Host Interface Spec for Winlink Supported Protocols/TNCs

Rick Muething, KN6KB John Wiseman, G8BPQ
Revised: 0.5 Nov 9, 2017

Please send questions, comments or suggestions to:
Rick Muething rmuething@cfl.rr.com or
John Wiseman john.wiseman@cantab.net

1. Scope and Contents

This document defines the basic Host interface specification for those wishing to use TNC supported protocols in the Winlink system. It is meant to cover (except where noted) both physical hardware/firmware TNCs and virtual (Software) TNCs. It describes how the host program (Client or Server Host Application) interfaces to the TNC driver. Two primary communication mechanisms are used: 1) serial (USB, RS-232/422, Bluetooth) and 2) TCPIP.

This is not meant to be a universal driver for all protocols. The commands for a specific TNC are usually protocol specific. There may however be some commands that are similar or the same across various protocols and TNC implementations. Each TNC or protocol (Packet, Robust Packet, Pactor (1-4), ARDOP etc.) will also have their own spec detailing the commands supported and their usage.

2. Host to TNC Connection

The host to TNC connection is designed to work using one of two basic data protocols:

1) Serial bit stream interface. Examples include RS-232, RS-422, USB using a virtual serial COM port and Bluetooth using a virtual COM port.
2) TCPIP connection either on the same computer or over a WiFi or Ethernet connection. The TCPIP interface uses two ports: one for Host to TNC and TNC to Host commands, and one (usually the next sequential port) for TNC to Host and Host to TNC data.

Serial protocol connections are described in this specification Section 3, TCPIP connections in Section 4.

3. WA8DED Host mode Host to TNC Protocol. (Serial data protocols)

The Serial data host to TNC protocol will use an expanded WA8DED host mode. (see Appendix A) This popular Host to TNC interface is used in many packet and in most Pactor modems in some form. The Preferred form is the Extended CRC Host mode interface as also supported by the SCS Pactor modems. This provide the highest security and a recovery mechanism for lost or contaminated host <> TNC serial data.

3.1 Initialization

The host program would normally use some initialization file (e.g. "TNC.ini") that resides with the host program. That file (which might be manipulated by user menus in the host program or edited directly by the user) should contain the minimum information to allow the host program to start the TNC automatically. This includes:

Serial COM Port and baud rate if using Serial interface

Or

Bluetooth interface details for pairing if the Bluetooth interface to the TNC is used.

Or

TCPIP Address and port numbers

All remaining required property values and setting may be supplied using the WA8DED host commands for the specific supported protocol (See Appendices A and B) or the TCPIP interface referenced in Appendix C.

4. TCPIP Host to TNC Interface

The TCPIP Host to TNC interfaces takes advantage of certain automatic retry and synchronization features inherent in the TCPIP protocol which speeds and simplifies the interface. This TCPIP connection can be made through the local host (between programs running on a single host) or over a TCPIP Ethernet link over a network. The host program normally would include a parameter setup menu (or ini file) with appropriate parameters for the TCPIP interface such as the Host address and TCPIP ports used. One port is used for Host to TNC and TNC to host commands. The *next* sequential port is used for data from TNC to host and host to TNC. The commands supported and data formats used may be a function of the specific protocol or Virtual TNC implementation and in general should not be expected to be consistent or compatible across various virtual TNCs and/or On-the-air protocols.

4.1 Host to TNC Commands and TNC to Host Commands and status:

TNC commands sent from the host to the TNC or from the TNC to the host are sent on the Command port and must be in the following format:

Command<Cr>

Commands are case insensitive and must be ASCII encoded and must not contain any embedded <Cr> (Carriage return) which acts as a delimiter.

4.2 Host to TNC and TNC to Host data :

The data channel is done using the next sequential TCPIP port number from the command port. Data sent from Host to the TNC and TNC to Host uses the format:

nnddddddddd

Where "nn" represents a 2 byte (binary) data count. There is no CRC used or required on a TCPIP connection. The data byte stream "ddddddddd" above may contain bytes of ***any*** binary data 00 to FF (hex) or values 0 to 255 (decimal).

Appendix A Serial Interface Protocol.

(From John Wiseman)

Overview

The Host to TNC protocol used over Serial and Bluetooth links operates in two modes, Text or Hostmode. Text mode is primarily used for initialization and testing, but can be used for simple iterative QSO's using a standard ASCII terminal program. There is no error checking on the text interface. Hostmode is used for automatic operation, and provides error detection and recovery over the serial/Bluetooth line. ARDOP Hostmode is based on the SCS CRC Hostmode (as used in the PTC and Dragon Controllers) which is itself an extension of WA8DED Hostmode.

Host mode Protocol Overview

The protocol is polled master/slave, with a single bit sequence toggle to detect lost or duplicated frames. The host program is the master and the TNC the slave. The polling frequency isn't defined, but a maximum interval of 100mS is recommended to minimize latency.

The link is considered Idle when the master has received an in-sequence response to its previous transmission. The master can transmit at any time the link is idle. If it has data to send, it sends it, otherwise it sends a General Poll message. The slave will respond with a list (possibly empty) of channels with data available. The master then polls each channel in the list.

If the master doesn't receive a valid response from the slave in a reasonable time, it will repeat its last transmission. If it doesn't get a response after a reasonable number of retries it will report a link failure to the higher level software, which will abort any transmission in progress, then try to reestablish the link.

If the master receives an out of sequence response following a timeout and retry it will assume a delayed response from the slave, discard the repeat, and continue to wait for a valid frame. If it receives an out of sequence response at any other time it will assume a protocol error and reset and restart the link.

If the slave receives an out of sequence message it will assume that its previous response was lost, discard the message and resend the previous response unchanged.

Packet Formats

All packets have a two byte header of 0xAAAA and a two byte CRC-16 checksum on end. The top bit of the Opcode field is a sequence toggle.

The basic packet format is:

```
-----------------------------------------------
|Header|Chan|Opcode| Payload           | CRC |
```

```
 ---------------------------------------------
 |AA AA | XX |  XX  | XX XX XX .... XX |XX XX|
 ---------------------------------------------
```

Payload can have two formats, either a Null terminated ASCII string or a Byte Count of 0 to 255 followed by 1 to 256 bytes of data.

There are two opcodes for Host to TNC packets, and eight for TNC to Host, though not all are used in ARDOP Native Mode.

From Host to TNC

Opcode 0 - Data
Opcode 1 - Command

From TNC to Host

Opcode 0 - Response Success (no data follows)
Opcode 1 - Response Success, followed by null terminated message
Opcode 2 - Response Failure, followed by null terminated message
Opcode 7 - Data, preceded by (length-1)

Channel 32 is used for Commands, Channel 33 for Data and Channel 34 for Debug information. Typical messages
are shown below (control fields in Hex). Note that the Command format is used for hostmode
protocol level commands. ARDOP commands (such as "ARQCALL") are sent as data on the
Command channel. The Debug channel is for information to be written to a debug log.

General Poll

```
 ---------------------------------
 |Header|Chan|Opcode|Payload| CRC |
 ---------------------------------
 |AA AA | FF |  01  | 00 47 |XX XX|
 ---------------------------------
```

Response is a null terminated list of channels with available data. Value is Channel plus 1

```
 ---------------------------------
 |AA AA | FF |  01  | 21 00 |XX XX|            Channel 32 has data (0x21 = 32 + 1)
 ---------------------------------
```

Poll to TNC Command Channel

```
---------------------------------
|Header|Chan|Opcode|Payload| CRC |
---------------------------------
|AA AA | 20 |  01  | 00 47 |XX XX|
---------------------------------
```

Response is an async response message from TNC

```
-------------------------------------------------
|Header|Chan|Opcode| Payload              | CRC |
-------------------------------------------------
|AA AA | 20 |  07  |<0F>c:NEWSTATE ISS <0D>|XX XX|
-------------------------------------------------
```

Command to TNC

```
-------------------------------------------------
|Header|Chan|Opcode| Payload                | CRC |
-------------------------------------------------
|AA AA | 20 |  00  |C:MYCALL G8BPQ <00>     |XX XX|
-------------------------------------------------
```

response

```
-------------------------------------------------
|AA AA | 20 |  01  |c:MYCALL now G8BPQ <00>  |XX XX|
-------------------------------------------------
```

Data to be transmitted

```
-------------------------------------------------
|Header|Chan|Opcode| Payload                | CRC |
-------------------------------------------------
|AA AA | 21 |  00  |<12>D:(Len)Message to Send|XX XX|
-------------------------------------------------
```

response

```
-------------------------------------------------
|AA AA | 21 |  00 |                         |XX XX|
-------------------------------------------------
```

Received Data from TNC

Data to be transmitted

```
-------------------------------------------------
|Header|Chan|Opcode| Payload                | CRC |
-------------------------------------------------
|AA AA | 21 |  07  |<13>d:(Len)ARQMsg Received|XX XX|
```

There is no response to inbound frames, apart from the implied ACK of the next host frame having an inverted tog

Appendix B: SCS CRC Hostmode

From John Wiseman

See the PTC-IIIusb Manual Chapter 10 for details of CRC Hostmode .

WA8DED Packet Format

There are two data formats, null terminated or counted.

Byte 0	Channel Number
byte 1	Type Code (see table below)
byte 3-end	Null Terminated string (max length 255)
or	
byte 3	Info Length - 1
byte 4-end	Info (length 1 to 256)

SCS CRC Hostmode adds two byte of 170 to the front and a crc-16 checksum to the end. It also defines the top bit (bit 7) of the Type Code as a sequence toggle and next bit (bit 6) as a sequence reset flag. Data transparency is ensured by adding a null byte after any occurence of 170 in the message or crc.

Command Codes

Host to Tnc

CHANNEL	CODE	DESCRIPTION
n	0	Information (preceeded by length-1)
n	1	Command (preceeded by length-1)

Tnc to Host

CHANNEL	CODE	DESCRIPTION
n	0	Success (nothing follows)
n	1	Success (message follows, null terminated)
n	2	Failure (message follows, null terminated)
n	3	Link Status (null terminated)
n	4	Monitor Header (null terminated)
n	5	Monitor Header (null terminated)
n	6	Monitor Information (preceeded by length-1)
n	7	Connect Information (preceeded by length-1)

Appendix C: TCPIP Interface

The TCPIP command interface is an optional TNC<>Host interface (software or virtual TNC or hardware/firmware TNC) where commands and data are sent and received via TCPIP ports instead of a single serial interfaces. The TCPIP protocol includes automatic CRC checking, first in first out and automatic repeat as part of the TCPIP protocol. Such an interface may be local (same computer), on a Local Area Net, or on a remote computer via a standard TCPIP (internet) connection. ASCII commands and binary data are sent to the TCPIP address/ports. This is intended to be used in a computerized host interface (normally not keyboard commands as in 1980's type Packet TNCs). TCPIP commands and data are not CRC protected (as in the serial interfaces of Appendices A and B) as the TCPIP protocol itself has built in CRC and auto repeat if necessary. All commands are plain (readable) text using 7 bit ASCII character encoding. Commands are always terminated with a <Cr> (carriage return) and must not contain a <Cr> character. Commands that do not include parameters will cause an "echo back" of the command and current parameters as described below. Commands or replies from the TNC always begin with "c:" (lower case c) and end with a "<Cr>. *Commands and data are no longer acknowledged by a "RDY" response as in prior versions. Commands are no longer required to be preceded by a "C:" or "c:" as in prior versions*. All commands and data will respond with an echo of the command followed by the current value as below:

Host CMD Sent:	**TNC Response:**
Command sent to query current value:	
ARQBW<Cr>	ARQBW 2000MAX <Cr>
Command sent to change current value:	
ARQBW 1000MAX<Cr>	ARQBW NOW 1000MAX<Cr>
Response to command that neither accepts or returns a value:	
ABORT<Cr>	ABORT<Cr>

All commands and data are buffered and processed on a first in first out basis. Host programs should normally sequence commands based on the responses above. If a command is incorrect or otherwise generates a TCPIP fault it will prompt a FAULT response allowing the host program to repeat the command or take alternate action. Command execution and reply is usually < 100 ms but a few commands may take a few seconds to execute.

When a command from the host is received with improper syntax or parameter values the Virtual TNC will reply with "c:FAULT <fault description + Echo back of command><CR This should cause the Host to correct or repeat the command**.**

It is recommended a debug logging option be made available which will log received commands and replies to aid in debugging new implementations. Data logging should also be made available for debugging to log summary (size) of data sent by the Host or

TNC. Since such logging greatly increases the size of the debug log it should only be enabled when necessary as in the integration with a new Host program.