The KISS TNC: A simple Host-to-TNC communications protocol

Mike Chepponis, K3MC, Phil Karn, KA90

ABSTRACT

The KISS[1] TNC provides direct computer to TNC communication using a simple protocol described here. Many TNCs now implement it, including the TAPR TNC-1 and TNC-2 (and their clones), the venerable VADCG TNC, the AEA PK-232/PK-87 and all TNCs in the Kantronics line. KISS has quickly become the protocol of choice for TCP/IP operation and multi-connect BBS software.

1. Introduction

Standard TNC software was written with human users in mind unfortunately, commands and responses well suited for human use are ill-adapted for host computer use, and vice versa. This is especially true for multi-user servers such as bulletin boards which must multiplex data from several network connections across a single host/TNC link. In addition, experimentation with new link level protocols is greatly hampered because there may very well be no way at all to generate or receive frames in the desired format without reprogramming the TNC.

The KISS TNC solves these problems by eliminating as much as possible from the TNC software, giving the attached host complete control over and access to the contents of the HDLC frames transmitted and received over the air. This is central to the KISS philosophy: the host software should have control over all TNC functions at the lowest possible level.

The AX.25 protocol is removed entirely from the TNC, as are all command interpreters and the like. The TNC simply converts between synchronous HDLC, spoken on the full or half-duplex radio channel, and a special asynchronous, full duplex frame format spoken on the host/TNC link. Every frame received on the IDLE link is passed intact to the host once it has been translated to the asynchronous format; likewise, asynchronous frames from the host are transmitted on the radio channel once they have been converted to HDLC format.

Of course, this means that the bulk of AX.25 (or another protocol) must now be implemented on the host system. This is acceptable, however, considering the greatly increased flexibility and reduced overall complexity that comes from allowing the protocol to reside on the same machine with the applications to which it is closely coupled.

It should be stressed that the KISS TNC is intended only as stopgap. Ideally, host computers would have HDLC interfaces of their own, making separate TNCs unnecessary. [15] Unfortunately, HDLC interfaces are rare, although they are starting to appear for the IBM PC. The KISS TNC therefore becomes the "next best thing" to a real HDLC interface, since the host computer only needs an ordinary asynchronous interface.

2. Asynchronous Frame Format

The "asynchronous packet protocol" spoken between the host and TNC is very simple, since its only function is to delimit frames. Each frame is both preceded and followed by a special FEND (Frame End) character, analogous to an HDLC flag. No CRC or checksum is provided. In addition, no RS-232C handshaking signals are employed.

The special characters are:

Abbreviation	Description	Hex value
FEND	Frame End	C0
FESC	Frame Escape	DB
TFEND	Transposed Frame End	DC
TFESC	Transposed Frame Escape	DD

The reason for both preceding and ending frames with FENDs is to improve performance when there is noise on the asynch line. The FEND at the beginning of a frame serves to "flush out" any accumulated garbage into a separate frame (which will be discarded by the upper layer protocol) instead of sticking it on the front of an otherwise good frame. As with back-to-back flags in HDLC, two FEND characters in a row should not be interpreted as delimiting an empty frame.

3. Transparency

Frames are sent in 8-bit binary; the asynchronous link is set to 8 data bits, 1 stop bit, and no parity. If a FEND ever appears in the data, it is translated into the two byte sequence FESC TFEND (Frame Escape, Transposed Frame End). Likewise, if the FESC character ever appears in the user data, it is replaced with the two character sequence FESC TFESC (Frame Escape, Transposed Frame Escape).
As characters arrive at the receiver, they are appended to buffer containing the current frame. Receiving a FEND marks the end of the current frame. Receipt of a FESC puts the receiver into "escaped mode", causing the receiver to translate a following TFESC or TFEND back to FESC or FEND, respectively, before adding it to the receive buffer and leaving escaped mode. Receipt of any character other than TFESC or TFEND while in escaped mode is an error; no action is taken and frame assembly continues. A TFEND or TESC received while not in escaped mode is treated as an ordinary data character.

This procedure may seem somewhat complicated, but it is easy to implement and recovers quickly from errors. In particular, the FEND character is never sent over the channel except as an actual end-of-frame indication. This ensures that any intact frame (properly delimited by FEND characters) will always be received properly regardless of the starting state of the receiver or corruption of the preceding frame.
This asynchronous framing protocol is identical to "SLIP" (Serial Line IP), a popular method for sending ARPA IP datagrams across asynchronous links. It could also form the basis of an asynchronous amateur packet radio link protocol that avoids the complexity of HDLC on slow speed channels.

4. Control of the KISS TNC

Each asynchronous data frame sent to the TNC is converted back into "pure" form and queued for transmission as a separate HDLC frame. Although removing the human interface and the AX.25 protocol from the TNC makes most existing TNC commands unnecessary (i.e., they become host functions), the TNC is still responsible for keying the transmitter's PTT line and deferring to other activity on the radio channel. It is therefore necessary to allow the host to control a few TNC parameters, namely the transmitter keyup delay, the transmitter persistence variables and any special hardware that a particular TNC may have.
To distinguish between command and data frames on the host/TNC link, the first byte of each asynchronous frame between host and TNC is a "type" indicator. This type indicator byte is broken into two 4-bit nibbles so that the low-order nibble indicates the command number (given in the table below) and the high-order nibble indicates the port number for that particular command. In systems with only one HDLC port, it is by definition Port O. In multi-port TNCs, the upper 4 bits of the type indicator byte can specify one of up to sixteen ports. The following commands are defined in frames to the TNC (the “Command” field is in hexadecimal):

Command	Function	Comments
0	Data frame	The rest of the frame is data to be sent on the HDLC channel.
1	TXDELAY	The next byte is the transmitter keyup delay in 10 ms units. The default start-up value is 50 (i.e.,500ms).
2	P	The next byte is the persistence parameter, p, scaled to the range 0 - 255 with the following formula: P = p * 256 - 1 The default value is P = 63 (i.e., p = 0.25).
3	SlotTime	The next byte is the slot interval in 10 ms units. The default is 10 (i.e., 100ms).
4	TXtail	The next byte is the time to hold up the TX after the FCS has been sent, in 10 ms units. This command is obsolete, and is included here only for compatibility with some existing implementations.
5	FullDuplex	The next byte is 0 for half duplex, nonzero for full duplex. the default is 0 (i.e., half duplex).
6	SetHardware	Specific for each INC. In the TNC-1, this command sets the

		modem speed. Other implementations may use this command for other hardware-specific functions.
FF	Return	Exit KISS and return control to a higher-level program. This is useful only when KISS is incorporated into the TNC along with other applications.

The following types are defined in frames to the host:

Type	Function	Comments
0	Data frame	Rest of frame is data from the HDLC channel

No other types are defined; in particular, there is no provision for acknowledging data or command frames sent to the TNC. KISS implementations must ignore any unsupported command types. All KISS implementations must implement commands 0,1,2,3 and 5; the others are optional.

5. Buffer and Packet Size Limits

One of the things that make the KISS TNC simple is the deliberate lack of TNC/host flow control. The host computers run higher level protocol (typically TCP, but AX.25 in the connected mode also qualifies) that handles flow control on an end-to-end basis. Ideally, the TNC would always have more buffer memory than the sum of all the flow control windows of all of the logical connections using it at that moment. This would allow for the worst case (i.e., all users sending simultaneously). In practice, however, many (if not most) user connections are idle for long periods of time, so buffer memory may be safely "overbooked". When the occasional "bump" occurs, the TNC must drop the packet gracefully, i.e., ignore it without crashing or losing packets already queued. The higher level protocol is expected to recover by "backing off" and retransmitting the packet at a later time, just as it does whenever a packet is lost in the network for any other reason. As long as this occurs infrequently, the performance degradation is slight; therefore the TNC should provide as much packet buffering as possible, limited only by available RAM.

Individual packets at least 1024 bytes long should be allowed. As with buffer queues, it is recommended that no artificial limits be placed on packet size. For example, the K3MC code running on a TNC-2 with 32K of RAM can send and receive 30K byte packets, although this is admittedly rather extreme. Large packets reduce protocol overhead on good channels. They are essential for good performance when operating on high speed modems such as the new WA4DSY 56 kbps design.

6. Persistence

The P and SlotTime parameters are used to implement true p-persistent CSMA. This works as follows:

Whenever the host queues data for transmission, the TNC begins monitoring the carrier detect signal from the modem. It waits indefinitely for this signal to go inactive. When the channel clears, the TNC generates a random number between 0 and 1.[2] If this number is less than or equal to the parameter p, the TNC keys the transmitter, waits .01 * TXDELAY seconds, and transmits all queued frames. The TNC then unkeys the transmitter and goes back to the idle state. If the random number is greater than p, the TNC delays .01 * SlotTime seconds and repeats the procedure beginning with the sampling of the carrier detect signal. (If the carrier detect signal has gone active in the meantime, the TNC again waits for it to clear before continuing). Note that p=1 means "transmit as soon as the channel clears"; in this case the p-persistence algorithm degenerates into the 1-persistent CSMA generally used by conventional AX.25 TNCs.

P-persistence causes the INC to wait for an exponentially-distributed random interval after sensing that the channel has gone clear before attempting to transmit. With proper tuning of the parameters p and SlotTime, several stations with traffic to send are much less likely to collide with each other when they all see the channel go clear. One transmits first and the others see it in time to prevent a collision, and the channel remains stable under heavy load. See references [1] through [13] for details.

We believe that optimum p and SlotTime values could be computed automatically. This could be done by noting the channel occupancy and the length of the frames on the channel. We are proceeding with a simulation of the p-persistence algorithm described here that we hope will allow us to construct an automatic algorithm for p and SlotTime selection.

We added p-persistence to the KISS INC because it was a convenient opportunity to do so. However, it is not inherently associated with KISS nor with new protocols such as TCP/IP. Rather, persistence is a channel access protocol that can yield dramatic performance improvements regardless of the higher level protocol in use we urge it be added to every INC, whether or not it supports KISS.

7. Implementation History

The original idea for a simplified host/TNC protocol is due to Brian Lloyd, WB6RQN. Phil Karn, KA9Q, organized the specification and submitted an initial version on 6 August 1986. As of this writing, the following KISS INC implementations exist:

TNC type	Author	Comments
TAPR TNC-1 & clones	Marc Kaufman, WB6ECE	Both download and dedicated ROM versions.
TAPR TNC-2 & clones	Mike Chepponis, K3MC	First implementation, most widely used. Exists in both downloadable and dedicated ROM versions.
VADCG INC & Ashby INC	MikeBruski, AJ9X	Dedicated ROM.
AEA PK-232 PK-87	Steve Stuart, N6IA	Integrated into standard AEA firmware as of 21 January 1987. The special commands "KISS ON" and "KISS OFF" (!) control entry into KISS mode.
Kantronics	Mike Huslig	Integrated into standard Kantronics firmware as of July 1987.

The AEA and Kantronics implementations are noteworthy in that the KISS functions were written by those vendors and integrated into their standard TNC firmware. Their TNCs can operate in either KISS or regular AX.25 mode without ROM changes. The TNC 1 and TNC-2 KISS versions were written by different authors than the original AX.25 firmware. Because of the specialized development environment used for the TNC 1 code, and because original source code for the TNC-2 was not made available, the KISS authors wrote their code independently of the standard AX.25 firmware. Therefore these TNCs require the installation of nonstandard ROMs. Two ROMs are available for the TNC-2. One contains "dedicated" KISS TNC code; the TNC operates only in the KISS mode. The "download" version contains standard N2WX firmware with a bootstrap loader overlay. When the TNC is turned on or reset, it executes the loader. The loader will accept a memory image in Intel Hex format, or it can be told to execute the standard N2WX firmware through the "H"[3] command. The download version is handy for occasional KISS operation, while the dedicated version is much more convenient for full-time or demo KISS operation.

The code for the TNC-1 is also available in both download and dedicated versions. However, at present the download ROM contains only a bootstrap; the original ROMs must be put back in to run the original TNC software.

8. Credits

The combined "Howie + downloader" ROM for the TNC-2 was contributed by WA7MXZ. This document was expertly typeset by Bob Hoffman, N3CVL.

9. Bibliography

1. Tanenbaum, Andrew S., "Computer Networks" pp. 288-292. Prentice-Hall 1981.

2. Tobagi, F. A.: "Random Access Techniques for Data Transmission over Packet Switched Radio Networks," Ph.D. thesis, Computer Science Department, UCLA, 1974.

3. Kleinrock, L., and Tobagi, F.: "Random Access Techniques for Data Transmission over Packet-Switched Radio Channels," Proc. NCC, pp. 187-201, 1975.

4. Tobagi, F. A., Gerla, M., Peebles, R.W., and Manning, E.G.: "Modeling and Measurement Techniques in Packet Communications Networks," Proc. IEEE, vol. 66, pp.1423-1447, Nov. 1978.

5. Lam, S. S.: "Packet Switching in a Multiaccess Broadcast Channel"' Ph.D. thesis, Computer Science Department, UCLA, 1974.

6. Lam, S. S., and Kleinrock, L.: "Packet Switching in a Multiaccess Broadcast Channel: Dynamic Control Procedures," IEEE Trans. Commun., vol COM-23, pp. 891-904, Sept. 1975.

7. Lam, S. S.: "A Carrier Sense Multiple Access Protocol for Local Networks," Comput. Networks, vol 4, pp. 21-32, Feb. 1980

8. B. Tobagi, F. A.: "Multiaccess Protocols in Packet Communications Systems," IEEE Trans. Commun., vol COM-28, pp. 468488, April 1980c.

9. Bertsekas, D., and Gallager, R.: "Data Networks", pp. 274-282 Prentice-Hall 1987.

10. Kahn, R. E., Gronemeyer, S. A., Burchfiel, J., and Kungelman, R. C. "Advances in Packet Radio Technology," Proc. IEEE. pp. 1468-1496. 1978.

11. Takagi, H.: "Analysis of Polling Systems," Cambridge, MA MIT Press 1986.

12. Tobagi, F. A., and Kleinrock, L. "Packet Switching in Radio Channels: Part II The Hidden Terminal Problem in CSMA and Busy-Tone Solution," IEEE Trans. Commun. COM-23 pp. 1417-1433. 1975.

13. Rivest, R. L.: "Network Control by Bayessian Broadcast," Report MIT/LCS/TM-285. Cambridge, MA. MIT, Laboratory for Computer Science. 1985.

14. Karn, P. and Lloyd, B.: "Link Level Protocols Revisited," ARRL Amateur Radio Fifth Computer Networking Conference, pp. 5.25-5.37, Orlando, 9 March 1986.

15. Karn, P., "Why Do We Even Need TNCs Anyway," Gateway, vol. 3 no. 2, September 5, 1986.

[1] "Keep It Simple, Stupid"

[2] To conform to the literature, here p takes on values between 0 to 1. However, fractions are difficult to use in a fixed point microprocessor so the KISS TNC actually works with P values that are rescaled to the range 0 to 255. To avoid confusion, we will use lower-case p to mean the former (0-1) and upper-case P whenever we mean the latter (0-255).

[3] For "Howie", of course.

NOTES on the Split-Screen Server Program by HB9CVV Sep-87

Version 2.2a of SSS, dated SEP-87

First the good news: this program is hamware, and it comes with the sources. You may do with it whatever you want - no contribution of any kind is required.

The bad news is that this file you are just reading is everything you get as documentation. And, the sources do not exhibit much of good programming style. SSS was started as a quick and dirty weekend project and has really grown since then, it is of course still dirty. It was intended for use within the family only (my mother is DK1SL), but I gave it to other hams when they wanted it and some of them seem to like it.

```
************************************************************************
*       PLEASE edit all *.PAR and *.INI files to insert YOUR call, name, qth,          *
*       beacontexts, function-key strings etc. The files supplied contain MY OWN        *
*       TEXTS and should not be used unmodified. In fact I am somewhat tired to read    *
*       MY connect message and my Functionkey texts sent by users of this program       *
*       who were apparently not able to look into things before they started to use it. *
*                                                                                        *
*       If you don't know, HOW to edit a file then get some help from a friend. If you do *
*       not know WHICH lines should be edited, I strongly propose you to read the        *
*       manuals supplied with your TNC before you go on the air. This would be of        *
*       great help to anybody, you included.                                             *
************************************************************************
```

To compile with TURBO PASCAL V3.x:

- Select KPC2400.PAS, PK-232.PAS, TNC2.PAS or TNCW.PAS as the MAIN-file.

- Select compilation to COM file.

- Set Max Heap to 0A00

- Compile.

Again, please do not forget to modify the INI, STR, and CFG files please.

=.=

The Split-Screen Server program consist of include-files with filenames prefixed "SSS_" and one main file per supported TNC.

Currently there are the following main files to serve the different TNCs:

TNC

For TNC-2 and clones. The TNC-2 has shown some problems with correct echoing when a burst of characters is sent to the TNC. To overcome this, when a line is sent, is is moved from the lower window to the upper window by the program rather than being echoed by the TNC-2. Additionally this is written in high-intensity, so you can easily distinguish between YOUR text and incoming text. That’s why ECHO OFF is sent to the TNC-2 during Init.

You can still select a version with echo from TNC, see variable Tnc_echo_on and the comments in TNC2.PAS. Note that with echo-on from TNC I managed with some tricks to get a clean echo (however I run the TNC at 2400 Bd or 9600 Bd).

TNCW

For TNC-1 or the TNC-2 with the WADDED software. Echoing by the TNC - no problems. But NO file-sending, because no Hardware-handshake available from WAG - there is no software handshake either (in normal terminal mode). So dont TRY to send files by modifying the

programm. Binary filetransfers are also not available.

PK232

This version uses echoing because AMTOR etc needs it. The HW handshake does not work (opposed to the manual !). Note Newer Software in PK-232 (dated October 86 or later) works correct.

Note I never used the PK-232 in RTTY, so it might work or might not, and CW is not supported at all.

Please switch the PK-232 modes (Packet/Arq, etc) via the SSS device mode-switching (ALT/M) rather than doing it direct. The reason is that SSS will also switch from LINE-input mode (used for Packet) to WORD-input mode (used for RTTY/AMTOR). If you switch direct by the PK-232 directives, SSS will NOT be aware of this.

CTRL/D, the dot ".", the comma ".", the space " " and CR are the delimiters for WORD-input mode which is used for AMTOR, i.e. the word is sent when such a delimtier is typed in. You can no longer edit (correct) the word after you have typed the delimiter.

In LINE mode the delimiter is CR only so you can edit the whole line.

Note that LINE/WOAD is implicitly selected by SSS (LINE for packet, else WORD) see also below.

KPC2400

For Kantronics KPC-2400. Basically the same as TNC2 but the help- windows reflect variations of the command language.

KAM

There was a half-finished module for the KAM, however I sold this device and I am unable to support any TNC devices which are not available. Consequently I had to remove it.

I have no intention to ever support the TNC-1 with TAPR software. This is also true for the GLB PK-1 (which I consider to be a disaster anyhow).

OPERATION

You issue commands by pressing the ESC-key which causes the TNC to prompt for a command. On the lower screen you will also see the string CMD>. Please note that anything you have typed in so far is not flushed when you enter command mode (by pressing ESC-key) but is restored after you left the command level. Please realize that S8S itself has no idea if the TNC is in Command-mode or Conversation-mode (by the way, this is true also for any other communications program). It means that you first have to get the command prompt (i.e. 'cmd:' for TNC-2) before you can give a command.

SSS always strips bit 8 of each incoming character - the only exception is during binary file transfers of course.

Pressing the function keys F9 and F10 display Help-windows. Read them please.

Program control (common to all devices):

All program control is now done by an ALT/<char> sequence (i.e. pressing ALT-KEY and a character key simultaneously:

ALT/A:

Abort the transmission of a Textfile.

ALT/B:

Toggle Bells ON / OFF. If BELL is OFF then any bell-character (07 hex or ^G, i.e. ctrl/g) is presented as tilde character (~) and consequently will not sound. However any ctrl/f (08 hex) received converted to BELL. Those who know it may send you a control-f and sound your bell.

This is indicated by BELL=Y or BELL=N in the screen divider line.

ALT/C:

Enable/Disable capturing all text (as shown in the upper window) to a specified file. Please refer to ALT/E also. If Capturing is OFF then ALT/C will cause the program to ask you for a file specifier. If you enter CR then it will use filename of the form "mmddhhmm.CAP", where this is constructed from month, day, hour and minute.

If the specified file already exists, you get a message and capturing remains disabled. IF Capturing is ON then ALT/C will close the file.

The Capture state is indicated by CAPT=N or CAPT=Y in the screen divider line.

ALT/Y:

Clear the lower window & flush the text there.

ALT/F:

Forced line send in LINE mode. If enabled then the line is sent to the TNC whenever after colomn 70 a space or dot is seen. The program simply appends a CR and sends the string typed so far to the TNC.

This is indicated by S80=Y or S80=N ('send at col 80') on the screen divider line.

ALT/S:

Send a TEXT file. You are asked for a filename. Please NOTE that the TNC is used in normal conversation mode, so the file may contain ONLY printable characters, i.e. you can ONLY send a text file. Bit 8 of each byte is cleared and only the codes 32-128 and 10 are transmitted. No linefeed is transmitted. We expect, as in conversation mode, the receiver to have AUTOLF ON. When the TNC device (i.e. the older PK-232) must use Software hand-shake, then you will see the ctrl/s and ctrl/q echo’s from the TNC in the upper Window (in form of graphic characters) - don't mind, it’s a good indication that everything is okay.

Also note that if you have NOT wired the RTS/CTS lines from a TNC-2 to the computer, you must select SW-handshake. Read the source, change the constant parameter and re-compile.

If you are not able to do this, change the cable to connect RS-232 lines 2-8 and 20.

DO NOT try to send non-TEXT files with ALT/S!

ALT/X:

Exit the program and return to DOS.

ALT/M:

Switch device modes (where applicable, i.e. PK-232 - a window appears and let you select Packet or Amtor, etc.).

ALT/D:

To see a directory.

ALT/Z:

Temporarily return to DOS level. This is done by invoking a second command processor. Additionally note that during operation on the DOS level, the SSS remains active. That is, incoming characters from the TNC are read and buffered, but of course NOT processed. The buffer size is 8kBytes - you better return to SSS by typing "EXIT" at the DOS level before that buffer overflows.

If you have capturing enabled and it goes to a floppy disk, NEVER change that floppy disk as long as capturing is enabled. You would probably never want to do this unless you go temporarily to DOS level anyhow. If you want to swap floppies when temporarily on DOS level then DISABLE capturing before you do an ALT/Z.

ALT/P:

Toggle Printer ON/OFF. Printout goes to LPT1. Change variable PrNum in SSS_DECL.PAS to select another unit. Note that I use the BIOS calls for printing, not the simpler Write to LST (via DOS). The reason is that in case of an printer error you would get the famous "write fault to device PRN: - Abort, Retry, Ignore ?" message and the system would wait forever for an answer.

Using the BIOS, SSS detects any printer error and simply stops printing. So if you want printout better have no paper-out and the printer switched online.

The printer state is indicated by PRT=Y or PRT=N in the screen divider line.

ALT/E:

Toggle the skipping of empty received lines, when monitoring is on, the TNCs seem always to append an extra CR-LF to a received frame, which gives a lot of empty lines on the screen. This is the reason why this (and previous) version did skip empty received lines. However, when capturing text file, skipping all empty lines is not very favorable. With ALT/E you can now toggle it. Don't forget to set EMPTY-L to YES whenever you intend to receive a text file by capturing and when you want to get the file as-is, i.e. all empty lines in that file as well.

This is indicated by EMPTY-L=Y or EMPTY L=N in the screen divider line.

ALT/! :

Send a BREAK to the TNC to force it into command-mode.

ALT/L:

List string directory. You can select one of the strings displayed by typing the associated identification character. This string is then simply sent to the TNC device. For the TNC-2, the strings are found in the file TNC2STR.PAR, for PK-232 in PK232STR.PAR, etc.

Such a string file can contain up to 20 lines with a maximum of 80 characters. You can put there whatever you want. Obviously, if a string is "C DK1SL V HB9F, HB9W", i.e. intended for a connection command you better have the TNC device in command-mode before you send it over.

PgUp:

Binary File Transfer, Upload. The YAPP(tm) protocol from WA7MBL is used.

PgDn:

Binary File Transfer, Download. The YAPP(tm) protocol from WA7MBL is used.

==========

On binary file transfers, using the YAPP(tm) protocol:

Please read the comments in SSS_XFER.PAS about ABORTING a pending file transfer to get an idea about the difference when aborting with <ESC> vs. ALT/K.

During an ongoing Binary file transfer you may use 'talkback' that means you can send messages to, and receive messages from the other station. Just start typing the message. The message is limited to approx 50 chars, SSS beeps when you exceed the maximum length. You must then send it with the usual <CR>. Note that such messages are inserted into the datastream packets and consequently it may take a while till it is their turn to be sent.

ALSO NOTE THAT BINARY FILETRANSFER IS ONLY POSSIBLE WHEN YOU HAVE PROPERLY CONNECTED ALL RS-232 LINES (2-8,20) TO THE TNC. Furthermore, you must have the device parameters set as indicated by the appropriate <devive>.INI file

Binary file xfer is not possible with TNC-1/WA8DED as is Textfile-send. This is by no means a drawback of the WA8DED software, this code must be used in HOST-mode for file transfers, and thus would require another type of server software.

The TNC-2 and other devices use different lines for hardware handshake but SSS is aware of this and should perform ok.

If binary transfer does not work, most of the problems are caused by incorrect TNC parameters. The parameters especially all RS-232 line related parameters MUST be set as shown in the <device>.INI files. Please realize that even if you change these parameters, the change will be effective only AFTER a RESTART command.

I realized that the TNC-2 shows some strange behavior when it toggles the hardware handshake lines, and I reworked the CTS ON/OFF recognition by SSS a bit to correct this in Version 2.2a. Also, anybody (including me) would assume that on a quality link the parameters MAXFRAME=7 and PACLEN 0 (i.e. 256) would give optimum performance. Again the TNC-2 at least mt TNC-2 with V 1.1,4 Software) shows a strange behavior which is apparently related to its internal buffer management. The optimum parameters which I found are: MAXFRAME 4 and PACLEN 0.

Please note that the YAPP(tm) protocol is not perfect, for a trouble free binary file transfer, the receiving station should enter the receive mode before the transmitting station enters the transmit mode. If the receiver is once in binary file receive mode, it will try to discard anything but a request from the other transmitting station, however this is not fool proof. So better DO NOT SEND texts to your partner in normal conversation mode when he has already switched to binary-file receive.

Also, during a transfer, the above mentioned TALKBACK should no longer be used when a transfer is about to finish.

Word-mode and Line-mode is implicitly switched only by PK232 when the device modes are altered. Packet operation is in Line-mode (line is sent by a OR), AMTOR and other modes use WORD-mode (any typed string is sent to the device whenever a SPACE, DOT or CR is seen).

If you start TNC2 by "C>TNC2 R" then the lines of a file TNC2.INI are read and sent to the TNC during init. Same for PK232, file is PK232.INI.

The stream switch character of the TNC-2/PK-232 is set to ^P, and switching can be done by the <INS>key on the Keyboard.

Whenever the link disconnects during a transmission, the current text which is already in the TNC's buffer goes out via the UNPROTO path. The other remaining text which is still sent to the TNC (unless you abort it with an ALT/A).

Whenever a file transmission is finished or aborted, the program switches to TNC command mode, possibly executes some commands and tries to switch to CONY mode again. This may fail if there are still packets unsent and the TNC lacks buffers. In this case you get a "too many packets outstanding" message. Wait till the STA-Led goes off (to indicate that no more packets are unacknowledged) and then go to CONV mode manually.

This program has by no means any idea about the TNC's state, basically it is a simple Terminal program specialized for TNC operations. A more close control would well be desirable, however, a true (proof) close control of a TNC is only possible with a little help from the TNC, and the only TNC programs which permit such a control is the TNC-1 Software from WA8DED when used in host-mode (that's why host mode exists), and the PK-232's host-mode.

The introduction of the DOS-level operations (see ALT/D and ALT/Z) requires to compile with a Maximum HEAP size of less than A000 (the default) to give space for the second shell. A value of 0A00 will work fine. Note that if you compile in memory, i.e. NOT to a file you cannot set the HEAP-size, and the (too large) defaults apply. You can run the program, but whenever you do an ALT/D or ALT/Z then YOU ARE DEAD.

==========

Revision History

New in the December version is ALT/E and time-stamping of incoming characters. This is required to write the characters received so far in ANY CASE if for 200mS there no more characters have been received. The FILESEND has been reworked a bit. The program uses CPACTIME ON now, this is more economical and permits to flush buffer in case of an abort.

The January Version has the Binary-file transfer facility implemented (PgDn, PgUp).

The March Version got the string directories (ALT/L).

The June Version got the KAM support. This required to re-arrange the 'function-key-text' to become an ALT-Function-key text, i.e. the text from the PAR file is now called by pressing ALT/F1 to ALT/F6 rather than F1 to F6 as before. For the PK-232, when switching the device modes by ALT/M, the device echo is altered during the mode switch. In packet, where SSS is in LINE-Mode, the echo from the PK-232 is switched off and done locally with high intensity. In non-packet modes, where SSS is in WORD-mode, echoing from the device is enabled. However, because I sold my KAM, I can not maintain software for it - KAM.* will remain as is.

The September Version got some reworking of the binary file transfer modes, and some cosmetic changes for the information windows of the TNC/WA8 software. It reflects TNC-1/WA8DED V1.3 and TNC-2/WA8DED V2.1. Also the configuration of SSS is now done by <device>.CFG files which permit to modify the window colours and the port configuration, and the defaults. These files also contain the texts for the function-keys (previously found in <device>RIG.PAR files - which do no longer exist). Also, deleting characters if you have typed more than one line will now work as expected (cursor is moved back to the previous line). If SSS echos, i.e. moves your transmit string to the upper receive window (this is the default) the echoed lines can be selected to be either intensified or reverse (usefull for laptop computers with LCD display). You select this in the CFG file.

You are free to give this program to other hams. Any positive feedback is welcome. Please note however that this program is given to you "as-is", I feel no obligation to give ANY support, i.e. I might fix bugs, or I might not.

If you are not 'satisfied' with the program, then remember that you didn't have to pay for it, and please use another terminal program.

My address is:

HB9CVV
Peter H. Heinrich Allmendstr. 25
CH-2562 Port
Switzerland
CIS [74170,32]

Have fun, vy 73 de Peter.

A NOTE ON COM-PORTS
==========

Besides the standard addresses and interrupts for COM1 and COM2, SSS uses the addresses found in the <device>.CFG file for the COM ports 3 to 8, which reflect the addresses used by the MS-400 board. You can change this for your requirements if necessary

In this file you also specify the default values for port number and baud rate
==========