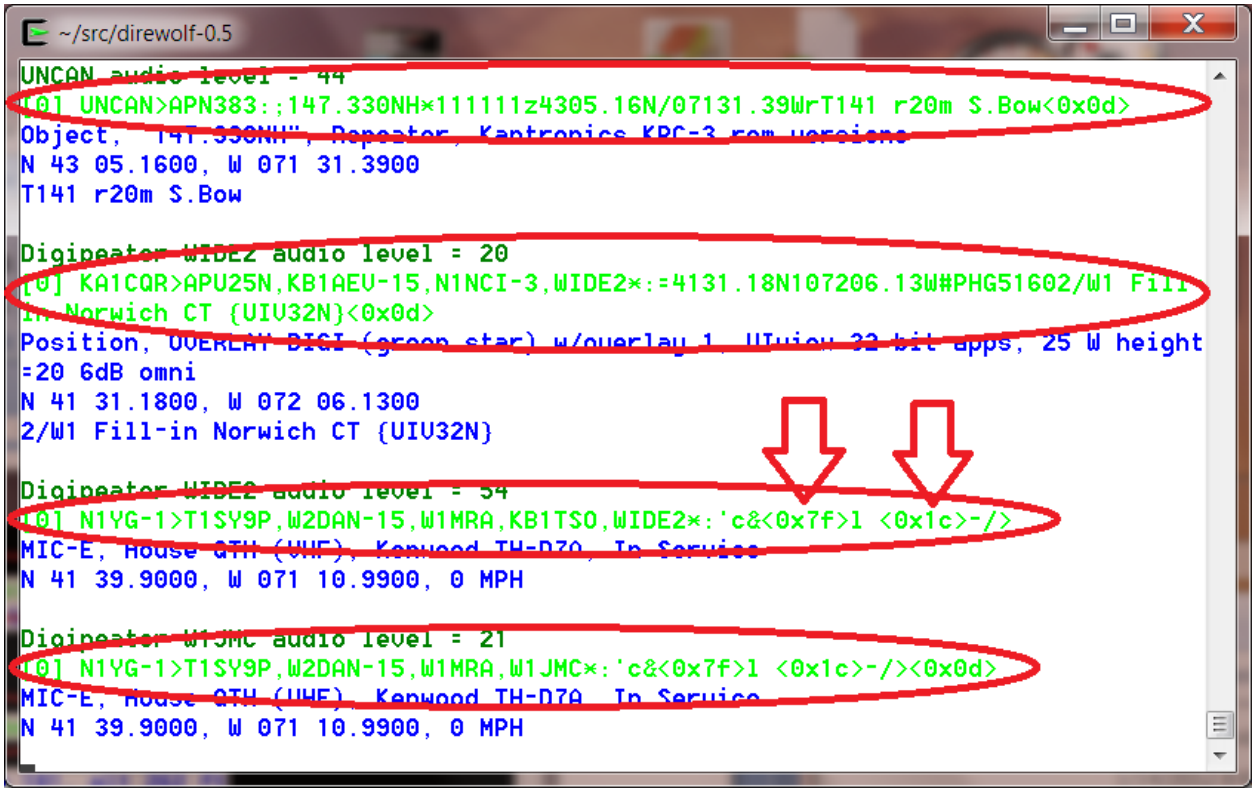
Notice that the audio levels vary quite a bit. If the level is too high, clipping will occur resulting in signal distortion and a much lower chance of being demodulated properly.

Dire Wolf has an automatic gain control and can handle a very wide range of audio signal levels. Other systems are not as forgiving.

A station using Dire Wolf can monitor the audio levels and advice those which are significantly different than most others.

The second line of each group has the raw received data. It has the following parts:

- "[0]" indicates it was received on the first (or only) radio channel.
- The source station.
- The "destination" which is a misleading name. For the MIC-E encoding it is part of the location. In most other cases, it identifies the type of device or software application.
- Digipeaters. "*" indicates it is the station we are actually receiving.
- Finally the information part of the packet. notice that unprintable characters are represented by their hexadecimal representation such as "<0x1c>". This is the same convention used by http://aprs.fi



Finally we have decoded information in blue.

The first line contains the message type, symbol, and other station attributes such as equipment/application type.

The second line is the location and optional speed and direction of travel.

The final line has any comment or weather information.

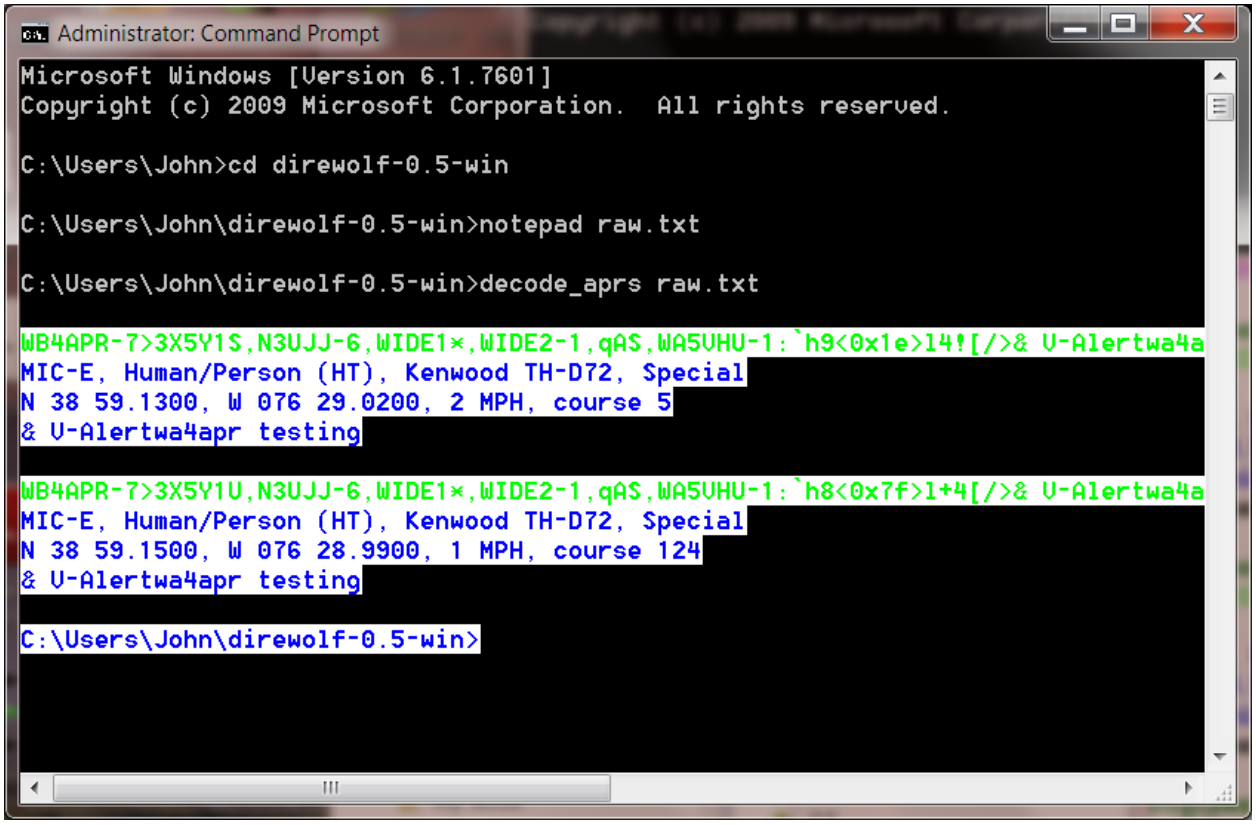
6.1 Raw Packet Decoder: decode_aprs.exe

Part of the Dire Wolf application is packaged as a separate raw packet decoder. As an example, you might find something like this in the raw data section of http://aprs.fi or http://findu.com .

WB4APR-7>3X5Y1S,N3UJJ-6,WIDE1*,WIDE2-1,qAS,WA5VHU-1:`h9<0x1e>l4![/>& V-Alertwa4apr testing=
WB4APR-7>3X5Y1U,N3UJJ-6,WIDE1*,WIDE2-1,qAS,WA5VHU-1:`h8<0x7f>l+4[/>& V-Alertwa4apr testing=

What do all those strange characters mean?

Put the raw packets into a text file. Remove any leading time stamps.
Run decode_aprs with the name of file on the command line.



One interesting thing to note here is that some message types use non-printable characters. In this case, we use the form **<0x**>** where ** is the hexadecimal representation. In the example above, we find two unprintable characters <0x1e> <0x7f>.

7 Data Rates

Packet radio can be sent over many different speeds and modulation methods. Here is a brief overview that might help clear up some of the confusion.

7.1 Bits per Second (bps) vs. Baud

The terms "Bits per Second" (bps) and Baud are often used interchangeably because they are often the same number.

Baud refers to the maximum number of "symbols" (signal states) per second. With two tone frequency shift keying a "symbol" represents a single bit so the numbers are the same. With more advanced modulation techniques we can send multiple bits at the same time. In this case, bits per second will be some multiple of the Baud.

7.2 1200 bps

This is the original method from when packet radio got started about 30 years ago and still the most popular. It is based on the Bell 202 standard which switches between 1200 and 2200 Hz tones to represent the two signal states. This is called Audio Frequency Shift Keying (AFSK). It is simple, easy to implement, and should work with any transceiver designed for voice. It isn't very fussy about the audio amplifier passband characteristics so you can simply use the microphone and speaker connections.

7.3 300 bps

Below 28 MHz, we are legally limited to 300 baud data (here, maybe different in other countries). HF operation typically uses AFSK with a difference of 200 Hz between the two tones. When AFSK is sent with an SSB transmitter it becomes FSK of the RF signal.

A slight mistuning of the receiver frequency will result in a corresponding difference in the audio tones. Dire Wolf can tolerate this mistuning by using multiple demodulators tuned to different audio frequency pairs.

A few references:

Packet Radio on HF http://wiki.complete.org/PacketRadioOnHF
Others… ?

Google for "hf aprs" for many discussions on this topic.

7.4 9600 bps

Rather than converting the digital data to audio, it is also possible to use the digital signal for direct FSK on the RF carrier. Here are some early designs from the previous century.

- K9NG - need to find link…
- G3RUH - http://www.amsat.org/amsat/articles/g3ruh/109.html
- KD2BD -http://www.amsat.org/amsat/articles/kd2bd/9k6modem/

The audio amplifiers – in both the transmitter and receiver – are designed for voice operation and don't have the necessary bandwidth for digital signals. Trying to use the microphone and speaker connections will only result in disappointment.

Some newer radios have "data" connectors that bypass the audio stages. (I think that is confusing. They should be labeled external modem.) Other equipment will need to be modified. The received signal needs to be taken from the discriminator before amplification stages have the chance to corrupt it. For transmitting, a direct connection needs to be made into the modulator. Here are some useful tips for 9600 baud operation:

http://www.wb4hfn.com/Resources/9600MAN.TXT
ftp://ftp.tapr.org/general/9600baud/

7.5 2400 bps

There are different – and incompatible – ways to get 2400 bits per second through a voice radio.

The MFJ-2400 packet modem uses the CCITT v.26 / Bell 201 modem standard. Rather than using multiple tones, this uses a single 1800 Hz tone but the phase is shifted to convey data. This is called Phase Shift Keying (PSK). In this case, the phase is shifted in multiples of 90° to send two bits at the same time. The phase changes at a maximum rate of 1200 "symbols" per second. The signal state changes at 1200 baud and two bits are sent at once so we end up with 2400 bits per second.

Dire Wolf does not have PSK capability.

AFSK could also be used but you'd probably need to get the two tones a little further apart for good results. I've seen references to ham radio 2400 baud AFSK with 1200/2400 and 1775/3250 tone pairs. That last one would probably have some trouble getting through the audio stages of most transceivers.

7.6 4800 bps

There are even more ways to get 4800 bits/second.

Using the same 1200 baud, 3 bits can be sent at once using 8 different phases or introducing multiple amplitudes.

I've heard of people using AFSK with 2400 and 4800 tones but it would be necessary to modify radios for greater audio bandwidth.

Finally, the Hamilton Area Packet Network “HAPN-T” board pushes the digital signal through the radio in the same way we would for 9600 baud operation. The literature doesn’t mention anything about data scrambling so it would probably not be compatible with the K9NG/G3RUH scheme.

8 Configuration File & command line options

The default configuration provides standard 1200 baud AFSK reception and will be adequate for many people. Those desiring more features and flexibility can change the operation by editing the configuration file and restarting Dire Wolf. Some of the options available include:

- Selecting alternate audio devices.
- Dual channel (stereo) operation for use with two transceivers.
- Audio sampling rate to balance between performance and CPU power required.
- Transmission rates other than 1200 baud. e.g. 300 for HF use.
- AFSK tones other than 1200 & 2200 Hz
- Digipeating.
- APRStt Gateway
- Internet Gateway (IGate).
- Beaconing.

The configuration file (direwolf.conf) contains documentation and examples in comments. Normally the configuration file is read from the current working directory. On Linux the user's home directory is also searched. The "-c" command line option can be used to read a file from a different location.

Other command line options are described at the end of this section.

Configuration commands are a keyword followed by parameters.
Command keywords are case insensitive. i.e. upper and lower case are equivalent.
Command parameters are case sensitive. i.e. upper and lower case are different.

Example: The next two are equivalent

```
PTT /dev/ttyS0 RTS
ptt /dev/ttyS0 RTS
```

But this not equivalent because device names are case sensitive.

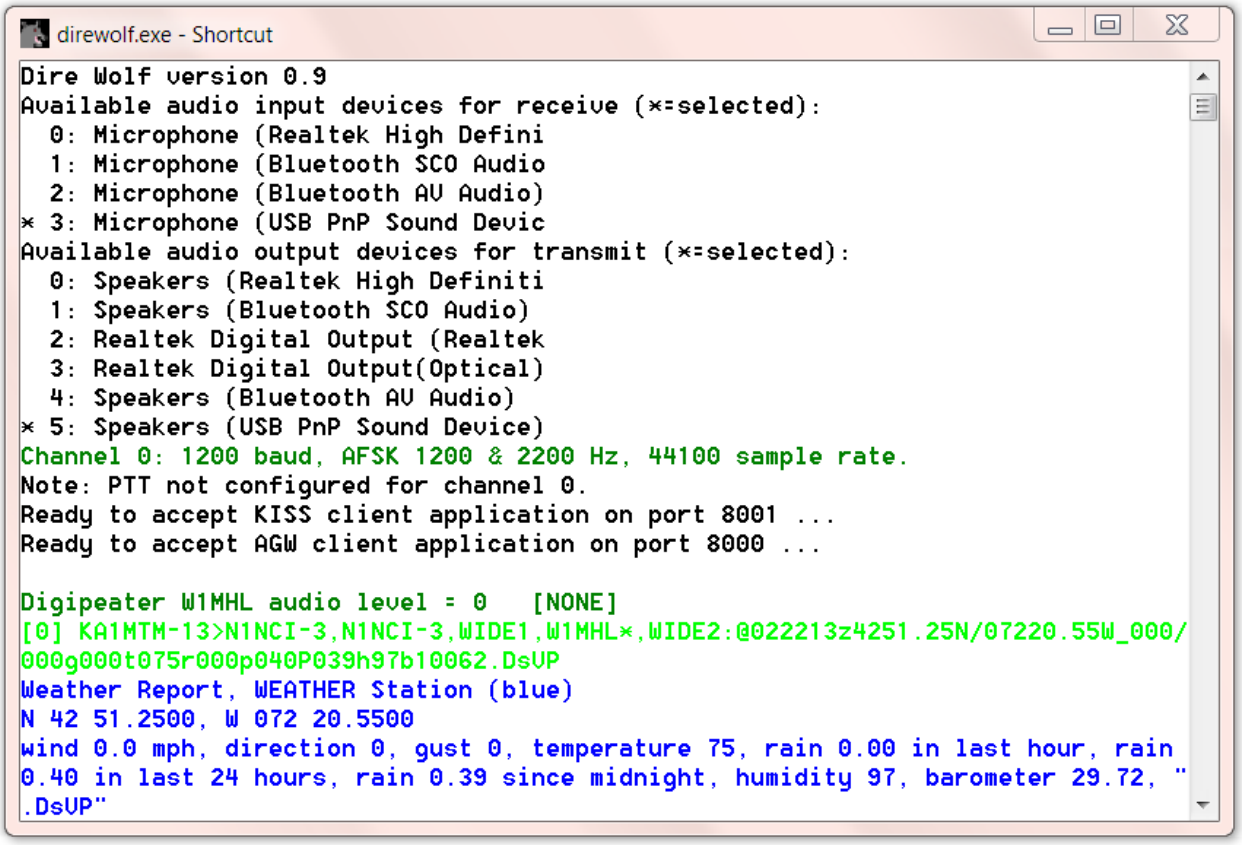
```
PTT /dev/TTYs0 RTS
```

8.1 Audio Device

Normally the system default audio device is used. There are situations where you would want to select a different device for connection to your radio. This might be a PCI card installed internally or an external USB audio adapter.

8.1.1 Audio Device selection - Windows

When Dire Wolf starts up, it displays the available audio devices.



Input devices and output devices are listed with an assigned number. Notice that the same physical device can have different numbers for input and output. In the example above, the USB audio device is 3 for input and 5 for output. To select this device, add this to the configuration file:

```
ADEVICE  3  5
```

For the Windows version, you can also specify some substring from the description. To select the USB audio device, you could alternatively use this:

```
ADEVICE  USB USB
```

You could also shorten it to use the same device for both input and output.

```
ADEVICE  USB
```

8.1.2 Audio Device selection – Linux ALSA

Linux ALSA audio devices are much more flexible and therefore more complicated and confusing.

Instead of getting close to the hardware, we want to use a higher level, more abstract view, which hides these details. Instead of the lower case L option, use upper case L instead. The two following commands produce more than 300 lines so this has been trimmed down to emphasize the relevant parts.

```
john@hamshack:~/direwolf-0.9$ arecord -L

default
    Playback/recording through the PulseAudio sound server
sysdefault:CARD=ICH5
    Intel ICH5, Intel ICH5
    Default Audio Device
front:CARD=ICH5,DEV=0
    Intel ICH5, Intel ICH5
    Front speakers

...
plughw:CARD=ICH5,DEV=3
    Intel ICH5, Intel ICH5 - ADC2
    Hardware device with all software conversions
sysdefault:CARD=Device
    USB PnP Sound Device, USB Audio
    Default Audio Device
front:CARD=Device,DEV=0
    USB PnP Sound Device, USB Audio
    Front speakers

...
hw:CARD=Device,DEV=0
    USB PnP Sound Device, USB Audio
    Direct hardware device without any conversions
plughw:CARD=Device,DEV=0
    USB PnP Sound Device, USB Audio
    Hardware device with all software conversions
sysdefault:CARD=Live
    SB Live! Value [CT4780], ADC Capture/Standard PCM Playback
    Default Audio Device
front:CARD=Live,DEV=0
    SB Live! Value [CT4780], ADC Capture/Standard PCM Playback
    Front speakers

...
plughw:CARD=Live,DEV=2
    SB Live! Value [CT4780], Multichannel Capture/PT Playback
    Hardware device with all software conversions
```

Output choices.

```
john@hamshack:~/direwolf-0.9$ aplay -L

default
    Playback/recording through the PulseAudio sound server
sysdefault:CARD=ICH5
    Intel ICH5, Intel ICH5
    Default Audio Device

...
hw:CARD=Device,DEV=0
    USB PnP Sound Device, USB Audio
    Direct hardware device without any conversions
plughw:CARD=Device,DEV=0
    USB PnP Sound Device, USB Audio
    Hardware device with all software conversions
sysdefault:CARD=Live
    SB Live! Value [CT4780], ADC Capture/Standard PCM Playback
    Default Audio Device

...
```

```
plughw:CARD=Live,DEV=3
    SB Live! Value [CT4780], Multichannel Playback
    Hardware device with all software conversions
```

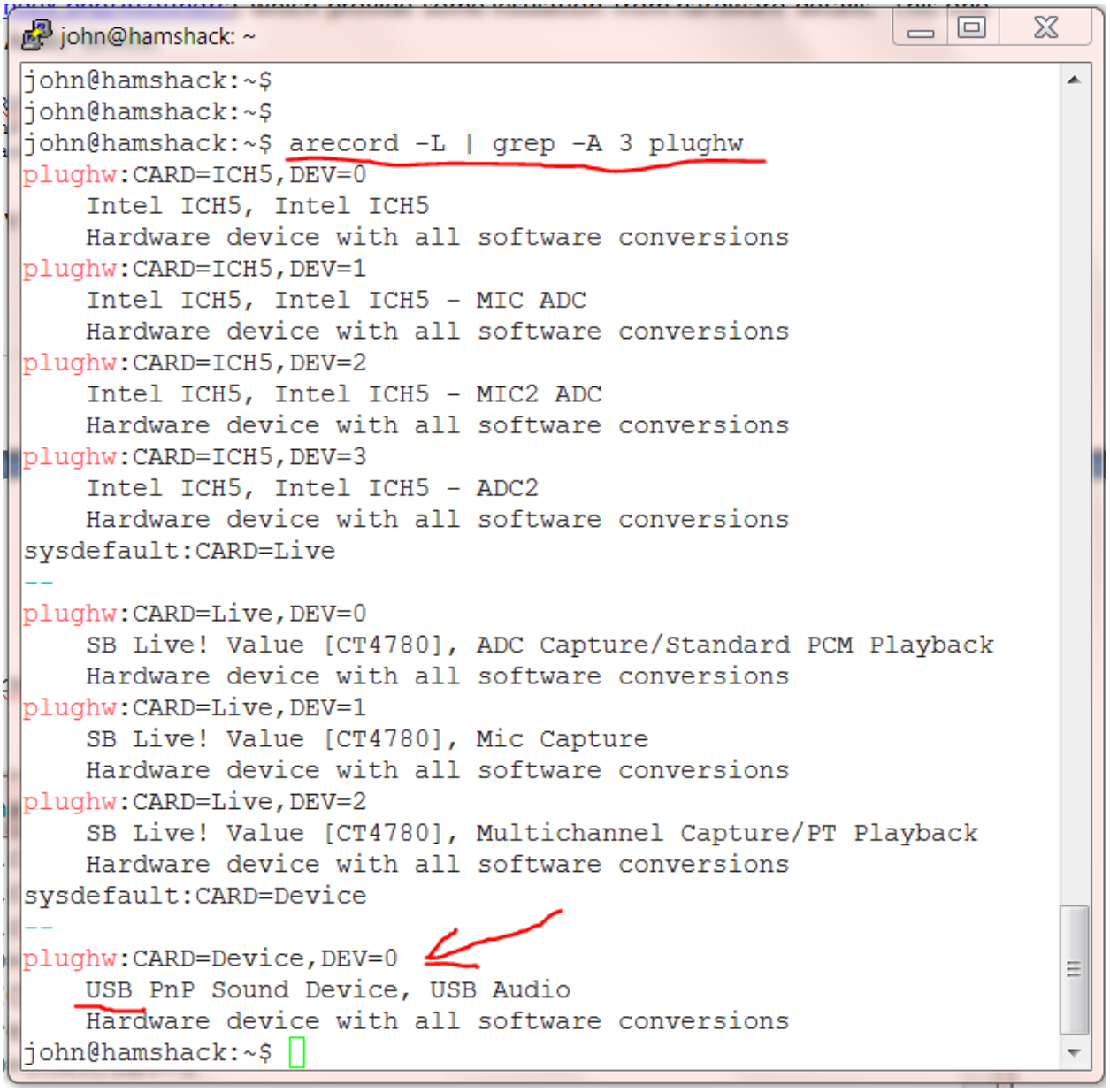
Too many choices! This is very confusing.

My recommendation is to use one of the "plughw" plugins (see http://www.alsa-project.org/main/index.php/Asoundrc) which provide some insulation from hardware details. This one mentions the USB Audio device.

```
plughw:CARD=Device,DEV=0
    USB PnP Sound Device, USB Audio
    Hardware device with all software conversions
```

Here is an easy way to get a list of just the "plughw" devices:

```
arecord -L | grep -A 3 plughw
```

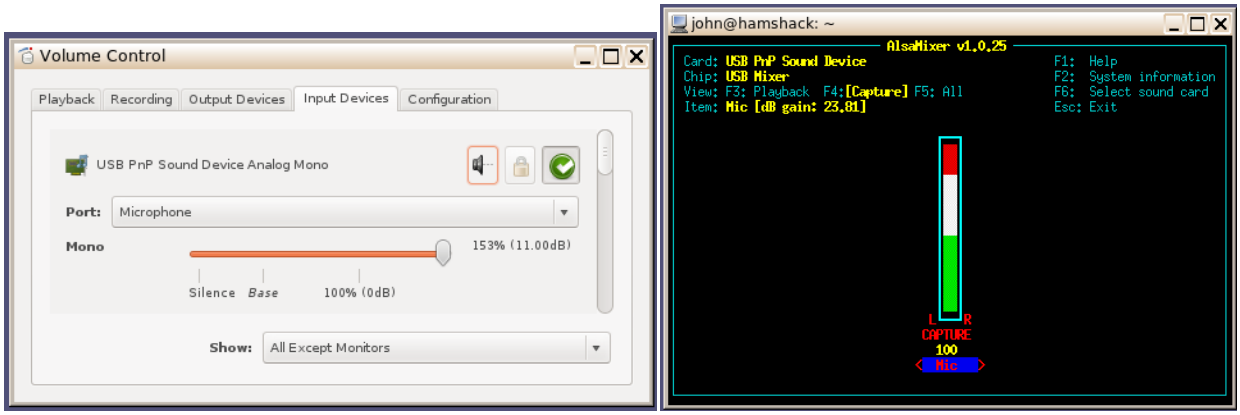


In this case, I want to pick the USB device. Copy the "plughw:…" line and put it in the configuration file preceded by ADEVICE.

To make a long story short, this would be a suitable configuration file setting for selecting the USB audio device on my computer. Yours might have different names.

```
ADEVICE  plughw:CARD=Device,DEV=0
ACHANNELS  1
```

Use **pavucontrol**, **alsamixer**, or similar application to set the audio signal levels.



8.1.3 You might want to skip this section.

This section describes an experiment that didn't work out so well. I'm including it because more advanced readers might find it educational. Others will just get more confused.

Most people will want to skip this section and continue with "Audio Device Properties."

You can get a list of the hardware devices with the "**arecord -l**" and "**aplay -l**" commands (NOTE: option is lower case L.)

```
john@hamshack:~/direwolf-0.9$ arecord -l

**** List of CAPTURE Hardware Devices ****
card 0: ICH5 [Intel ICH5], device 0: Intel ICH [Intel ICH5]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: ICH5 [Intel ICH5], device 1: Intel ICH - MIC ADC [Intel ICH5 - MIC ADC]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: ICH5 [Intel ICH5], device 2: Intel ICH - MIC2 ADC [Intel ICH5 - MIC2 ADC]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: ICH5 [Intel ICH5], device 3: Intel ICH - ADC2 [Intel ICH5 - ADC2]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: Device [USB PnP Sound Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: Live [SB Live! Value [CT4780]], device 0: emu10k1 [ADC Capture/Standard PCM
Playback]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: Live [SB Live! Value [CT4780]], device 1: emu10k1 mic [Mic Capture]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: Live [SB Live! Value [CT4780]], device 2: emu10k1 efx [Multichannel Capture/PT
Playback]
  Subdevices: 1/1
  Subdevice #0: subdevice #0


john@hamshack:~/direwolf-0.9$ aplay -l

**** List of PLAYBACK Hardware Devices ****
```

```
card 0: ICH5 [Intel ICH5], device 0: Intel ICH [Intel ICH5]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: ICH5 [Intel ICH5], device 4: Intel ICH - IEC958 [Intel ICH5 - IEC958]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: Device [USB PnP Sound Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: Live [SB Live! Value [CT4780]], device 0: emu10k1 [ADC Capture/Standard PCM
Playback]
  Subdevices: 32/32
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  …
  Subdevice #30: subdevice #30
  Subdevice #31: subdevice #31
card 2: Live [SB Live! Value [CT4780]], device 2: emu10k1 efx [Multichannel Capture/PT
Playback]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  …
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 2: Live [SB Live! Value [CT4780]], device 3: emu10k1 [Multichannel Playback]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

In this example, we have 3 audio devices.

Card 0 is on the motherboard.
Card 1 is a cheap USB audio adapter.
Card2 is a PCI card.

Troubleshooting tip:

What if “aplay –l” complains, “no soundcards found…”?

I had a situation where user “root” could see the devices but an ordinary user could not. The solution was to add the user to the “audio” group like this.

sudo addgroup john audio

Sometimes you want to get close to the hardware but this is not one of them. The chip inside of the cheap USB audio adapter is physically capable of only single channel (mono) input and two channel (stereo) output. This presents a little problem. If we specify single channel operation,

ADEVICE hw:1,0
ACHANNELS 1

We get the following error message because the output side is capable of only 2 channel (stereo) operation.

Could not set number of audio channels.
Invalid argument
for hw:1,0 output.

If we try to use 2 channels, with this configuration,

```
ADEVICE  hw:1,0
ACHANNELS  2
```

We get a different error because the input side is capable of only single channel operation.

```
Could not set number of audio channels.
Invalid argument
for hw:1,0 input.
```

By using **plughw** instead of **hw**, an extra layer of software hides these inconvenient hardware details.

8.1.4 Audio Device properties

Two options are available

ARATE *sample-rate*

Where,
sample-rate is number of audio samples per second.
The default is 44100.
Other standard values are 22050 and 11025.

CHANNELS *num-channels*

Where,
num-channels is 1 for mono (default) or 2 for stereo,
allowing use of two radio channels.

All of the performance tuning has been done with the standard audio sample rate of 44100. A lower rate has less demanding CPU requirements but performance will be slightly degraded. Use a lower rate only if your computer is too slow to keep up.

8.1.5 Use with Software Defined Radios

When using software defined radios (SDR), the audio will be coming from another application rather than a "soundcard."

Gqrx (2.3 and later) has the ability to send streaming audio through a UDP socket to another application for further processing. As explained in http://gqrx.dk/doc/streaming-audio-over-udp, select the Network tab of the audio settings window. Enter the host name or address where Dire Wolf will be running. Use "localhost" if both are on the same computer. Pick some unused UDP port. Here we use the same number as in the gqrx documentation.

Use the following Dire Wolf configuration file options:

```
ADEVICE udp:7355 default
ARATE 48000
CHANNELS 1
```

Alternatively, you can override the configuration file settings with command line options like this:

```
direwolf -n 1 -r 48000 -b 16 udp:7355
```

"-n 1" sets number of audio channels to 1.
"-r 48000" means audio sample rate of 48000 per second.
"-b 16" means 16 bits per sample, signed, little endian.

Other SDR applications might produce audio on stdout so it is convenient to pipe into the next application. In this example, the final "-" means read from stdin.

```
rtl_fm -f 144.39M -o 4 - | direwolf -n 1 -r 24000 -b 16 -
```

See http://kmkeen.com/rtl-demod-guide/index.html for rtl_fm documentation.

Here is another possible variation you might want to try. In one window, start up Dire Wolf listening to a UDP port. Note that rtl_fm has a default sample rate of 24000.

```
direwolf -n 1 -r 24000 -b 16 udp:7355
```

In a different window, run rtl_fm and use the netcat utility to send the audio by UDP.

```
rtl_fm -f 144.39M -o 4 - | nc -u localhost 7355
```

Note that the SDR and Dire Wolf can be running on different computers, even different operating systems. You could use the command above on Linux but change localhost to the address of a Windows machine where Dire Wolf is running.

8.2 Radio channel configuration

As mentioned above you can have 1 or 2 radio channels. Specify options for one or two channels like this:

CHANNEL 0
(options for first (left) or only channel: MYCALL, MODEM, PTT, etc.)

CHANNEL 1
(options for second (right) channel if two specified earlier)

8.2.1 Radio channel – MYCALL

Multiple radio channels can use the same or different station identifiers. This is required for digipeating. Example:

MYCALL WB2OSZ-5

The APRS specification requires that the call use only upper case letters and digits. The substation id (SSID), if specified, must be in the range of 1 to 15.

8.2.2 Radio channel - Modem configuration & multiple decoders

Each radio channel can be configured separately for different speeds and audio tones for the AFSK modems. The general form of the configuration option is:

MODEM *baud [mark space [A][B][C] [number offset]]*

This replaces the 3 separate options HBAUD, MARK, and SPACE in earlier versions and adds a new capability.

The default configuration is 1200 baud, 1200 & 2200 Hz tones for VHF FM use. The equivalent configuration option is:

MODEM 1200 1200 2200

The following would be a suitable configuration for 300 baud HF SSB operation using the popular 1600 / 1800 Hz tone pair.

MODEM 300 1600 1800

Starting with version 0.9 it is possible to have multiple decoders running in parallel. For 1200 baud (standard for VHF FM), there are 3 different decoder types fine tuned in different ways. There are a few cases where one will successfully decode a marginal signal that the other two can't. By running two or three at the same time, decoder performance is increased. "A" is the one from previous versions. "B" is a little better but takes more processing power. "C" is even better but takes more processing power. You can choose your own processing power vs. performance tradeoff.

MODEM 1200 1200 2200 A
MODEM 1200 1200 2200 B
MODEM 1200 1200 2200 C
MODEM 1200 1200 2200 BC
MODEM 1200 1200 2200 ABC

Don't be scared about running all 3 unless you have a really old slow computer. It only takes about 10% of the CPU time of a typical 3 year old PC.

When using HF SSB, any mistuning or poor calibration can cause the audio frequencies to shift. These are less likely to be decoded properly. For this situation, we have a different style of multiple decoders per channel. This time they are tuned to different audio frequency pairs. With this example, we have 7 different modems, spaced 30 Hz apart.

MODEM 300 1600 1800 7 30

When the application starts up, the modem configuration is confirmed along with the audio frequencies for each. This should be able to tolerate mistuning of 100 Hz in each direction.

```
Channel 0: 300 baud, AFSK 1600 & 1800 Hz, 44100 sample rate.
        0.0: 1510 & 1710
        0.1: 1540 & 1740
        0.2: 1570 & 1770
        0.3: 1600 & 1800
        0.4: 1630 & 1830
        0.5: 1660 & 1860
        0.6: 1690 & 1890
```

When multiple modems are configured per channel, a simple spectrum display reveals which decoders picked up the signal properly.

|	means a frame was received with no error.
:	means a frame was received with a single bit error. (FIX_BITS 1 or higher configured.)
.	means a frame was received with multiple errors. (FIX_BITS 2 or higher configured.)
_	means nothing was received on this decoder.

Here are some samples and what they mean.

___|___	Only the center decoder (e.g. 1600/1800 Hz) was successful.
_|||___	3 different lower frequency modems received it properly. Assuming USB operation, the transmitting station is probably a little low in frequency.
___|||:	3 different higher frequency modems received it with no error. The highest one received it with a single bit error.

Here are some typical signals heard on 10.1476 MHz USB.

```
KA0MOS-9 audio level = 49   [NONE]   __||||_
[0.3] KA0MOS-9>APZ111,WIDE:>TNOSaprs 1.11.2 TNOS APRS IGATE Conqu
Status Report, CAR, Experimental
TNOSaprs 1.11.2 TNOS APRS IGATE Conquerall Bank Nova Scotia

KA0MOS-9 audio level = 50   [NONE]   ___|||:
[0.4] KA0MOS-9>APZ111,WIDE:!4419.82N/06429.32W&144.390,144.970
Position, HF GATEway, Experimental
N 44 19.8200, W 064 29.3200
144.390,144.970

WA8LMF-13 audio level = 54   [SINGLE]   ____:__
[0.4] WA8LMF-13>APU25N,ECHO:}KJ4ERJ-15>APZTLE,TCPIP,WA8LMF-13
+4A$[%q4zN{!wWQ!<0x0d>
Third Party Header, REC. VEHICLE, UIview 32 bit apps

WA8LMF-13 audio level = 48   [NONE]   ___||__
[0.3] WA8LMF-13>APU25N,ECHO:>151334z_Live 30Meter APRS Activi
Status Report, REC. VEHICLE, UIview 32 bit apps
_Live 30Meter APRS Activity  Map: http://wa8lmf.net/map
```

The beginning of the monitor line shows the radio channel and which modem was used.

You can optionally specify a single letter to select the decoder type when specifying multiple frequencies.

```
MODEM 300 1600 1800 A 7 30
MODEM 300 1600 1800 B 7 30
MODEM 300 1600 1800 C 7 30
```

G3RUH data scrambling is used with there are no AFSK tones specified:

```
MODEM 9600
```

As mentioned in an earlier section. This won't work with the microphone and speaker connection on your transceiver. The audio amplifiers, designed for voice, do not have enough bandwidth and distort the signal so it is not usable.

8.2.3 Radio Channel – Push to Talk (PTT)

There are three different methods available for activating your transmitter.

- Serial port control lines.
- General Purpose I/O pins (Linux only).
- VOX (voice operated transmit) – External hardware activates the transmitter when transmit audio is present.

To use a serial port (either built-in or a USB to RS232 adapter cable), use an option of this form:

PTT *device-name* [-]*rts-or-dtr*

For Windows the device name would be COM1, COM2, etc.

For Linux, the device name would probably be something like /dev/ttyS0 or /dev/ttyUSB0. You can also use the Windows format. COM1 is converted to /dev/ttyS0, COM1 is converted to /dev/ttyS1, and so on.

Normally the higher voltage is used for transmit. Prefix the control line name with "-" to get the opposite polarity.

On Linux you can use General Purpose I/O (GPIO) pins if available. This is mostly applicable to a microprocessor board, such as a Raspberry Pi or BeagleBone, not a general purpose PC.

PTT GPIO [-]*pin-number*

There are more details in the separate Raspberry Pi APRS document.

Examples:

```
PTT  COM1  RTS
PTT  COM1  -DTR
PTT  /dev/ttyUSB0  RTS
PTT  GPIO  25
```

Note that it is possible to get two separate transmit controls from a single serial port by using both the RTS and DTR signals.

8.2.4 Radio Channel – Transmit timing

After turning on transmitter, send "flag" characters for TXDELAY * 10 milliseconds for transmitter to stabilize before sending data. The default of "30" actually means 300 milliseconds. This is for compatibility with most other implementations.

```
TXDELAY 30
```

Keep transmitting for TXTAIL * 10 milliseconds after sending the data. This is needed to avoid dropping PTT too soon and chopping of the end of the data. There is latency between the time we send data to a sound card and when it actually comes out so we need to provide a little extra time to be safe. "10" actually means 100 milliseconds, again for compatibility with others.

```
TXTAIL 10
```

SLOTTIME and PERSIST are used to generate a random time between the time when the channel is clear and when we start transmitting.

They have the same traditional meanings as in nearly every TNC going back 30 years. You probably want to keep the defaults. This delay is not used when transmitting digipeated frames.

```
SLOTTIME 10
PERSIST 63
```

8.2.5 Radio Channel – Allow frames with bad CRC

Normally we want to reject any received frame if the CRC is not perfect. Some TNCs have a "passall" option that skips the FCS check and allows all sorts of random garbage to get thru. Dire Wolf can optionally try to fix a small number of corrupted bits. "Fix" is probably too strong of a word. It's really a good guess and there is no guarantee that it is right. The default is currently:

```
FIX_BITS  1
```

See section called "One Bad Apple Don't Spoil the Whole Bunch" for more discussion.

8.3 Client application interface

Three different interfaces are provided for client applications such as APRSIS32, UI-View32, Xastir, APRS-TW, YAAC, SARTrack, and many others.

8.3.1 AGWPE network protocol

In most case, Dire Wolf can be used as a drop in replacement for AGWPE. By default, it listens on network port 8000. This can be changed with a command resembling:

```
AGWPORT  8000
```

8.3.2 Network KISS

The KISS protocol can also be used with a network port so Dire Wolf and the client application can be running on different computers. The default is:

```
KISSPORT  8001
```

8.3.3 Serial port KISS - Windows

A configuration option like this:

NULLMODEM COM3

will provide a dumb KISS TNC on COM3. You need to provide either a "null modem" cable to another serial port, used by the application, or configure a virtual null modem cable.

See later section, with "**com0com**" in the title, for an in depth discussion of how this works.

8.3.4 Serial port KISS - Linux

This feature does not use the configuration file. Instead it is activated by using the –p option on the command line.

A "pseudo terminal" is created, providing a virtual KISS TNC. The Linux chapter, "KISS TNC emulation – serial port" section, provides some examples of how to use this with some popular applications.

8.4 Digipeater operation

8.4.1 Digipeater - Configuration Details

Digipeater configuration is achieved with commands of the form:

DIGIPEAT *from-chan to-chan aliases wide [preemptive]*

where,

from-chan	is the channel where the packet is received.
to-chan	is the channel where the packet is to be re-transmitted.
aliases	is an alias pattern for digipeating ONCE. Anything matching this pattern is effectively treated like WIDE1-1. 'MYCALL' for the receiving channel is an implied member of this list.
wide	is the pattern for normal WIDEn-N digipeating where the ssid is decremented.
preemptive	is one of the preemptive digipeating modes: OFF, DROP, MARK, or TRACE. Default is off.

Pattern matching uses "extended regular expressions." Rather than listing all the different possibilities (such as "WIDE3-3,WIDE4-4,WIDE5-5,WIDE6-6,WIDE7-7"), a pattern can be specified such as "^WIDE[34567]-[1-7]$". This means:

^	beginning of call. Without this, leading characters don't need to match and ZWIDE3-3 would end up matching.
WIDE	is an exact literal match of upper case letters W I D E.
[34567]	means ANY ONE of the characters listed.
-	is an exact literal match of the "-" character (when not found inside of []).
[1-7]	is an alternative form where we have a range of characters rather than listing them all individually.
$	means end of call. Without this, trailing characters don't need to match. As an example, we would end up matching

WIDE3-15 besides WIDE3-1.

Google "Extended Regular Expressions" for more information.

Duplicates are not transmitted if the same thing was transmitted within the DEDUPE number of seconds. The default is

DEDUPE 30

Duplicate checking is performed by comparing the source, destination, and information part. In other words, the via path is ignored.

8.4.2 Digipeater - Typical configuration

Enable digipeating by editing the configuration file (direwolf.conf) and modifying the two lines that look similar to this:

- MYCALL NOCALL

  Obviously, you would want to change this to your own call.
  For example: MYCALL WB2OSZ-5

- #DIGIPEAT 0 0 ^WIDE[3-7]-[1-7]$ ^WIDE[12]-[12]$

  Remove the "#" character at the beginning of the line. Lines beginning with "#" are comments and they are ignored.

Restart Dire Wolf so it will read the modified configuration file.

What does this all mean?

- The first 0 means the rule applies to packets received on radio channel 0.

- The second 0 means anything matching the rule is transmitted on channel 0.

- Next we aliases that need to match exactly. We use ^WIDE[3-7]-[1-7]$ to "trap" larger values of N as discussed in

  Fixing the 144.39 APRS Network
  The New n-N Paradigm
  http://www.aprs.org/fix14439.html

- The final parameter specifies patterns to be processed with the new n-N paradigm if not caught by the aliases.

8.4.3 Digipeater – example 2 – routing between two states.

In this hypothetical example, we are on top of a tall hill between Massachusetts and New Hampshire.

- Radio channel 0: Directional antenna towards MA
- Radio channel 1: Directional antenna towards NH

Each channel does its normal digipeating out to the same channel. Anything with **MA***n-n* in the path should be sent to channel 0 regardless of where it came from.

```
DIGIPEAT 0 0 ^WIDE[3-7]-[1-7]$ ^WIDE[12]-[12]$|^MA[1-7]-[1-7]$
DIGIPEAT 1 0 ^WIDE[3-7]-[1-7]$ ^WIDE[12]-[12]$|^MA[1-7]-[1-7]$
```

Similarly we want anything for NH to be digipeated only to radio channel 1.

```
DIGIPEAT 0 1 ^WIDE[3-7]-[1-7]$ ^WIDE[12]-[12]$|^NH[1-7]-[1-7]$
DIGIPEAT 1 1 ^WIDE[3-7]-[1-7]$ ^WIDE[12]-[12]$|^NH[1-7]-[1-7]$
```

8.4.4 Digipeater algorithm

If the first unused digipeater field, in the received packet, matches the first pattern, the original digipeater field is ***replaced by*** MYCALL of the destination channel.

```
Example:    W9XYZ>APRS,WIDE7-7
Becomes:    W9XYZ >APRS,WB2OSZ*
```

In this example, we trap large values of N as recommended in http://www.aprs.org/fix14439.html

If not caught by the first pattern, see if it matches the second pattern. Matches will be processed with the usual WIDEn-N rules.

If N >= 2, the N value is decremented and MYCALL (of the destination channel) is ***inserted*** if enough room.

```
Example:    W9XYZ >APRS,WIDE2-2
Becomes:    W9XYZ >APRS,WB2OSZ*,WIDE2-1
```

If N = 1, we don't want to keep WIDEn-0 in the digipeater list so the station is ***replaced by*** MYCALL.

```
Example:    W9XYZ >APRS,WIDE2-1
Becomes:    W9XYZ >APRS,WB2OSZ*
```

If N = 0, the hop count has been used up and the packet is not digipeated.

8.4.5 Digipeater - Compared to other implementations

Based on observations, some other popular implementations ***always insert*** their call rather than replacing when the hop count is all used up. Example:

	Unconditional insert	**Adaptive insert / replace**
Original digipeater path	WIDE1-1,WIDE2-2	WIDE1-1,WIDE2-2
After 1 hop	W1ABC,WIDE1*,WIDE2-2	W1ABC*,WIDE2-2
After 2 hops	W1ABC,WIDE1,W2DEF*,WIDE2-1	W1ABC,W2DEF*,WIDE2-1
After 2 hops	W1ABC,WIDE1,W2DEF,W3GHI,WIDE2*	W1ABC,W2DEF,W3GHI*
Implemented by	KPC-3+, TM-D710A	Dire Wolf

The unconditional insert approach has a rather unfortunate consequence. The final packet looks like it was relayed by **five** different digipeaters.

- W1ABC
- Unknown station not implementing tracing.
- W2DEF
- W3GHI
- Unknown station not implementing tracing.

The packet is longer than it needs to be and wastes radio channel capacity.

This also creates an ambiguous situation where we are not sure about the path taken. Here is a real example that demonstrates the different cases and something new and unexpected.

We start off with the original packet. There is no "*" in the header, so we are hearing the originating station.

```
N1TBN-9 audio level = 9   [NONE]
[0] N1TBN-9>T2SU5U,WIDE1-1,WIDE2-1:`c.<m>Lk/]"4G}449.075MHz=<0x0d>
MIC-E, truck, Kenwood TM-D710, In Service
N 42 35.5500, W 071 18.3200, 15 MPH, course 48, alt 157 ft, 449.075MHz
```

Next we see the same packet (below) after it was digipeated by WB2OSZ-5 and AB1OC-10. Notice how the original WIDE1-1 was replaced by WB2OSZ-5 because the remaining hop count was all used up.

```
Digipeater WIDE2 audio level = 11   [SINGLE]
[0] N1TBN-9>T2SU5U,WB2OSZ-5,AB1OC-10,WIDE2*:`c.<m>Lk/]"4G}449.075MHz=<0x0d>
MIC-E, truck, Kenwood TM-D710, In Service
N 42 35.5500, W 071 18.3200, 15 MPH, course 48, alt 157 ft, 449.075MHz
```

The "*" appears after WIDE2 so that is what the radio is hearing. If we didn't know the earlier history, we wouldn't know whether WIDE2-0 (the -0 is not displayed) was left there by AB1OC-5 or a different later station that did not identify itself.

Here is something totally unexpected. Below we see the packet was digipeated twice and we are hearing W1HML, as indicated by the "*" after it.

```
Digipeater W1MHL audio level = 13   [NONE]
[0] N1TBN-9>T2SU5U,WB2OSZ-5,W1MHL*,WIDE2:`c.<m>Lk/]"4G}449.075MHz=<0x0d>
MIC-E, truck, Kenwood TM-D710, In Service
N 42 35.5500, W 071 18.3200, 15 MPH, course 48, alt 157 ft, 449.075MHz
```

The really strange part is a WIDE2-0, at the end, which is not marked as being used. When the remaining count is reduced to zero, the digipeater should be marked as being used.

In version 1.0, we start to list the **possible** actual station heard when "*" is after something of the form WIDEn-0. Example:

```
Digipeater WIDE2 (probably AB1OC-10) audio level = 10   [NONE]
[0] KB1DDC>TR3R8X,W1XM,WIDE1,AB1OC-10,WIDE2*:'c&*l <0x1c>-/]"3r}<0x0d>
MIC-E, House, Kenwood TM-D700, En Route
N 42 32.8800, W 071 10.1400, 0 MPH, alt 0 ft
```

8.4.6 Preemptive Digipeating

Normally the digipeater function looks only at the first unused item in the digipeater list. The preemptive option allows processing of any unused field, not just the first one, if my call or an alias matches. Note that the option does not apply to the "generic XXXXn-N" specification.

Example: The received packet contains these digipeaters:

CITYA*, CITYB, CITYC, CITYD, CITYE

The first one has already been used. My alias list includes CITYD.

Normally, this would not be retransmitted because CITYB is not in the alias list. When the preemptive option is selected, "CITYD" is matched even though it is not the first unused. As you would expect, CITYD is replaced by my call before retransmission. What happens to CITYB and CITYC? That depends on the option specified:

- DROP – All prior path data is lost.
- MARK – Prior path data is marked as being used.
- TRACE – Prior path data will reflect the actual path taken.

Results, for this example, are summarized below.

Option	Path after digipeating	Comment
OFF	(none)	No match. Not digipeated.
DROP	WB2OSZ*, CITYE	Erases history before getting here. Gives incorrect impression that original station was heard directly rather than via CITYA.
MARK	CITYA, CITYB, CITYC, WB2OSZ*, CITYE	Gives incorrect impression that packet traveled through CITYB and CITYC.

TRACE	CITYA, WB2OSZ*, CITYE	Accurate tracing of path used to get here.

8.5 Beaconing

Dire Wolf has several configuration commands for setting up periodic transmissions.

8.5.1 Position & Object Beacons

Two configuration commands are available for periodic beacons to announce yourself or other things in your region with fixed positions.

PBEACON - for a "position report." This is generally used for your own location.

OBEACON - for an "object report." This is generally used for other entities.
The big difference is that the "object report" contains an **object name**, usually different than your radio call.

These have many options so it would be very cumbersome and error prone to have everything in fixed positions. Instead we use **keyword=value** pairs. The available keywords are:

Keyword	Description	Example values	Comment
DELAY	Time, in minutes or minutes:seconds, to delay before sending first time. Default is 1 minute.	1 0:30	One minute. Half minute.
EVERY	Time, in minutes or minutes:seconds, between transmissions. Default is 10 minutes.	10 9:45	Ten minutes. 9 ¾ minutes
SENDTO	Radio channel for transmission or "IG" to send to Internet Gateway. Default is the first, or only, radio channel 0.	1 IG	Second radio channel. Internet Gateway.
VIA	Digipeater path. Default none.	WIDE1-1 WIDE1-1,WIDE2-1	Upper case only. No spaces.
OBJNAME	Name for object, up to 9 characters. Applies only to **O**BEACON.	EOC Hamfest	Any printable characters including embedded spaces.
LAT	Latitude in signed decimal degrees (negative for south) or degrees ^ minutes hemisphere.	42.619 42^37.14N	Both examples are equivalent.
LONG	Longitude in signed decimal degrees (negative for west) or degrees ^ minutes hemisphere.	-71.34717 71^20.83W	Both examples are equivalent.
SYMBOL	Two different styles are available: (a) Exactly two characters specifying symbol table / overlay and the symbol code. (b) A substring of the description found	S# "Jet ski"	More details below.

	in symbolsX.txt or symbols-new.txt.		
OVERLAY	A single upper case letter or digit overlay character.	S	
POWER	Transmitter power in watts.	50	
HEIGHT	Antenna height in feet.	20	
GAIN	Antenna gain in dBi.	6	
DIR	One of 8 directions, N, NE, E, SE, S, SW, W, or NW, for a directional antenna. Default is omni-directional.	NE	
FREQ	Where to contact you by voice or radio frequency for some other entity. MHz.	146.955	
TONE	CTCSS tone required for specified radio frequency. Hz.	74.4	
OFFSET	Transmit offset in MHz.	-0.60	MHz.
COMMENT	Name, location, announcements, etc.		
COMPRESS	Use 1 for compressed format. Note that power/height/gain gets converted to single radio range value in the compressed format.	0 1	Human readable. Compressed.

Note: Entire configuration item must be on a single line. Some of the examples, below, are on multiple lines due to page width limitation.

Any values containing spaces must be surrounded by quotation marks.

Example: Typical home station. The ASCII character set does not contain the degree symbol so we use ^ instead to separate degrees and minutes. If no symbol is given, it defaults to house.

```
PBEACON  LAT=42^37.14N  LONG=71^20.83W
```

You might want to identify your station once every ten minutes with different ranges. This would use the WIDE2-2 path twice an hour and no digipeating the other four times per hour.

```
PBEACON  DELAY=1  EVERY=30  VIA=WIDE2-2 LAT=42^37.14N  LONG=71^20.83W
PBEACON  DELAY=11  EVERY=30              LAT=42^37.14N  LONG=71^20.83W
PBEACON  DELAY=21  EVERY=30              LAT=42^37.14N  LONG=71^20.83W
```

The easy way to specify a symbol is with a substring of the description found in the included files **symbolsX.txt** or **symbols-new.txt** and on optional overlay. Examples:

```
PBEACON  LAT=42^37.14N  LONG=71^20.83W  SYMBOL="Jet Ski"
PBEACON  LAT=42^37.14N  LONG=71^20.83W  SYMBOL="digi"  OVERLAY=S
```

For more precise control, you can specify exactly two characters with a particular pattern. The first character indicates:

/ = primary symbol table
\ = alternate symbol table

A-Z 0-9 = alternate symbol table with specified overlay.

These two are equivalent:

```
PBEACON  LAT=42^37.14N  LONG=71^20.83W  SYMBOL=\#  OVERLAY=S
PBEACON  LAT=42^37.14N  LONG=71^20.83W  SYMBOL=S#
```

To advertize a voice repeater in your neighborhood:

```
OBEACON  OBJNAME=146.955ma LAT=42^34.61N  LONG=71^26.47W SYMBOL=/r
        OFFSET=-0.600 TONE=74.4 COMMENT="www.wb1gof.org"
```

Remember it must be a single line in the configuration file even though it is two lines on this page. Note how "/r" was used to get the repeater symbol. If you used "SYMBOL=repeater", it would end up matching the "Mic-E Repeater" description and the symbol code would come out as "m."

In this case, FREQ= would be redundant because the frequency is part of the object name. See http://aprs.org/localinfo.html for recommendations.

The **symbols-new.txt** file is still evolving. You can download the latest from http://www.aprs.org/symbols/symbols-new.txt

8.5.2 Custom Beacon

For unusual situations, or if you enjoy composing obscure APRS packets by hand, the custom beacon type is available.

The timing, transmission channel, and digipeater via path are the same as for the position and object beacons. The difference is that you can put anything you want in the information part.

Keyword	Description	Example values	Comment
DELAY	Time, in minutes or minutes:seconds, to delay before sending first time. Default is 1 minute.	1 0:30	One minute. Half minute.
EVERY	Time, in minutes or minutes:seconds, between transmissions. Default is 10 minutes.	10 9:45	Ten minutes. 9 ¾ minutes
SENDTO	Radio channel for transmission or "IG" to send to Internet Gateway. Default is the first, or only, radio channel 0.	1 IG	Second radio channel. Internet Gateway.
VIA	Digipeater path. Default none.	WIDE1-1 WIDE1-1,WIDE2-1	Upper case only. No spaces.
INFO	Handcrafted "information" part for packet.		

Some examples …

8.6 Internet Gateway (IGate)

Dire Wolf can serve as a gateway between the radio network and servers on the Internet. This allows information to be retrieved from locations such as http://aprs.fi or http://findu.com. Information can optionally be relayed from the servers, through your station, and on to the radio.

First you need to specify the name of a Tier 2 server. The current preferred way is to use one of these regional rotate addresses:

- noam.aprs2.net - for North America
- soam.aprs2.net - for South America
- euro.aprs2.net - for Europe and Africa
- asia.aprs2.net - for Asia
- aunz.aprs2.net - for Oceania

Each name has multiple addresses corresponding to the various servers available in your region. Why not just specify the name of one specific server? This approach offers several advantages:

- Simplicity – You don't need to change your configuration as new servers become available or disappear.
- Resilience – If your current server becomes unavailable, another one will be found automatically.
- Load balancing – Picking one at random helps to spread out the load.

Visit http://aprs2.net/ for the most recent information. You also need to specify your login name and passcode. For example:

```
IGSERVER  noam.aprs2.net
IGLOGIN  WB2OSZ-5  123456
```

If you want to transmit information from the servers, you need to specify two additional pieces of information. First, you need to specify the radio channel and the via path for the packet header. Examples:

```
IGTXVIA  0  WIDE1-1,WIDE2-1
IGTXVIA  1  WZ9ZZZ
IGTXVIA  0
```

In the first case packets will be transmitted on the first radio channel with a path of WIDE1-1,WIDE2-1. In the second case, packets are transmitted on the second radio channel and directed to a known nearby digipeater with wide coverage. In the third case, there will be no digipeating.

You will probably want to apply a filter for what packets will be obtained from the server. Read about filters here: http://www.aprs2.net/wiki/pmwiki.php/Main/FilterGuide Example:

```
IGFILTER  m/50
```

Finally, we don't want to flood the radio channel. The IGate function will limit the number of packets transmitted during 1 minute and 5 minute intervals. If a limit would be exceeded, the packet is dropped and warning is displayed in red.

```
IGTXLIMIT  6  10
```

If you want your station to appear at http://findu.com or http://aprs.fi , you need to send a beacon advertising your position. If you send it over the radio, another IGate client station needs to hear you and pass the information along to a server.

To put your own station on the map, without relying on someone else to hear you, send a beacon to the IGate server by specifying "SENDTO=IG" in the beacon configuration. Use overlay R for receive only, T for two way.

```
PBEACON sendto=IG delay=0:30 every=60:00 symbol="igate"
                              overlay=R lat=42^37.14N long=071^20.83W

PBEACON sendto=IG delay=0:30 every=60:00 symbol="igate"
                              overlay=T lat=42^37.14N long=071^20.83W
```

8.7 APRStt Gateway

The APRStt Gateway function allows a user, equipped with only a DTMF ("touch tone") pad, to enter information into the global APRS network. Various configuration options determine how the touch tone sequences get translated to APRS "object" packets. They are easily recognized because they all begin with TT.

TTPOINT
TTVECTOR
TTGRID
TTUTM
TTCORRAL
TTMACRO
TTOBJ

See separate document, APRStt Implementation Notes" for all the details.

8.8 Command Line Options

Command line options can be used to specify the configuration file location or override some of the settings in the configuration file.

-c *fname* Configuration file name.

-r *n* Audio sample rate. e.g. 44100

-n	*n*	Number of audio channels. 1 or 2.
-b	*n*	Bits per audio channel. 8 bit unsigned or 16 bit signed little endian.
-B	*n*	Data rate in bits/sec. Standard values are 300, 1200, 9600. If < 600, AFSK tones are set to 1600 & 1800. If > 2400, K9NG/G3RUH style encoding is used. Otherwise, AFSK tones are set to 1200 & 2200.
-d	*x*	Debug communications with client application a = AGWPE network protocol k = KISS serial port n = KISS network u = Redisplay non-ASCII characters in hexadecimal
-t	*n*	Text colors. 1 = normal, 0 = disable text colors.
-x		Send transmit level calibration tones.

After any options, there can be a single command line argument for the source of received audio. This can overrides the audio input specified in the configuration file. Choices are:

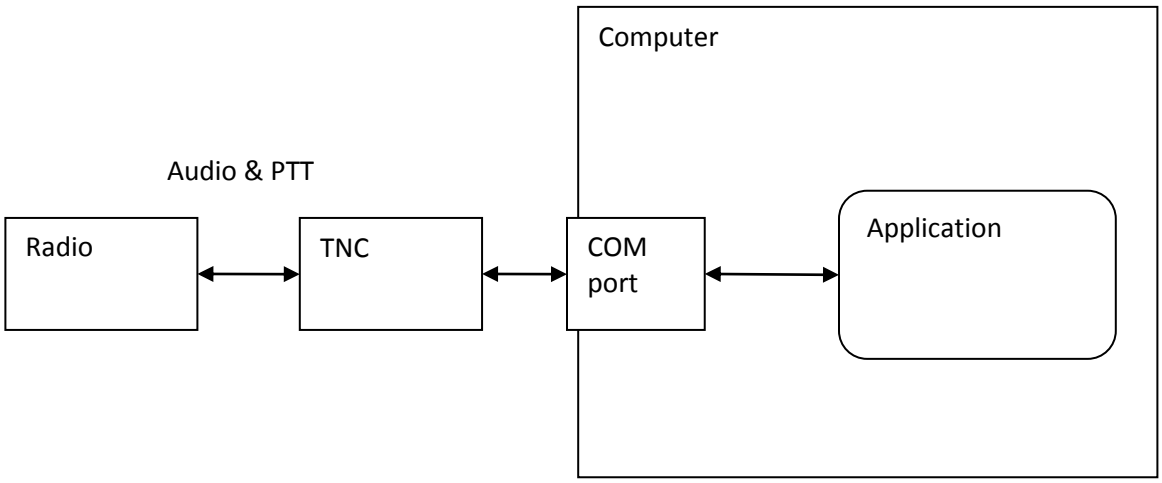
- "-" or "stdin" for reading from stdin.
- "UDP:" followed by an optional port number to read from a UDP socket.

The Software Defined Radio section contains some examples.

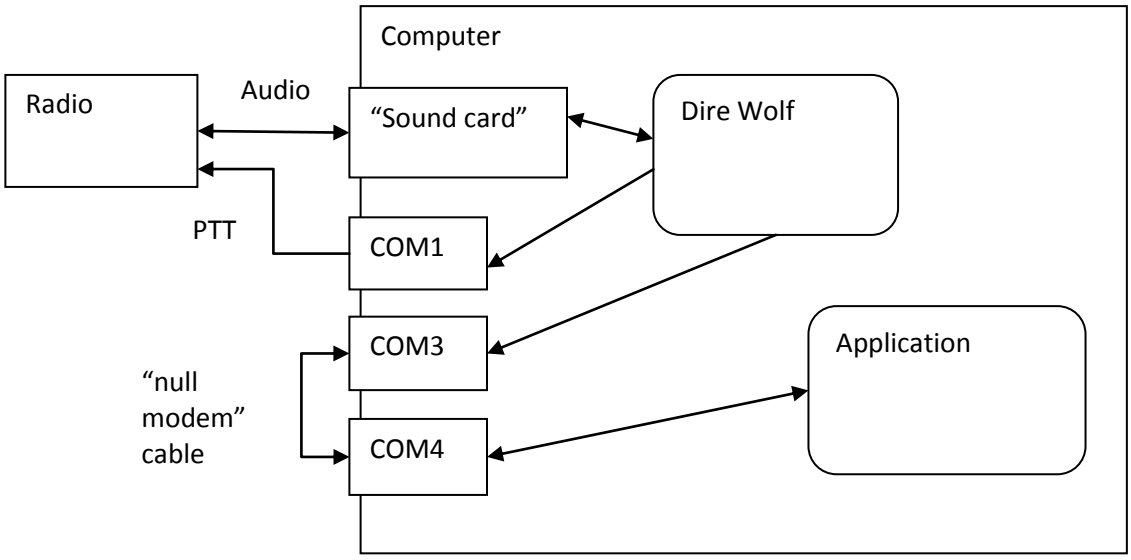
9 Advanced Topics - Windows

9.1 Install com0com (optional)

Many Windows packet radio applications can communicate with a physical TNC connected to a serial port, as illustrated below.

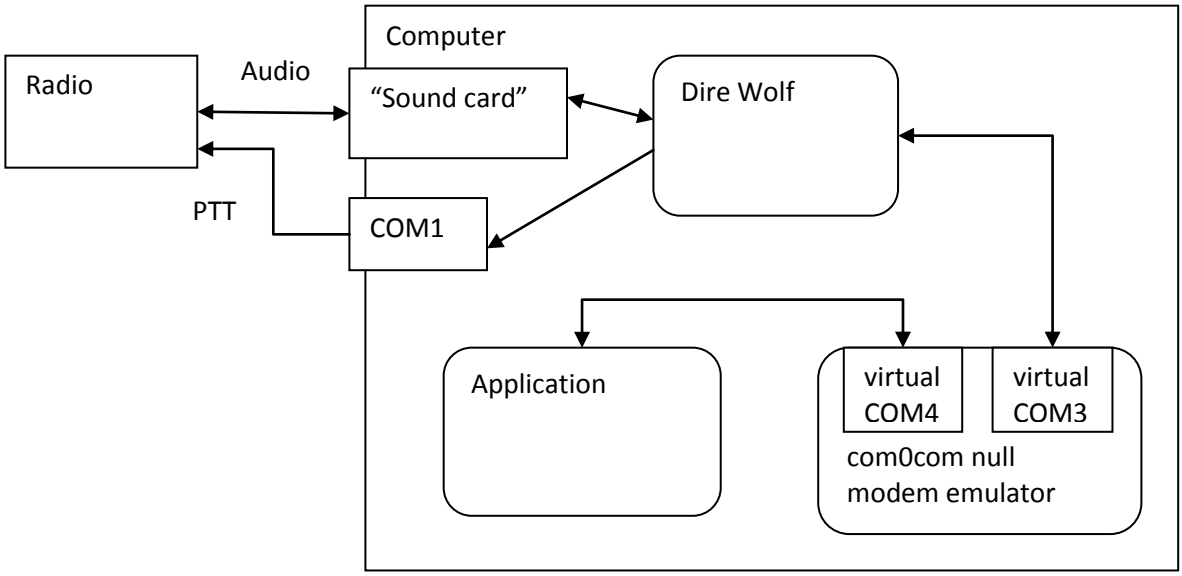


Dire Wolf is a software replacement for a separate TNC. One way of using it is illustrated below.



The packet radio application expects to find a TNC on COM4. COM3, connected to Dire Wolf, behaves like a KISS TNC. The two serial ports are connected to each other with a "null modem" cable. Anything coming out of the COM3 port goes into COM4 and vice versa.

Rather than having two physical serial ports, connected by an external cable, we can use a pair of virtual ports.



Special software tricks both Dire Wolf and the packet radio application into thinking they are using a pair of physical serial ports connected to each other.

This step is <u>not</u> necessary if you only want to use the "AGW TCPIP socket interface" or KISS over a network connection.

Down load and install the "Null-modem emulator" from http://sourceforge.net/projects/com0com/

Click on the "View all files" button then pick "com0com" and the most recent version, currently 2.2.2.0.

If you have the 64 bit version of Windows 7, download the file with "x64-fre" in the name. Otherwise, get the file with "i386-fre" in the name.

Follow the instructions for installation.

This creates two virtual serial ports named CNCA0 and CNCB0. We want to rename them to COM3 and COM4. There is an opportunity to run the Setup Command Prompt at the end of the installation. You can also run it at a later time with:

Start → All Programs → com0com → Setup Command Prompt

Enter these commands, <u>exactly</u> as shown:

```
change CNCA0 PortName=COM3,EmuBR=yes
change CNCB0 PortName=COM4,EmuOverrun=yes
quit
```

It is very important that you apply the options as shown. Without them, Dire Wolf might hang trying to write to COM3 if nothing is connected to COM4.

Verify that it works correctly.

(1) Open a Cygwin command window and type this:

```
cat /dev/com4
```

(2) Open another Cywin command window and type this:

```
echo "testing 123" > /dev/com3
```

(3) Notice that "testing 123" appears in the window from step (1).

Edit the configuration file, "direwolf.conf." Look for the line that looks like this:

```
# NULLMODEM COM3
```

and remove the "#" character from the beginning of the line.

If you followed the instructions here, Dire Wolf will make the virtual COM3 behave like a KISS TNC. Configure your application to use COM4 and it will think it is attached to an external TNC.

If you already have a COM3 or COM4, use other numbers and make the appropriate substitutions in all of the configuration steps.

9.2 Build Dire Wolf from source (optional)

The Windows version contains prebuilt executable files so you don't need to build it from source. Some people might want to. Here is how.

9.2.1 Windows

The Windows version is built with the MinGW compiler from http://www.mingw.org/. "cd" into the source directory and run "make" with the Windows-specific Makefile.

```
cd direwolf-0.9-src
make -f Makefile.win
```

The result should be several new executable files including "direwolf.exe" and "decode_aprs.exe."

9.2.2 Linux

See section 5.

10 Receive Performance

10.1 WA8LMF TNC Test CD

How does Dire Wolf perform compared with other approaches? The de facto standard of measurement is the number of packets decoded from Track 2 of WA8LMF's TNC Test CD obtained from http://wa8lmf.net/TNCtest/index.htm

Here are some results that have been found. WARNING! Do not take them too seriously. They should only be taken as ballpark figures. These tests were not performed under identical carefully controlled scientific conditions. Very large differences are probably significant. However, any small differences are completely meaningless and could be misleading.

Reference	**TNC**	**Packets decoded**
KI4MCW https://sites.google.com/site/ki4mcw/Home/arduino-tnc	Arduino Duemilanove (328p)	871
	TNC-X	818
	Argent Data OpenTracker 1+	729
	AGWPE 2005.127	500
	Linux PC soundmodem	412
	Linux PC multimon	130
N4MSJ http://groups.yahoo.com/group/tnc-x/message/542	KPC-3	986
	MFJ-1274	883
	AEA PK90	728
	Early Beta TT4	920
4X6IZ http://www.tau.ac.il/~stoledo/Bib/Pubs/QEX-JulAug-2012.pdf	AX25 Java Soundcard Modem	964
N1VG http://www.tapr.org/pipermail/aprssig/2007-May/019449.html	Tracker 2	910
	KPC-3 (non-plus)	967
	uTNT	970
	Tracker 2 with TCM3105	991
	AEA PK-90	728
	MFJ-1274	883
WB2OSZ	Dire Wolf version 0.5	931
	Dire Wolf version 0.6 "FIX_BITS" option is set to 0 for all of these tests.	965
	Dire Wolf version 0.9 A B C ABC	965 968 971 976

Version 0.9 has three different decoders fine tuned in different ways. The original one, from earlier versions, is called “A.” The additional decoders, called “B” and “C,” offer slightly better performance at the cost of greater CPU requirements.

Another, called “F” (for fast) is really “A” but it handles only the default case of 1200 baud data and 44,100 sample rate. It is optimized for low end processors that don’t have vector math instructions and doesn’t offer much benefit with Intel x86 type processors.

Decoder	Packets decoded from WA8LMF test CD	Relative amount of CPU time required.	Comment
A	965	43	Same as previous versions.
B	968	53	
C	971	63	
A & B & C	976	111	Best decoding, most CPU required.
F	965	37	Mostly benefits microprocessor systems. Only for 1200 baud, 44100 sample rate.

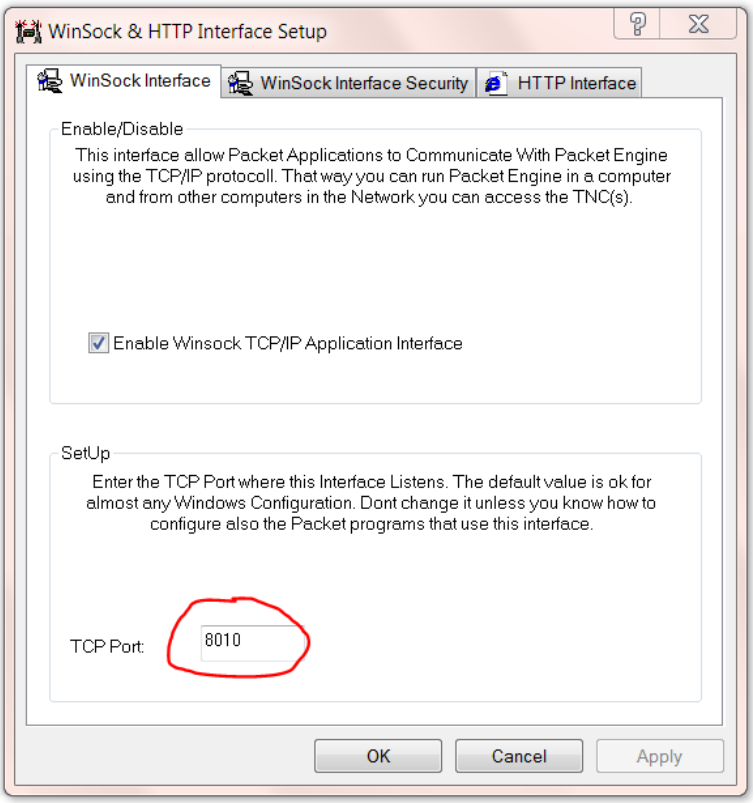
You can select one or more to run in parallel with a configuration file setting. Best results are obtained when using A, B, and C at once. It still takes less than 10% of a typical home computer, now 3 years old, so why not use it if you have plenty of CPU power to spare?

10.2 1200 Baud software TNC comparison

Here we compare 1200 baud decoder performance against two other popular “soundcard TNC” applications by running them all at the same time with the same live audio. First we need to configure them so they all use different TCP ports for communication with client applications.

10.2.1 Prepare AGWPE

Download AGWPE “Hamware” version 2013.415 from http://www.sv2agw.com/downloads/
Configure it to use TCP port 8010 rather than the default 8000.



10.2.2 Prepare UZ7HO SoundModem

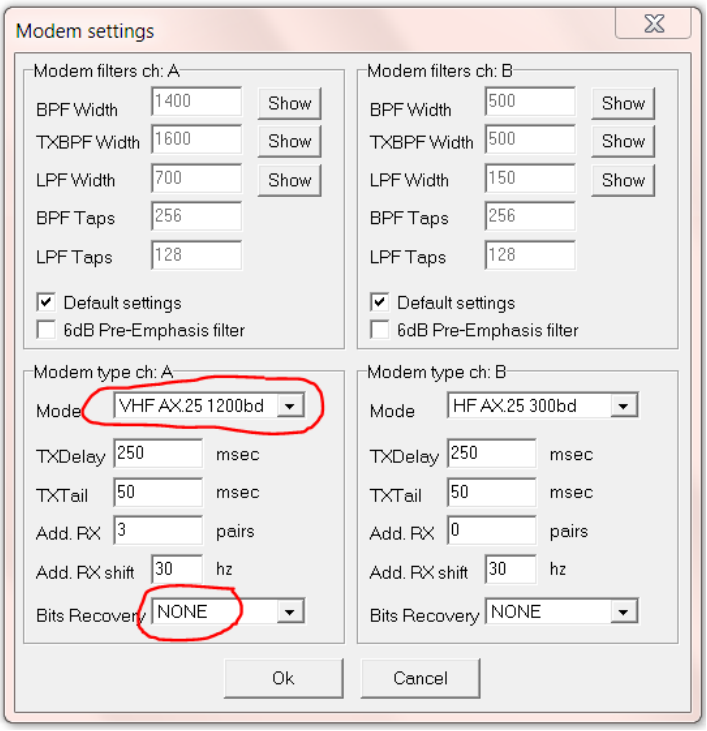
Download from http://uz7ho.org.ua/packetradio.htm

Edit the soundmodem.ini file. Look for the server port number and change the default 8000 to 8020.

```
[AGWHost]
Server=1
```

```
Port=8020
```

This did not seem to have any effect. You will notice that we still use the default port of 8000 later. Pick 1200 baud modem. Be sure Bits Recovery is set to none so we have a fair comparison.



10.2.3 Prepare Dire Wolf

Be sure to use Dire Wolf version 0.9 or later. Edit the configuration file to contain:

```
MODEM 1200 1200 2200 abc
AGWPORT  8040
FIX_BITS 0
```

Using the default of "FIX_BITS 1" would be cheating. This is similar to the UZ7HO SoundModem "Bits Recovery" option and is explained in a later section with "One Bad Apple" in the name.

10.2.4 Compare them.

Run the "aclients" application with command line arguments like this

```
aclients   8010=AGWPE   8000=UZ7HO   8040=DireWolf
```

This connects to all 3 TNC applications at the same time and prints the packets in columns. From a distance, it's easy to see the general trend of how they compare. Periodically total numbers of packets received are printed.

After running for almost a full day, we find this at the end:

```
W1BRI>APW261,WA1PLE-15,AB1OC-10,WIDE2
AE1P>TRUT6U,N1NCI-3*,WIDE2-1:'d+)l  -
AE1P>TRUT6U,N1NCI-3,W1MHL,WA1PLE-15*:
AE1P>TRUT6U,N1NCI-3,AB1OC-10,WIDE2*:'
UNCAN>APN383:;147.135NH*111111z4258.9
W1XM-15>APOT30:!4221.62N/07105.36Wr 0
N3OBQ>APRSWX,N3XJT-1,WIDE1,KB1AEV-15,
K1DF-7>APNU19,UNCAN,WIDE1*:!4309.95NS
K1UI-1>N1RCW-2,K1CKK-2,W1MHL,W1MRA,WA
K1UI-1>N1RCW-2,K1CKK-2,W1MHL,N1EDF-1,
WX1PBD>APU25N,WIDE2-2:@031154z4232.32
W1BRI>APW261,WA1PLE-15,AB1OC-10,WIDE2
KB1ZXL-1>T2QW1R,W1MHL*,WIDE2-1:`c0km6
KB1ZXL-1>T2QW1R,N1EDF-1,WIDE1,W1MHL*,
W1OEM>APU25N,KB1AEV-15,N1NCI-3,WIDE2*
AE1P>TRUT6U,N1NCI-3*,WIDE2-1:'d+)l  -
AE1P>TRUT6U,N1NCI-3,W1MHL,WA1PLE-15*:
AE1P>TRUT6U,N1NCI-3,AB1OC-10,WIDE2*:'
UNCAN>APN383:;147.135NH*111111z4258.9
W1MHL>BEACON:!4223.32N/07115.23W#PHG3
W1XM-15>APOT30:!4221.62N/07105.36Wr 0
N3OBQ>APRSWX,N3XJT-1,WIDE1,KB1AEV-15,
K1DF-7>APNU19,UNCAN,WIDE1*:!4309.95NS
K1UI-1>N1RCW-2,K1CKK-2,W1MHL,W1MRA,WA
K1UI-1>N1RCW-2,K1CKK-2,W1MHL,N1EDF-1,
WX1PBD>APU25N,WIDE2-2:@031154z4232.32
WX1PBD>APU25N,AB1OC-10*,WIDE2-1:@0311
W1BRI>APW261,WA1PLE-15,AB1OC-10,WIDE2
KB1ZXL-1>T2QW1R,W1MHL*,WIDE2-1:`c0km6
KB1ZXL-1>T2QW1R,N1EDF-1,WIDE1,W1MHL*,
W1OEM>APU25N,KB1AEV-15,N1NCI-3,WIDE2*
AE1P>TRUT6U,N1NCI-3*,WIDE2-1:'d+)l  -
AE1P>TRUT6U,N1NCI-3,W1MHL,WA1PLE-15*:
AE1P>TRUT6U,N1NCI-3,AB1OC-10,WIDE2*:'
UNCAN>APN383:;147.135NH*111111z4258.9
W1MHL>BEACON:!4223.32N/07115.23W#PHG3
W1XM-15>APOT30:!4221.62N/07105.36Wr 0
N3OBQ>APRSWX,N3XJT-1,WIDE1,KB1AEV-15,
N3OBQ>APRSWX,N3XJT-1,WIDE1,NORFCT,N1N
K1DF-7>APNU19,UNCAN,WIDE1*:!4309.95NS
K1UI-1>N1RCW-2,K1CKK-2,W1MHL,W1MRA,WA
K1UI-1>N1RCW-2,K1CKK-2,W1MHL,N1EDF-1,
WX1PBD>APU25N,WIDE2-2:@031154z4232.32

Totals after 1404 minutes, AGWPE 12551, UZ7HO 18797, DireWolf 18874
```

10.2.5 Summary

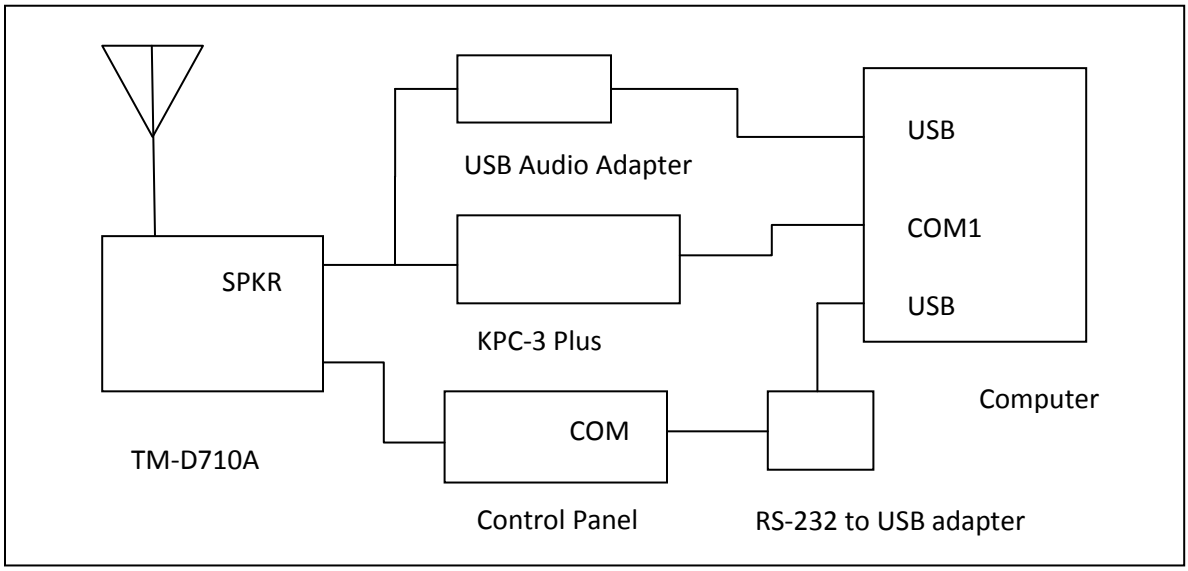
The UZ7HO SoundModem and Dire Wolf decode nearly the same number of packets, differing by less than a half of a percent. There are cases where each is successful while the other is not. AGWPE missed 33% of the packets decoded by the others. Your mileage may vary.

10.3 1200 Baud hardware TNC comparison

Here we compare 1200 baud decoder performance against two popular hardware based solutions. For this experiment we need:

- A cheap USB Audio Adapter (http://www.amazon.com/gp/product/B001MSS6CS)
- Kantronics KPC-3 Plus
- Kenwood TM-D710A
- Serial communication cable for D710A (http://www.amazon.com/gp/product/B000068OER is a lower cost alternative to the official Kenwood PG-5G) – connect to COM port on control panel.
- Audio Y cable, RS232-cables.

Wire up everything as shown below.



10.3.1 Prepare KPC-3 Plus

Using some sort of terminal emulator application, connect to /dev/ttyS0. Disable any sort of digipeater settings or beaconing (DIGIPEAT, UITRACE, UIDIGI, UIFLOOD, BEACON, BLT) so it is not distracted by trying to transmit. Beacons also show up like monitored transmissions. Enable monitoring:

```
MONITOR ON
```

You should see received packets being displayed. Exit from the terminal application.

10.3.2 Prepare D710A

Use the TNC button on the control panel to select “PACKET12” (not APRS) mode. Enable the COM port with menu 604.

Using some sort of terminal emulator application, connect to /dev/ttyUSB0. Disable any sort of digipeater settings or beaconing so it is not distracted by trying to transmit. Enable monitoring:

```
MONITOR ON
```

You should see received packets being displayed. Exit from the terminal application.

10.3.3 Prepare Dire Wolf

Be sure to use Dire Wolf version 0.9 or later. In this test we are using Linux and an external USB Audio Adapter. The configuration file was modified to contain these.

```
ADEVICE  plughw:CARD=Device,DEV=0
MODEM 1200 1200 2200 abc
FIX_BITS 0
```

The audio device might be different for other people repeating this experiment.

Using the default of “FIX_BITS 1” would be cheating. This is explained in a later section with “One Bad Apple” in the name.

10.3.4 Compare them.

Run the “aclients” application with command line arguments like this

```
aclients  /dev/ttyS0=KPC3+  /dev/ttyUSB0=D710A  8000=DireWolf
```

It starts off looking like this with all receiving the same thing:

```
john@hamshack:~/direwolf-0.9$ ./aclients  /dev/ttyS0=KPC3+  /dev/ttyUSB0=D710A  8000=DireWolf
Client 2 now connected to DireWolf on localhost (127.0.0.1), port 8000
Client 0 now connected to KPC3+ on /dev/ttyS0
Client 1 now connected to D710A on /dev/ttyUSB0
KB1LOY-1>T2QV0X,W1MRA,UNCAN,WIDE2*: <   KB1LOY-1>T2QV0X,W1MRA,UNCAN,WIDE2* <U   KB1LOY-1>T2QV0X,W1MRA,UNCAN,WIDE2*:'c
KB1LOY-1>T2QV0X,W1MRA,N8VIM,WIDE2*: <   KB1LOY-1>T2QV0X,W1MRA,N8VIM,WIDE2* <U   KB1LOY-1>T2QV0X,W1MRA,N8VIM,WIDE2*:'c
KB1LOY-1>T2QV0X,W1MRA,AB1OC-10,WIDE2*   KB1LOY-1>T2QV0X,W1MRA,AB1OC-10,WIDE2*   KB1LOY-1>T2QV0X,W1MRA,AB1OC-10,WIDE2*
KB1LOY-1>T2QV0X,N1OMJ,WIDE1,W1MHL*,WI   KB1LOY-1>T2QV0X,N1OMJ,WIDE1,W1MHL*,WI   KB1LOY-1>T2QV0X,N1OMJ,WIDE1,W1MHL*,WI
KN1Q>TRRS5P,W1MHL*,WIDE2-1: <UI>:`c*~   KN1Q>TRRS5P,W1MHL*,WIDE2-1 <UI>:`c*~l   KN1Q>TRRS5P,W1MHL*,WIDE2-1:`c*~l!2>/"
KN1Q>TRRS5P,W1MHL,UNCAN,WIDE2*: <UI>:   KN1Q>TRRS5P,W1MHL,UNCAN,WIDE2* <UI>:`   KN1Q>TRRS5P,W1MHL,UNCAN,WIDE2*:`c*~l!
KN1Q>TRRS5P,W1XM,WIDE1*,WIDE2-1: <UI>   KN1Q>TRRS5P,W1XM,WIDE1*,WIDE2-1 <UI>:   KN1Q>TRRS5P,W1XM,WIDE1*,WIDE2-1:`c*~l
```

Let it run for most of a day and we find these totals:

```
K1UI-1>N1RCW-2,K1CKK-2,N1RCW-3,W1CLA-   K1UI-1>N1RCW-2,K1CKK-2,N1RCW-3,W1CLA-   K1UI-1>N1RCW-2,K1CKK-2,N1RCW-3,W1CLA-
W1MHL>APN382: <UI>:!4223.32N/07115.23                                          W1MHL>APN382:!4223.32N/07115.23W#PHG3
KN1Q>TRRS5P,W1MHL*,WIDE2-1: <UI>:`c*~                                          KN1Q>TRRS5P,W1MHL*,WIDE2-1:`c*~l!<>/"
KN1Q>TRRS5P,W1MHL,UNCAN,WIDE2*: <UI>:   KN1Q>TRRS5P,W1MHL,UNCAN,WIDE2* <UI>:`   KN1Q>TRRS5P,W1MHL,UNCAN,WIDE2*:`c*~l!
KN1Q>TRRS5P,W1MHL,W1JMC*: <UI>:`c*~l!                                          KN1Q>TRRS5P,W1MHL,W1JMC*:`c*~l!<>/"4r
N8VIM-2>APT312,N8VIM,WIDE1*,WIDE2-1:    N8VIM-2>APT312,N8VIM,WIDE1*,WIDE2-1 <   N8VIM-2>APT312,N8VIM,WIDE1*,WIDE2-1:>
N8VIM-2>APT312,N8VIM,WIDE1,W1JMC*: <U   N8VIM-2>APT312,N8VIM,WIDE1,W1JMC* <UI   N8VIM-2>APT312,N8VIM,WIDE1,W1JMC*:>Tr
K1UI-1>N1RCW-2,K1CKK-2,W1CLA-1*,WIDE2   K1UI-1>N1RCW-2,K1CKK-2,W1CLA-1*,WIDE2   K1UI-1>N1RCW-2,K1CKK-2,W1CLA-1*,WIDE2
K1UI-1>N1RCW-2,K1CKK-2,W1CLA-1,UNCAN,   K1UI-1>N1RCW-2,K1CKK-2,W1CLA-1,UNCAN,   K1UI-1>N1RCW-2,K1CKK-2,W1CLA-1,UNCAN,
                                                                               KC4HAY-14>APT311,W1MHL*,WIDE2-1:/0800
KC4HAY-14>APT311,W1MHL,W1JMC*: <UI>:/   KC4HAY-14>APT311,W1MHL,W1JMC* <UI>:/0   KC4HAY-14>APT311,W1MHL,W1JMC*:/080039
W1AW>APU25N,KB1AEV-15,W1UWS-1,UNCAN,W   W1AW>APU25N,KB1AEV-15,W1UWS-1,UNCAN,W   W1AW>APU25N,KB1AEV-15,W1UWS-1,UNCAN,W
```

```
W4HIX-1>APOT21,KB1POR-2,UNCAN,WIDE2*:   W4HIX-1>APOT21,KB1POR-2,UNCAN,WIDE2*    W4HIX-1>APOT21,KB1POR-2,UNCAN,WIDE2*:
W4HIX-1>APOT21,W1MHL*,WIDE2-1: <UI>:!   W4HIX-1>APOT21,W1MHL*,WIDE2-1 <UI>:!4   W4HIX-1>APOT21,W1MHL*,WIDE2-1:!4240.7
W4HIX-1>APOT21,W1MHL,W1JMC*: <UI>:!42   W4HIX-1>APOT21,W1MHL,W1JMC* <UI>:!424   W4HIX-1>APOT21,W1MHL,W1JMC*:!4240.77N
KN1Q>TRRS5P,W1MHL*,WIDE2-1: <UI>:`c*~   KN1Q>TRRS5P,W1MHL*,WIDE2-1 <UI>:`c*~l   KN1Q>TRRS5P,W1MHL*,WIDE2-1:`c*~l!<>/"
KN1Q>TRRS5P,W1MHL,UNCAN,WIDE2*: <UI>:   KN1Q>TRRS5P,W1MHL,UNCAN,WIDE2* <UI>:`   KN1Q>TRRS5P,W1MHL,UNCAN,WIDE2*:`c*~l!
KN1Q>TRRS5P,W1MHL,W1JMC*: <UI>:`c*~l!   KN1Q>TRRS5P,W1MHL,W1JMC* <UI>:`c*~l!<   KN1Q>TRRS5P,W1MHL,W1JMC*:`c*~l!<>/"4r
W1TG-1>APU25N,UNCAN*,WIDE2-1: <<UI>>:   W1TG-1>APU25N,UNCAN*,WIDE2-1 <UI C>:=   W1TG-1>APU25N,UNCAN*,WIDE2-1:=4256.20
W1TG-1>APU25N,W1CLA-1,W1MHL*,WIDE2: <   W1TG-1>APU25N,W1CLA-1,W1MHL*,WIDE2 <U   W1TG-1>APU25N,W1CLA-1,W1MHL*,WIDE2:=4
                                        W1AEC-1>BEACON,W1MHL,W1JMC* <UI>:!413   W1AEC-1>BEACON,W1MHL,W1JMC*:!4136.79N
N8VIM>BEACON,WIDE2-2: <UI>:!4240.85N/   N8VIM>BEACON,WIDE2-2 <UI>:!4240.85N/0   N8VIM>BEACON,WIDE2-2:!4240.85N/07133.
N8VIM>APN391,WIDE2-2: <UI>:$ULTW00630   N8VIM>APN391,WIDE2-2 <UI>:$ULTW006300   N8VIM>APN391,WIDE2-2:$ULTW006300C7029
N8VIM>APN391: <UI>:$ULTW006300C7029D6   N8VIM>APN391 <UI>:$ULTW006300C7029D6F   N8VIM>APN391:$ULTW006300C7029D6FA3277
N8VIM>BEACON,W1MHL*,WIDE2-1: <UI>:!42   N8VIM>BEACON,W1MHL*,WIDE2-1 <UI>:!424   N8VIM>BEACON,W1MHL*,WIDE2-1:!4240.85N
N8VIM>APN391,W1MHL*,WIDE2-1: <UI>:$UL   N8VIM>APN391,W1MHL*,WIDE2-1 <UI>:$ULT   N8VIM>APN391,W1MHL*,WIDE2-1:$ULTW0063
N8VIM>BEACON,W1MHL,W1CLA-1,WIDE2*: <U   N8VIM>BEACON,W1MHL,W1CLA-1,WIDE2* <UI   N8VIM>BEACON,W1MHL,W1CLA-1,WIDE2*:!42
                                                                                N8VIM>APN391,W1MHL,W1CLA-1,WIDE2*:$UL
UNCAN>APN383: <UI>:;147.330NH*111111z   UNCAN>APN383 <UI>:;147.330NH*111111z4   UNCAN>APN383:;147.330NH*111111z4305.1

Totals after 1352 minutes, KPC3+ 13418, D710A 11390, DireWolf 15053
```

10.3.5 Summary

Using the largest number as a score of 100%, we find that the KPC-3 Plus gets a score of 90% and the TM D710A gets a score of 76%.

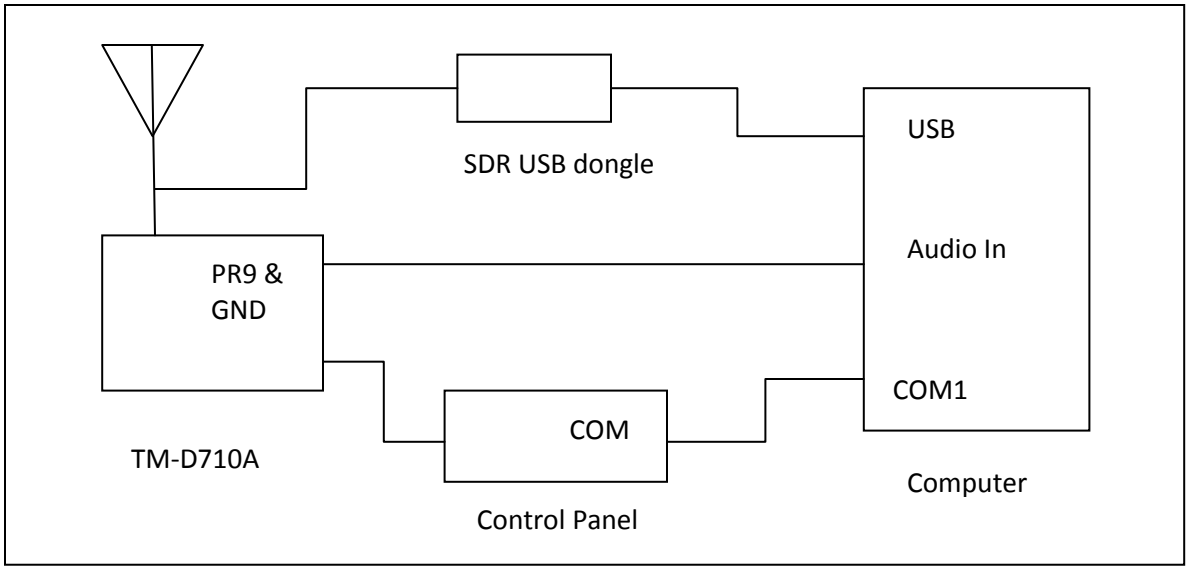
“aclients” is included with the distribution so others can try similar experiments.

10.4 9600 Baud TNC comparison

Here we compare 9600 baud decoder performance. For this experiment we need:

- Kenwood TM-D710A.
- Software Defined Radio USB dongle. (such as http://www.amazon.com/Receiver-RTL2832U-Compatible-Packages-Guaranteed/dp/B009U7WZCA/ref=pd_cp_e_0)
- Serial communication cable for D710A (http://www.amazon.com/gp/product/B000068OER is a lower cost alternative to the official Kenwood PG-5G) – connect to COM port on control panel.
- Radio data cable with 6 pin mini-DIN connector – same type of connector used for PS/2 keyboard and mouse. The data communications cable from the Kenwood PG-5H package does not appear to be suitable. It uses the PR1 pin. We need the PR9 pin.
- Tee adapter to connect single antenna to two receivers.

Wire up everything as shown below.



Connect the PR9 pin from the DATA connector on the transceiver to the audio input on the computer. This has wider bandwidth than the PR1 signal or the speaker output.

10.4.1 Prepare D710A

Use the TNC button on the control panel to select “PACKET96” (not APRS) mode. See later note if you see PACKET12 instead. Enable the COM port with menu 604.

Using some sort of serial port terminal emulator application, such as gtkterm, connect to /dev/ttyS0. Disable any sort of digipeater settings or beaconing so it is not distracted by trying to transmit. We also don’t want to fry the SDR USB dongle! If the control panel shows PACKET12, change the speed by typing this command:

```
HBAUD 9600
```

Enable monitoring:

```
MONITOR ON
```

You should see received packets being displayed. Exit from the terminal application.

10.4.2 Prepare Dire Wolf, first instance

Be sure to use Dire Wolf version 1.0 or later. In this test we are using Linux and an internal soundcard. The configuration file was modified to contain these.

```
ADEVICE  plughw:0,0
MODEM 9600
FIX_BITS 0
```

The audio device might be different for other people repeating this experiment.

When a data speed and no tones are specified, it uses N9GH/G3RUH style encoding.

Using the default of "FIX_BITS 1" would be cheating. This is explained in a later section with "One Bad Apple" in the name.

10.4.3 Prepare Dire Wolf, second instance

This one will be using an SDR dongle rather than a sound card. Make a copy of direwolf.conf and call it direwolf.conf2. Make the following changes:

```
ADEVICE  default
FIX_BITS 0
AGWPORT 8002
KISSPORT 8003
```

Start up the SDR and Dire Wolf in a single command line like this:

```
rtl_fm -f 144.99M -o 4 -s 48000 | direwolf -c direwolf.conf2 -n 1 -r 48000 -B 9600 -
```

Note how we use the command line to specify the audio input device (- at the end) and data rate (-B 9600). Both applications must use 1 audio channel and the same sample rate (48000).

10.4.4 Compare them.

After many days of listening, no indigenous 9600 baud activity was heard so I had to generate my own by walking around the neighborhood with a tracking device.

Run the “aclients” application with command line arguments like this

```
aclients  /dev/ttyS0=D710  8000=DireWolf-soundcard  8002=DireWolf-SDR
```

It starts off looking like this with all receiving the same thing:

```
Client 1 now connected to DireWolf-soundcard on localhost (127.0.0.1), port 8000
Client 2 now connected to DireWolf-SDR on localhost (127.0.0.1), port 8002
Client 0 now connected to D710 on /dev/ttyS0
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=

Totals after 2 minutes, D710 10, DireWolf-soundcard 10, DireWolf-SDR 9
```

To decrease the signal strength we take the transmitter for a little walk around the neighborhood. As the distance increases, the SDR is not doing so well.

```
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=

Totals after 4 minutes, D710 19, DireWolf-soundcard 19, DireWolf-SDR 16

WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=           WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=

Totals after 6 minutes, D710 25, DireWolf-soundcard 25, DireWolf-SDR 17
```

For a while, the other two continue receiving when the SDR gets nothing.

```
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
                                        WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=

Totals after 8 minutes, D710 32, DireWolf-soundcard 33, DireWolf-SDR 17
```

During one part of the journey, nothing is heard.

```
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
                                        WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=

Totals after 10 minutes, D710 34, DireWolf-soundcard 35, DireWolf-SDR 17


Totals after 12 minutes, D710 34, DireWolf-soundcard 35, DireWolf-SDR 17


Totals after 14 minutes, D710 34, DireWolf-soundcard 35, DireWolf-SDR 17


Totals after 16 minutes, D710 34, DireWolf-soundcard 35, DireWolf-SDR 17
```

As we begin to return to home, 2 out of 3 resume receiving the signal.

```
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=

Totals after 18 minutes, D710 38, DireWolf-soundcard 39, DireWolf-SDR 17

WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=

Totals after 20 minutes, D710 43, DireWolf-soundcard 44, DireWolf-SDR 17
```

As we get closer, all three receive the signal.

```
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=          WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=          WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=          WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=          WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=          WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=          WB2OSZ>TRSW1S:'c0nl r[/>"4U}=
WB2OSZ>TRSW1S <UI R>:'c0nl r[/>"4U}=    WB2OSZ>TRSW1S:'c0nl r[/>"4U}=          WB2OSZ>TRSW1S:'c0nl r[/>"4U}=

Totals after 24 minutes, D710 59, DireWolf-soundcard 60, DireWolf-SDR 25
```

This is not the most realistic test scenario – only one transmitter is involved - but it does provide useful data for comparison. Why is the tracker location not changing? I think it is because I put the HT in my pocket sideways and the GPS receiver was looking sideways rather than up to the sky.

10.4.5 Summary

The software defined radio approach had rather disappointing results. Did you expect miracles for $20?

There is potential for improvement by:

- Finding better configuration options.
- Using higher quality hardware such as the FUNcube Pro dongle.
- Using other SDR software such as gqrx.

Remember this is for 9600 baud operation. Results with 1200 baud could be much different.

Dire Wolf, using the soundcard, and the internal TNC of the TM-D710A had essentially the same results.

10.5 One Bad Apple Don't Spoil the Whole Bunch…

There is an old proverb, "One bad apple spoils the barrel," which applies to AX.25 frames used for APRS and traditional packet radio. Each frame contains a 16 bit frame check sequence (FCS) used for error detection. If any one bit is corrupted along the way, the FCS is wrong and the entire frame is discarded. The Osmond Brothers offered the advice, "Give it one more try before you give up…" That can also apply to AX.25 frames. From my observations, single bit errors are fairly common. Why not give it one more try before giving up?

My original attempt at receiving APRS signals performed the HDLC decoding real time on the bits from the AFSK demodulator. If the FCS was wrong, the frame was discarded. The original bit stream was gone. No second chances.

In version 0.6, the HDLC decoder was rearranged to operate in two different phases. The first phase only looked for the special 01111110 "flag" patterns surrounding the frames. The raw received data was stored in an array of bits without undoing the "bit stuffing" at this time. This stream of bits was then processed in the second phase. This provides an opportunity to give it another try if it didn't go well the first time.

For single bit errors, we can try to invert each of the bits – one at a time! – and recalculate the FCS. My experimentation found this recovered a lot of packets that would normally be discarded. Experimental results are summarized in a table later.

What about two or three adjacent bits getting clobbered along the way? If something is good, then more must be better. Right? The next experiment was to try modifying groups of two or three adjacent bits.

Why stop at modifying only adjacent bits? What about two non-adjacent (or "separated") single bit errors? This also allowed a fair number of additional frames to be decoded but at a much larger cost. The processing time is proportional to the square of the number of bits so it climbs rapidly with larger packets. This often takes several seconds rather than the couple milliseconds for all the others.

There is one little problem with flipping various bits trying to find a valid FCS. We get a lot of false positives on the FCS check and end up with bogus data. Callsigns contain punctuation characters. The information part has unprintable characters.

The 16 bit FCS has 65,536 different possible values. Even if totally random data goes into the checking process, you will end up with a valid FCS one out of every 65,536 times. When you try hundreds or even thousands of bit flipping combinations and process lots of packets, a fair number will just happen to get past the FCS check and produce bad data.

My solution was to run the results through an additional sanity check. A valid AX.25 frame will have:

- An address part that is a multiple of 7 bits.
- Between 2 and 10 addresses.
- Only upper case letters, digits, and space in the addresses.
- For APRS, the information part has only printable ASCII characters or these:

- 0x0a line feed
- 0x0d carriage return
- 0x1c used by MIC-E
- 0x1d used by MIC-E
- 0x1e used by MIC-E
- 0x1f used by MIC-E
- 0x7f used by MIC-E
- 0x80 seen in "{UIV32N}<0x0d><0x9f><0x80>"
- 0x9f seen in "{UIV32N}<0x0d><0x9f><0x80>"
- 0xb0 degree symbol, ISO Latin1
  (Note: UTF-8 uses two byte sequence 0xc2 0xb0.)
- 0xbe invalid MIC-E encoding.
- 0xf8 degree symbol, Microsoft code page 437

After applying this extra step of validity checking, no bad data was ever observed for the single bit fixing case. In very large sample sizes, there were a few cases of bad data getting thru when flipping more than one adjacent bit. Obvious errors are fairly common when flipping two non-adjacent bits.

In this example, the first decoder was able to achieve a valid FCS and plausible contents by flipping two non-adjacent bits. The third decoder received it with a correct CRC. Results were different so the duplicate detection did not combine them.

Digipeater WB6JAR-10 audio level = 23 [TWO_SEP] .__
[0] N6VNI-14>APRS,WB6JAR-10*,WIDE,**QIDE-6**:!3356.05N/11758.**6**1Wk Geo & Kris LaHabra,CA

Digipeater WB6JAR-10 audio level = 23 [NONE] _:|
[0] N6VNI-14>APRS,WB6JAR-10*,WIDE,**WIDE**:!3356.05N/11758.**0**1Wk Geo & Kris LaHabra,CA

In this example, the first and third decoders both found combinations of two bit changes that resulted in a valid FCS and plausible data. The second one does not look right with "/V" in the GPS sentence. The first one might or might not be correct. Checking the GPS checksum is left as an exercise for the reader.

N6QFD-9 audio level = 14 [TWO_SEP] .__
[0] N6QFD-9>GPSTJ,WIDE2-
2:$GPRMC,020114,A,3409.7103**,U**,11804.0209,W,14.6,89.2,23110**5**,13.5,E,A*30<0x0d><0x0a>

N6IFD-9 audio level = 14 [TWO_SEP] __.
[0] N6IFD-9>GPSLJ,WIDE2-
2:$GPRMC,020114,A,3409.7103**/V**,11804.0209,W,14.6,89.2,23110**9**,13.5,E,A*30<0x0d><0x0a>

Most of my earlier testing was done with Track 2 of the WA8LMF TNC Test CD (http://wa8lmf.net/TNCtest/index.htm). With the test CD, I got the following results for Dire Wolf version 0.6. Versions 0.7 and 0.8 had no changes in this area. In version 0.9, we use all 3 decoders running in parallel.

	Version 0.6		Version 0.9	
Bits changed	**Number of packets received**	**Percentage increase**	**Number of packets received**	**Percentage increase**
None	965	- - -	976	- - -
Single	+ 12	1.2	+ 21	2.1
Two adjacent	+ 2	0.2	+ 1	0.1
Three adjacent	+ 0	0	+ 0	0.0
Two separated	+ 12	1.2	+ 24	2.4

For version 0.6, overall, 2.6 % more packets were decoded for a total of 991.

Results were more impressive when listening to local stations live. About 23% additional packets were successfully decoded after flipping some bits and giving them another chance.

	Version 0.6	
Bits changed	**Number of packets received**	**Percentage increase**
None	6998	- - -
Single	+ 962	13.7
Two adjacent	+ 57	0.8
Three adjacent	+ 7	0.1
Two separated (not adjacent)	+ 572	8.2

Why such large disparities in the % increase? What is so much different about the local stations heard vs. the sample on the Test CD? I looked for a pattern in the packets that would normally be rejected but were recovered by flipping a single bit.

It doesn't seem to be correlated with a small number of stations. I tabulated where the signals came from (digipeater heard, not original source station) and they are from all over, not just a few stations. It doesn't seem to be correlated with audio deviation of the transmitted signal. Audio levels varied over a 9 to 1 ratio. A lot of people still don't get the concept of setting a proper transmit audio level.

Is it correlated to the type of system transmitting? Again, there doesn't seem to be a pattern. A wide variety of system types are represented.

For now, the reason is still a mystery but one thing is certain. The Dire Wolf "sound card TNC" now decodes a lot more packets that were formerly missed.

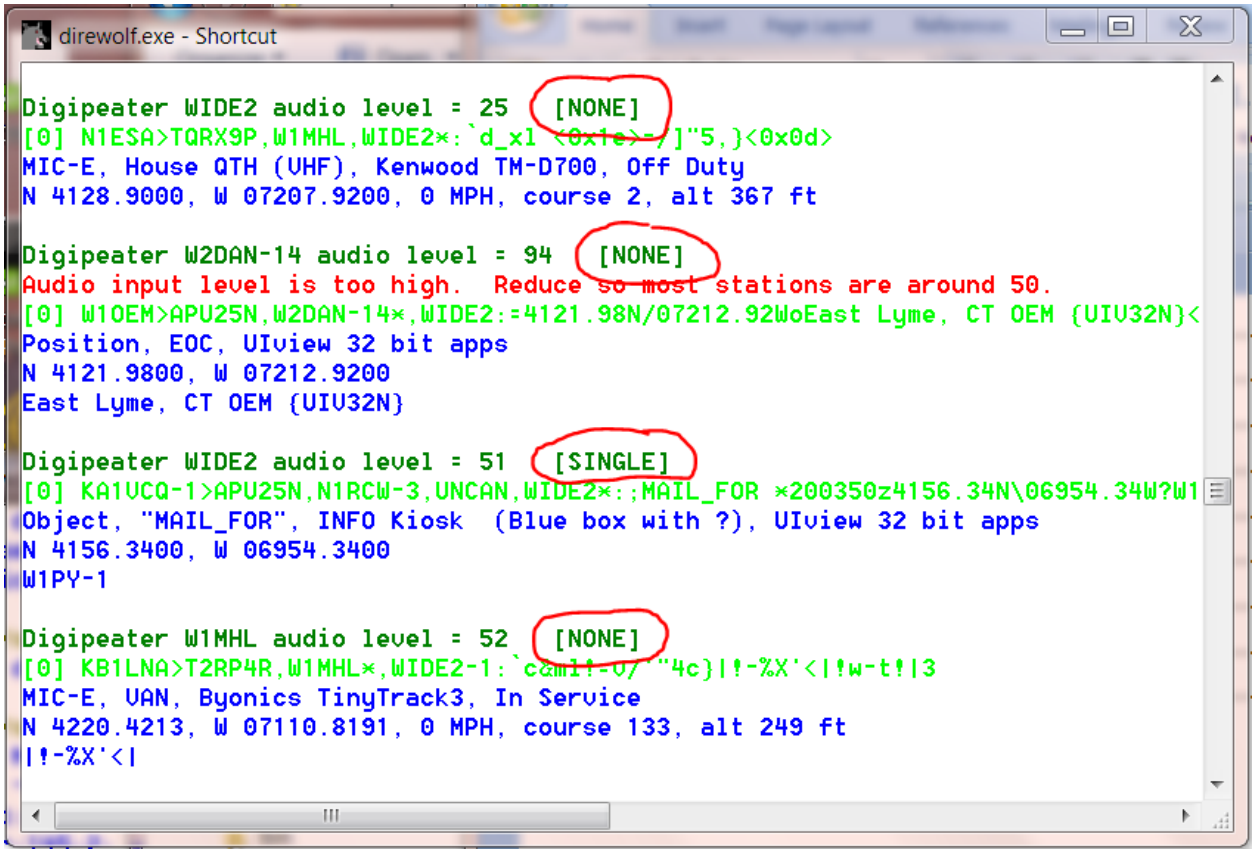
The quick cases, of flipping adjacent bits, are done immediately for quick response. This takes a negligible amount of CPU time.

When this fails, the raw frames are put into a queue and processed by a lower priority background task that examines the much larger number combinations of two non-adjacent single bit errors. This can

take much longer, perhaps seconds, because the number of bit flipping combinations is proportional to the **square** of the frame length.

By default, only single bit correction is enabled. You can experiment with the others with a configuration file setting.

The audio level line contains the number of bits that were changed to get a valid FCS on the frame. In most cases this will be NONE. Here is an example, where a frame that would normally be rejected, was recovered by changing a SINGLE bit.



The Digipeater and IGate functions will process only packets received with a correct CRC to avoid relaying possibly corrupted data. Forwarding possibly corrupted data would be a disservice to the community.

This feature was turned off for all of the performance comparisons with other hardware and software TNCs.

11 UTF-8 characters

11.1 Background

AX.25, like most other computer communication, uses the ASCII character set. ASCII was developed in the 1960's and has a total of 94 printable characters. This didn't keep people happy for very long. As computer usage grew, different vendors starting to add more characters in many different inconsistent ways. Numerous incompatible standards were only partial solutions.

For example, the degree symbol ° was represented by

11111000	in Microsoft code page 437
10110000	in ISO Latin1 (8859-1)

Skipping over several decades of history and countless incompatible standards, UTF-8 is now the preferred way to handle communication for all the additional characters. ASCII is a subset of UTF-8 so they can be used at the same time. Character codes with 0 in the most significant bit are the traditional ASCII characters:

0xxxxxxx	Latin letters, digits, common symbols, and control functions such as new line.

Vast numbers of additional characters are represented by sequences of two or more bytes. The first byte has 11 in the two most significant bits. One or more additional bytes have 10 in the most significant bytes.

11xxxxxx 10xxxxxx …

For example, the degree symbol is now:

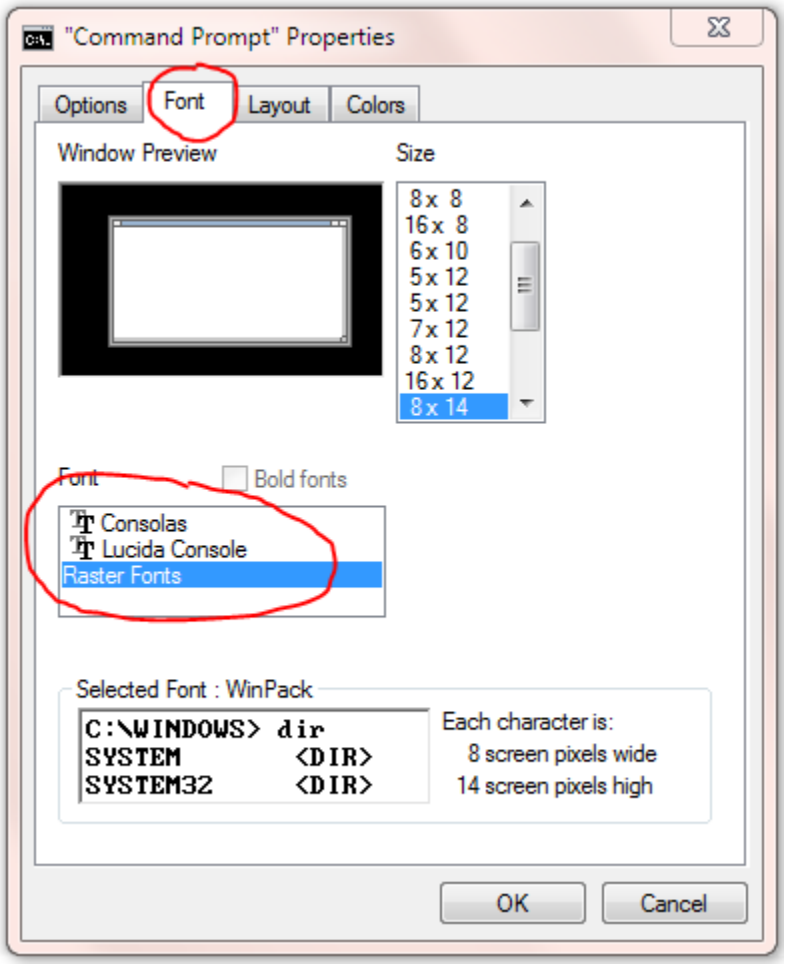
11000010 10110000

When Dire Wolf is used as a TNC for other client applications, UTF-8 is fully supported. Characters from the radio get sent to the application. Characters from the application get sent to the radio.

The only issue arises when trying to display the characters so a person can see them. Dire Wolf does not have a graphical user interface (GUI). It is just a text-based application that depends on some sort of terminal emulator to change internal character codes into viewable images. Some very old terminal emulators don't understand UTF-8. Others might have the capability but need special configuration settings.

11.2 Microsoft Windows

The Microsoft Windows "Command Prompt" has a default of "Raster Fonts." This has a very limited set of characters available. Select one of the other two.



Run direwolf with the upper case -U option to display a test string.

Here are results for the 3 different fonts:

- Consolas

```
UTF-8 test string: mañana º Füße
```

- Lucida Console

```
UTF-8 test string: mañana ° Füße
```

- Raster Fonts

```
UTF-8 test string: ma╠▒ana ┬▒ F╠╝╠fe
```

11.3 Linux

UTF-8 is usually the default on newer systems but there might be cases where you need to set the LANG environment variable.

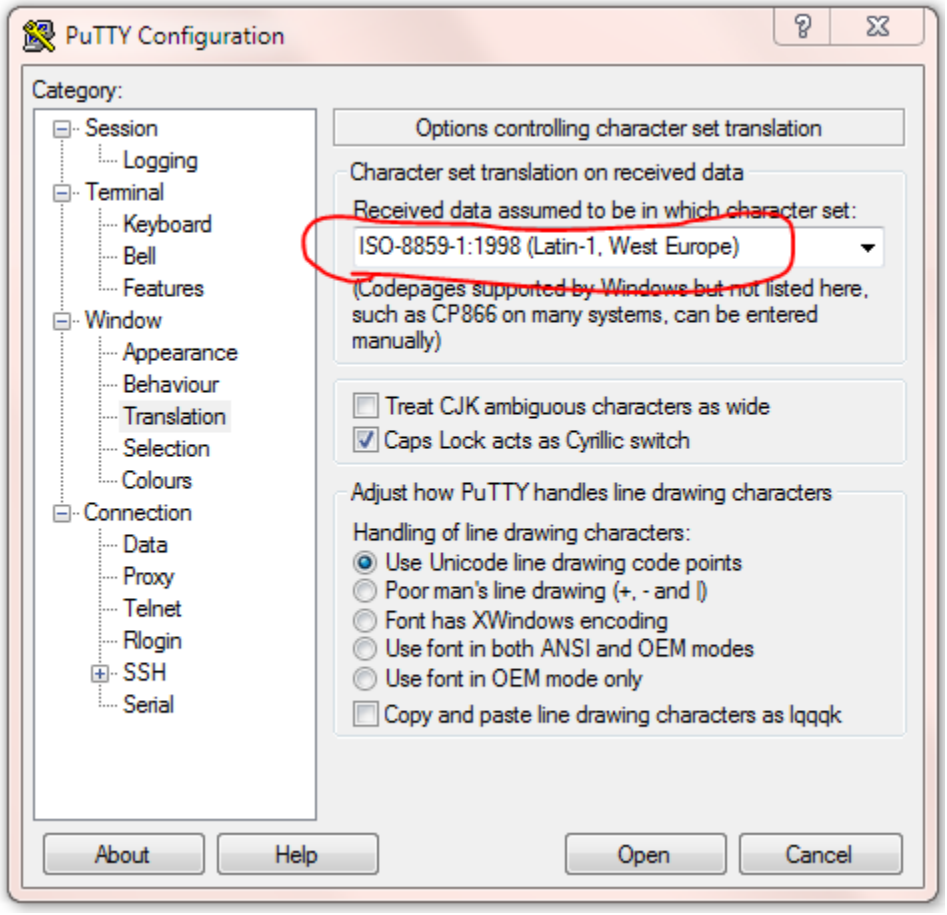
The default on **Raspian** is correct. This is using **LXTerminal**.

```
pi@raspberrypi ~ $ echo $LANG
en_GB.UTF-8
pi@raspberrypi ~ $ direwolf -t 0 -U
Dire Wolf version 1.0

  UTF-8 test string: mañana ° Füße
```

The defaults on **Ubuntu** are also correct. There are reports that a certain command line option is required to make **xterm** process UTF-8 but that doesn't seem to be true anymore.

```
john@hamshack:~$ echo $LANG
en_US.UTF-8
john@hamshack:~$ direwolf -t 0 -U
Dire Wolf version 1.0

  UTF-8 test string: mañana ° Füße
```

If using **PuTTY** to access a remote Linux system, be sure to change the character set to UTF-8.



If PuTTY is using ISO Latin-1, it will look like this:

```
john@hamshack:~$ echo $LANG
en_US.UTF-8
john@hamshack:~$ direwolf -t 0 -U
Dire Wolf version 1.0

  UTF-8 test string: maÃ±ana Â° FÃ¼Ãe
```

Linux has many flavors and an overabundance of terminal emulators so we can't cover all the possibilities here. Google for something like linux terminal utf-8 for more help.

11.4 Debugging

The "-d u" command line option turns on debugging for messages containing non-ASCII characters.

After the normal monitor format, just the information part of the packet is repeated. Any non-ASCII characters are displayed in hexadecimal so you can take a closer look at the bytes in the packet.

Here we see where the character string “ελληνικά” has been replaced by the numerical values of the bytes: ce b5 ce bb ce bb ce b7 ce bd ce b9 ce ba ce ac.

```
WB2OSZ audio level = 49    [NONE]
[0] WB2OSZ>APDW10:!4237.14N/07120.83W-It's all ελληνικά to me.
!4237.14N/07120.83W-It's all <0xce><0xb5><0xce><0xbb><0xce><0xbb><0xce><0xb7><0x
ce><0xbd><0xce><0xb9><0xce><0xba><0xce><0xac> to me.
Position, House, DireWolf, WB2OSZ
N 42° 37.1400, W 071° 20.8300
It's all ελληνικά to me.
```

This extra line appears only when non-ASCII characters are present.

11.5 Configuration File

To be continued…

12 Feedback

Send your feedback to wb2osz *at* Comcast *dot* net. Be sure to mention the version number and whether you are using Windows or Linux.