

Dire Wolf User Guide

**Decoded
Information from
Radio
Emissions for**

**Windows
Or
Linux
Fans**

Version 1.3 -- February 2016

Contents

1	Introduction	1
2	Features	2
3	Connection to Radio.....	3
3.1	q	3
3.2	For Best Results.....	4
4	Installation & Operation Microsoft Windows XP or later	6
4.1	Run Dire Wolf	7
4.2	Select better font	8
4.3	AGW TCPIP socket interface	9
4.3.1	APRSISCE/32.....	9
4.3.2	Ui-View	9
4.3.3	YAAC (Yet Another APRS Client).....	10
4.3.4	SARTrack	10
4.4	Kiss TNC emulation serial port	10
4.4.1	APRSISCE/32.....	11
4.4.2	UI-View32.....	11
4.4.3	YAAC (Yet Another APRS Client).....	11
4.5	Kiss TNC emulation network	11
4.5.1	APRSISCE/32.....	12
5	Installation & Operation Linux	13
5.1	Download source code	13
5.1.1	Download with web browser.....	13
5.1.2	Using git clone.....	13
5.2	Build & Install	14
5.3	Select UTF-8 character set	17
5.4	Run Dire Wolf	18
5.5	AGW TCPIP socket interface	18
5.5.1	Xastir	18
5.6	Kiss TNC emulation serial port	18
5.6.1	Xastir	19
5.6.2	Linux AX25.....	19
5.6.2.1	Troubleshooting kissattach failure.....	21
5.6.2.2	First Work-around.....	21

5.6.2.3	Second Work-around	21
5.6.2.4	Unexpected transmissions	22
6	Macintosh OS X	23
6.1	Install Xcode/Command line tools	23
6.2	Install Macports	23
6.3	Install Support tools and PortAudio Library	24
6.4	Compiling Direwolf	24
6.5	Read the instructions to configure direwolf.conf file.	24
6.6	Running direwolf	25
6.7	Read the rest of the User Guide.	26
6.8	In case of difficulties	26
7	Basic Operation	27
7.1	Start up configuration information	27
7.2	Information for receiving and transmitting	28
7.3	Periodic audio device statistics	33
8	Data Rates	35
8.1	Bits per Second (bps) vs. Baud	35
8.2	1200 bps	35
8.3	300 bps	35
8.4	9600 bps	36
8.5	2400 bps	37
8.6	4800 bps	38
9	Configuration File & command line options	39
9.1	Audio Device	39
9.1.1	Audio Device Selection All Platforms	41
9.1.2	Audio Device selection - Windows	41
9.1.3	Audio Device selection Linux ALSA	42
9.1.4	Audio Device selection Mac OS X	44
9.1.5	Audio Device properties	45
9.1.6	Use with Software Defined Radios	45
9.1.6.1	gqrx	45
9.1.6.2	rtl_fm	46
9.1.6.3	SDR#	47
9.1.6.4	SDR Troubleshooting	50

9.2	Radio channel configuration	50
9.2.1	Radio channel MYCALL.....	51
9.2.2	Radio channel - Modem configuration , general form	51
9.2.3	Radio channel - Modem configuration for 1200 baud.....	52
9.2.4	Radio channel - Modem configuration for 300 baud HF	52
9.2.5	Radio channel - Modem configuration for 9600 baud.....	54
9.2.6	Radio Channel - Allow frames with bad CRC.....	54
9.2.7	Radio channel DTMF Decoder.....	55
9.2.8	Radio Channel Push to Talk (PTT).....	56
9.2.8.1	PTT with serial port RTS or DTR	56
9.2.8.2	PTT with General Purpose I/O (GPIO)	57
9.2.8.3	PTT with Parallel Printer Port.....	57
9.2.8.4	PTT using hamlib	57
9.2.8.4.1	Hamlib PTT Example 1: Use RTS line of serial port.	58
9.2.8.4.2	Hamlib PTT Example 2: Use GPIO of USB audio adapter. (e.g. DMK URI).....	59
9.2.9	Radio Channel Data Carrier Detect (DCD)	61
9.2.10	Radio Channel Transmit Inhibit Input	61
9.2.11	Radio Channel Transmit timing.....	62
9.2.11.1	Should I use wired PTT or VOX?	63
9.2.11.2	Frame Priority and KISS Protocol	64
9.3	Logging of received packets	65
9.4	Client application interface.....	65
9.4.1	AGWPE network protocol	65
9.4.2	Network KISS.....	66
9.4.3	Serial port KISS - Windows	66
9.4.4	Serial port KISS - Linux.....	66
9.5	APRS Digipeater operation.....	67
9.5.1	Digipeater - Configuration Details	67
9.5.2	Digipeater - Typical configuration.....	68
9.5.3	Digipeater example 2 routing between two states.....	69
9.5.4	Digipeater algorithm	69
9.5.5	Digipeater - Compared to other implementations	70
9.5.6	Preemptive Digipeating.....	71
9.5.7	The Ultimate APRS Digipeater	72

9.6	Packet Filtering.....	73
9.6.1	Logical Operators	73
9.6.2	Filter Specifications	73
9.6.2.1	Wildcarding	74
9.6.2.2	Range Filter	74
9.6.2.3	Budlist Filter	74
9.6.2.4	Object Filter.....	75
9.6.2.5	Type Filter	75
9.6.2.6	Symbol Filter	75
9.6.2.7	Digipeater Filter	75
9.6.2.8	Via digipeater unused Filter	76
9.6.2.9	Group Message Filter	76
9.6.2.10	Unproto Filter.....	76
9.6.3	SATgate example.....	77
9.7	GPS Interface.....	78
9.7.1	Direct connect to GPS receiver	78
9.7.2	GPSD Server	78
9.8	Beaconing.....	79
9.8.1	Position & Object Beacons.....	79
9.8.2	Custom Beacon	82
9.8.3	Tracker Beacon.....	83
9.8.4	SmartBeaconing™	84
9.9	Internet Gateway (IGate)	85
9.9.1	IGate - Select server and log in	85
9.9.2	IGate Configure transmit.....	85
9.9.3	IGate Sending directly to server.....	86
9.9.4	IGate Client-side filtering	86
9.9.5	SATgate mode	87
9.9.6	IGate Debugging Options	87
9.10	APRStt Gateway	89
9.11	Transmitting Speech	90
9.11.1	Install Text-to-Speech Software.....	90
9.11.2	Configuration	90
9.11.3	Sample Application: ttcac.....	91

9.12	Transmitting Morse Code	92
9.13	Logging	92
9.13.1	Conversion to GPX format	93
9.14	Command Line Options.....	95
10	Advanced Topics - Windows	97
10.1	Install com0com (optional)	97
10.2	Build Dire Wolf from source on Windows (optional).....	100
11	Receive Performance	101
11.1	WA8LMF TNC Test CD	101
11.2	Evolution	101
11.3	1200 Baud hardware TNC comparison	103
11.3.1	Prepare KPC-3 Plus.....	103
11.3.2	Prepare D710A	104
11.3.3	Prepare Dire Wolf	104
11.3.4	Compare them.	105
11.3.5	Summary	106
11.4	9600 Baud TNC comparison.....	107
	Prepare D710A.....	107
11.4.1	Prepare Dire Wolf, first instance.....	108
11.4.2	Prepare Dire Wolf, second instance.....	108
11.4.3	Compare them.	109
11.4.4	Results.....	111
11.5	qq q ' [q]	113
12	UTF-8 characters	118
12.1	Background	118
12.2	Microsoft Windows.....	118
12.3	Linux	120
12.4	Debugging	121
12.5	Configuration File.....	122
13	Other Included Applications	123
13.1	aclients Test program for side-by-side TNC performance comparison	123
13.2	atest - Decode AX.25 frames from an audio file	123
13.3	decode_aprs - Convert APRS raw data to human readable form.....	123
13.4	gen_packets - Generate audio file for AX.25 frames	124
13.5	ll2utm, utm2ll Convert between Latitude/Longitude & UTM Coordinates	124

1 Introduction

In the early days of Amateur Packet Radio, it was necessary to use a [] with specialized hardware. Those days are gone. You can now get better results at lower cost by q nd using software to decode the signals.

Dire Wolf is a software "soundcard" modem/TNC and [APRS](#) encoder/decoder. It can be used stand-alone to observe APRS traffic, as a q [digipeater], [APRStt](#) gateway, or Internet Gateway (IGate). It can also be used as a virtual TNC for other applications such as [APRSIS32](#), [UI-View32](#), [Xastir](#), [APRS-TW](#), [YAAC](#), [UISS](#), [Linux AX25](#), [SARTrack](#), [RMS Express](#), and many others. Both KISS and AGWPE network protocols are supported for use by applications.

Software and documentation can be found here:

Main page -- Scroll down to README section - <https://github.com/wb2osz/direwolf/>

Releases -- <https://github.com/wb2osz/direwolf/releases>

Documentation for most recent stable release --
<https://github.com/wb2osz/direwolf/tree/master/doc>

Documentation for most recent (unstable) development version --
<https://github.com/wb2osz/direwolf/tree/dev/doc>

Wiki -- <https://github.com/wb2osz/direwolf/wiki>

2 Features

- Software replacement for hardware based Packet TNC.
- 300, 1200, 9600 bits per second data rates.
- Over 1000 error-free frames decoded from WA8LMF TNC Test CD.
- Compatible with Software defined radios such as gqrx, rtl_fm, and SDR#.
- Operation with up to 3 soundcards and 6 radios.
- APRStt gateway using latitude/longitude or UTM/MGRS/USNG coordinates.
- Tool Kit for developing Touch Tone to Speech applications.
- 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.
- Logging of received packets and conversation to GPX format for mapping.
- Beacons of fixed positions or GPS location. (GPS currently on Linux only.)
- Very flexible Digipeating including selective routing between channels.
- Packet filtering for digipeater or Internet Gateway.
- Separate raw packet decoder: decode_aprs
- Support for UTF-8 character set.
- Runs in three different environments:
 - Microsoft Windows XP or later. Pentium 3 or equivalent or later.
 - Linux, regular PC or single board computers such as Raspberry Pi.
 - Macintosh OS X.

See the **CHANGES.md** file for revision history.

* APRS is a registered trademark of APRS Software and Bob Bruninga, WB4APR.
SmartBeaconing™ is a trademark of HamHUD.net.

3 Connection to Radio

Receive Audio In:

For receiving all you need to do is connect your receiver speaker to _____ 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 _____

Transmit Audio Out:

If you want to transmit, you will need to get audio from the computer to the microphone input of your transceiver.

PTT signal to activate transmitter:

If you have a serial port (either built-in or a USB to RS232 adapter cable), the RTS or DTR line can be used to activate the transmitter. [_____ opto-isolator.) GPIO pins can be used on suitable Linux systems. Otherwise you will need a VOX circuit with a very short turn off delay.

I highly recommend using some sort of hardware timer to limit transmission time. Without this, you might end up with your transmitter stuck on for a very long time. Alternatively some radios have a configurable transmit timeout setting to limit transmission time.

Others have documented this type of interface extensively _____ q _____ any homebrew plans and commercial products are available. A few random examples:

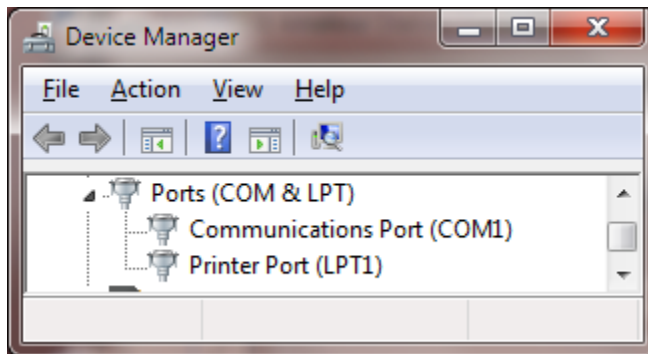
- <http://www.ebay.com/itm/EASY-DIGI-USB-Sound-Card-Interface-NO-MORE-USB-RS232-ADAPTERS-/221668996763>
- <https://sites.google.com/site/kh6tyinterface/>
- <http://www.qsl.net/wm2u/interface.html>
- <http://zs1i.blogspot.com/2010/02/zs1i-soundcard-interface-ii-project.html>
- <http://www.kb3kai.com/tnc/soft-tnc.pdf>
- <http://www.dunmire.org/projects/DigitalCommCenter/soundmodem/mySoundCardInterface.png>

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

3.1 Don't have a serial port?

_____ K _____

I 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.

3.2 For Best Results

For receiving:

- Leave squelch open.

Squelch delay will cut off the start of

- Turn off any battery saver feature.

This feature powers the receiver on and off rapidly to extend battery life. You will miss the beginning of transmissions that come during the power down part of the cycle.

-

This is actually one receiver scanning between two frequencies, not two independent receivers.

For transmitting:

- Set proper transmit audio level.

: Too high will cause distortion and make decoding less likely.

need to listen to other signals and set ours around the average of what others are sending.

- Avoid use of VOX built into your transceiver.

This is designed for voice operation and will keep the transmitter on about a half second after the transmit audio has ended. This is much too long. Others will probably start transmitting before you stop.

For an explanation, see the section [: **Transmit Delay**](#).

- If using the Signalink USB, turn the delay down to the minimum (fully counterclockwise).

According to the documentation, this should turn off the transmitter around 15 or 30 milliseconds after the transmit audio has ended.

Mobile Rigs:

The [Signalink USB](#) is designed specifically for connection to an external modem. If available, use this instead of the speaker and microphone connections. This connection has flatter audio response.

The next 3 sections contain information specific to different operating systems. Proceed to the corresponding one for your situation.

- (4) Windows XP or later
- (5) Linux
- (6) Macintosh OS X

4 Installation & Operation – Microsoft Windows XP or later

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

If using OS X, skip section 4 and proceed to section 6.

Download the desired **direwolf-*-win.zip** file from <https://github.com/wb2osz/direwolf/releases>

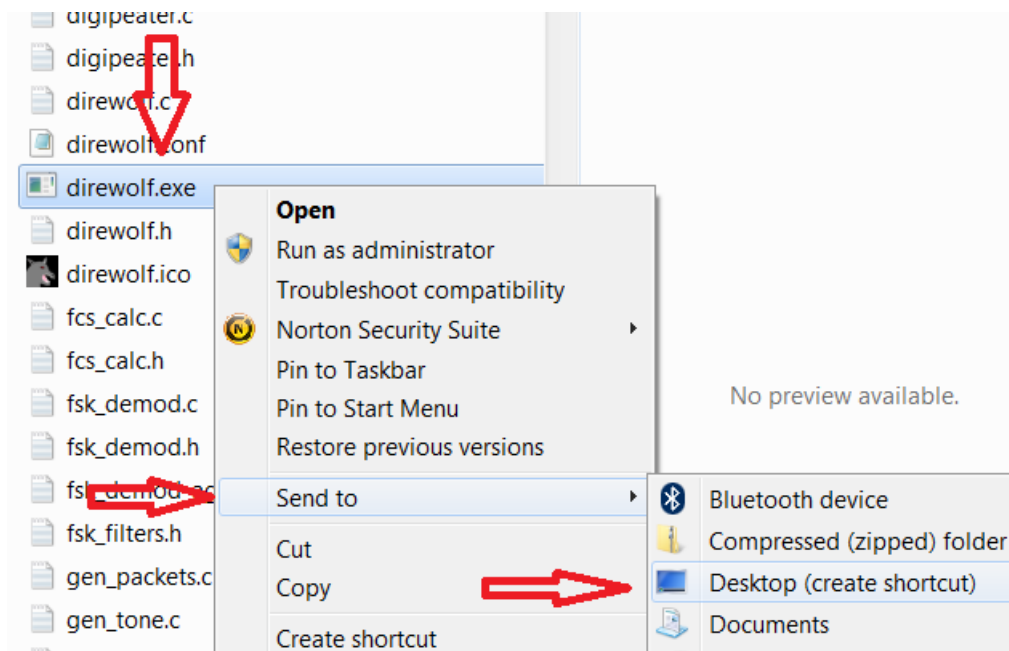
A Pentium 3 processor or equivalent or later is required for the prebuilt version. If you want to use a computer from the previous century, see instructions in Makefile.win.

Put the Dire Wolf distribution file, direwolf-1.3-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

You should end up with a new folder containing:

- **direwolf.exe** -- The application.
- **User-Guide.pdf** -- This document.
- and many

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:

```
~/src/direwolf-1.2
Dire Wolf version 1.2
Available audio input devices for receive (*=selected):
  * 0: Microphone (C-Media USB Headpho (channel 2)
    1: Microphone (Bluetooth SCO Audio
    2: Microphone (Bluetooth AV Audio)
  * 3: Microphone (Realtek High Defini (channels 0 & 1)
Available audio output devices for transmit (*=selected):
  * 0: Speakers (C-Media USB Headphone (channel 2)
    1: Speakers (Bluetooth SCO Audio)
    2: Realtek Digital Output(Optical)
    3: Speakers (Bluetooth AV Audio)
  * 4: Speakers (Realtek High Definiti (channels 0 & 1)
    5: Realtek Digital Output (Realtek
Channel 0: 1200 baud, AFSK 1200 & 2200 Hz, C+, 44100 sample rate, DTMF decoder enabled.
Channel 1: 9600 baud, K9NG/G3RUH, 44100 sample rate x 2.
Channel 2: 300 baud, AFSK 1600 & 1800 Hz, D, 44100 sample rate / 3.
      2.0: D 1510 & 1710
      2.1: D 1540 & 1740
      2.2: D 1570 & 1770
      2.3: D 1600 & 1800
      2.4: D 1630 & 1830
      2.5: D 1660 & 1860
      2.6: D 1690 & 1890
Note: PTT not configured for channel 2. (Ignore this if using VOX.)
Virtual KISS TNC is connected to COM3 side of null modem.
Ready to accept AGW client application 0 on port 8000 ...
Ready to accept KISS client application on port 8001 ...

Digipeater W1MRA audio level = 35(32/16) [NONE] ____|_|_|_|_
[0.5] NR1X-14>APDR12,W1MV-1,WIDE1,W1MRA*,WIDE2:=4206.61N/07046.25Wu091/002/A=000019 aprsdroid
Position, Truck (18 wheeler), APPrDroid replaces old APAND1.
N 42 06.6100, W 070 46.2500, 2 MPH, course 91, alt 19 ft
  aprsdroid with bluetooth tnc-x

Digipeater WIDE1-1 audio level = 35(32/14) [NONE] ____|_|_|_|_
[0.6] N3LEE-7>T2TS5Q,WIDE1-1*,WIDE2-1: <CHD1 <0x1c>[/ "6"}N3LEE Mobile 146.520
"146.520" in comment looks like a frequency in non-standard format.
For most systems to recognize it, use exactly this form "146.520MHz" at beginning of comment.
MIC-E, Human, Unknown manufacturer, In Service
N 42 43.5100, W 071 44.4000, 0 MPH, alt 650 ft, 146.520 MHz
N3LEE Mobile 146.520
```

It starts with some informational messages in black.

- Audio devices being used and their mapping to radio channels. The current version allows up to three audio devices. This allows up to six radio channels when all are operating in stereo.

Next we have some troubleshooting information about the radio channel configuration. Dire Wolf supports the most popular 1200, 9600, and 300 baud standards. For 300 baud HF SSB operation, multiple decoders can be configured to compensate for signals off frequency.

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 and some other useful troubleshooting information.
- The raw data is displayed in green and deciphered information is in blue.
- Sometimes you will see error messages in red when invalid data is received or other problems are noticed.

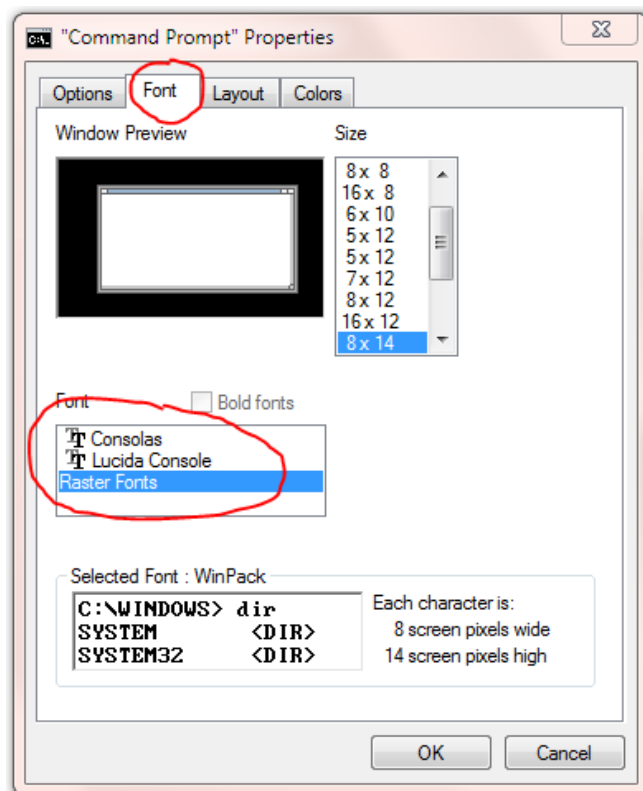
We will learn more about these in later chapters.

The rest of section 4 describes how to use Dire Wolf with other packet radio applications such as APRSISCE/32 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 and 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

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 **UTF-8 Characters**. **Choose one of the others.** For more details, see section called **UTF-8 Characters**.



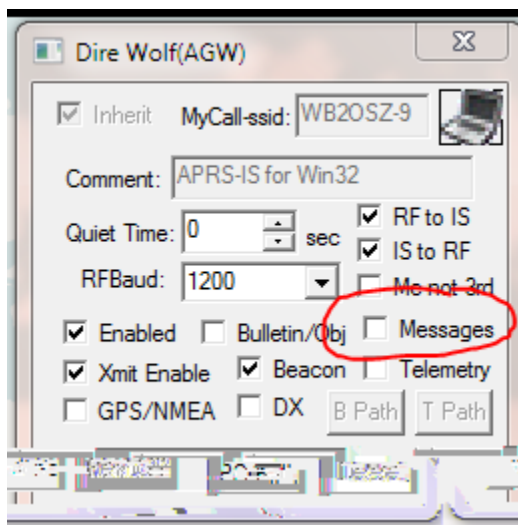
4.3 AGW TCPIP socket interface

3 different client applications can connect at the same time. default port 8000. Up to

4.3.1 APRSISCE/32

1. First, start up Dire Wolf.
2. Run APRSISCE/32.
- 3.
- 4.
- 5.
- 6.
- 7.

option in the TNC port configuration. ' It is necessary to enable the messages



By default it is off. of warning.

4.3.2 Ui-View

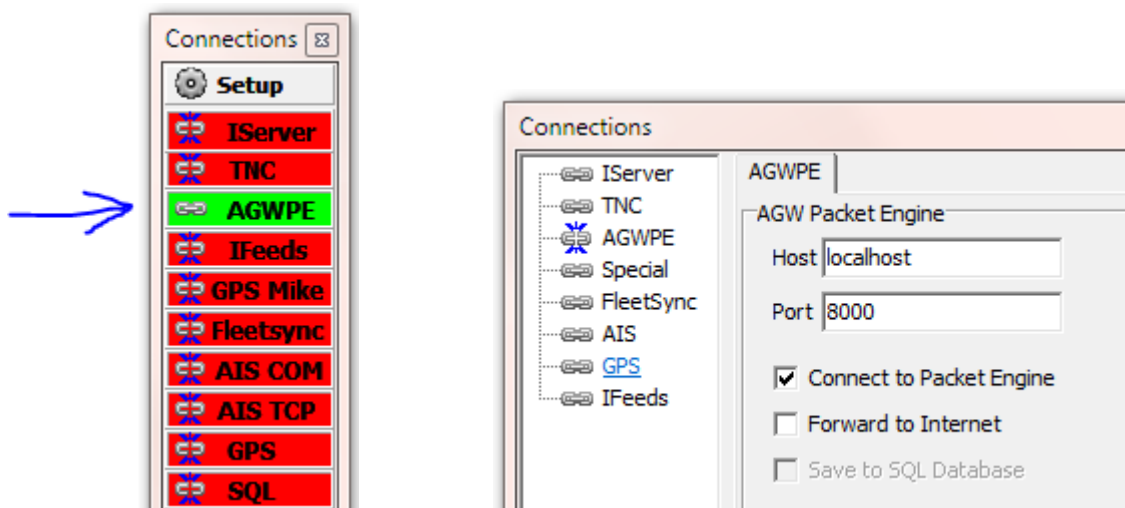
1. First, start up Dire Wolf.
2. Run UI-View32
3. From the Setup menu, pick Comms Setup.
- 4.
5. Take defaults of localhost and 8000. Click on OK.
6. Click on OK for Comms Setup.

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. Q
5. K
6. From the Port type list, choose AGWPE.
7. q
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




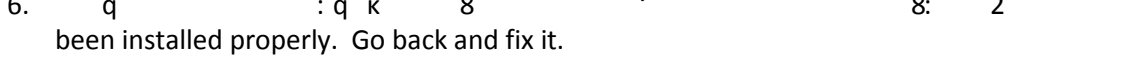
Dire Wolf can act like a packet radio TNC using the KISS protocol by serial port.

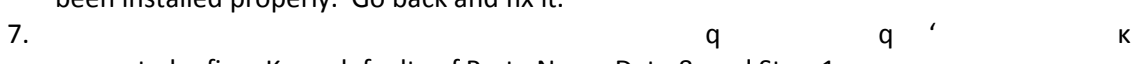
You can use a serial port to emulate a hardware TNC. A cable can be attached to different computer running an application expecting a KISS TNC. More often, you will run both on the same computer and want to connect them together without two physical serial ports and a cable between them.

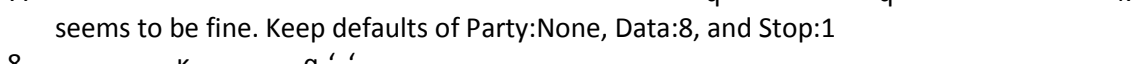
To use this feature, you must install com0com as explained later in the Advanced Topics section. If you followed the instructions, other applications will think they are talking with a TNC on the COM4 serial port.

Here are detailed configuration steps for a couple popular applications.


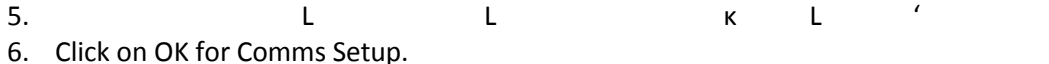
4.4.1 APRSISCE/32

1. First start up Dire Wolf.
2. Run APRSISCE/32.
3. 
4. 
5. 
6. 

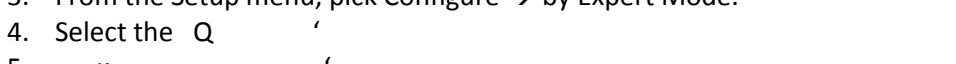
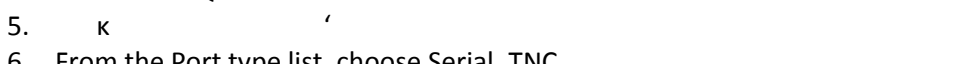
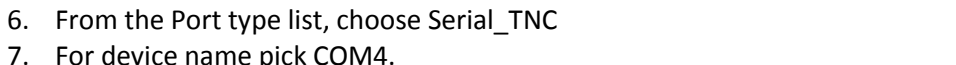
been installed properly. Go back and fix it.
7. 


seems to be fine. Keep defaults of Party:None, Data:8, and Stop:1
8. 

4.4.2 UI-View32

1. First, start up Dire Wolf.
2. Run UI-View32
3. From the Setup menu, pick Comms Setup.
4. 
5. 
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 Q 
5. 
6. From the Port type list, choose Serial_TNC
7. For device name pick COM4.
8. 

qq 
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 APRSISCE/32

1. First start up Dire Wolf.
2. Run APRSISCE/32.
3. : q κ q q ' '
4. q L ' κ '
5. κ : Q QQ '
6. q 223'
7. κ q ' '

Skip sections 5 (Linux) and 6 (OS X) and proceed to section 7 for Basic Operation.

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**.

The ALSA sound system is used for Linux. If you have some other Unix-like operating system that does not have ALSA, you will need to install it. 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. If you are using the Raspberry Pi, you will need to install the Raspberry Pi firmware and the Raspberry Pi kernel.

5.1 Download source code

Choose either of these methods, depending on your preference. If this is new to you and you are not familiar with the command line, the first method might be less confusing.

5.1.1 Download with web browser

Go to the releases page, <https://github.com/wb2osz/direwolf/releases>, with your web browser. Choose the desired release, and download the source as either zip or a compressed tar file. They are equivalent. Choose whichever one you pick except in the next step here.

If you picked the zip format, unpack it with a command like this:

If you use the compressed tar format, unpack it like this:

The exact file name will depend on the release version. Adjust your actual command accordingly. In either case, change your current working directory to the source directory. e.g.

Skip section 5.1.2.

5.1.2 Using git clone

Follow these steps to clone the git repository and checkout the desired version.

At this point you should have the most recent stable version. There are times when you might want to get a specific older version. To get a list of them, type:

You should see a list of releases something like this:

To select a specific version, specify the tag like this:

In some cases, you might want the latest (sometimes unstable) development version to test a bug fix or get a preview of a new (possibly incomplete) feature that will be in the next release. In that case, type:

5.2 Build & Install

It might be necessary to install an additional package with this command:

```
Failure to install the libasound2-  
alsa/asoundlib.h: No such file or directory'
```

Optional Step: update tocalls file

The APRS packet destination field is often used to identify the manufacturer/model of the sender. These are not hardcoded into Dire Wolf. Instead they are read from a file called tocalls.txt at application start up time.

The original standard symbols (house, car, etc.) are built in but the "new" symbols, using overlays, are often updated. These are also read from files so you can use the latest versions without updating the rest of the application.

If you want to use the latest versions of these files, instead of the versions bundled in with the software release, type this:

The risk is that new additions to the file could be in some incompatible format and won't be processed correctly.

Optional Step: gpsd support

Dire Wolf can send beacons based on the current location taken from a GPS receiver. When using Linux, the preferred method is to use “gpsd” which allows multiple applications to share a GPS receiver. Install it:

The “make” step should realize whether the necessary files are available and you should see a message looking something like one of these indicating whether gpsd support was built in.

Optional Step: hamlib support

Dire Wolf can use “hamlib” for more flexible PTT control. You can install it from a package or build from source as described here:

http://hamlib.sourceforge.net/manuals/1.2.15/_rdmedevel.html

Boiled down version :

*Edit **Makefile.linux** and look for this section:*

Remove the # from the beginning of the last two lines.

Compile and install the application.

You should now have files in these locations, under /usr/local, owned by root.

/usr/local/bin/direwolf	The main application.
/usr/local/bin/decode_aprs	q http://aprs.fi or http://findu.com
/usr/local/bin/tt2text text2tt ll2utm utm2ll log2gpx gen_packets aclients ttcalc dwspeak.ch	Utilities related to APRStt gateway, UTM coordinates, log file to GPX conversion, test frame generation, TNC comparison, and a Touch Tone to Speech sample application.
/usr/local/bin/telem-balloon.pl telem-bits.pl telem-data91.pl telem-data.pl telem-eqns.pl telem-parm.pl telem-unit.pl telem-volts.py /usr/local/share/doc/direwolf/ APRS-Telemetry-Toolkit.pdf telem-balloon.conf telem-m0xer-3.txt telem-volts.conf	APRS Telemetry Toolkit.
/usr/share/applications/direwolf.desktop	Application definition with icon, command to execute, etc.
/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.
/usr/share/direwolf/symbolsX.txt symbols-new.txt	Descriptions and codes for APRS symbols.
/usr/share/direwolf/dw-icon.png	Icon for the desktop.
/usr/local/share/doc/direwolf/ A-Better-APRS-Packet-Demodulator-Part-1-1200-baud.pdf A-Better-APRS-Packet-Demodulator-Part-2-9600-baud.pdf APRStt-Implementation-Notes.pdf APRStt-interface-for-SARTrack.pdf CHANGES.md LICENSE-dire-wolf.txt LICENSE-other.txt	Various documentation, mostly in PDF form. README.md is an overview.

Raspberry-Pi-APRS.pdf Raspberry-Pi-APRS-Tracker.pdf Raspberry-Pi-SDR-IGate.pdf README.md User-Guide.pdf	
/usr/local/share/doc/direwolf/examples/ direwolf.conf dw-start.sh sdr.conf telem-m0xer-3.txt telem-balloon.conf telem-volts.conf	Sample configuration files and other examples.
/usr/local/man/man1/*	q -line help.

Some of these files might not apply to your system depending on the type of desktop environment.

If this is the first time you are installing Dire Wolf perform this step:

When upgrading from an earlier version, you will probably want to skip this step because it will wipe out your earlier configuration file.

This step should have copied the initial configuration file into your home directory.

~/direwolf.conf	Configuration file. Search order is current working directory then the
-----------------	---

If you are installing from a DEB or RPM package, /usr/bin will probably be used instead of q to your home directory or other desired location.

5.3 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.

```

K      '      K      q  ...

af_ZA.utf8
en_GB.utf8
fr_CH.utf8

```


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

5.4 Run Dire Wolf

R

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 using it with some other application at this time, skip ahead to section 7, Basic Operation.

5.5 AGW TCPIP socket interface

q QQ κ q 222'

5.5.1 Xastir

- 1.
2. Run Xastir from another window.
3. : q κ
4. κ
5. q : q κ κ Q κ
6. Take all the defau κ L
7. κ
8. κ Q κ
9. Watch all the stations appear on the map.

Q q q '47
q protocol. You can overcome this restriction by using the KISS TNC interface.

5.6 Kiss TNC emulation – serial port

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

What is a pseudo terminal? Dire Wolf acts like a traditional TNC speaking the KISS protocol over a serial port. Some packet applications want to talk to a TNC over a serial port. One possible approach would be to have Dire Wolf talk to one serial port and the application would talk to another serial port. The two serial port connectors would be attached to each other with [] that data going out of one would go into the other.

In this case, Dire Wolf creates a pseudo terminal and talks to one end. The other is available for use by an application such as Xastir or kissattach. The visible end will have a device name like `/dev/pts/99`.

annoying if you want a single script to start up Dire Wolf and associated applications that use the serial KISS interface.

5.6.1 Xastir

- ### 5.6.2 Linux AX25

For Red Hat / Fedora / CentOS,

Page
19

This is important and not obvious. → Remove any blank lines from the file. ←

You should see a message something like this:

Received frame queue is out of control. Length=' .
Reader thread is probably frozen.
This can be caused by using a pseudo terminal (direwolf -p) where another.
application is not reading the frames from the other side.

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:**

You could use something like this instead if you want to start up multiple applications from one script.

After a successful **kissattach**, continue appropriately for your situation. Simple example for testing:

: - κ q qlay the KISS protocol messages. You might see something like this for a ping command to one of the 44.x.x.x addresses:

5.6.2.1 Troubleshooting – kissattach failure

Sometimes kissattach has an issue with the Dire Wolf pseudo terminal. This shows up most often on Raspbian but sometimes occurs with other versions of Linux.

The root cause and a proper solution have not been found yet. For now, two different work-arounds are available.

5.6.2.2 First Work-around

IZ1YPS came up with this interesting work-around.

(1) Start up direwolf with -p option as you normally would.

(2) `q` `q` `[q ']` kissattach, use `/dev/ptmx` instead. Example:

```
sudo /usr/sbin/kissattach /dev/ptmx radio
```

It should respond with something like this:

Remember that last line because it will be used in the final step.

(3) Connect them with mkiss.

```
sudo mkiss /tmp/kisstnc
```

The last command line argument is the result from step 2. If you wanted to script those last two steps, you could do it like this:

```
x=`sudo /usr/sbin/kissattach /dev/ptmx radio 44.56.4.118 | tail -1`
```

5.6.2.3 Second Work-around

Rather than using the pseudo terminal feature of Dire Wolf, use the TCP network KISS port instead.

8 `q` `,` `q` `:` `...`
pseudo terminal for use by other applications.

without -q q ‘ q ...

Now create a two way connection between port 8001 and a new pseudo terminal in a different command window.

Use the result with `kissattach`.

5.6.2.4 Unexpected transmissions

Why might you transmitting apparent trash when no beacons were configured? The issue is that if you enable a TCP/IP address on your Linux `ax?` interface, broadcasting programs like Samba, Avahi (Bonjour), etc. will send their traffic out over RF! The solution here is to either reconfigure those applications to only bind to specific interfaces (not all interfaces) or setup iptables packet filters to intercept that broadcast traffic before it hits the `ax?` interface.

You can find a lot of good information on Linux AX.25 here:

<http://www.trinityos.com/HAM/CentosDigitalModes/hampacketizing-centos.html>

6 Macintosh OS X

A port to the Macintosh was provided by Robert, KK5VD. This is new for version 1.3 and has not been well tested yet.

Requirements for compiling/installing Direwolf on Mac OS X.

- Built/Tested using Mac OS X 10.10 and XCode 6.3 Development/Command line tools.
- Installation of Macports package manager is required as the dependencies within the makefile.macosx are structured for it.

6.1 Install Xcode/Command line tools.

Xcode can be found on Apple's Developer website. You will need an ID and password to gain access.

<https://developer.apple.com/downloads/index.action> will lead you to a login window from your browser.

Obtain a login ID or enter your current one. Once logged in, uncheck all checked boxes on the left with the exception of Development tools.

Locate Xcode and command line tools for your operating system version. Do not select anything that's a beta release.

Download both and install Xcode first followed by the command line tools.

Execute the terminal program located here: /Applications/Utilities/Terminal.app

This program is used to enter command line commands.

After installing, run the following command from the command line to activate the command line tools (MacOSX 10.10 and possibly other versions).

At the prompt enter your password. (You must have admin rights).

6.2 Install Macports

Install macports package manager from this URL: <https://www.macports.org/install.php>

Select and install the one that is relevant to your operating system version.

6.3 Install Support tools and PortAudio Library.

From the command line, enter the following:

- `coreutilis` Includes the program `ginstall`, as Apple's version doesn't work correctly with the makefile provided by Direwolf.
- `portaudio` Portable Audio Interface library for accessing Apple's CoreAudio sound system.
- `+universal` This flag will cause the build system to create both 32bit and 64bit versions of the library.

6.4 Compiling Direwolf.

Obtain the source code by one of the methods described in the Linux section, above.

From the command line.

```
cd <DireWolf Source Code Location>
```

```
-k          means keep going even if errors.
```

If there are no reported errors.

Perform next step only if this is the first time DireWolf is being installed. It will cause an existing customized version of `direwolf.conf` file to be overwritten.

6.5 Read the instructions to configure `direwolf.conf` file.

The configuration file is located in either the current directory path or the HOME directory.

After editing the configuration file **SAVE A COPY TO ANOTHER LOCATION!** Move the edited file to `~/` (home) directory.

The one significant difference from the other operating systems is that the audio device names can contain spaces. If they do, they must be quoted like the example below.

Configuration file format:

<Device Input Name>:<Device Input Number> <Device Output Name>:<Device Output Number>

Example:

You are probably wondering: How do I know what devices are available? A listing of audio devices is presented on start up of Dire Wolf. The remaining configuration options are described in the relevant sections of this User Guide. Where there are Windows / Linux differences use the Linux version.

Example Device listing:

6.6 Running direwolf.

From the command line enter.

Direwolf reads the configure file that is located in the same directory where the program was executed.
i.e. ~/direwolf.conf

There are a number of command line parameter available to the user. These are listed later in this User Guide.

6.7 Read the rest of the User Guide.

This should answer most of your questions.

If something is missing or unclear post a question on one of the discussion groups or contact the author.

6.8 In case of difficulties

If you are not a programmer and/or not familiar with using command line build tools, you have a lot to learn. ☹

The author of Dire Wolf (WB2OSZ) does `q` `q` `y` issues specifically related to this platform. If you are having Mac-specific issues, post your question to one of the discussion groups:

https://groups.yahoo.com/neo/groups/direwolf_packet/info

<https://groups.yahoo.com/neo/groups/linuxham/info>

Or e-mail the person providing the port to this platform:

Robert, `kk5vd(at)yahoo(dot)com`

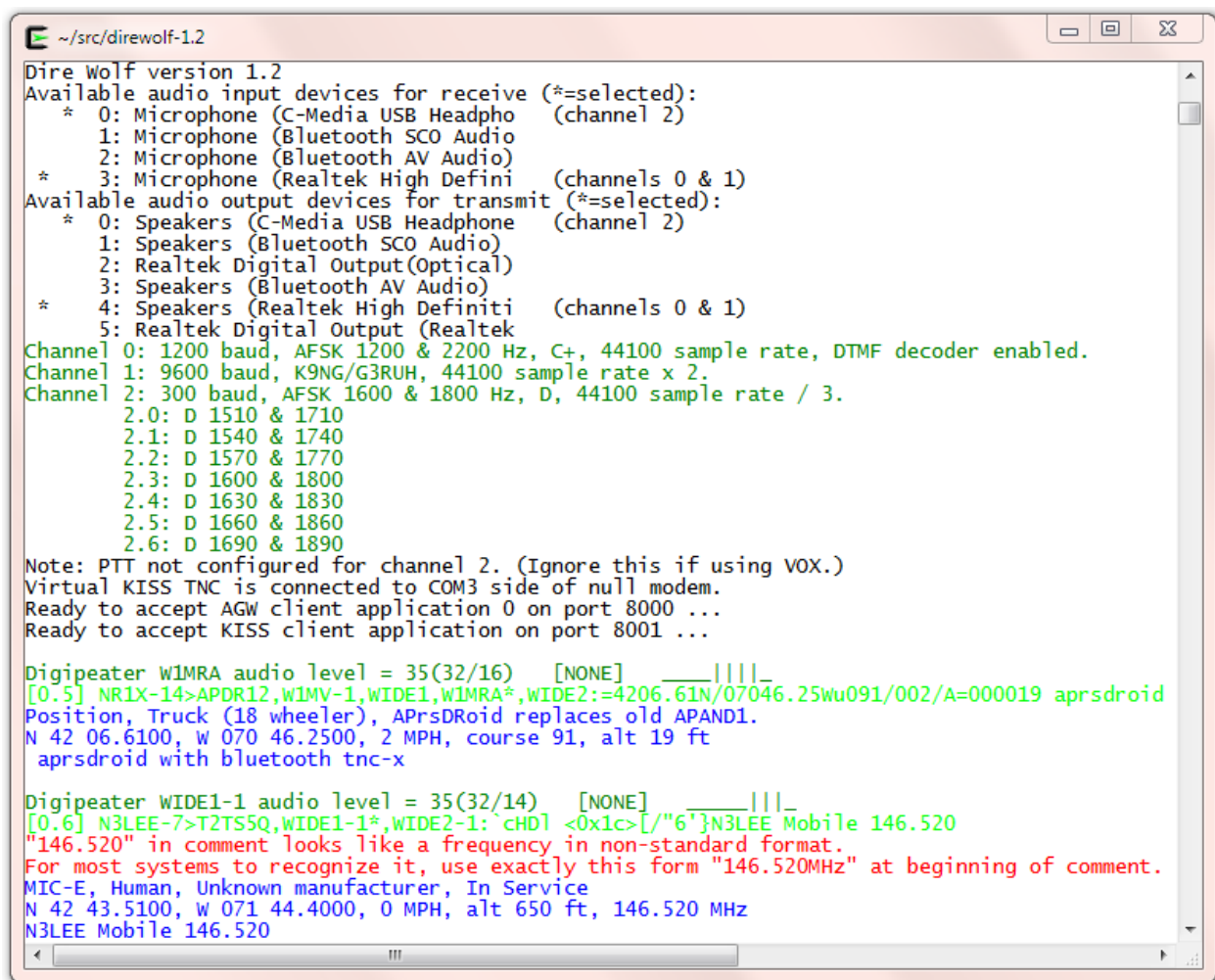
7 Basic Operation

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. Some of these illustrations might be from an earlier version and look slightly different than the current version.

7.1 Start up configuration information

You should see something like this for the Windows version:



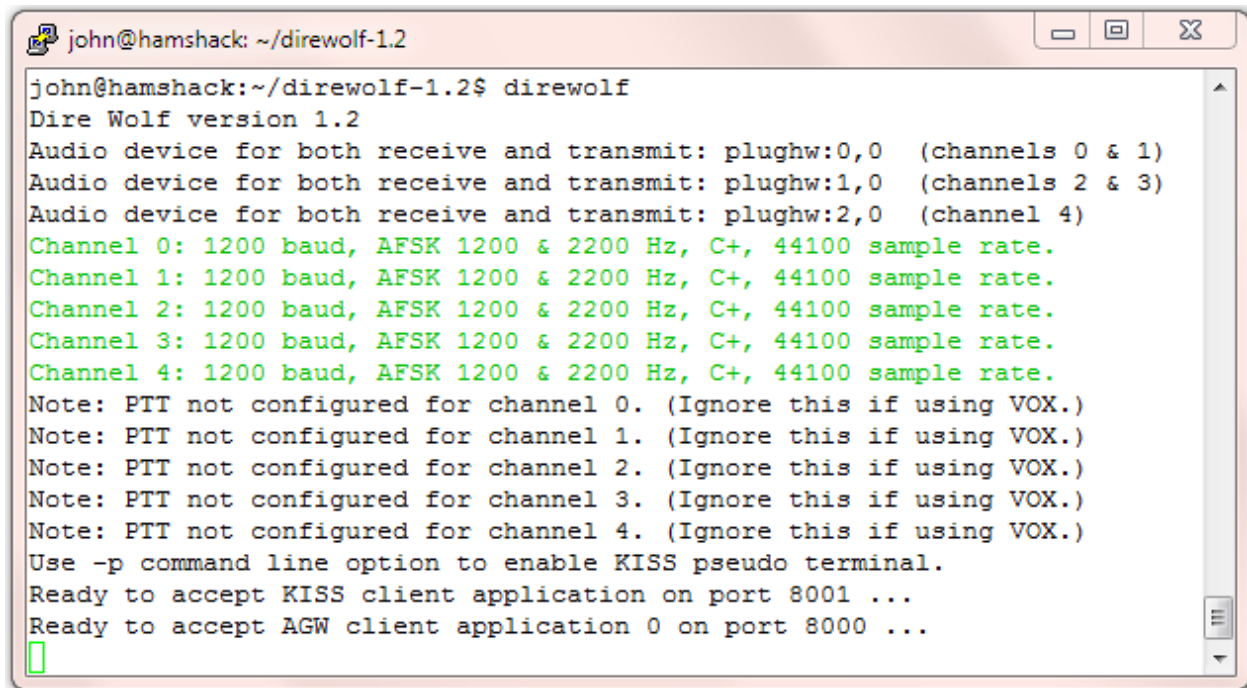
```
~/src/direwolf-1.2
Dire Wolf version 1.2
Available audio input devices for receive (*=selected):
  * 0: Microphone (C-Media USB Headpho (channel 2)
    1: Microphone (Bluetooth SCO Audio
    2: Microphone (Bluetooth AV Audio)
  * 3: Microphone (Realtek High Defini (channels 0 & 1)
Available audio output devices for transmit (*=selected):
  * 0: Speakers (C-Media USB Headphone (channel 2)
    1: Speakers (Bluetooth SCO Audio)
    2: Realtek Digital Output(Optical)
    3: Speakers (Bluetooth AV Audio)
  * 4: Speakers (Realtek High Definiti (channels 0 & 1)
    5: Realtek Digital Output (Realtek
Channel 0: 1200 baud, AFSK 1200 & 2200 Hz, C+, 44100 sample rate, DTMF decoder enabled.
Channel 1: 9600 baud, K9NG/G3RUH, 44100 sample rate x 2.
Channel 2: 300 baud, AFSK 1600 & 1800 Hz, D, 44100 sample rate / 3.
  2.0: D 1510 & 1710
  2.1: D 1540 & 1740
  2.2: D 1570 & 1770
  2.3: D 1600 & 1800
  2.4: D 1630 & 1830
  2.5: D 1660 & 1860
  2.6: D 1690 & 1890
Note: PTT not configured for channel 2. (Ignore this if using VOX.)
Virtual KISS TNC is connected to COM3 side of null modem.
Ready to accept AGW client application 0 on port 8000 ...
Ready to accept KISS client application on port 8001 ...

Digipeater W1MRA audio level = 35(32/16) [NONE] ____|_|_|_
[0.5] NR1X-14>APDR12,W1MV-1,WIDE1,W1MRA*,WIDE2:=4206.61N/07046.25Wu091/002/A=000019 aprsdroid
Position, Truck (18 wheeler), APRsDRoid replaces old APAND1.
N 42 06.6100, W 070 46.2500, 2 MPH, course 91, alt 19 ft
  aprsdroid with bluetooth tnc-x

Digipeater WIDE1-1 audio level = 35(32/14) [NONE] ____|_|_|_
[0.6] N3LEE-7>T2TS5Q,WIDE1-1*,WIDE2-1: CHD1 <0x1c>[/6'}N3LEE Mobile 146.520
"146.520" in comment looks like a frequency in non-standard format.
For most systems to recognize it, use exactly this form "146.520MHz" at beginning of comment.
MIC-E, Human, Unknown manufacturer, In Service
N 42 43.5100, W 071 44.4000, 0 MPH, alt 650 ft, 146.520 MHz
N3LEE Mobile 146.520
```

It starts off listing the available audio devices. In this case, we have a cheap USB Audio adapter and the others are 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:



```
john@hamshack: ~/direwolf-1.2
john@hamshack:~/direwolf-1.2$ direwolf
Dire Wolf version 1.2
Audio device for both receive and transmit: plughw:0,0 (channels 0 & 1)
Audio device for both receive and transmit: plughw:1,0 (channels 2 & 3)
Audio device for both receive and transmit: plughw:2,0 (channel 4)
Channel 0: 1200 baud, AFSK 1200 & 2200 Hz, C+, 44100 sample rate.
Channel 1: 1200 baud, AFSK 1200 & 2200 Hz, C+, 44100 sample rate.
Channel 2: 1200 baud, AFSK 1200 & 2200 Hz, C+, 44100 sample rate.
Channel 3: 1200 baud, AFSK 1200 & 2200 Hz, C+, 44100 sample rate.
Channel 4: 1200 baud, AFSK 1200 & 2200 Hz, C+, 44100 sample rate.
Note: PTT not configured for channel 0. (Ignore this if using VOX.)
Note: PTT not configured for channel 1. (Ignore this if using VOX.)
Note: PTT not configured for channel 2. (Ignore this if using VOX.)
Note: PTT not configured for channel 3. (Ignore this if using VOX.)
Note: PTT not configured for channel 4. (Ignore this if using VOX.)
Use -p command line option to enable KISS pseudo terminal.
Ready to accept KISS client application on port 8001 ...
Ready to accept AGW client application 0 on port 8000 ...
```

It starts with:

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

7.2 Information for receiving and transmitting

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, : ']
- 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

$$q \quad ' \quad 2 \quad : \quad 3 \quad ' \quad q \quad q$$

$$q \quad k \quad ' \quad q \quad q \quad k \quad \text{tion.}$$
- **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.

```

Positionless Weather Report, FIRE TRUCK, Byons WXTrac
wind 4.0 mph, direction 280, gust 4, temperature 51, rain 0.00 in last hour, rain
"tU2k"

Digipeater WIDE2 audio level = 37
[0] K1DES>APRS,KQ1L-5,UNCAN,WIDE2*:!4416.38nn06935.21w<0x0d>
Warning: Lower case n found for latitude hemisphere. Specification requires upper
Warning: Lower case w found for longitude hemisphere. Specification requires upper
Symbol table identifier is not '/' (primary), '\' (alternate), or valid overlay
Symbol code is not a printable character.
Position, --no-symbol-- w/overlay n, Generic, (obsolete. Digis should use APNxxx
N 4416.3800, W 06935.2100

Digipeater W2DAN-14 audio level = 72
[0] KA1SUW-1>T1TY7R,W2DAN-14*,WIDE2-1:`c3#1!t#/"4')=<0x0d>
MIC-E, DIGI (white center), Kenwood TM-D710, In Service
N 4149.7200, W 07123.0700, 0 MPH, course 188, alt 52 ft
[0H] KA1SUW-1>T1TY7R,W2DAN-14,WB2OSZ-5*:`c3#1!t#/"4')=<0x0d>

Digipeater W2DAN-14 audio level = 71
[0] KA1SUW-1>T1TY7R,W2DAN-14,WB2OSZ-5*:`c3#1!t#/"4')=<0x0d>
Digipeater: Drop redundant packet.

```

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

The APRS specification requires upper case letters for the hemisphere. Many systems will also

```

Digipeater N1NCI-3 audio level = 10 [NONE]
[0] N1EOE>APN391,N1NCI-3*,WIDE2-1:!4216.95n/07243.20w#phg6230/ Easthampton MA<0x0d>
Warning: Lower case n found for latitude hemisphere. Specification requires upper case N or S.
Warning: Lower case w found for longitude hemisphere. Specification requires upper case E or W.
Position, DIGI (white center), Kantronics KPC-3 rom versions
N 42 16.9500, W 072 43.2000
phg6230/ Easthampton MA

```

Q q q
temperature information in a specific format.


```

Digipeater W1MHL audio level = 29 [NONE]
[0] WA1JSE-1>APU25N,N1RCW-3,W1MHL*,WIDE2-1::146.955- *172203z4144.03N/07011.00WmK1PBO Rptr 88.5 EL 20024<0x0d>
"88.5" in comment looks like it might be a CTCSS tone in non-standard format.
For most systems to recognize it, use exactly this form "T088" at near beginning of comment, after any frequency.
Object, "146.955-", Mic-E Repeater, Uiview 32 bit apps
N 41 44.0300, W 070 11.0000, 146.955 MHz, PL 88.5
K1PBO Rptr 88.5 EL 20024

```

Q q κ qq

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.

```

~/src/direwolf-0.5
W2DAN-14 audio level = 72
[0] W2DAN-14>APN302,WIDE2-2:!4150.45NL07140.18W#PHG7560/RIAFMRS DIGI - w2dan@arrl
1.net<0x0d>
Position, OVERLAY DIGI (green star) w/overlay L, Kantronics KPC-3 rom versions,
49 W height=320 6dB omni
N 41 50.4500, W 071 40.1800
/RIAFMRS DIGI - w2dan@arrl.net

- w2da [0] W2DAN-14>APN302,W1MHL,W1JMC*:!4150.45NL07140.18W#PHG7560/RIAFMRS DIGI
n@arrl.net<0x0d>
Position, OVERLAY DIGI (green star) w/overlay L, Kantronics KPC-3 rom ver
49 W height=320 6dB omni
N 41 50.4500, W 071 40.1800
/RIAFMRS DIGI - w2dan@arrl.net

- w2da [0] W2DAN-14>APN382,UNCON,WIDE2*:!4150.45NL07140.18W#PHG7560/RIAFMRS DIGI
n@arrl.net<0x0d>
Position, OVERLAY DIGI (green star) w/overlay L, Kantronics KPC-3 rom ver
49 W height=320 6dB omni
N 41 50.4500, W 071 40.1800
/RIAFMRS DIGI - w2dan@arrl.net

```

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:

- 2 [] channel.
- The source station.
- 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.
- q ' are actually receiving.
- Finally the information part of the packet. notice that unprintable characters are represented q 2 3 ' This is the same convention used by <http://aprs.fi>

```

~/src/direwolf-0.5
UNCAN audio level = 44
[01 UNCAN>APN383::147.330NHx111111z4305_16N/07131.39WrT141.r20m.S.Bow<0x0d>
[01 UNCAN>APN383::147.330NHx111111z4305_16N/07131.39WrT141.r20m.S.Bow<0x0d>
Digipeater WIDE2 audio level = 20
[01 KA1CQR>APU25N,KB1AEU-15,N1NCI-3,WIDE2*:=4131.18N107206.13W#PHG51602/W1 Fill
in Norwich CT {UIU32N}<0x0d>
Position, OVERLAY DIGI (green star) w/overlay 1, UIU32N 22 bit apps, 25 W height
=20 6dB omni
N 41 31.1800, W 072 06.1300
2/W1 Fill-in Norwich CT (UIU32N)
Digipeater WIDE2 audio level = 54
[01 N1YG-1>T1SY9P,W2DAN-15,W1MRA,KB1TS0,WIDE2*:'c&<0x7f>1 <0x1c>-/'>
MIC-E, House QTH (UHF), Kenwood TH-D70, In Service
N 41 39.9000, W 071 10.9900, 0 MPH
Digipeater W1JMC audio level = 21
[01 N1YG-1>T1SY9P,W2DAN-15,W1MRA,W1JMC*:'c&<0x7f>1 <0x1c>-/'><0x0d>
MIC-E, House QTH (UHF), Kenwood TH-D70, In Service
N 41 39.9000, W 071 10.9900, 0 MPH

```

Finally we have decoded information in blue.

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

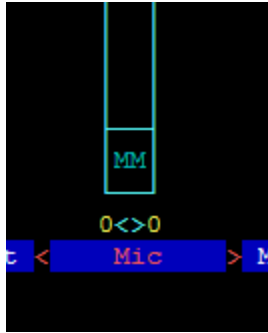
The final line has any comment or weather information.

This can be a useful troubleshooting tool if packets are not being decoded as expected. Is received audio getting to the decoder? Is the audio interface producing the proper sample rate?

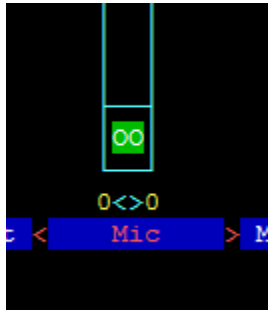
ADEVICE0	means it is the first audio device (sound card). ADEVICE1 for the second, etc.
44.1 k	is the approximate average sample rate during the interval. If this is off significantly, there is something wrong with the audio input system. For example, one time use of a USB hub for an audio adapter caused this to be 42.8 k. Many samples were getting lost.
CH0 90	shows a audio input is working for channel 0. If no frames are being decoded, leave the squelch open and set audio input gain so this is somewhere in the 30 to 150 range.
CH1 0	shows that channel 1 has no audio input. In this case we are running in the audio device in stereo with the right channel disconnected.

If you are getting insufficient audio, check the cabling and the audio input gain.

$$q : q' \quad \text{if} \quad q = q'$$



Select it by using the ← and → keys. Press Q.



Press the ← key.

You want any auto gain control to be off.



Press the ← and → keys to select it and press

8 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.

8.1 Bits per Second (bps) vs. Baud

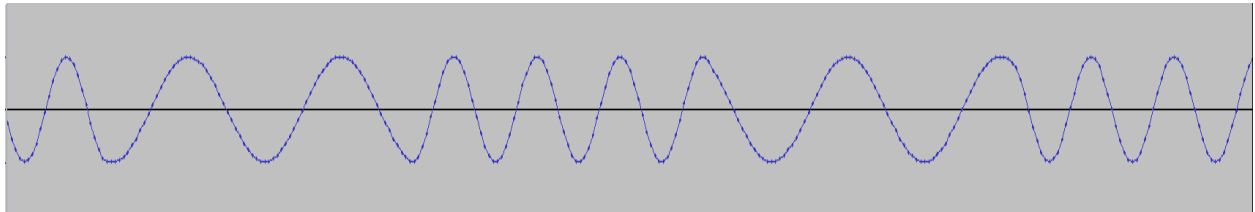
q [q]
same number.

Baud refers to the maximum number of (signal states) per second. With two tone frequency shift keying q 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.

8.2 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

q : K
amplifier passband characteristics so you can simply use the microphone and speaker connections.



8.3 300 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>

q

q'

8.4 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 - κ'
- 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 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/>
https://groups.yahoo.com/neo/groups/direwolf_packet/conversations/topics/768

' 22

8 22

wave if the right combination of bits is present.

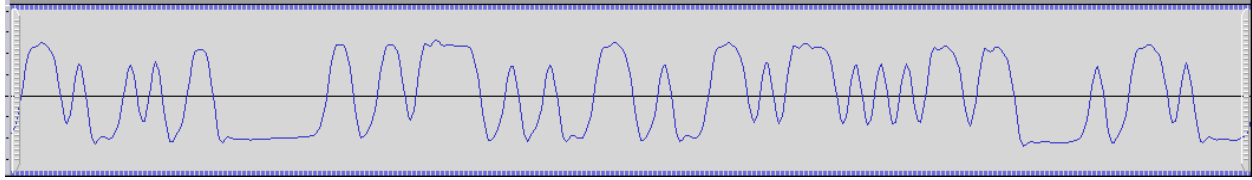
Take a look at the response of a chip often used in the cheap USB Audio adapters:
<http://www.hardwaresecrets.com/datasheets/CM108.pdf> Look in section 9.3.2 of the data sheet and notice that the frequency response is flat, within 1 dB, from roughly 50 Hz to 15 kHz. This is what we want. Wide and flat to minimize distortion.

Can you get better results with an expensive higher quality (whatever that means) audio interface? I κ' from side-by-side comparisons.

Compare the CM108 frequency response with the SignalLink USB:
http://www.frenning.dk/OZ1PIF_HOMEPAGE/SignalLinkUSB-mods.html Response is bumpy and falls off a cliff above 2.5 KHz. Good for 1200 baud but there is no possible way this could ever work for 9600 baud.

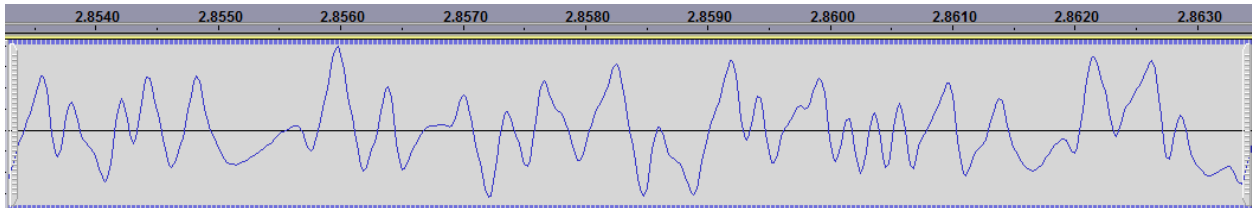
We also need low frequency response. VHF/UHF FM receivers usually have a high pass filter, with a cut off of around 300 Hz so the

Q q L ...



Notice how we have a nice horizontal line where several bits in a row have the same value.

The next graph was captured at the same time from an RS-UV3 where the high pass filter was not disabled.



Instead of a nice horizontal line, it droops back to the center because the lower frequencies are lost.

You can go faster if your radio and soundcard have enough bandwidth. For more discussion, see related document [*Going-beyond-9600-baud.pdf*](#).

8.5 2400 bps

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

A L q 4822 L 3422 4822 3997 5472 q

That last one would have some trouble getting through the audio stages of most transceivers.

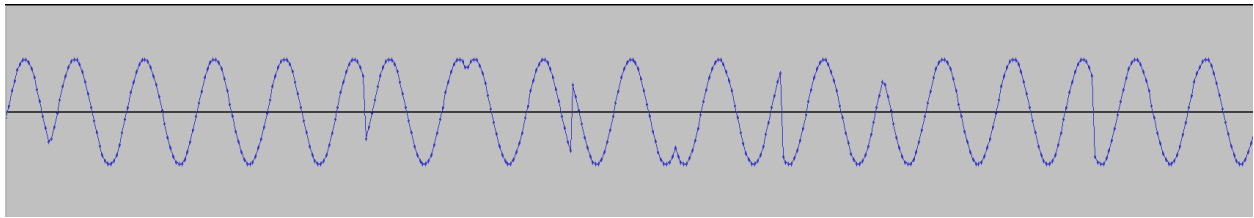
K 3 2 4822 q

- [MFJ-2400](#) which is an optional board for the MFJ-1270 or MFJ-1274.
- [AEA PK232-2400](#).
- [Kantronics KPC-2400](#).

q K : q q 4822 q
never gained much popularity.

The first two listed above used the EXAR [XR-2123](#) PSK modem chip which implements the [V.26](#) / Bell 201 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 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.



For more information, see the accompanying document, ***2400-4800-PSK-for-APRS-Packet-Radio.pdf***.

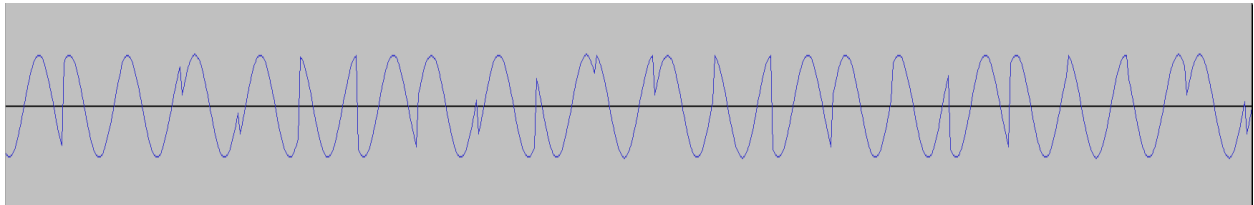
8.6 4800 bps

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

greater audio bandwidth. If you have that bandwidth, you can do better than AFSK.

The Hamilton Area Packet Network HAPN- pushes the digital signal through the radio in the same way we would for 9600 baud operation. The literature scrambling so it would probably not be compatible with the K9NG/G3RUH scheme.

If we can distinguish between 4 different phases, why not go for 8? With a few minor modifications, a [V.27](#) style demodulator can be implemented. This uses the same 1800 Hz audio carrier. It can change phase 1600 times per second so it is 1600 baud. The 8 phases can represent 3 bits at a time so we have $1600 \times 3 = 4800$ bits per second.



For more information, see the accompanying document, ***2400-4800-PSK-for-APRS-Packet-Radio.pdf***.

9 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).
- Beaconsing.

Normally the configuration file is read from the current working directory. On Linux the directory is also searched. - q location.

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

Configuration commands are generally 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.
- Any parameter values containing spaces must be enclosed by quotes.

Example: The next two are equivalent

But this not equivalent because device names are case sensitive.

9.1 Audio Device

Often q : : audio device is used.

If you have multiple soundcards, you might want to use a different dedicated interface rather than the same one that goes to your speakers.

Starting with version 1.2, up to three audio devices can be used at the same time. This allows operation with up to six radio channels. If you need two channels, there are some advantages to using a separate

9.1.1 Audio Device Selection – All Platforms

A radio channel (or pair of channels when using stereo) normally uses the same physical interface for both input (receive) and output (transmit). In this case, it can be listed once, in the ADEVICE configuration, as in these examples:

You could also list the **same interface twice**, once for input and once for output. These are equivalent to the previous example where the interface was listed once.

It is also possible to use different audio interfaces for receive and transmit. Examples:

The output interface must be something recognized by the sound system for your particular Operating System, often referred to as a platform. Windows, Linux, and Mac OSX differences are covered in the next few sections.

The sections after that cover additional forms that can be used for the input only. These are typically used with Software Defined Radios (SDR).

- `- [] k`
standard input.
- `UDP:nnn` means read from the specified UDP port.

In both cases, the audio streams must be 16 bit signed little endian. You must make sure that the number of samples per second agree for the digital audio source and in the Dire Wolf configuration.

9.1.2 Audio Device selection - Windows

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


```

Dire Wolf version 1.2
Available audio input devices for receive (*=selected):
  * 0: Microphone (C-Media USB Headpho (channel 2)
    1: Microphone (Bluetooth SCO Audio
    2: Microphone (Bluetooth AV Audio)
  * 3: Microphone (Realtek High Defini (channels 0 & 1)
Available audio output devices for transmit (*=selected):
  * 0: Speakers (C-Media USB Headphone (channel 2)
    1: Speakers (Bluetooth SCO Audio)
    2: Realtek Digital Output(Optical)
    3: Speakers (Bluetooth AV Audio)
  * 4: Speakers (Realtek High Definiti (channels 0 & 1)
    5: Realtek Digital Output (Realtek

```

Input devices and output devices are listed with a number assigned by the operating system. In the configuration you can select them with either the number or a substring of the description. One number (or string) can be used for both or the receive/transmit sides can be listed separately.

For this example, these two different forms would be equivalent:

or

The numbers can change as USB devices are added and removed so picking some unique substring of the description is more predictable.

Many people will `q` `q`

`4` means operate the preceding device in stereo mode for two radio channels.

`3` `q` `q`

Mono operation (one channel per device) is assumed if not specified.

9.1.3 Audio Device selection – Linux ALSA

Linux ALSA audio devices are much more flexible and therefore more complicated and confusing. Here is the simplified version that will be appropriate in most cases.

`q` `q` `arecord -l` (the option is lower case L)

card 0

card 1

q q aplay -l (the option is lower case L)

card 0

card 1

The first (card 0) is on the system board.
The second (card 1) is a cheap USB audio adapter. Remember these numbers so we can reference the desired one later.

Troubleshooting tip:

q q : ,

The solution was to add the user name q κ ‘

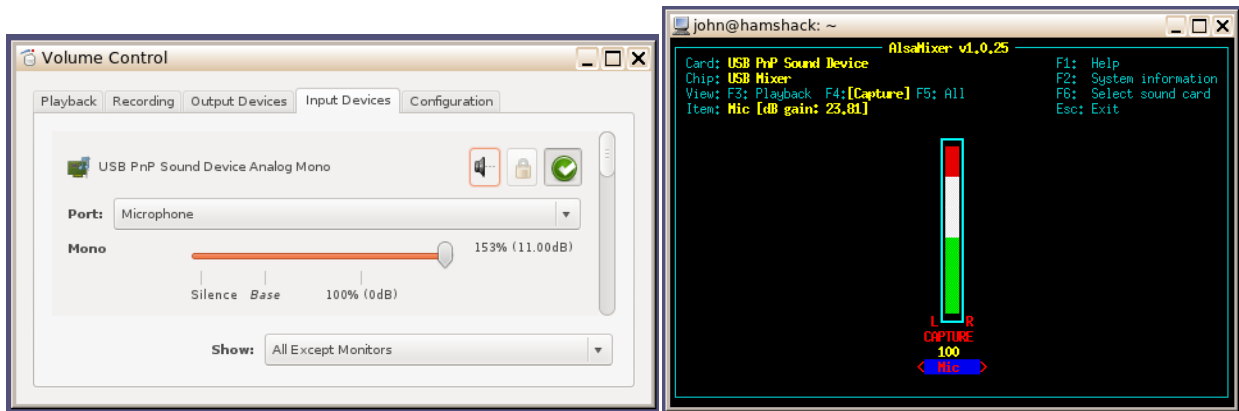
sudo addgroup john audio

This will not take effect immediately. Log out and log in again.

In this example, I want to pick the USB device. Recall that the card number was 1 so we want to put
q .3:2 he configuration file like this:

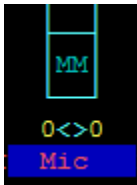
:

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



If the user has PulseAudio installed, the installing of pavucontrol is mandatory to make sure the right audio routing is done. In many respects, pavucontrol can do everything that alsamixer can do but not the other way around. Unfortunately, there are reports that pavucontrol can create blocking issues and programs will crash if it's running or they will stop running when pavucontrol is loaded.

Troubleshooting tip:



On a new system you might find the audio input device initially muted. If you see : q ← and → κ q unmute it. Use ↑ key to set near maximum gain.

Once you have the proper levels set, save them with:

Otherwise, you might find them reset to some other default the next time you reboot.

You might need to use "sudo alsactl restore" to make sure proper sound levels are always restored.

9.1.4 Audio Device selection – Mac OS X

The Macintosh version uses Port Audio. The audio device names may contain spaces. If they do, the parts with spaces must be quoted so we now which spaces are part of the names and which spaces separate them.

When Dire Wolf starts up, you should see a list of the audio devices available.

9.1.5 Audio Device properties

Two options are available. They apply to the most recent **ADEVICE n** command.

ARATE *sample-rate*

Where,

sample-rate is number of audio samples per second.

The default is 44100. Other standard values are 22050 and 11025.

When using a normal audio interface (built in to motherboard or USB adapter),

κ

‘

This would be necessary when using a software defined radio which tend to use rates like 48000 or 24000. This can also be specified on the command line for the first device only.

ACHANNELS *num-channels*

Where,

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

Q q

‘

use

9.1.6 Use with Software Defined Radios

When using software defined radios (SDR), the audio will be coming from another application rather

‘

9.1.6.1 gqrx

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

‘

q

‘

Q κ

Qq

‘

the same number as in the gqrx documentation.

Use the following Dire Wolf configuration file options:

This means that Dire Wolf will obtain audio from a UDP stream for receiving. If you transmit on that
 :

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

- - 3 3'
- - 8 222 audio sample rate of 48000 per second.
- - 3 3 q q : : ' ,

Note that these command line options apply only to the first audio device (ADEVICE0) and the first channel (CHANNEL 0).

9.1.6.2 rtl_fm

Other SDR applications might produce audio on stdout so it is convenient to pipe into the next
 qq ' q : - ' ,

Instead of command line options, you could do the same thing in the configuration file like this:

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.

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

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.

If you see some warning about audio input level being too high, don't worry about in this case.

It's only a potential problem when using the analog input of a sound card. If the analog audio input is too large, it can exceed the range of the A/D converter, resulting in clipping, distortion of the signal, and less reliable demodulation. The warning level is overly cautious. The input level can go much higher before it reaches the A/D range limit.

In this case, where 16 bit digital audio is going from one application to another, there is no danger of overflowing the signal range.

9.1.6.3 SDR#

point to a Google search:

<http://www.google.com/search?q=sdrsharp+tutorial>

Here are some other good locations to help you get started.

<http://www.atouk.com/SDRSharpQuickStart.html>

<http://www.qsl.net/yo4tnv/docs/SDRSharp.pdf>

<https://learn.adafruit.com/getting-started-with-rtl-sdr-and-sdr-sharp/sdr-number-fm-radio>

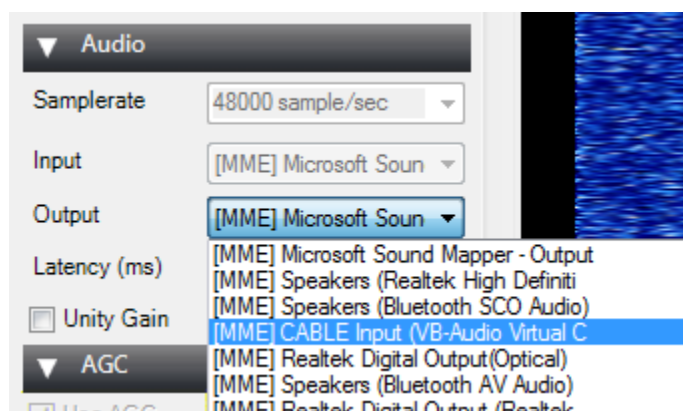
First verify that you can hear the desired signal through the speaker. Check the frequency calibration against a signal of known frequency. My cheap RTL-SDR dongle was off by 64 ppm which is more than 9 kHz on the 2 meter band.

How can we send the received audio to another application instead of the speaker? You could install a second. There is

other without any hardware in the middle.

Install VB-CABLE Driver from <http://vb-audio.pagesperso-orange.fr/Cable/> and reboot.

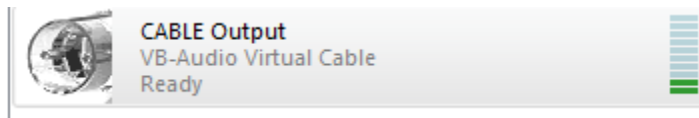
Run SDR# and notice the audio settings. It probably has 48000 samples per second. Instead of using the



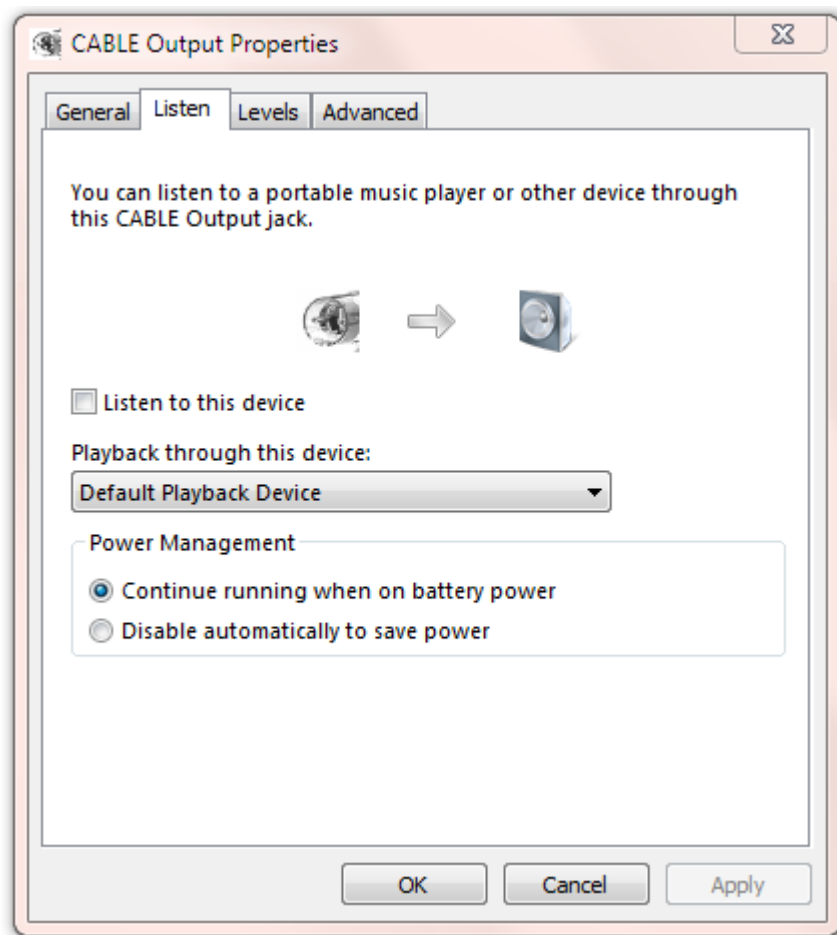
on the speaker icon and pick Recording Devices from the pop up menu.



You should see CABLE Output and the level indicator on the right should show some activity. Select it and click the Properties button.

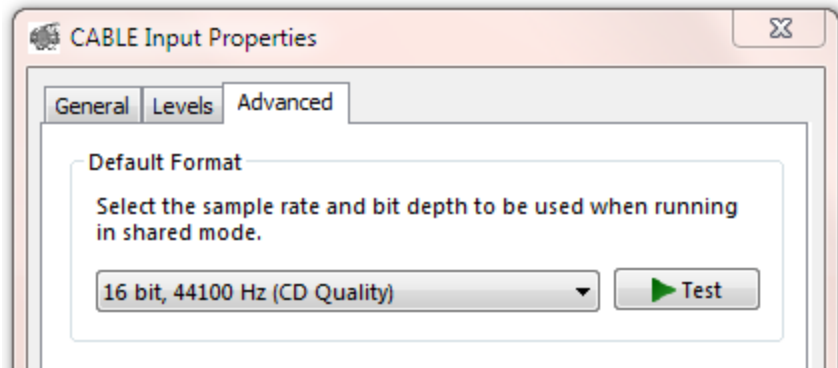


Pick the Listen tab.



K qq ' DR# through the speaker. Leave it on for now during the testing. You might want to turn it off again after it is all working.

If you look at the CABLE device Advanced properties,



you 88322 q ' 8 222
 q ' k q q
 caring.

I found the mismatch to be disturbing and changed it to different values for sample rate, bits per
 q : ' k ' :
 setting seems to be ignored and the bytes just get pushed through.

It is important that the applications producing and consuming the audio stream agree. The delivery

Put this in your direwolf.conf file:

When you start up Dire Wolf, you should see something like this:

*** 3: CABLE Output (VB-Audio Virtual (channel 0)**

Channel 0: 1200 baud, AFSK 1200 & 2200 Hz, E+, 48000 sample rate.

The lines of interest have been highlighted in red.

- The audio input is from the CABLE output device.
- The sample rate matches the value seen in SDR#.

You should now be able to decode the packets you hear.

9.1.6.4 SDR Troubleshooting

`q k : - 32`
will print out the audio sample rate and level each 10 seconds. In this example, we see that audio is
`q`
configuration file so it defaults to 44100. The decoder is expecting 44.1k samples per second, but the
actual rate is 48k so the decoding will fail.

```
* 3: CABLE Output (VB-Audio Virtual (channel 0)

Channel 0: 44100 sample rate.

Sample rate approx. 48.0 k audio level CH0 61
```

The reported sample rate will vary a little, especially for short collection intervals. This is because the audio samples are transferred in large blocks for efficiency rather than a steady stream of one at a time.

9.2 Radio channel configuration

As mentioned above you can have up to six radio channels. Specify options for each channel like this:

CHANNEL 0

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

CHANNEL 1

(options for second channel if first device is operating in stereo.)

Each of the following MYCALL, MODEM, PTT, and so on, applies to the most recent CHANNEL command.

9.2.1 Radio channel – MYCALL

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

The AX.25 specification requires that the call is a maximum of 6 upper case letters and digits. The substation id (SSID), if specified, must be in the range of 1 to 15.

9.2.2 Radio channel - Modem configuration , general form

The format has changed in version 1.2 to be simpler and more intuitive. The old format will still be accepted but you will see a message with a suggestion to upgrade to the new format.

Each radio channel can be configured separately for different speeds and modem properties. The general form of the configuration option is:

MODEM *speed* [*option*]...

In most cases, you can just specify the speed. For special situations you can override the defaults with these options:

Form	Purpose	Applicable to	Example, Comments
<i>mark:space</i>	Specify non-default tone pair for AFSK modes. Use 0:0 to for K9NG/G3RUH style baseband.	All.	2130:2230 for AEA/timewave PK-232 HF tones.
<i>num@offset</i>	<i>num</i> demodulators with center frequencies shifted by <i>offset</i> ‘	Mostly 300 baud on HF SSB.	5@30 could be used to compensate for other transmitters off frequency.
<i>/n</i>	Divide audio sampling frequency to reduce CPU speed requirement.	The multiple demodulator case, very old computer, or micrcocomputer.	/2 means use half the normal audio sample rate. Do not use for 9600 baud.
<i>ABCDEF+-</i>	Pick demodulator version and optional multiple slicers.	A, B, C, E, and F are for 1200 baud. D is optimized for 300 baud. qq to 1200 and 9600.	More details in the receive performance section. Short answer: A, B , C, F are obsolete. D is the default for 300 baud. E is the default for 1200. E+ will provide better compensation for imbalance of mark/space tone amplitudes.

			The letters are only for AFSK modes and do NOT apply to 9600 baud.
--	--	--	--

Note: The @ and + options are mutually exclusive. Both can't be used at the same time on the same channel.

9.2.3 Radio channel - Modem configuration for 1200 baud

The default configuration is 1200 baud, AFSK with 1200 & 2200 Hz tones for VHF FM use. If you omit the configuration, the default is the same as using either one of these:

`q` `q` `q` -emphasis / de-emphasis
`'` `q` ***A-Better-APRS-Packet-Demodulator.pdf***
 detail. This is on by default but you can deactivate it - '

Demodulator style E is the default. There is probably no reason to use anything else for 1200 baud.

Both send and receive must use the same speed and modulation type. In special situations, such as a satellite, you might want to receive 9600 baud and transmit 1200 baud. In this case you would need to use two different channels, one for transmit, and one for receive.

9.2.4 Radio channel - Modem configuration for 300 baud HF

The following are equivalent suitable configurations for 300 baud HF SSB operation using the popular 1600 / 1800 Hz tone pair.

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 can configure multiple decoders per channel, each tuned to a different pair of audio frequencies. With this example, we have 7 different modems, spaced at 30 Hz apart.

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 Conquer
Status Report, CAR, Experimental
TNOSaprs 1.11.2 TNOS APRS IGATE Conquerall Bank Nova Scotia

2W&144.390,144.970  KA0MOS-9 audio level = 50  [NONE]  ___|_|_|_|:
[0.4] KA0MOS-9>APZ111,WIDE:!4419.82N/06429.3
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, UView 32 bit apps

WA8LMF-13 audio level = 48 [NONE] ____||__
[0.3] WA8LMF-13>APU25N,ECHO:>151334z_Live 30Meter APRS Activi
Status Report, REC. VEHICLE, UView 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.

q 522 q -emphasis /
de-emphasis that causes the two tone amplitudes to be way out of balance.

Running several demodulators in parallel can consume a lot of CPU time. You will probably want to use
n q q Q '

9.2.5 Radio channel - Modem configuration for 9600 baud

K9NG / G3RUH style baseband with scrambling is used with there are no AFSK tones specified:

Using the “/n” option would be a very bad idea in this case. We need the high sample rate to capture the high baud rate.

The demodulator types (A, B, : '] L ' qq 22 '

q '
offset can be introduced with using a software defined radio (SDR). This will be explained in a later
q -Better-APRS-Packet- '

As mentioned in an earlier section, t κ q q κ
your transceiver. The audio amplifiers, designed for voice, do not have enough bandwidth and distort
the signal so it is not usable.

9.2.6 Radio Channel - Allow frames with bad CRC

Normally we want to reject any received frame if the CRC is not perfect. Dire Wolf can optionally try to
q ' q ' guessing and
there is no guarantee that it is right.

One Bad Apple Don't Spoil the Whole Bunch '

Previously it was a global setting that applied to all channels. In Version 1.2, it applies to the most recent CHANNEL so different radio channels can have different settings .

The general format is:

FIX_BITS

Where,

effort_level indicates the amount of effort to modify the frame to get a valid CRC.

0 means no attempt.

1 means try changing single bits. (default)

2 means try changing two adjacent bits.

larger not recommended because there is a high probability of getting bad data.

sanity_check adds a heuristic to guess whether the fix up attempt was successful.

APRS tests whether it looks like a valid APRS packet. (default)

AX25 only checks the address part. Suitable for non-APRS packet.

NONE bypasses the sanity check.

PASSALL means allow the frame through after exhausting all fix up attempts.

Occasionally you might see something resembling a valid packet but most of the Time it will just be random noise. Examples:

audio level = 45(32/16) [PASSALL]

[0] <0xc0>k<0xe3>)<0x15><0xe5><0xe7>y<0xd6>r<0xeb>Um<0x8a>#

audio level = 28(23/19) [PASSALL]

[0] <0xa4><0xa6>"<0xa7>f<0xa2><0xa0><0x96>b<0x9a><0x92><0x88>@<0xe4><0x96><0x84>
b<0xa0><0x9e><0xa4><0xe4><0xae>b<0x9a><0xa4><0x82>@<0xe0><0xae><0x92><0x84>
<0x8a>d@c<0x03><0xf0>'cFwmH'>/=KEN<0x10>FROM COMTOOCOOK,N.H.<0x0d>

Only error-free frames are digipeated or passed along to an APRS-IS server. Propagating possibly corrupt data would not be acting responsibly. Note that these frames **are** passed along to attached applications. If they pass along data to someone else, it could be corrupt.

9.2.7 Radio channel – DTMF Decoder

[] : : with this command:

You can confirm that the option is selected from the message at application start up time:

DTMF decoder enabled.

9.2.8 Radio Channel – Push to Talk (PTT)

There are up to five different methods available for activating your transmitter.

- Serial port control lines.
- General Purpose I/O pins (Linux only).
- Parallel Printer Port (Linux only).
- [q :]'
- VOX (voice operated transmit) External hardware activates the transmitter when transmit audio is present.

When PTT has not been configured, you will see a message like this at start up time:

note. nfigure an output control line when using VOX so just ignore the informational

9.2.8.1 PTT with serial port RTS or DTR

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 [[-]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.

‘ Q -
opposite polarity. Some interfaces want RTS and DTR to be driven with opposite polarity to minimize chances of transmitting at the wrong time. Starting with version 1.2, you can now specify two control lines with the same or opposite polarity. Example:

Alternatively, the RTS and DTR signals from one serial port could control two transmitters. E.g.

Examples:

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

9.2.8.2 PTT with General Purpose I/O (GPIO)

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. Precede the pin

```
PTT GPIO [-]pin-number
```

Example:

```
PTT GPIO 25
```

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

9.2.8.3 PTT with Parallel Printer Port

The old fashioned parallel printer port can also be used on Linux. In this case, use LPT, followed by an
q - : umber. This works only with the primary parallel
printer port on the motherboard or possibly a PCI card configured to use I/O address 0x378. It will not
work with a USB to parallel printer port adapter.

Examples:

```
PTT LPT 0
PTT LPT -2
```

9.2.8.4 PTT using hamlib

If the Linux version was built to use **hamlib**, you can also use this form for greater flexibility:

```
PTT RIG model port
```

Where,

model identifies the type of radio.

2 is used to communicate with

Port is name of serial port connected to radio.
In the case where model is 2, this would be a host name/address
and optional port number. Default port is 4532

Examples:

- Yeasu FT-817 on /dev/ttyUSB0:
- rigctld on localhost:
- Try to guess what is on /dev/ttyS0:

For more details, see <http://sourceforge.net/p/hamlib/wiki/Hamlib/>

FAQ: <http://sourceforge.net/p/hamlib/wiki/FAQ/>

This would be a good place to go with questions: <http://sourceforge.net/p/hamlib/discussion/>

9.2.8.4.1 Hamlib PTT Example 1: Use RTS line of serial port.

Of course, it would be a lot easier to use the built-in functionality for this simple case. This is just an exercise on our journey to being able to use the flexibility for more interesting cases.

‘ In one terminal window, start up a daemon with the desired configuration.

```
q      mother board.
κ      :      q      q      ‘
      q      ‘
```

In another window,

We observe that the RTS ‘ ‘

```
κ      ‘
      ‘
      .8754
```

is the default port. You might also see examples with 127.0.0.1 which is equivalent but obscure and

```
κ      κ      ‘      :      qtion
      4’
```

Again, we should observe the RTS line of serial port `/dev/ttyS0` changing. To use this for PTT, put this in your Dire Wolf configuration file:

up. In this case it is running on the same host but it could be running on a different computer.

9.2.8.4.2 Hamlib PTT Example 2: Use GPIO of USB audio adapter. (e.g. DMK URI)

A few people have asked about support for the DMK URI. This uses a C-Media CM108/CM119 with one interesting addition: a GPIO pin is used to drive PTT. Here is some related information.

DMK URI:

http://www.dmkeng.com/URI_Order_Page.htm
<http://dmkeng.com/images/URI%20Schematic.pdf>
<http://www.repeater-builder.com/voip/pdf/cm119-datasheet.pdf>

Homebrew versions of the same idea:

<http://images.ohnosec.org/usbfbob.pdf>
<http://www.gsl.net/kb9mwr/projects/voip/usbfbob-119.pdf>
<http://rtplib.weebly.com/uploads/1/6/8/7/1687703/usbfbob.pdf>
<http://www.repeater-builder.com/projects/fob/USB-Fob-Construction.pdf>

We will need to use one of

Create a shell script, with the following content:

When we run it, we see that we want `/dev/hidraw3` in this case:

The default permissions allow use only by root, so we need to change this:

Now we can start up rigctld:

Test it from another window:

- q : K ...

Finally, put this in your direwolf configuration file in the appropriate channel section:

Doing this every time you reboot would be annoying. How can we automate the process? We can enhance our earlier script to include the necessary start up steps:

–

If you want it to run automatically after each reboot, add this to `/etc/rc.local`, using your script location.

Watch the debug output with a command like this:

9.2.9 Radio Channel – Data Carrier Detect (DCD)

The carrier detect signal can be sent to any of the output locations available for PTT. The same serial port can be used for both PTT and DCD. For example:

In this case, you could connect an LED to the serial port like this:

Pin 5 (GND) ---- (cathode) LED (anode) ---- 680 ohm resistor ---- Pin 4 (DTR)

9.2.10 Radio Channel – Transmit Inhibit Input

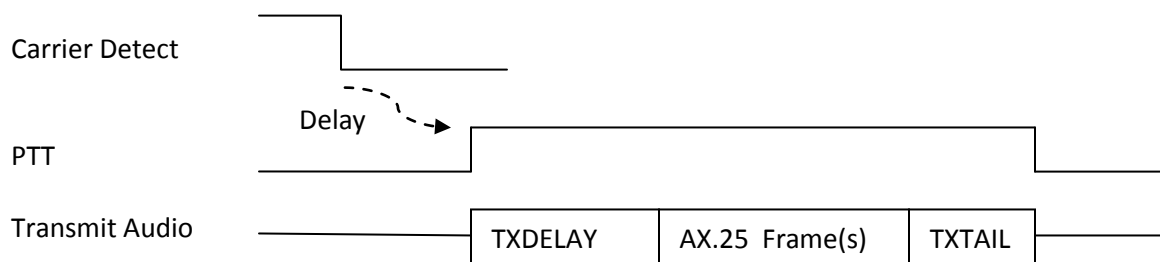
For the Linux version only, it is possible to have a GPIO control input to prevent transmitting. This might be used with a squelch signal from the receiver. A site with multiple radios could use this to give priority to the other radio service when it is active.

As with PTT, minus in front of the GPIO number means invert the signal.

9.2.11 Radio Channel – Transmit timing

Transmit timing is determined by 5 parameters which can be different for each channel. The defaults are:

DWAIT 0	x 10 mSec per unit = 0 mSec.
SLOTTIME 10	x 10 mSec per unit = 100 mSec.
PERSIST 63	probability for transmitting after each slottime.
TXDELAY 30	x 10 mSec per unit = 300 mSec.
TXTAIL 10	x 10 mSec per unit = 100 mSec.



When a frame is ready for transmission, we first have to wait until the channel is clear. The technical term for this is Carrier Sense Multiple Access (CSMA). In plain English, if you want to say something, wait until no one else is speaking.

- First we wait for the DWAIT time. Normally this is zero. This is used only for specific situations
- For digipeated frames the transmission can begin immediately after the channel is clear.
- For other frames, SLOTTIME and PERSIST are used to generate a random delay. This is to minimize the chances of two different stations starting to transmit at the same time. The process is:

(a) Wait for SLOTTIME.

(b) If a random number, in the range of 0 to 255, is less than or equal to PERSIST, start to transmit. Otherwise go back to step (a).

For the default values, we have delays with the following probabilities:

Delay, mSec	Probability	
100	.25	= 25%

200	$.75 * .25$	= 19%
300	$.75 * .75 * .25$	= 14%
400	$.75 * .75 * .75 * .25$	= 11%
500	$.75 * .75 * .75 * .75 * .25$	= 8%
600	$.75 * .75 * .75 * .75 * .75 * .25$	= 6%
700	$.75 * .75 * .75 * .75 * .75 * .75 * .25$	= 4%
etc.	...	

- If a signal is detected during any of the steps above, we go back to the top and start over.

The Push to Talk (PTT) control line is asserted.

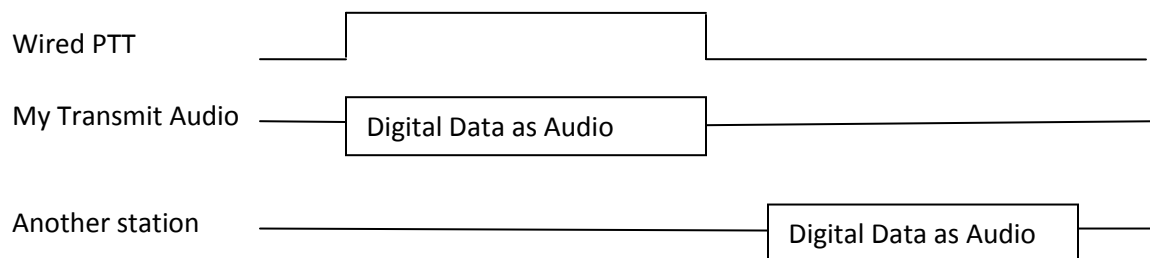
because the transmitter takes a little while to stabilize after being activated' q [23333332] a time period of TXDELAY. For historical reasons, going back more than 30 years, the times are in 10 mS units so 30 actually means 300 milliseconds.

latency between sending an audio waveform to q
q sound has been completed so we need to keep the PTT on a little longer. q
also sent during this time to keep the channel busy. A TXTAIL of 10 (x10 mSec = 100 mSec) is probably a little on the generous side but better safe than sorry.

9.2.11.1 Should I use wired PTT or VOX?

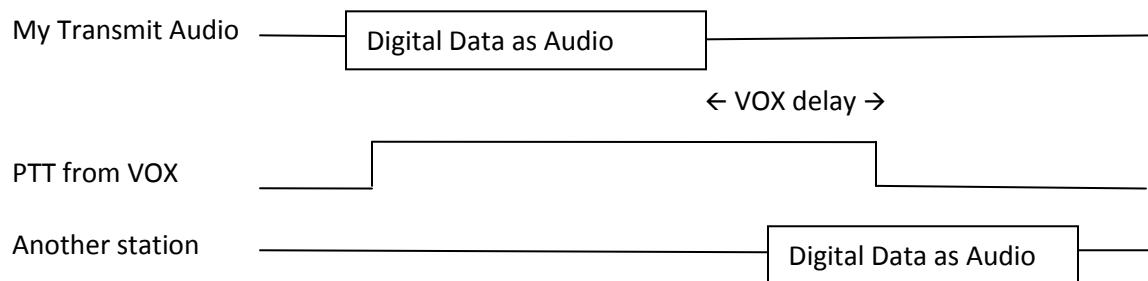
It might be tempting to use the VOX built into your transceiver and avoid the extra circuitry for the PTT signal. Is this a good idea?

transmitter is turned on, we send our digital data as an audio signal, and then turn off the transmitter when the audio is finished. Another station hears an empty channel, waits a little while, and starts transmitting.



VOX stands for Voice Operated Transmit. It is designed for voice which contains small gaps between words. Each word so there is a built in delay before turning off the transmitter. The default setting is 500 milliseconds and the minimum setting is 250. Another setting for the amount of time would not be unreasonable to assume they picked something around the default time for another brand.

Keeping the transmitter on a half second after the sound ends is fine for voice. But what about digital data? We saw in the previous section that another station waiting for a clear channel, will usually transmit within a half second after no longer hears another digital signal.



switched back to receive yet. You are also interfering with others by sending a quiet carrier. My recommendation is to avoid VOX unless you can be sure the transmitter will turn off very soon (e.g. less than 50 mSec.) after the audio signal is no longer present. If you insist on using VOX, be sure the

9.2.11.2 Frame Priority and KISS Protocol

The AX.25 protocol spec defines two priorities for transmission of frames.

- **Priority**

These are sent as soon as the channel is not busy. On your screen you will see magenta lines
[nH]: n is the channel, meaning **H**igh priority. APRS uses this for digipeated frames.

- **Normal**

After the channel is clear, we wait for a random time to reduce chances of a collision. On your
[nL]: n is the channel, meaning **L**ow priority.

Unfortunately the KISS protocol does not have a way to convey this distinction. If it looks like the client application is sending an APRS frame, we apply a heuristic to decide. When a digipeater field has the : q . Otherwise it goes in the normal low priority queue.

ng similar should be attempted for non-APRS AX.25 frames. It might be possible to apply some heuristic for control vs. information frames but we might make the situation worse by sending them out of the original order.

9.3 Logging of received packets

Specify the directory where log files should be written' ' K ' Examples:

A new log file is started each day. The log file has the name *yyyy-mm-dd.log*, where *yyyy-mm-dd* is the current date. The file format is described later in this section.

9.4 Client application interface

Three different interfaces are provided for client applications such as APRSISCE/32, UI-View32, Xastir, APRS-TW, YAAC, SARTrack, AX.25 for Linux, RMS Express, and many others.

9.4.1 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:

Only the raw mode (similar to KISS) interface is available at this time. This is fine for all APRS applications and some others such as RMS Express.

Some other packet applications, such as Outpost, require the AX.25 connected mode. This is currently not available in Dire Wolf. If you try to use connected mode, you will get an error message like this:

```
Can't process command from AGW client app.  
Connected packet mode is not implemented.
```

The **ttcalc** sample application uses this protocol and can be used as a starting point for writing your own applications.

In a system exposed to the Internet, you might want to disable the port for security reasons. Do this by specifying 0 for the port number:

9.4.2 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:

This can also be disabled by specifying 0:

9.4.3 Serial port KISS - Windows

A configuration option like this:

q L 5' q

serial port, used by the application, or configure a virtual null modem cable.

: **com0com** : epth discussion of how this works.

9.4.4 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.

$$\begin{array}{ccccccc} q & & : q & & L & & q : L \\ q & & : q & & q & & q q \end{array}$$

9.5 APRS Digipeater operation

Dire Wolf has only APRS style digipeating built in. It doesn't know anything about connected mode.

When operating as a KISS TNC (with either KISS or AGW interface) it passes all the frames along and doesn't care what is in them. They don't even need to be valid AX.25 format. You could use another application to supply the connected mode digipeating logic.

9.5.1 Digipeater - Configuration Details

Digipeater configuration is achieved with commands of the form:

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

where,

<i>from-chan</i>	is the channel where the packet is received.
<i>to-chan</i>	is the channel where the packet is to be re-transmitted.
<i>aliases</i>	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.
<i>wide</i>	is the pattern for normal WIDEn-N digipeating where the ssid is decremented.
<i>preemptive</i>	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.

As a typical example, you might have a dual port digipeater between the national standard APRS section) q k q (described in a later section)

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.

9.5.2 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]\$

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. This gets replaced by MYCALL when digipeated. It does not apply the rule of decrementing the last digit of WIDEn-n. 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>

If you wanted to process WIDE3- : qq : this instead:

- The final parameter specifies patterns to be processed with the new n-N paradigm if not caught by the aliases. If the last digit is greater than zero it is decremented by 1 when retransmitted.

9.5.3 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.

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

9.5.4 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 $N \geq 2$, 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.

9.5.5 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.

q k ' e 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,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

```

Th qq 4 ' K :

k 4-0 (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 3 :

```

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** WIDEn-0. Example:

```

Digipeater WIDE2 (probably AB1OC-10) audio level = 10 [NONE]
[0] KB1DDC>TR3R8X,W1XM,WIDE1,AB1OC-10,WIDE2*: `c&*1 <0x1c>-/] "3r}<0x0d>
MIC-E, House, Kenwood TM-D700, En Route
N 42 32.8800, W 071 10.1400, 0 MPH, alt 0 ft

```

9.5.6 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

' q qq XXXn- q '

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
q : ' q :
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.
- TYPE Prior path data will be reflected in the act (all 91 (th) taken) ETBTve, T2F1 11.04 Tff 0 0 1 04.28 3191(i)] T[-ahapDck

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	

9.6 Packet Filtering

Sometimes it is desirable for a digipeater or Internet Gateway to pass along some types of packets and block others.

A filter can be defined for each combination of where the packet came from (radio channel or IGate server) and where it is being sent to. The format of the configuration command is:

```
from-channel to-channel filter-expression
to-channel filter-expression
from-channel filter-expression
```

The filter expression is loosely based on <http://www.aprs-is.net/javaprsfilter.aspx> **Server-side Filter Commands** with the addition of logical operators to combine the filter results. For example, you could decide to digipeat only telemetry originating from WB2OSZ or object reports not within a certain distance of a given location.

```
q q q '
```

9.6.1 Logical Operators

The individual filter specifications return a true or false value depending whether the current packet satisfies the condition. These results can be combined into larger expressions to for very flexible configuration. The operators are:

	Logical OR. Result is true if either argument is true.
&	Logical AND. Result is true if both arguments are true.
!	Logical NOT. This inverts the value of the following part.
()	Parentheses are used for grouping.

9.6.2 Filter Specifications

The filter specifications are composed of a lower case letter, the punctuation character to be used as a field separator, and parameters. These two are equivalent:

Other implementations allow only the separator character. This extra flexibility comes in handy when q '

Everything is case sensitive. This means that upper and lower case are not equivalent.

All Filter Specifications must be followed by a space. This is so we can distinguish between special characters that are part of the filter or a logical operator.

9.6.2.1 Wildcarding

Most of the filters ‘ This
operates on character strings without any knowledge of the callsign-SSID syntax. If you wanted to
4 regardless of any SSID, your first reaction might be to use

This would not be correct because it would also match W2UBA, W2UBZ, and many others. The correct form would be:

This will match only that callsign (implied SSID of zero) or that callsign followed by any SSID.

9.6.2.2 Range Filter

r/ / /

This allows **position** and **object** reports with a location within the specified distance of given location.

Latitude and longitude are in decimal degrees. (negative for south or west.)
Distance is in kilometers.

Note that this applies only to packets containing a location. It will return a false result for other types such as messages and telemetry. If you wanted to digipeat stations only within 50 km you might use something like this:

This would reject other **q q κ** ‘ :
operator to also allow all types other than position and object:

9.6.2.3 Budlist Filter

b/ / ...

Allow all packets from the specified calls. These must be exact matches including the SSID. Wildcarding is allowed.

[] q :

q κ q ‘

9.6.2.4 Object Filter

o/ / ...

Allow objects and items whose name matches one of them listed. Wildcarding is allowed.

9.6.2.5 Type Filter

t/

List one or more of the following letters for types of packets to be allowed.

- p - Position Packets
- o - Object
- i - Item
- m - Message
- q - Query
- s - Status
- t - Telemetry
- u - User-defined
- n - NWS format
- w - Weather

9.6.2.6 Symbol Filter

s/ / /

pri q ‘

alt one or more symbols from the alternate symbol set.

over ‘ Overlays apply only to the alternate symbol set.

Examples:

- Allow house and car from primary symbol table.
- Allow alternate table digipeater, with or without overlay.
- Allow alternate table digipeater, only if no overlay.
- Allow alternate table digipeater, with overlay S, L, or 1

9.6.2.7 Digipeater Filter

d/ / ...

Allow packets that have been repeated by any of the listed digipeaters. Wildcarding is allowed.

: q q κ only if heard directly, use this:

That means, when digipeating from channel 0 to channel 0 allow packets only if they have **not** been digipeated through some other station.

9.6.2.8 Via digipeater unused Filter

v/ / ...

Allow packets that q -been- ' is allowed.

9.6.2.9 Group Message Filter

g/ / ...

q κ ' Wildcarding is allowed.

9.6.2.10 Unproto Filter

u/ / ...

Allow packets with any of the specified strings in the AX.25 destination field. APRS uses this field in a variety of ways. Most often it is the system type from the tocalls.txt file. For example, to select packets from the Kantronics KPC-3+, version 9.1, use:

This does not apply to the MIC-E packet types because they use the destination field for part of the position.

,

9.6.3 SATgate example

Suppose you wanted to run a SATgate to forward anything either from some station called "RS0ISS" or anything digipeated by it. That's easy. The filter could look like:

That means when forwarding from RF channel 0, through the IGate function, accept packets that are from RS0ISS or have been digipeated by RS0ISS. The "b" filter matches the source address. The "d" filter only considers digipeater addresses that have already been used.

Consider the case where you want to run a normal terrestrial IGate and a SATgate at the same time. Suppose that some nearby earth station sent a packet via RS0ISS-4. A nearby observer might see something like this:

The first one was heard directly from the source. The second one is the retransmission by the digipeater named RS0ISS-4. In this context, the "*" means that the digipeater address has been used. i.e. We are hearing the digipeater, not the source station directly. If we send both of these to an APRS Internet Server, the second one will be dropped as a duplicate. How could we filter out the first one and let the second through?

One suggestion, from a discussion group, was to ignore all packets heard directly from the source and process only those which have been digipeated. In this case the filter would be:

Here "*" is a wildcard meaning match anything. Dropping anything heard directly would interfere with the normal IGate behavior. The "v" filter is useful in this case.

"v" also looks at the digipeater addresses but considers only those which have NOT been used. In this example, "v/RS0ISS*/ARISS" produces a match if any **unused** digipeater address matches RS0ISS (with any SSID, due to wildcard), or exactly ARISS. The "!" inverts the following value.

We end up dropping the first example packet because it is addressed to travel via certain digipeaters but hasn't yet. The second example packet, and other normal terrestrial packets - whether heard directly or via digipeater - are allowed to continue to the APRS Internet Server.

q

9.7 GPS Interface

Your location, from a GPS receiver, can be used in a tracker beacon described in the following Beacons section. There are two types of interface available:

- Direct connect to serial or USB port. Available on all platforms.
- GPSD server. Available only on Linux. This allows multiple applications to share one receiver.

9.7.1 Direct connect to GPS receiver

Use the GPSNMEA configuration option with the device name. This might be a serial port or a USB device that looks like a serial port. The standard baud rate of 4800 is used. Examples:

Dire Wolf reads NMEA sentences (\$GPRMC and \$GPGLL) from the receiver and parses them to extract position information.

Advantages: Simple configuration. Available on all platforms.

```
... Q                                q  qq  '

```

9.7.2 GPSD Server

An alternative method is available on Linux. q (<http://www.catb.org/gpsd/>) allows multiple applications to share the GPS receiver at the same time. q ,

Add the GPSD item to the configuration file. If the GPSD server process is running on a different computer, you can specify the host name or address. Examples:

```
-- equivalent to first example.
-- use server running on different computer.
```

The accompanying document, **Raspberry-Pi-APRS-Tracker.pdf**, goes into more detail. The same general principles apply to other types of Linux systems.

9.8 Beacons

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

- PBEACON - Position
- OBEACON - Object
- CBEACON - Handcraft your own Custom beacon
- TBEACON - Tracker beacon with GPS location

9.8.1 Position & Object Beacons

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

```
PBEACON - q '
OBEACON - q '
                q 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. Use an extremely long interval (like 1000000 for around two years) here to get a one time transmission.	10 9:45	Ten minutes. 9 ¾ minutes
SENDTO	send to Internet Gateway. Default is the first, or only, radio channel 0. received on that channel.	1 IG R0	Second radio channel. Internet Gateway. Simulated channel 0 reception.
DEST	Explicit destination field for AX.25 packet. Normally you will want the default which identifies the software version.	CQ	Q special cases explained later.

VIA	Digipeater path. Default none.	WIDE1-1 WIDE1-1,WIDE2-1	Upper case only. No spaces.
MESSAGING	Set the APRS Messaging attribute for a position report. i.e. Data Type Indicator	0 1	Default value. Set attribute.
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.
ZONE	Zone with latitude band for UTM coordinates.	19T	
EASTING	UTM coordinate.	307504	
NORTHING	UTM coordinate.	4721177	
ALTITUDE or ALT	Altitude in meters.	90	
SYMBOL	Two different styles are available: (a) Exactly two characters specifying symbol table / overlay and the symbol code. (b) A substring of the description.	S# K κ	More details below.
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.		
COMMENTCMD	Run specified command and insert result after the fixed part of comment.		Original intention was to insert base 91 compressed telemetry.
COMPRESS	Use 1 for compressed format. Note that power/height/gain gets converted to single radio range value in	0 1	Human readable. Compressed.

	the compressed format.		
--	------------------------	--	--

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. All three of these are different ways to represent the same location.

```
PBEACON LAT=42^37.14N LONG=71^20.83W
PBEACON LAT=42.619 LONG=-71.34717
PBEACON zone=19T easting=307504 northing=4721177
```

The included coordinate conversion utilities can be use to convert one form to the other. In the following examples, the first line is the command you type. The second line is the response.

```
$ ll2utm 42.619 -71.34717
UTM zone = 19, hemisphere = N, easting = 307504, northing = 4721177
MGRS = 19TCH12 19TCH0721 19TCH075212 19TCH07502118 19TCH0750421177
USNG = 19TCH02 19TCH0621 19TCH075211 19TCH07502117 19TCH0750321177
```

```
$ utm2ll 19T 307504 4721177
from UTM, latitude = 42.618996, longitude = -71.347166
```

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. Examples:

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

q - [qq]

line option.

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.

```
q      '      q      :      q
-  q      q      /  '      q
```

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 offset often causes confusion. When it appears in the pack : 32 κ ' 722 7 ' q q found here: <http://www.aprs.org/info/freqspec.txt>

Here is one possible way to send messages through the International Space Station. It is similar to Q

```
PBEACON delay=00:01 every=00:30 symbol="/'" lat=32^39.30N long=097^23.06W
comment="Hello from Texas, sutton.matthew@gmail.com" via=ARISS
dest=CQ messaging=1
```

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

9.8.2 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. The first character of the information part is the data type indicator.

Keyword	Description	Example values	Comment
DELAY	Time, in minutes or minutes:seconds, to delay before sending first time.	1 0:30	One minute. Half minute.

	Default is 1 minute.		
EVERY	Time, in minutes or minutes:seconds, between transmissions. Default is 10 minutes.	10 9:45	Ten minutes. 9 ¾ minutes
SENDTO	send to Internet Gateway. Default is the first, or only, radio channel 0. received on that channel.	1 IG R0	Second radio channel. Internet Gateway. Simulated channel 0 reception.
DEST	Explicit destination field for AX.25 packet. Normally you will want the default which identifies the software version.	CQ	Q special cases explained later.
VIA	Digipeater path. Default none.	WIDE1-1 WIDE1-1,WIDE2-1	Upper case only. No spaces.
INFO	q packet. This is a constant value.		
INFOCMD	q for packet. This allows each to be different as determined by a user-supplied script.		

A couple examples:

See **APRS Telemetry Toolkit**

q

9.8.3 Tracker Beacon

Information from a GPS receiver can be used to report the location of a moving entity.

First you must use either the **GPSNMEA** (all platforms) or **GPSD** (Linux only) configuration items to establish communication with the GPS receiver.

The **TBEACON** command has the same options as **PBEACON**, above, except latitude, longitude, course, and speed are obtained from the GPS receiver. If you specify **ALTITUDE** greater than 0, the actual altitude will be taken from the GPS location.

Example: Driving around in a car.

```
TBEACON DELAY=0:30 EVERY=2:00 VIA=WIDE1-1 SYMBOL=car
```

This will wait 30 seconds then transmit once every 2 minutes after that.

In this case, the FREQ options can be used to indicate that you are listening to a certain voice channel.

TBEACON SYMBOL=car FREQ=146.955 OFFSET=-0.600 TONE=74.4

9.8.4 SmartBeaconing™

A fixed transmission schedule might not be ideal. If you are moving quickly, you might want to send position updates more quickly. If sitting still, there is no reason to transmit very often. Sending redundant information over and over just clutters up the radio channel. A display application which

off when there is a change of direction.

SmartBeaconing™ adjusts th used by Kenwood, Yaesu/Standard, and in many other applications. These 3 examples are all equivalent. In the first example, reasonable defaults are supplied for use in a land vehicle. In other two, all parameters are specified.

Remember that a beacon time of just a number is interpreted as minutes. So if you want 1800 seconds, 2.3 22 ‘

What do the numbers mean?

- Fast Speed & Fast Rate -- For speeds above 60 MPH, a beacon will be sent every 1 ½ minutes.
- Slow Speed & Slow Rate -- For speeds below 5 MPH, a beacon will be sent every 30 minutes.
- For speeds in between, a rate proportionally in between will be used.

Additional beacons will be sent more frequently when direction changes significantly.

- Send no more frequently than 10 seconds apart.
- Send if direction has changed more than 30 degrees since last report at high speed.
- Requires sharper turns at lower speeds.

replaced by a variable time depending on motion.

More details can be found in these references or just Google for APRS SmartBeaconing™ to find discussions and recommendations.

<http://www.hamhud.net/hh2/smartbeacon.html>
<http://info.aprs.net/index.php?title=SmartBeaconing>

9.9 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.

9.9.1 IGate - Select server and log in

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 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:

9.9.2 IGate – Configure transmit

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

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.aprs-is.net/javaprsfilter.aspx> A simple example:

Important!

Here we are simply passing along the filter specification and not processing or checking it in any way.

: ' e 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.

9.9.3 IGate – Sending directly to server

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 q ' Use overlay R for receive only, T for two way.

sendto=IG

sendto=IG

9.9.4 IGate – Client-side filtering

After setting an appropriate - : want, creating excessive clutter on the radio channel. It is possible to apply another stage of filtering in : - ' Q to be transmitted on radio channel 0, you could use a filter like this:

This can also be applied in the opposite direction to restrict what is passed from the radio channel(s) to the server. Examples:

Only "messages" from channel 0.

Only weather from channel 1.

Nothing from channel 2.

The differences between the two types of filtering are summarized below.

	Server-side filtering	Client-side filtering
Configuration file	IGFILTER	FILTER
Where defined	http://www.aprs-is.net/javaprsfilter.aspx	Packet Filtering document.
Where processed	Internet Server	Inside of Dire Wolf application.
Expressions with & ! ()	No	Yes
Precise control over what is passed through	No	Yes
Can be different for each radio channel	No	Yes

9.9.5 SATgate mode

If we hear a packet directly and the same one digipeated, we only send the first to the APRS IS due to duplicate removal. It may be desirable to favor the digipeated packet over the original. For example, you might be more interested in packets that have been forwarded by satellites rather than heard directly. For this situation, we have an option which delays a packet if we hear it directly and the via path is not empty.

When using this option, the digipeated packets will go to the server immediately. The original, heard directly, is sent after a delay, typically 10 seconds. Duplicate removal will drop the original if there is a corresponding digipeated version.

The configuration option for this feature is:

You can optionally add a delay time in seconds. The default is 10.

You can find more discussion here: <http://www.tapr.org/pipermail/aprssid/2016-January/045283.html>

9.9.6 IGate Debugging Options

To see more of what is going on behind the scenes, use one of these debugging options on the command line:

Show packets being sent to Server.
Show information about duplicate removal.

9.10 APRStt Gateway

The APRStt Gateway function allows a user, equipped with [] q : information into the global APRS network. Various configuration options determine how the touch tone Q q κ ' with TT.

TTPOINT
TTVECTOR
TTGRID
TTUTM
TTCORRAL
TTMACRO
TTOBJ
etc.

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

9.11 Transmitting Speech

There are many software applications that will convert text to speech. Dire Wolf can utilize these to transmit information with a synthesized voice. At the end of this section we have a simple application

[] q ' q

9.11.1 Install Text-to-Speech Software

First we need to install some software to convert text to speech. Googling around reveals some good lists of alternatives available.

http://download.cnet.com/windows/text-to-speech-software/3150-33660_4-0.html

http://elinux.org/RPi_Text_to_Speech_%28Speech_Synthesis%29

These examples use **eSpeak** but most of the others should also be fine with minor changes to the scripts. First we need to install the software.

Windows:

Download **setup_espeak-xxx.exe** from <http://espeak.sourceforge.net/download.html> and run it.

Raspbian Linux:

Instructions here: ***RPi Text to Speech (Speech Synthesis)***

(Not yet tested at the time this is being written.)

Other Debian / Ubuntu Linux:

Red Hat / Fedora / CentOS Linux:

9.11.2 Configuration

Next we need a little script to run the text-to-speech application with the desired options. Dire Wolf will invoke this script with two command line arguments:

- The radio channel number. In most cases you can ignore this. In more complex multi-channel situations you can use this to send the speech to the desired audio device.
- The text to be spoken.

Two simple examples are provided:

- **dwespeak.bat** for use with Windows
- **dwespeak.sh** for use with Linux:

Open a command window and run one of the following depending on your operating system. The quotation marks are very important. Do not omit them.

Next, edit the configuration file, `direwolf.conf`, and add one of these options, again using the appropriate script name for your operating system.

one possible way you could use this to announce an event:

```
CBEACON dest=SPEECH info="Club meeting tonight at 7 pm."
```

9.11.3 Sample Application: `ttcalc`

Here is a simple application that can be used as a starting point for developing your own applications. This is a sample application.

- (1) Perform steps above and verify that the **dwespeak.bat** or **dwespeak.sh** script produces speech output.
- (2) Edit the Dire Wolf configuration file and make sure it has the SPEECH option specified properly.
delay=0:15 every=0:30 Use

- (3) Enable the DTMF decoder on the desired channel in the configuration file. Example:

(4) `DTMF decoder enabled` with the startup status messages.

(5) `q κ` has connected.

(6) Transmit touch tones (from a different radio obviously) such as:

`4 * 6 #`

(7) `q κ` : will be transmitted in response.

This is not a very useful application. It is provided as a simple example to be used as a starting point for your innovation applications.

9.12 Transmitting Morse Code

ation part is sent in Morse Code. Under some circumstances you might want to use this to meet station identification requirements. If an SSID is specified, it is multiplied by 2 to get the speed in words per minute (WPM). E.g. SSID of -10 will be 20 WPM.

```
CBEACON dest=MORSE info="de WB2OSZ/R."
10 info="de WB2OSZ/R."
```

9.13 Logging

Simple, yet versatile, logging is available by specifying `l` (lower case L) on the command line or using the LOGDIR option in the configuration file. In either case, specify the directory (folder) where log files

`q ['] κ`

Rather than saving often unreadable raw data, the digested parts are saved in Comma Separated Value (CSV) format. The first line has the names of the fields.

```
chan, utime, isotime, source, heard, level, error, dti, name, symbol, latitude, longitude,
speed, course, altitude, frequency, offset, tone ,system, status, comment
```

Name	Example	Description
chan	0	Radio channel where packet was received.
utime	1403134556	UTC in seconds since January 1, 1970 in decimal.
isotime	2014-06-18T09:56:21Z	Time in ISO 8601 format.
source		Sending station of packet.
heard		Station heard on radio. Actually our best guess always be sure due to different interpretations of tracing. See Digipeater section for my discussion about this.
level	23	Audio level of station heard.
error	0	0 = packet received with correct CRC. 1 = able to get correct CRC by changing one bit. >2 = found good CRC by changing more than 1 bit. Result q qq
dti	!	Data Type Identifier first byte of information part. For q q q ition with APRS messaging.
name	EOC	Name from Object or Item report. Otherwise the sending station.
symbol	/-	Two characters: symbol table (or overlay) and symbol code.
latitude	12.345678	In degrees. Negative south.
longitude	-123.456789	In degrees. Negative is west.
speed	55	Speed in knots.
course	123	Direction of travel, degrees.
altitude	90	Meters above average sea level.
frequency	146.955	Voice frequency in MHz.
offset	-600	Voice transmit offset in kHz.
tone	74.4 D123	CTCSS tone or DCS code q
system	Kenwood TH-D72	Name of hardware or software. Usually derived from the destination address, such as APN383 for Kantronics KPC-3. For MIC- q κ :
status	En Route	Status from MIC-E packets.
telemetry	Seq=3307, Vbat=4.383 V, Vsolar=0.436 V, Temp=-34.6 C, Sat=12	Telemetry data.
comment		Comment.

Fields are quoted if the data value contains a comma or quotation character. A new log file is started each day. The log file has the name *yyyy-mm-dd.log*, where *yyyy-mm-dd* is the current date.

Data, in this convenient form, can be imported into a spreadsheet or fed into other conversion applications to obtain the desired subset and format.

9.13.1 Conversion to GPX format

A sample application is included for converting a log file to GPX format. The source code can be used as the starting point for other custom converters.

Specify one or more log file names on the command line. Redirect the output if you want to save the GPX information to a file. Example:

The GPX file can be uploaded to many popular mapping applications such as Google maps or OpenStreetMap. Here is one that is very easy to use: <http://www.gpsvisualizer.com/>. You have an account or log in. Simply upload your GPX file and the waypoints and tracks are displayed on a map. Click on a waypoint to see any additional information.



9.14 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. Case is significant. Pay careful attention to upper and lower case.

-c	<i>fname</i>	Configuration file name.
-r	<i>n</i>	Audio sample rate for first audio device. e.g. 44100
-n	<i>n</i>	Number of audio channels for first audio device. 1 or 2.
-b	<i>n</i>	Bits per audio channel for first audio device. 8 bit unsigned or 16 bit signed little endian.
-B	<i>n</i>	Data rate in bits/sec for channel 0. 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	<i>n</i>	Divide audio sample rate by <i>n</i> for channel 0.
-l	<i>logdir</i>	Name of directory for storing log files. q ' q k '
-d	<i>x</i>	Debug options a = AGWPE network protocol client k = KISS serial port client n = KISS network client u = Redisplay non-ASCII characters in hexadecimal p = Packet hex dump g = GPS interface t = Tracker beacon o = Output controls such as PTT and DCD i = IGate h = Hamlib verbose level. Repeat for more.
-q	<i>x</i>	Quiet (suppress output) options , d = Omit decoding of APRS packets.
-t	<i>n</i>	Text colors. 1 = normal, 0 = disable text colors.
-x		Send transmit level calibration tones.
-U		Print UTF-8 test string and exit.
-S		Print symbol tables and exit.

-a *n* Print audio device statistics each *n* seconds. **Q**

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

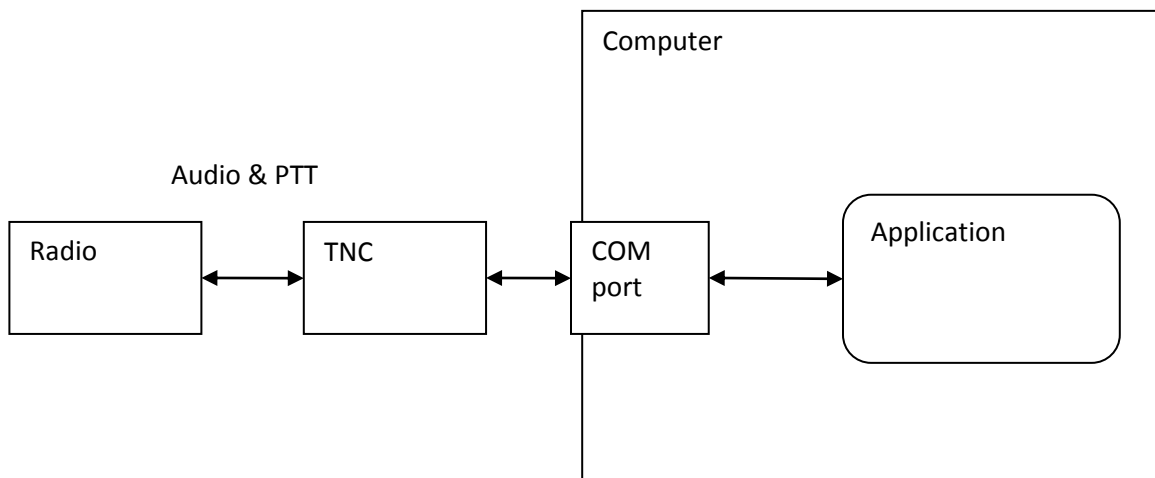
- **-** Standard input.
- **Q..** *q* *q* **Q** *κ* **'**

The Software Defined Radio section contains some examples.

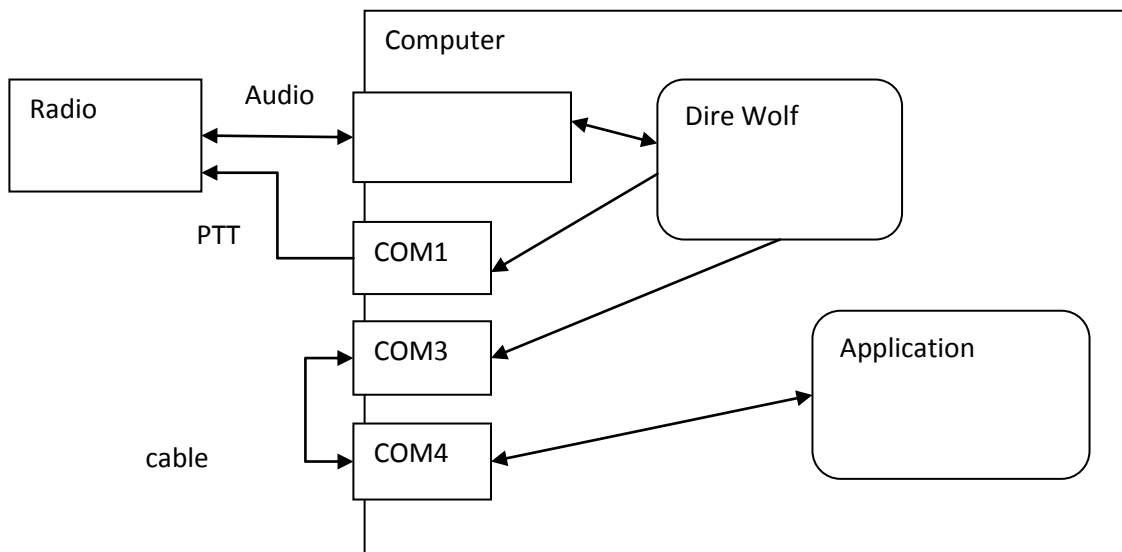
10 Advanced Topics - Windows

10.1 Install com0com (optional)

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

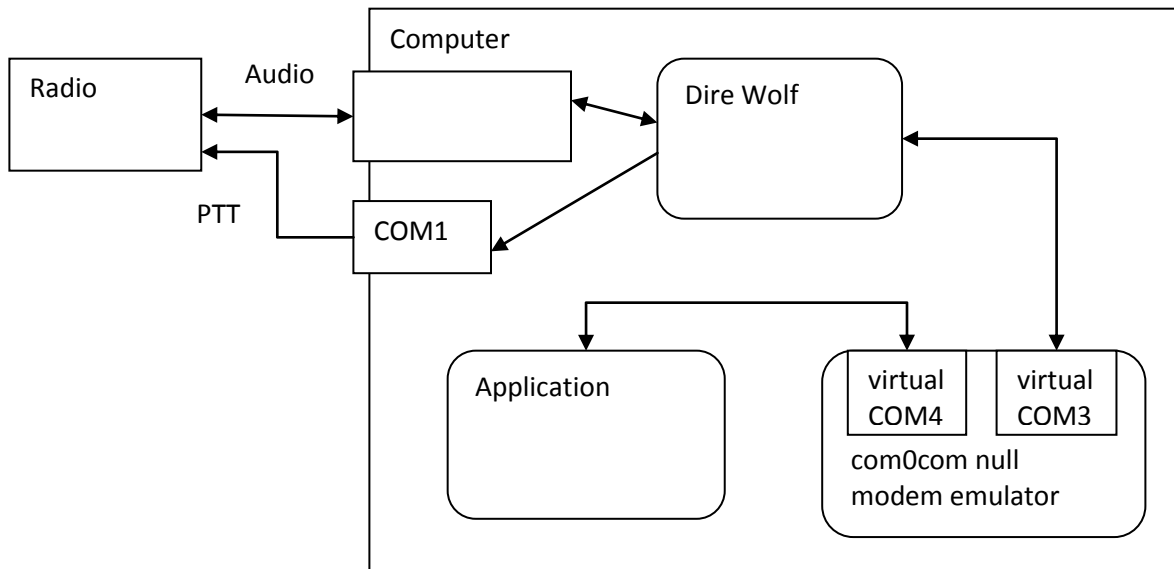


Dire Wolf is a software replacement for a separate TNC. One way of using it is illustrated below.



The packet radio application expects to find a TNC on COM4. COM3, connected to Dire Wolf, behaves as if the data 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 not necessary if you only want to use the “AGW TCPIP socket interface” or KISS over a network connection.

<http://sourceforge.net/projects/com0com/>

at the time this is being written.

8 5 - 9: 8- ' ise,

Follow the instructions for installation.

This creates two virtual serial ports named CNCA0 and CNCB0. In this example we will rename them to COM3 and COM4. If you already have a COM3 or COM4, use other numbers and make the appropriate substitutions in all of the configuration steps.

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, exactly as shown:

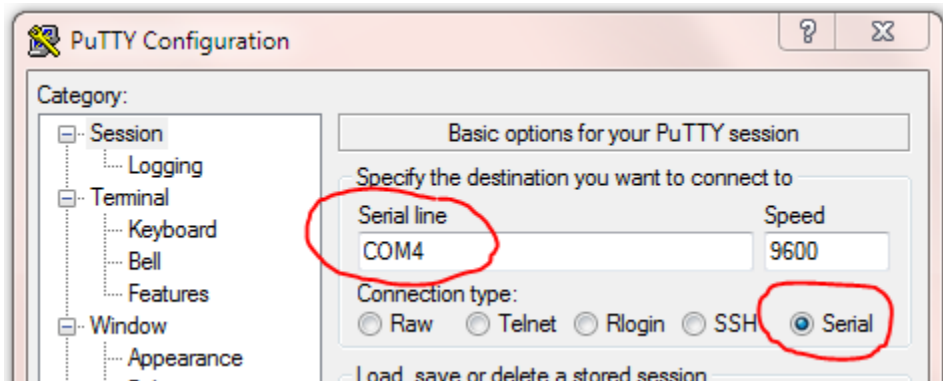
It is very important that you apply the options as shown. Without them, Dire Wolf might hang trying to write to COM3 if nothing is connected to COM4.

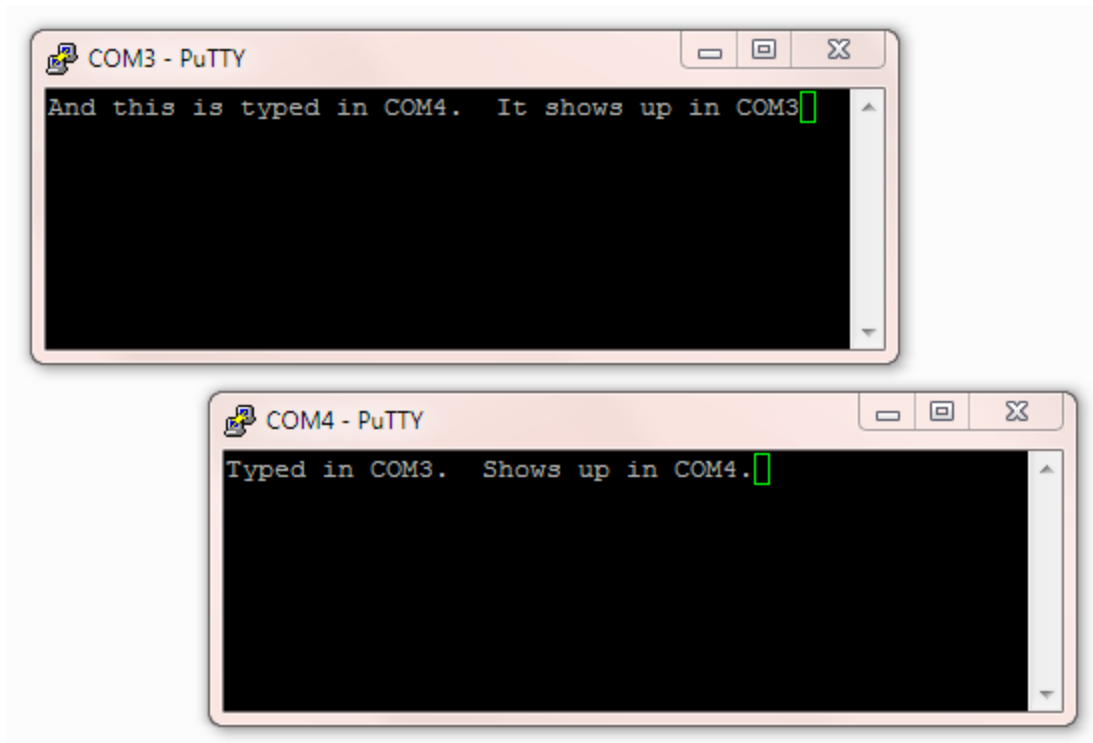
On Windows XP, you can verify correct operation by starting up two different instances of HyperTerminal.

Start → All Programs → Accessories → Communications → HyperTerminal

Connect one to COM3 and the other to COM4 (or other pair used in earlier setup). Anything typed into one should show up in the other.

Unfortunately, HyperTerminal is not available on Windows 7. See <http://windows.microsoft.com/en-us/windows/what-happened-hyperterminal#1TC=windows-7> PuTTY is a good alternative.





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

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.

10.2 Build Dire Wolf from source on Windows (optional)

The Windows version contains prebuilt executable files
people might want to. Here is how.

‘ Some

The Windows version is built with the MinGW compiler from <http://www.mingw.org/>
k -specific Makefile.

win

The result should be several new executable files including

q ‘ ‘

11 Receive Performance

11.1 WA8LMF TNC Test CD

See separate document, with similar name.

11.2 Evolution

Decoder A is the original performance at the cost of greater CPU requirements.

Decoder B [really A] but it handles only the default case of 1200 baud data and 44,100 sample rate. Decoder C offers a considerable CPU time reduction for ARM processors (Raspberry Pi, Beaglebone, etc.) but difference Intel x86 type processors.

Decoder	Packets decoded from WA8LMF test CD, track 2	Relative amount of CPU time required.	Comment
A	963	44	Same as earlier versions.
B	964	50	New for version 0.9
C	970	53	
D	923	33	300 baud results. It is not intended for use with 1200 baud.
E	988	67	New in version 1.2.
F	963	42	Mostly benefits microprocessor systems without vector processing. Not much benefit for x86 PC. Only for 1200 baud, 44100 sample rate.
A+	975	49	New in version 1.2
B+	982	56	
C+	984	60	
E+	1008	70	
F+	975	49	

Decoder D is a better APRS Packet Demodulator. It explains the problem and a couple solutions.

E 3422 q
you are using a really old slow PC, κ q q: you could try
CPU power. The configuration command would be like this:

Q q Q '
n option which uses less

This is now the default when running on the Raspberry Pi.

11.3 1200 Baud hardware TNC comparison

Here we compare 1200 baud decoder performance against two popular hardware based solutions. This

3'4

q

q

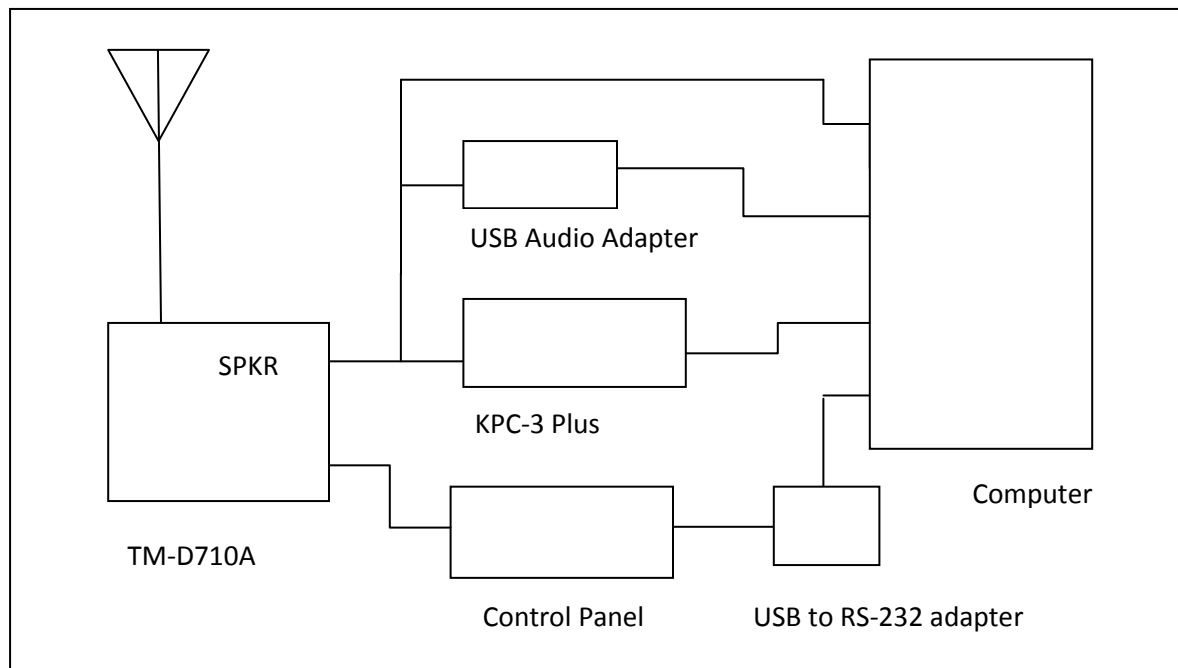
'

For this experiment we need:

- Antenna, outside on the roof.
- A cheap USB Audio Adapter (<http://www.adafruit.com/product/1475>)
- 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 cables, RS232-cables.
- PC running Ubuntu Linux.

Connect up everything as shown below.

modems, I used the speaker output because that is probably the more typical usage.



11.3.1 Prepare KPC-3 Plus

Using some sort of terminal emulator application, such as minicom, connect to `/dev/ttyS0`. Disable any 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:

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

11.3.2 Prepare D710A

q Q L 34 [Q] ' q
with menu 604.

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

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

11.3.3 Prepare Dire Wolf

In this test we are using Linux so that device name syntax is shown.

Two different configuration files were prepared. The first (direwolf.conf0) will use the default audio input on the motherboard.

Note that attempted bad bit fix-up is disabled so we count only error-free frames. This provides a fair apples-to-apples comparison against the other systems without this feature.

Prepare a second configuration file (direwolf.conf1) like this.

This provides the more typical usage with the default FIX_BITS value. A \$5 external USB Audio Adapter is being used to dispel the rumor that you need an expensive sound card for good results.

Start up two different Dire Wolf instances, with different configuration files, in different windows.

11.3.4 Compare them.

test fixture with command line arguments like this

Each command line argument is a serial port name or a TCP port number. Notice how we use two
q ' q
the results.

Packets are collected from 4 different sources and printed side-by-side in columns for each TNC. A gap means that TNC did not decode the frame that others did.

It starts off looking like this with the first couple packets being received by everybody.

' q 52 ' '

' and we find these totals...'

11.3.5 Summary

If we give the highest number a score of 100% and scale the others proportionally, the scores are:

- 100% Dire Wolf, single bit fix up
- 93% Dire Wolf, error-free frames only
- 70% Kantronics KPC-3 Plus
- 67% Kenwood TM-D710A

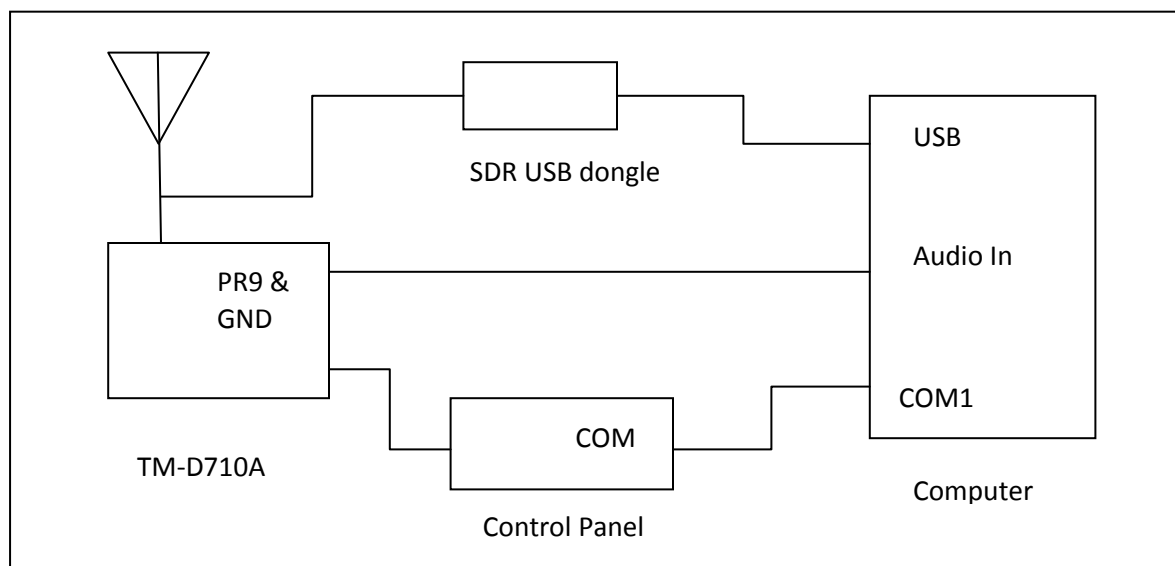
The exact proportions will vary depending on what stations you happen to hear.

11.4 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. In this case, we use an indoor antenna so we can get weak signals with the transmitter only a few blocks away.



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.

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 Kenwood TH-D72A transmitting beacons at the maximum rate (every 0.2 minute) at EL power.

Prepare D710A

Tune to 144.99 MHz. Use the TNC button
Enable the COM port with menu 604.

q Q L [Q] ‘

Using some sort of serial port terminal emulator application, such as minicom, connect to /dev/ttyS0. Disable any sort of digipeater settings or beaconing so it is not distracted by trying to transmit. We also
q Q L 34: q q
this command:

Enable monitoring:

Exit from the terminal application.

11.4.1 Prepare Dire Wolf, first instance

Be sure to use Dire Wolf version 1.2 or later. In this test we are using Linux and the system board
' : ' -96 with the
following:

When a data speed and no tones are specified, it uses N9GH/G3RUH style encoding.
The default of 3 is turned off so we get a fair apples-to-apples comparison.
Start up one instance of Dire Wolf like this.

11.4.2 Prepare Dire Wolf, second instance

This one will be using an SDR dongle rather than a sound card. Prepare another configuration file, direwolf.conf-sdr, with the following:

It is not necessary to set the modem characteristics because this will be done on the command line.
Start up the SDR and Dire Wolf in a single command line like this:

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).

The astute reader might notice the strange frequency 144.982 when we are using 144.99. It seems the SDR frequency is off a little. This was necessary to get similar + and - swings around the center frequency. Your results might be different.

When 144.99 was used, we see an imbalance like this:

When tuned too far the other way, at 144.977, we see:

IMPORTANT POINT:

The position of the vertical bars and underscores have much different meanings for 300, 1200, and 9600 baud operation.

- 300 Baud HF SSB, multiple tone pairs. (config file: MODEM 300 5@40)

Here we have 5 completely separate demodulators with different tone pairs. Each of the decoded signal positions corresponds to a different tone pair. Mistuning of an SSB signal will shift the audio frequencies up or down. The position in the middle indicates proper tuning. Positions toward the left mean the audio tones are too low.

- 1200 Baud AFSK, mu q q ' (config file: MODEM 1200 E+)

The character positions correspond to different gain ratios between the mark and space filters. Toward the right means more gain for the space tone.

- 9600 Baud K9NG/G3RUH, multiple slicing levels. (config file: MODEM 9600 +)

In this case, it becomes a tuning indicator. I think this is due to a DC bias from the SDR being off frequency or an artifact of discriminator non linearity on the edges but need to study this in more detail.

11.4.3 Compare them.

Start up a recording so we can go back and try more experiments with this data at a later time.

qq

K

It starts off with all receiving the same thing while the transmitter is near.

As the signal gets weaker, the left and right columns are missing some that are in the middle column.

What happened there? It looks like garbage. When listening to 9600 baud we occasionally see random noise that just happens to have a valid CRC. Remember that there are only 65536 possible values for the CRC and sometimes random noise will just happen to match. Below is what it looked like in the first receiving window. At the end we will subtract these from the totals.

The transmitter was out of range for a few minutes. When it starts to come back in range, the middle is the first to start receiving it properly.

Once again, random noise just happened to have a valid CRC. We will subtract this from the final totals. This is what it looked like in the receiving window:

As the transmitter returns home, all three can hear it properly.

Total scores, subtracting frames that look like garbage:

$$37 + 6 = 43$$

$$45 - 2 + 6 = 49$$

$$27 - 1 + 6 = 32$$

This is not the most realistic test scenario - only one transmitter is involved - but it does provide useful data for comparison.

11.4.4 Results

If we give the highest number a score of 100% and scale the others proportionally, the scores are:

- 100% Dire Wolf, audio input on system board.
- 88% Kenwood TM-D710A
- 65% Dire Wolf, software defined radio (SDR) adapter.

The exact percentage should not be taken too seriously because they can be manipulated by the amount of time spent in the zone where Dire Wolf could receive the signal but the TNC inside the TM-D710A could not.

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

Q. After getting an initial GPS fix, the HT was inside for quite a while where it would lose the GPS signal. Did the radio decide to shut off the GPS and use the last known location even when out walking around?

11.5 One Bad Apple Don't Spoil the Whole Bunch... ("FIX_BITS" option)

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. 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 as soon as they came out of 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 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 [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. For larger frames this could be seconds rather than milliseconds.

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 further refinement was to run the results through an additional sanity check. A good AX.25 frame will have:

- An address part that is a multiple of 7 bytes.
- Between 2 and 10 addresses.

- | | |
|------|--|
| 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 |

What about non-APRS frames? There is a configuration setting to perform a less stringent sanity check for general AX.25. The control and protocol bytes, of the frame, are not tested. The information part is

A less strict sanity check makes it more likely for corrupted data to get through.

Digipeater WB6JAR-10 audio level = 23 [TWO_SEP] .__
[0] N6VNI-14>APRS,WB6JAR-10*,WIDE,QIDE-6!:3356.05N/11758.61Wk Geo & Kris LaHabra,CA

Digipeater WB6JAR-10 audio level = 23 [NONE] _.:|
[0] N6VNI-14>APRS,WB6JAR-10*,WIDE,WIDE!:3356.05N/11758.01Wk Geo & Kris LaHabra,CA

N6QFD-9 audio level = 14 [TWO_SEP] .__
[0] N6QFD-9>GPSTJ,WIDE2-
2:\$GPRMC,020114,A,3409.7103,U,11804.0209,W,14.6,89.2,231105,13.5,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,14.6,89.2,231109,13.5,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.

Bits changed	Version 0.6		Version 0.9		Version 1.2	
	Number of packets received <i>(+ means change from previous line)</i>	Percentage increase <i>(in addition to preceding line)</i>	Number of packets received	Percentage increase	Number of packets received	Percentage increase
None	965	---	976	---	988	
Single	+ 12	1.2	+ 21	2.1	+ 15	1.5
Two adjacent	+ 2	0.2	+ 1	0.1	+ 3	0.3
Three adjacent	+ 0	0	+ 0	0.0	+2	0.2
Two separated	+ 12	1.2	+ 24	2.4	+9	0.9

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.

Bits changed	Version 0.6	
	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.

from (digipeater heard, not original source station) and they are from all over, not just a few stations.

3 q q cept of setting a proper transmit audio level.

variety of system types are represented.

By default, only single bit fix up is enabled. You can experiment with the others with a configuration file setting. In the configuration file, specify the maximum level of correction to be attempted:

- 0 [NONE] - Don't try to repair.
- 1 [SINGLE] - Attempt to fix single bit error. (default)
- 2 [DOUBLE] - Also attempt to fix two adjacent bits.
- 3 [TRIPLE] - Also attempt to fix three adjacent bits.
- 4 [TWO_SEP] - Also attempt to fix two non-adjacent (separated) bits.

Example: Limit attempt to fixing a single bit:

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.

```
direwolf.exe - Shortcut
Digipeater WIDE2 audio level = 25 [NONE]
[0] N1ESA>TQRX9P,W1MHL,WIDE2*:`d_xl <0x1e>-/'5,)<0x0d>
MIC-E, House QTH (UHF), Kenwood TM-D700, Off Duty
N 4128.9000, W 07207.9200, 0 MPH, course 2, alt 367 ft

Digipeater W2DAN-14 audio level = 94 [NONE]
Audio input level is too high. Reduce so most stations are around 50.
[0] W10EM>APU25N,W2DAN-14*,WIDE2:=4121.98N/07212.92W<East Lyme, CT OEM {UIU32N}<
Position, EOC, UIview 32 bit apps
N 4121.9800, W 07212.9200
East Lyme, CT OEM {UIU32N}

Digipeater WIDE2 audio level = 51 [SINGLE]
[0] KA1UCQ-1>APU25N,N1RCW-3,UNCAN,WIDE2*:;MAIL_FOR *200350z4156.34N\06954.34W?W1
Object, "MAIL_FOR", INFO Kiosk (Blue box with ?), UIview 32 bit apps
N 4156.3400, W 06954.3400
W1PY-1

Digipeater W1MHL audio level = 52 [NONE]
[0] KB1LNA>T2RP4R,W1MHL*,WIDE2-1:`c&ml!-0/'4c)|!-%X'<|!w-t!|3
MIC-E, UAN, Byonics TinyTrack3, In Service
N 4220.4213, W 07110.8191, 0 MPH, course 133, alt 249 ft
|!-%X'<|
```

It is important to remember that **we are not repairing errors**, we are only changing bits until we get a valid CRC. We could be adding additional corruption instead of repairing corruption.

κ **not a validity check.** We catch things that obviously look crazy but corrupted data could get through. One person compared this to playing a game of Russian Roulette. Most of the time κ : κ ‘

This feature is being provided for those who want to experiment with it. q
information such as emergency communications. When u 3
will get occasional bad data.

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.

Note that the possibly corrupted frames are passed along to any attached applications. If you are using some other application to perform the digipeater or IGate functions, you could be passing along corrupted data.

12 UTF-8 characters

12.1 Background

ASCII, like most other computer communication, uses the ASCII character set. ASCII was developed in 1963 with 94 printable characters. As computer usage grew, different vendors started 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.

33	32	'
----	----	---

For example, the degree symbol is now:

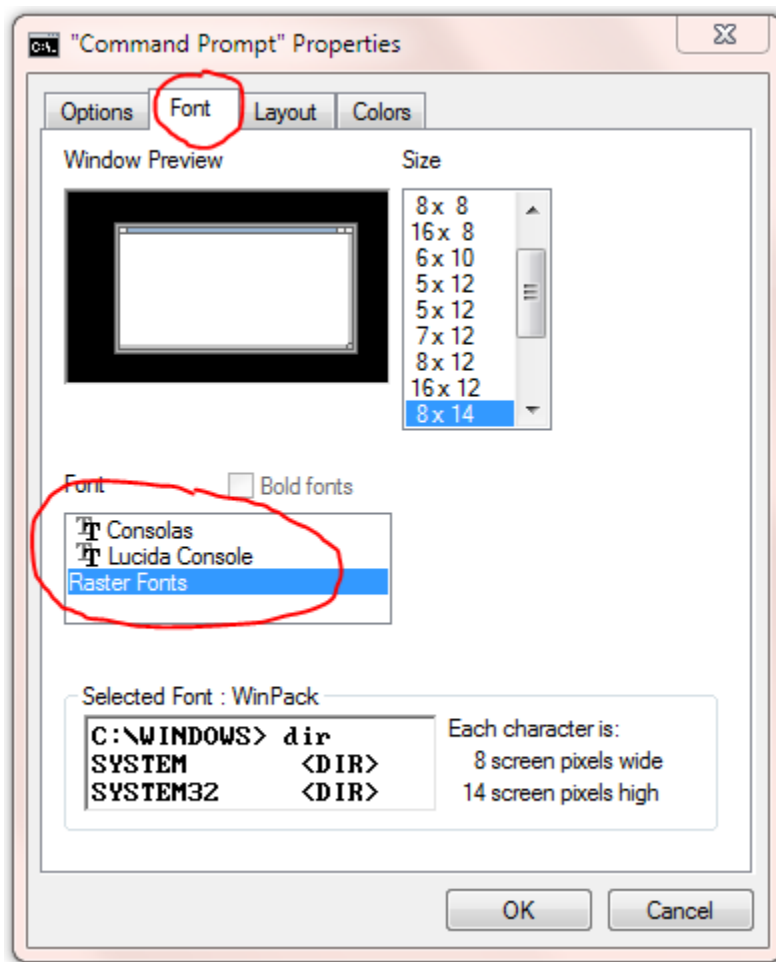
11000010 10110000

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 settings.

12.2 Microsoft Windows

The Q q has a default of ' 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 ° Füße

- Lucida Console

UTF-8 test string: mañana ° Füße

- Raster Fonts

UTF-8 test string: ma|ana T F|fe

12.3 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 **Raspbian** 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
```

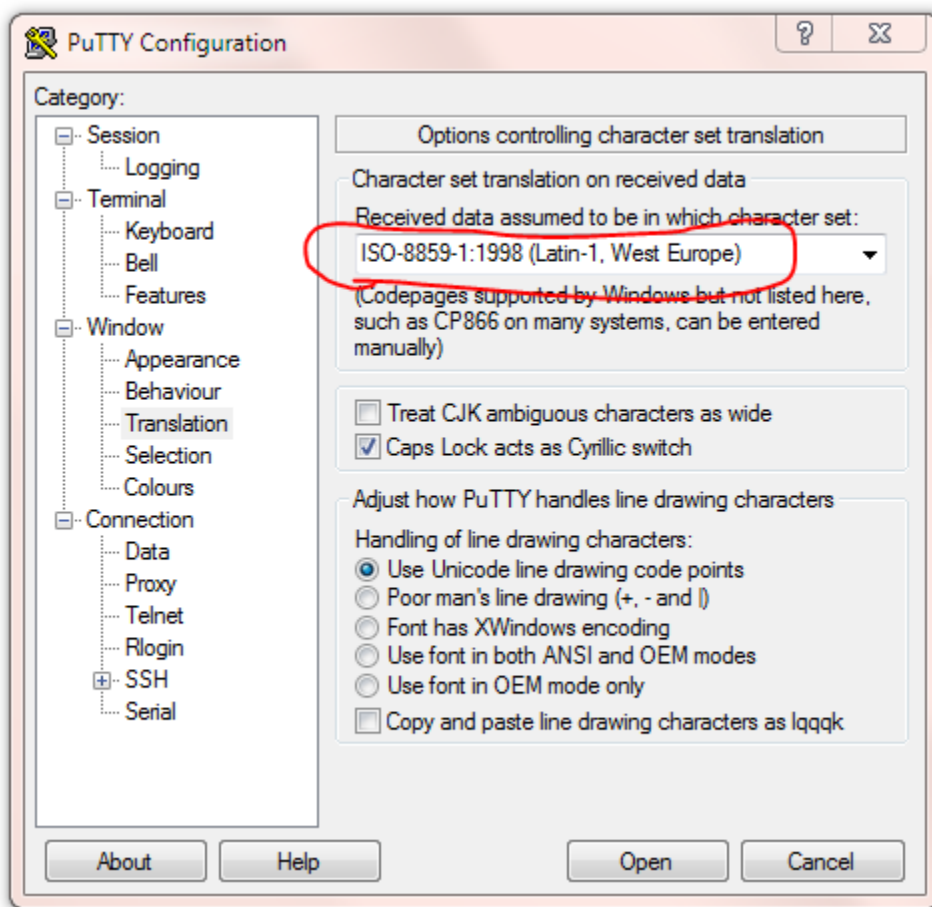
UTF-8 test string: mañana ° Füße

The defaults on **Ubuntu** are also correct. There are reports that a certain command line option is required to make **xterm** process UTF-

```
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: maÃtana Å° FÃlÃe
```

possibilities here. Google for something like linux terminal utf-8 for more help.

12.4 Debugging

-d u

q

ning 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.

q

13 Other Included Applications

The `qk` is available in the Linux version.

`qq` ' `q`

13.1 aclients – Test program for side-by-side TNC performance comparison

`aclients` is used to compare how well different TNCs decode AX.25 frames at the same time. The Receive Performance section contains a couple examples.

13.2 atest - Decode AX.25 frames from an audio file

`atest` is a test application which decodes AX.25 frames from an audio recording. This provides an easy way to test Dire Wolf decoding performance much quicker than normal real-time.

13.3 decode_aprs - Convert APRS raw data to human readable form

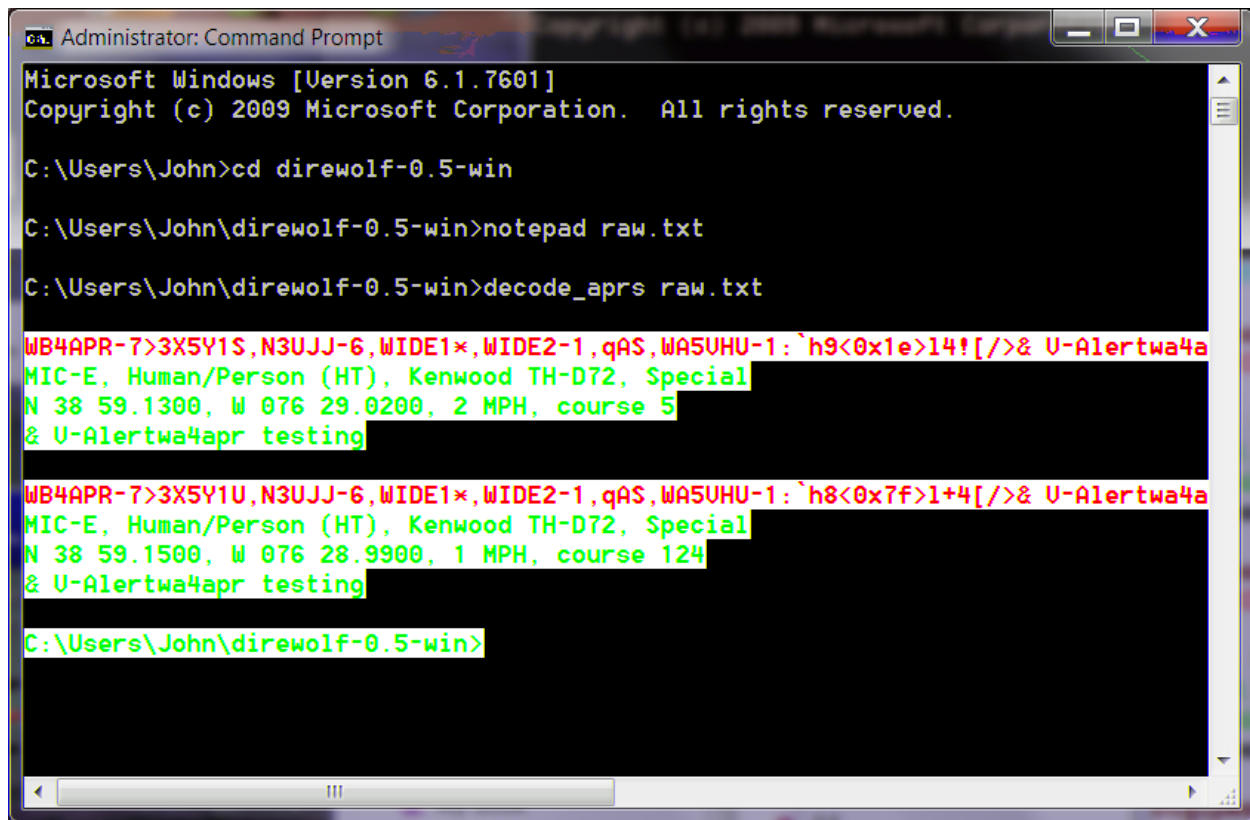
`decode_aprs` is useful for understanding sometimes obscure APRS packets and finding errors.

Suppose you 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>I4![/>& V-Alertwa4apr testing=
WB4APR-7>3X5Y1U,N3UJJ-6,WIDE1*,WIDE2-1,qAS,WA5VHU-1:`h8<0x7f>I+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.



```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\John>cd direwolf-0.5-win
C:\Users\John\direwolf-0.5-win>notepad raw.txt
C:\Users\John\direwolf-0.5-win>decode_aprs raw.txt

WB4APR-7>3X5Y1S,N3UJJ-6,WIDE1*,WIDE2-1,qAS,WA5UHU-1:`h9<0x1e>14![/]>& U-Alertwa4a
MIC-E, Human/Person (HT), Kenwood TH-D72, Special
N 38 59.1300, W 076 29.0200, 2 MPH, course 5
& U-Alertwa4apr testing

WB4APR-7>3X5Y1U,N3UJJ-6,WIDE1*,WIDE2-1,qAS,WA5UHU-1:`h8<0x7f>1+4[/]>& U-Alertwa4a
MIC-E, Human/Person (HT), Kenwood TH-D72, Special
N 38 59.1500, W 076 28.9900, 1 MPH, course 124
& U-Alertwa4apr testing

C:\Users\John\direwolf-0.5-win>
```

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>`.

13.4 gen_packets - Generate audio file for AX.25 frames

gen_packets is a test application which converts text to AX.25 audio for testing packet decoders.

It is very flexible allowing a wide range of audio sample rates, data speeds, and AFSK tones. It will even generate the scrambled signals commonly used for 9600 baud operation.

13.5 ll2utm, utm2ll - Convert between Latitude/Longitude & UTM Coordinates

These are explained in the separate APRStt Implementation Notes.

13.6 log2gpx - Convert Dire Wolf log files to GPX format

13.7 text2tt, tt2text – Convert between text and APRStt tone sequences

These are explained in the separate APRStt Implementation Notes.

14 Questions & Feedback

https://groups.yahoo.com/neo/groups/direwolf_packet/info is a good place to share information with other users. You might find your questions have already been answered here.

You can contact the author directly at wb2osz *at* Comcast *dot* net. Be sure to mention the version number and whether you are using Windows or Linux.