Cheap & Simple Digipeater for WINLINK using a TinyTrack4

Gordon Gibby KX4Z
August, 2016
Software Version Information:

TinyTrack4 Firmware:	alpha 0.68
UZ7HO soundmodem.exe:	0.94 or 0.95
WINLINK EXPRESS:	1.4.0.0 (August, 2016)
RMS PACKET	2.1.22.0 (latest version as of writing)
AGWPE	2013.1.27.2

Contents **Page**

Why this might be useful 1
Why this approach was chosen 2
Digipeaters versus NODES 2
How to Put It All Together 3
Winlink Client Setup 4
RMS Server Setup 6
Configure RMS Packet 9
Configure the Tiny Track 4 Digipeater 11
How to Observe This System Working 17
(A small intro into how to read packet lingo)
Typical Throughput 18

Why this might be useful

For emergency communications that are detailed, lengthy, or contain emotionally charged details, digital email can be the fastest, most appropriate and best transmission method. Digital techniques including WINLINK have far higher throughput than hand-copied or typewritten voice. For tactical quick messages, voice is great. For long detailed precise lists, voice is not great.

Emergency communications volunteers using VHF/UHF tools such as WINLINK EXPRESS and the WINLINK packet set of tools (including PACLINK and RMS PACKET) may have difficulty spanning their entire area of service with one server location, due to terrain & obstructions. The ability to use a DIGIPEATER or NODE to digitally relay the packets in both directions can make impossibly distant communications become quite workable, at the expense of increased time needed to move the information (because it has to be relayed) and thus reduced throughput.

Packet (AX.25) communications were a great development in ham radio. The schemes used to check for errors, request repeated packets, and acknowledge them are complex. My understanding of the AX.25 system is growing, but limited. If you are serious about using packet communications, the

information in the following explanations may be inadequate and you may want to consult additional explanations of packet. (Protocol: https://www.tapr.org/pub_ax25.html)

Why this approach was chosen

Digipeaters have been around for a long time. The technology is well worked-out and one can purchase an off-the-shelf Kantronics TNC that can easily be set up to do digipeating. However, they are not cheap, and if you are working over a larger service area and might want to have 4, 5, 6 or more emergency digipeaters available for emplacement (possibly over different frequencies), that can get to be quite pricey. You can also purchase used TNC's; however several of the ones that I purchased appeared to be "dead in the water." I wanted to find inexpensive but workable methods of creating digipeaters that could be maintained at volunteers' homes using simple outside antennas (perhaps hanging from tree limbs), which could easily be repaired or maintained by volunteers. I also wanted to figure out how to build inexpensive digipeaters that might even use just a $25 Baofeng UV5RA handitalkie (perhaps operated from a solar panel) that could be easily and rapidly put into service at temporary locations after a catastrophe to provide service to an area that previously had no such digital service. (E.G., deployed to provide help to a mutual-aid or distant disaster area, previously unserved with digital.) All of these ideas are 180 degrees away from the usual method of placing high quality expensive (and reliable!) equipment at commanding heights, but which if damaged in a disaster situation, are difficult to reach and repair.

Digipeaters versus Nodes

Digipeaters, while cheap, have a disadvantage. The packets they handle are not error checked at each "stop along the way". Instead, they are simply forwarded along and any missing or corrupted packets aren't discovered in the intermediate locations, but only at the "far end" where a request for a replacement has to be originated and then travel back the complete digipeater trail. A NODE system checks packets at each NODE and fixes any problems right then and there. That's an advantage that is gained by using more capable processing equipment. I'm also working on solutions for NODES, but this paper discusses an inexpensvie DIGIPEATER solution.

Now How To Put It Together

The TinyTrack4 is a very inexpensive (approx $75 pre-constructed[1]) TNC intended for APRS location packet broadcasting. The packets needed for those broadcasts are un-numbered, unconnectd, and are sent out without any intention to replace "missing" packets. They are simply sent out and if they fail, they fail. However, the TNC has a KISS interface mode and can be utilized as a capable TNC with any higher level software that can utilize a KISS mode TNC. That software can be simple or extensive; it can be communications, digipeater or even NODE software.

The goal of this article, however, is to use the digipeater capabilities of the TinyTrack4 – without the long term need for any additional computer. Simply connect the TinyTrack4 to a VHF FM transceiver and it can be a digipeater.

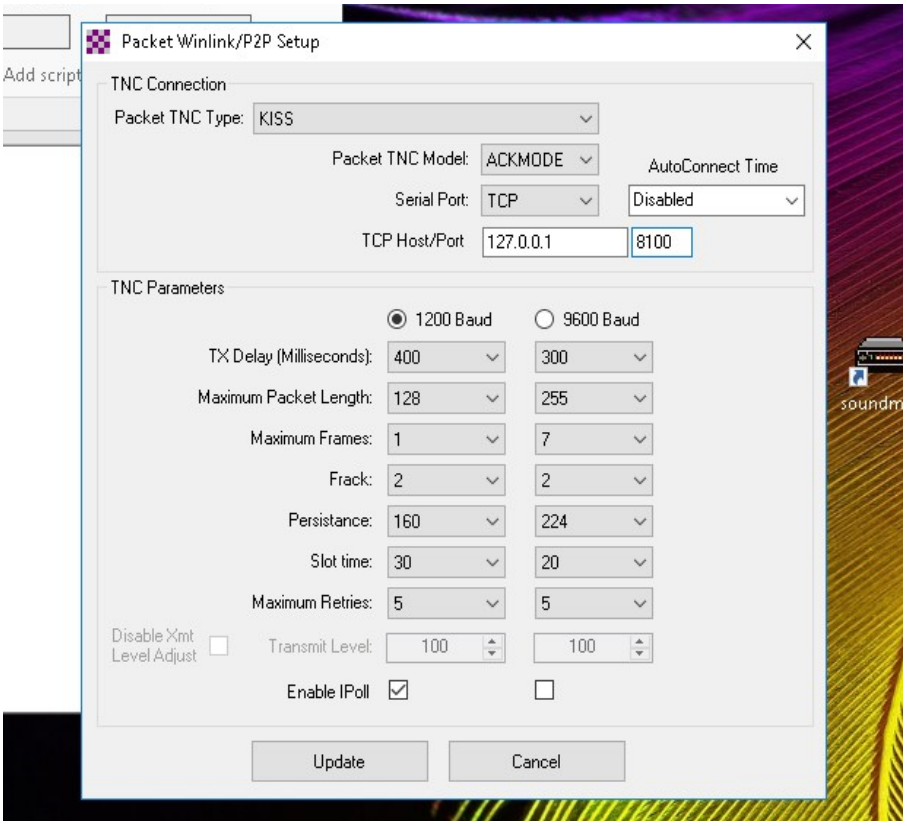
> **NOTE: MY TINYTRACK4 FIRMWARE (alpha 0.68 firmware) APPARENTLY HAS A BUG WHICH I HAD TO WORK AROUND.** It routinely fails to capture the "second" packet of a group of packets sent to it. Thus it never digipeats out that packet. This leads to a lot of requests for, and responses of RESENDS of packets. This probably results from the fact the TinyTrack4 was not developed primarily for "connected" numbered, error-corrected packets utilized with WINLINK. APRS sends out a SINGLE packet. The way to get around this "bug" is to set "Maximum Frames" [allowed to be sent before an ACKnowledgement] equal to ONE. Then the WINLINK system sends out a single frame, gets a short ACK packet, and sends the next packet. While this isn't the most efficient system, it WORKS. And it is way way faster than voice communications!!!
>
> *Note that this adaptation/workaround will NOT prevent the client from using other digipeaters, or connecting directly to a RMS PACKET, or using other RMS PACKET servers. It simply makes the digipeater system operate in a simple-minded way with which the digipeater succeeds.*

In this article we will discuss three radio installations to help beginners get this sort of system going;

> **CLIENT:** this is the individual user, who wants to be able to send and receive email (and even attachments) via the WINLINK system.
>
> **DIGIPEATER**: This is the system of the TinyTrack4 connected to a VHF FM transceiver (in my case, either an ICOM-28 or a Baofeng UV5RA).
>
> **SERVER**: this is the WINLINK RMS PACKET server software, connected to a VHF FM transceiver on the same frequency as the first two stations, and also connected either to the Internet or to an RMS RELAY (which may be connected to a RMS TRIMODE software which has an HF transceiver and can forward the email out over the HF long distance connections)

WINLINK CLIENT SETUP:

I happen to use sound-card based SIGNALINK digital interface. On my client, I use UZ7HO's soundmodem.exe, because it allows me to very easily see the signals AND the packet address and header information. I've used this with cheap transceivers such as the Baofeng UV5RA (be sure to turn STE (option 35) OFF and RP-STE (option 36) OFF, so that the receiver-transmit switchover delays are minimized for packet back and forth) and with much higher powered Yaesu FT-2900R's. To reduce RFI, you will want to put a few 3" loops in the USB cable between the computer and the Signalink and/or some ferrites; and you'll want to do the same for the cable between the Signalink and the mic/speaker/Ptt connections on the radio. It is very useful to purchase higher-quality USB cables that include SHIELDING. (I have also used hadware based TNCs successfully). The following demonstrates my setup within WINLINK EXPRESS for the PACKET connections:



Packet TNC Type: Note that I'm using UZ7HO soundmodem with a Signalink USB interface to the transceiver. Soundmodem.exe is connected as a KISS device, using TCP/IP on port 8100. You'll need to set your parameters to match whatever TNC you are using – which could be connected by TCP/IP (even on a different computer) – or by serial port.

Packet TNC Mode: ACKMODE (not sure what exactly this means, might mean more acknowledgment of packets)

Serial Port/TCP Host/Port – these will be different if you are using a serial-port connected TNC

TNC Properties:

TX Delay (millisenconds): 400 milliseconds will hopefully be enough time for your

transceiver to get ready to transmit.

Maximum Packet Length: This controls what this station SENDS; it cannot control what it RECEIVES. Usually you'll want to use the same value on the CLIENT as the SERVER. You don't set this on the Digipeater. If you are having troubles, start with 32 and work up to 64 and then possibly 128. Generally, higher numbers mean faster throughput for larger messages.

Maximum Frames: The number sent before we get any ACKnolwedgement. For the firmware in my TINYTRACK4 (with the bug described above) , I have to use ONE. It will work with a higher number, but it will just uselessly send packets that are going to be lost. 2 or 3 would be much more common in normal installations.

FRACK: 2

Persistence: change if you are a guru

Enable Ipoll: turn ON

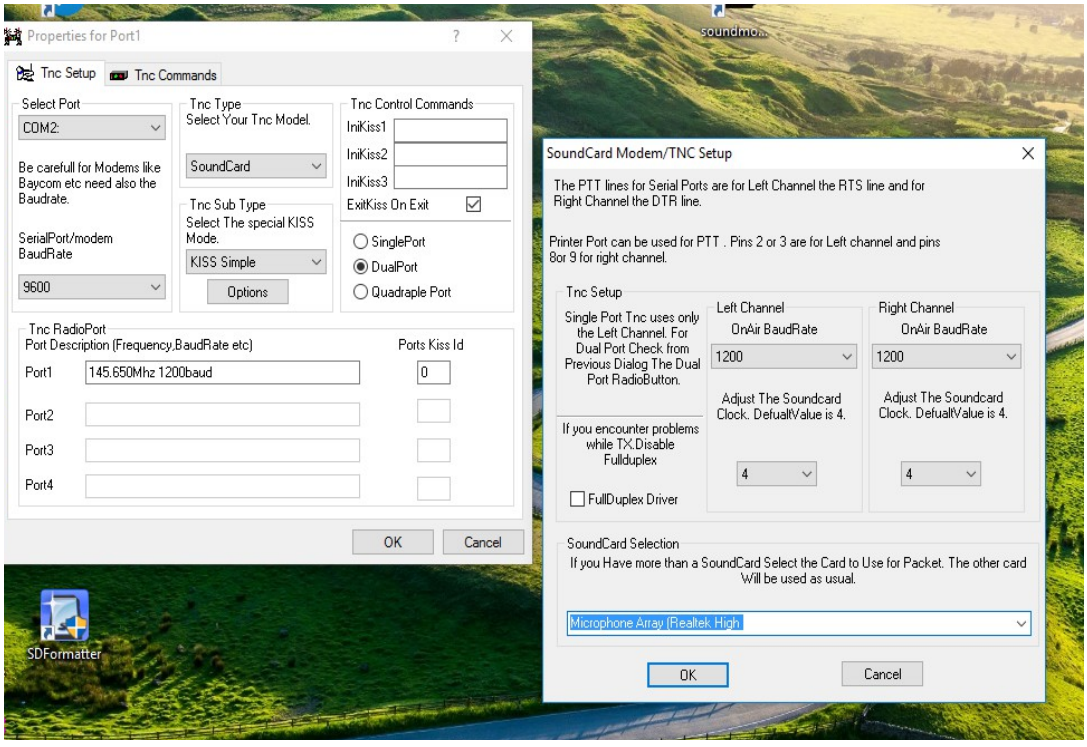
RMS SERVER Setup:

Because of what seems to be a bug in my version of RMS PACKET (in addition to the bug in the TinyTrack4) , I can't make it send packets of small size or limited MAXFRAMES before acknowledgement. At the suggestion of John Wiseman, I switched to use AGWPE software as an intermediary between my interface (in this case, a Signalink, but could have been a TNC) and the RMS PACKET software-- specifically because this gave me back control of MAXFRAMES and maximum frame length. The interface between RMS PACKET and AGWPE is best done via a TCP/IP socket on the computer, which could be at many different sockets. Typically 8000 is utilized for AGWPE.

Hopefully, at some point after this writing, RMS PACKET (& in particular, its KISS interface software) may be altered so that the normal controls for these functions have normal control, in which case the interposition of AGWPE may no longer be necessary.

1. Download AGWPE (the free hamware verion) from: http://www.sv2agw.com/downloads/

2. When installed, this application sits on the Windows Task Bar, in my case, in the "chevron" overflow area that must be clicked to gain access to the icon for the AGWPE app. You can either right or left click to gain access to its menus.
3. From the AGWPE menu, Click on "Properties" to gain access to following setup screens; you will need to "ADD PORT" and then configure that port as shown below.

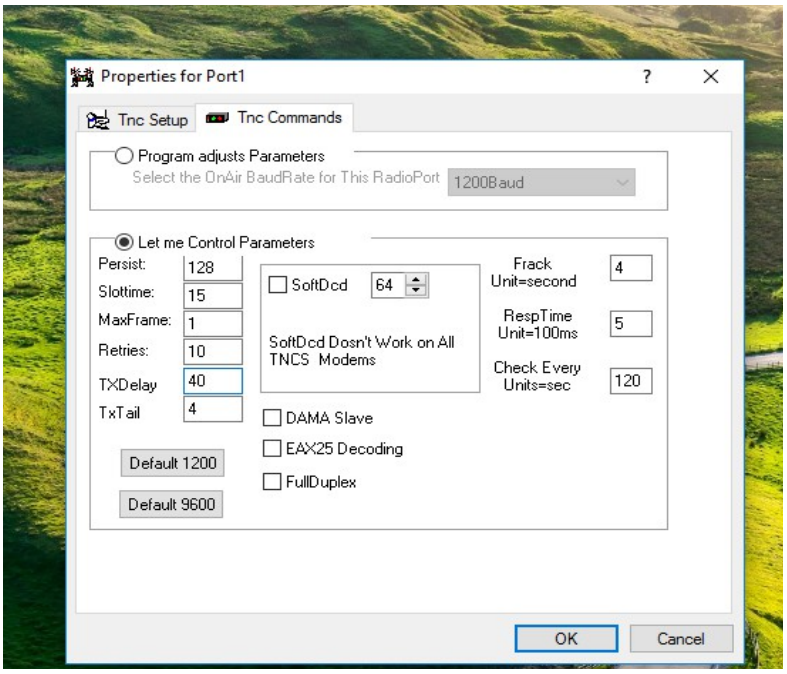


Because I use a TCP/IP interface between RMS PACKET and AGWPE, I was not concerned by the "Select Port" portion of the TNC SETUP.
In the TNC Type, select your TNC (in my case, "Soundcard")
On the next screen I had to select which soundcard (note that the figure shows "microphone" but on my actual server, the correct choice includes a reference to the USB Signalink)

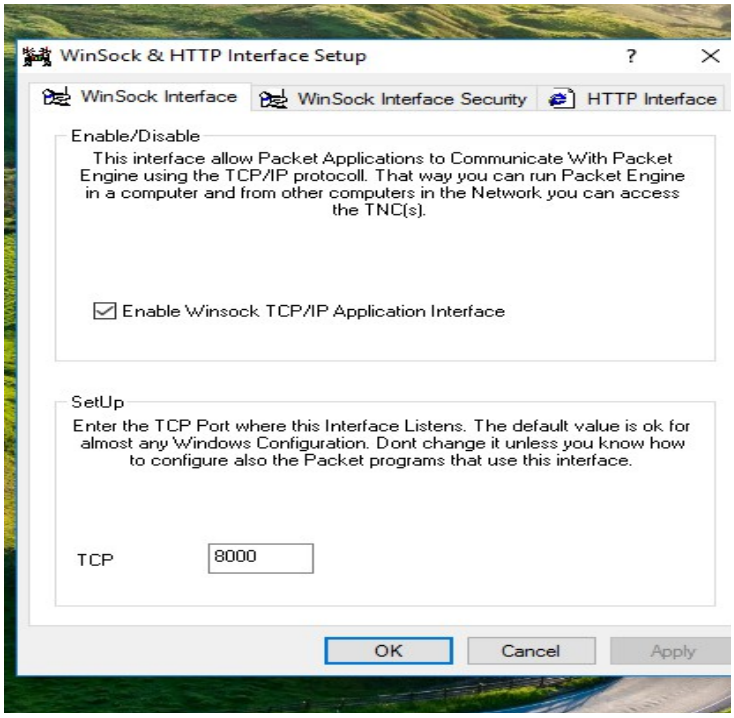
I chose half-duplex.

Next click on the TNC Commands Tab and choose "Let Me Control Parameters". The following are the ones I think need adjustment:

MaxFrames Set to 1 to cover the bug in the TT4
TxDelay Might want to increase that to 40 or 50 (400 or 500 msec for the transmitter to get going) until you have proven your transmitter turns on more quickly than this, particularly for older or handheld equipment.



Now, go back to clicking on the AGWPE Icon, select the menu item, “Setup Interface” to get the TCP/IP port configured. In my case, I'm using Port 8000



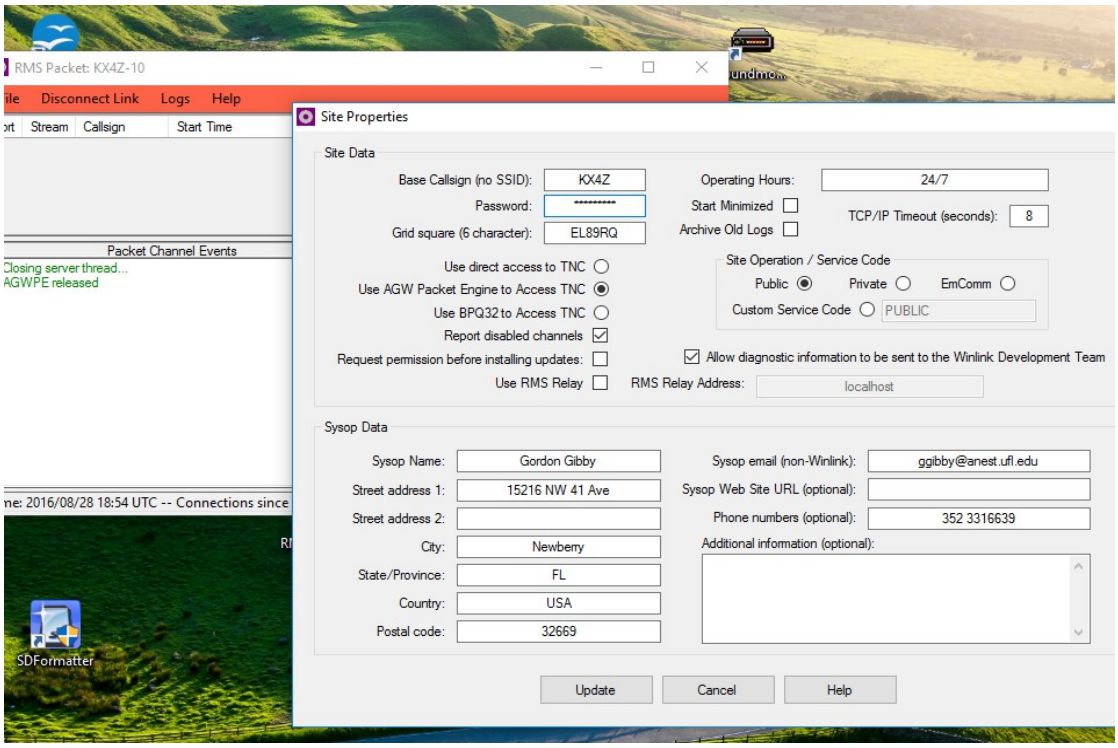
This should get your TNC (or soundcard/software-based TNC) set up properly with the required MaxFrames etc for the the TT4 digipeater.

You may have to do some trial and error to find the best setting of the received audio on your transceiver (*AGWPE has some visual tools to help with thi*s) and also of the level of transmitted audio going into your mic connection. I generally leave the “computer audio speaker” at 100% and adjust the Signalink TX volume. The standard way to set the transmit audio (and thus, the deviation) is to listen with a separate receiver and adjust the TX audio upwards until the received signal no longer gets louder, and then back off a bit back into the “linear” range. You might also compare with the received audio perception of loudness from other stations Overly strong audio is a bad thing.

Configure RMS PACKET.

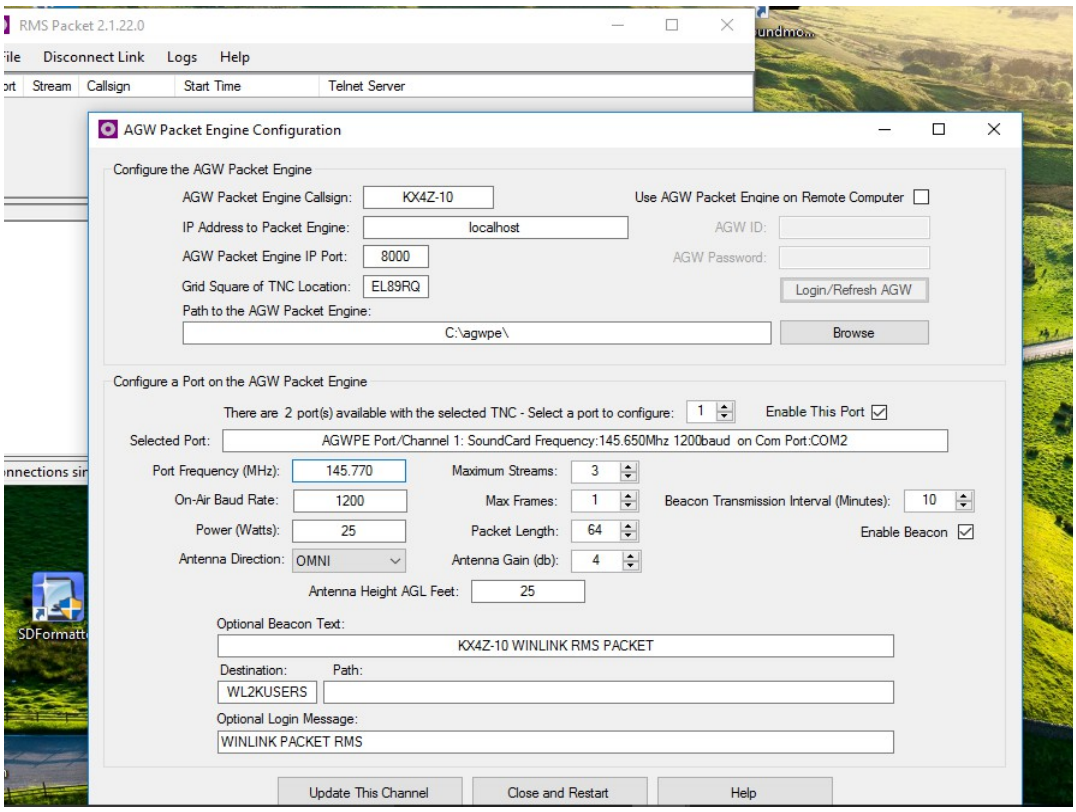
Left Click **FILE | Site Properties** and configure your WINLINK settings for connection to the WINLINK system:

To use the workaround discussed herein, click on **Use AGW Packet Engine to Access TNC**. The remainder of the information is standard WINLINK configuration information.



Left Click **FILE | Packet Channels** and further configure your server. The most important settings are:

AGW Packet Engine Port MUST match what you set within AGWPE for the TCP/IP interface!
Path to the AGW Packet Engine --- use Browse button to get this correct.
Must choose a port (usually #1) and enable it.
Much of the page is notification stuff that may show up on the winlink.org site to advertise your site.
Some of it has to do with setting a Beacon (transmitted information every so many minutes; be brief!)
Set the **Max Frames** to 1 and **Packet Length** to 64 (hoping that an upgraded software will pay attention to these settings.... If you find that 128 works properly, up the Max Packet Length to 128 for better performance). Note that the Packet Length only controls the maximum of what the server SENDS, not what it is sent.



CONFIGURE THE TINY TRAK 4 DIGIPEATER

You will want to read the TinyTrack4 Quick Start Guide as a minimum and gain access to the (optional) graphical user interface program to read and write programming to the device, which is found under the Primary Tiny Track 4 files" zip file. At this writing the application was "Tiny Track 4 Alpha Config 0.68"

You will need to construct a DB9 connector to go to your mic/speaker/ptt connections from your transceiver, and also to provide power (6-15 volts). A fuse or even a small lightbulb in series to limit the current in the event of a short, might be a nice protective touch. Byonics sells cables or you can make your own.

For the purposes of this digipeater, you do not need the GPS device.

In order to program this device, you'll need either a female-female null modem cable (pin 5 to 5, 2 to 3 and 3 to 2) or make one yourself. (And in my case, I had to also use a USB-serial converter as my computer had no serial port, only USB's.) The configuration of the Tiny Track is described on their web page.

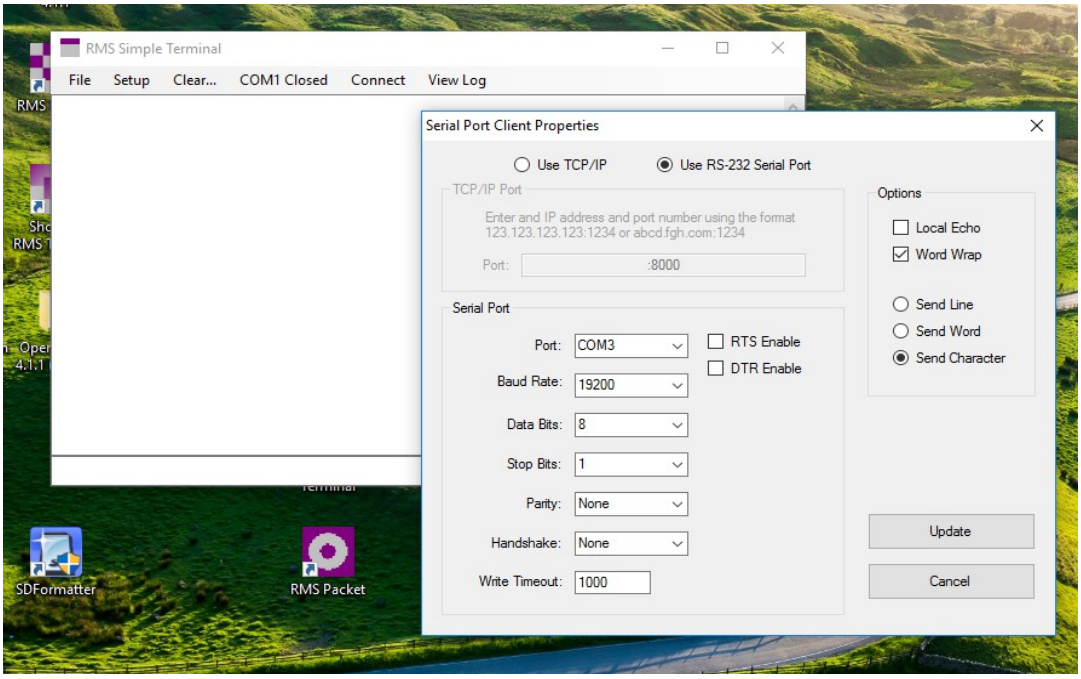
I first configured mine using RMS SIMPLE TERMINAL and the process described in their Tiny Track 4 Quick Start Guide. This makes it relatively easy to set the RX and TX gains (through software).

If you don't know which COMx port on your computer you're using, go through SETTINGS or CONTROL PANEL (depending on version ofWindows) to get to Device Manager and click on COM ports to see which are active.

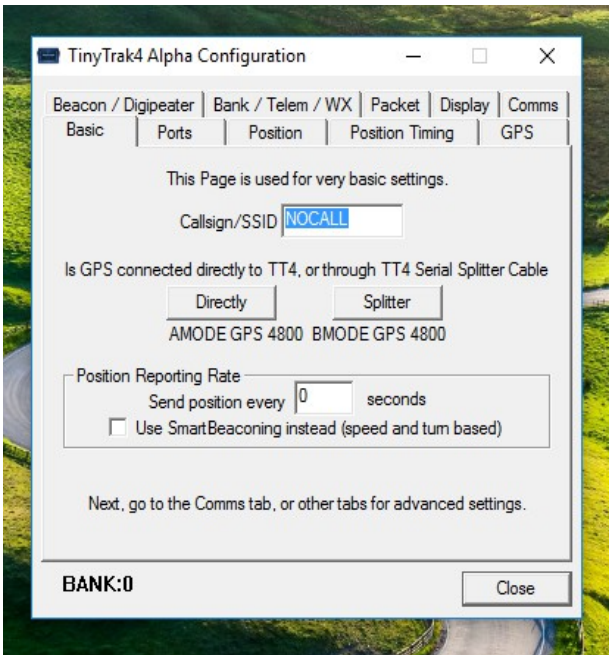
Be sure to set your terminal program for 19200, 8n1, no handshaking. In RMS SIMPLE TERMINAL you have to each time "connect" the serial port. You'll get used to that. After powering up the TT4 hit ESC three times to wake up your connection. Then use the commands (HELP / DISPLAY / RESTORE QUIT etc, and other information in the quick start to set the necessary settings. Remember that you can scroll the RMS SIMPLE TERMINAL window to see settings that have run off the screen.

Be certain to set the squelch (just to the point of quietness) on your transceiver and then carefully follow the instructions for setting RXAMP and TXLEVEL. Your particular RXAMP setting will depend on the audio gain of your transceiver; if this is adjusted by the volume control – make MARK so you'll put it back to the right setting and for long term, consider gluing that control!

If you overdrive the transmitter, your transmissions may well be unfruitful.



Later on, you might want to use the GUI program for touchups, as it is quick and easy. Go to http://www.byonics.com/tinytrak4/ and download the "Primary Tiny Track 4 files" zip file; and/or the Quick Start Guide. You can use RMS SIMPLE TERMINAL or any other terminal program, or the GUI applicatiion described above.



For both the terminal-based programming and the GUI-based, cycling the power off->on to the TT4 is what make it available to "talk." Using the terminal, you have to hit ESC three times quickly to get it to talk to you.

Setting the receive and transmit audio levels is EXTREMELY important. I first made the mistake of not having the squelch quieted on my transceiver when setting the RX level. Use your client station to generate some packets to allow you to measure the RXAMP and set them according to the instructions.

My preferred settings are below (obtained with RMS SIMPLE TERMINAL and a display command, with the ones that I think are important or useful bolded:

:display
BANK is 0
P300 is FALSE
TXTDISP is FALSE
NODISP is FALSE
PPATHING is FALSE
DMSDISP is FALSE
MICETMV is FALSE
ENTS is FALSE
TELHIRES is FALSE
TELVOLT is TRUE
TELTEMP is TRUE
PREEMPT is FALSE
DIGIID is FALSE
WXPOS is TRUE
TELREAD is TRUE
FRAWDISP is FALSE
HRAWDISP is FALSE
WYPTXT is FALSE
PKTICOM is TRUE
PKTOCOM is TRUE
RPATHDISP is FALSE
LEDS is TRUE
PAVPEN is FALSE
DEC96 is FALSE
DDIST is FALSE
HEADERLN is FALSE
DMETRIC is FALSE
SOFTRST is FALSE
MSGCMD is FALSE
MSGCAP is FALSE
LRNTPS is FALSE
GPSCHK is FALSE
INTCLK is TRUE

DECSTAT is FALSE
DIGIMY is TRUE
TOSV is TRUE
TALT is FALSE
TSPEED is TRUE
TIMESTAMP is TRUE
TIMEHMS is TRUE
SBEN is FALSE
TSWPT is TRUE
AMODE is TEXT
BMODE is GPS
ABAUD is 19200
BBAUD is 4800
BNKMODE is 0
SSIDROUTE is 0
ALTNET is APTT4
MYCALL is KX4Z-7
PATH1 is WIDE1-1
PATH2 is WIDE1-2
PATH3 is RELAY
TSTAT is /TinyTrak4 Alpha
BTEXT is digi status voltage is ^V emp is ^7
BPERIOD is 120 **2 minutes...lengthen for real usage**
TXD is 60
MTXD is 10
PERSIST is 65
SLOTTIME is 15
QUIET is 1
TRNKMODE is 0
CDMODE is TONES
CDLEVEL is 20
TXLEVEL is 100
TXTWIST is 50
RXAMP is 25
GWAYLEN is 9
GWAYMODE is NMEA
GRELAYBITS is 1
GRELAYRATE is 0
GKRELAY is 0
LOCATION is 1234.5678N 12345.6789W
GALT is 1000
TSYMCODE is >
TSYMTABLE is /
STATUSRATE is 1
PPERIOD is 0
MPPERIOD is 0
SBSSPEED is 5

SBFSPEED is 60
SBSPERIOD is 1800
SBFPERIOD is 90
SBTANGLE is 27
SBTSLOPE is 255
SBTTIME is 5
MMSG is 0
TSOFFSET is 17
TDAO is 0
TPROTOCOL is APRS
TPSWITCH is 0
TPERIOD is 0
TVOLTTWK is 128
TTEMPTWK is 128
WPERIOD is 0
ALIAS1 is WIDE1
ALIAS2 is WIDE2
ALIAS3 is RELAY
DUPETIME is 0
FILTERCALL is

How to Observe This System Working:

You need a monitor screen to observe the packets. I happen to use soundmodem.exe (UZ7HO) on my client, so it is easy to observe the packets. Watch the callsigns. You'll see a packet

client callsign to *target callsign* via *digipeater callsign* (followed by packet information)

That should immediately be followed by a packet showing (by a * after the digipeater) that a packet was generated by the digipeater in response –

client callsign to *target callsign* via *digipeater callsign** (followed by the same information, possibly with a LEN (length) one greater

Watch the Sx data inside the packet headers – the packets are numbered 0-7 and then around again. You should see the packets flow in a smooth order without REJ (reject) being issued by the target.

If you see lots of REJ and packets being sent over and over....then you have a problem.

A very nice and easy-to-read explanation of how to read packet headers can be found on one page here: http://www.soundcardpacket.org/8headers.aspx

Here's an example of a monitor capture by soundmodem from the client station. Client (KX4Z) transmissions are in RED. Digipeater (KX4Z-7) and Target (RMS PACKET KX4Z-10) are in black. Comments are added.

1:Fm KX4Z-7 To APTT4 Via WIDE1-1,WIDE1-2,RELAY <UI R Pid=F0 Len=41> [16:20:11R] [+++]
digi status voltage is 13.6V emp is 3.13v
1:Fm KX4Z To KX4Z-10 Via RELAY <SABM C P> [16:20:29T] SABM – request for connection P = immediate reply
1:Fm KX4Z To KX4Z-10 Via RELAY* <SABM C P> [16:20:30R] [+++] digipeated
1:Fm KX4Z-10 To KX4Z Via RELAY <UA R F> [16:20:31R] [+++]
1:Fm KX4Z-10 To KX4Z Via RELAY <I C P R0 S0 Pid=F0 Len=42> [16:20:32R] [+++]
KX4Z-10 Experimental WINLINK PACKET EMAIL **Packet #0 (S0) is sent by the RMS PACKET PID=F0 frame contains simple ascii text**

1:Fm KX4Z-10 To KX4Z Via RELAY* <UA R F> [16:20:32R] [+++] **F= packets received OK**
1:Fm KX4Z-10 To KX4Z Via RELAY* <I C P R0 S0 Pid=F0 Len=42> [16:20:33R] [+++]
KX4Z-10 Experimental WINLINK PACKET EMAIL **Packet #0 (S0) is sent by the digipeater**

1:Fm KX4Z To KX4Z-10 Via RELAY <RR R F R1> [16:20:33T] **RR = ready to receive Client KX4Z indicates wants #1 (R1)**
1:Fm KX4Z To KX4Z-10 Via RELAY* <RR R F R1> [16:20:34R] [+++] **digipeater repeats the request, leng 34 because of the ***
1:Fm KX4Z-10 To KX4Z Via RELAY <I C P R0 S1 Pid=F0 Len=21> [16:20:35R] [+++]
[WL2K-3.2-B2FWIHJM$] **RMS PACKET responds with Packet #1, length 21**

1:Fm KX4Z-10 To KX4Z Via RELAY* <I C P R0 S1 Pid=F0 Len=21> [16:20:36R] [+++]
[WL2K-3.2-B2FWIHJM$] **Gets didpeated,**

1:Fm KX4Z To KX4Z-10 Via RELAY <RR R F R2> [16:20:37T] **responds it is ready for #2**
1:Fm KX4Z To KX4Z-10 Via RELAY* <RR R F R2> [16:20:38R] [+++] **The request for 2nd packet is relayed**

1:Fm KX4Z-10 To KX4Z Via RELAY <I C P R0 S2 Pid=F0 Len=14> [16:20:39R] [+++]
;PQ: 41357024 **RMS PACKET responds with Packet #2 (S2)**

1:Fm KX4Z-10 To KX4Z Via RELAY* <I C P R0 S2 Pid=F0 Len=14> [16:20:40R] [+++]
;PQ: 41357024 **Which is diigipeated by the relay**

1:Fm KX4Z To KX4Z-10 Via RELAY <RR R F R3> [16:20:40T] **And the client says Ready for #3**
1:Fm KX4Z To KX4Z-10 Via RELAY* <RR R F R3> [16:20:41R] [+++] **Which gets digipeated as well**

1:Fm KX4Z-10 To KX4Z Via RELAY <I C P R0 S3 Pid=F0 Len=24> [16:20:43R] [+++]
SanDiego CMS via KX4Z > **So the RMS Packet sends out #3**

1:Fm KX4Z-10 To KX4Z Via RELAY* <I C P R0 S3 Pid=F0 Len=24> [16:20:44R] [+++]
SanDiego CMS via KX4Z > **Which gets digipeated**

1:Fm KX4Z To KX4Z-10 Via RELAY <RR R F R4> [16:20:44T] **And the client sayd read for #4**

Had there been an error, you would see REJ and the a Request for a repeat packet. Actually pretty easy to decipher.

TYPICAL THROUGHPUT

MAXFRAMES = 1
Digipeated ONCE with strong signal levels

My measurements indicate the following characters per second throughput. Sending (from the client) was slower than receiving from the server (unknown why)**. By comparison, very accurate handwritten or typewriten voice transmisisons are unlikely to exceed 30 words per minute, or 150 characters per minute** (Messages actually transferred wre approximately 400 characters, measured after compression. Speeds are of the actual character transferred; because of compression in this particular case the "uncompressed speed" would be far higher)

Max Packet Length	Sending char/minute	Receiving char/minute
32	270	(unmeasured)
64	500	920
128	500	2000

1 http://www.byonics.com/tinytrak4/