KISS (amateur radio protocol)

‹The template *Manual* is being considered for merging.›

KISS (Keep It Simple, Stupid[1]) is a protocol for communicating with a serial terminal node controller (TNC) device used for amateur radio. This allows the TNC to combine more features into a single device and standardizes communications. KISS was developed by Mike Cheponis and Phil Karn to allow transmission of AX.25 packet radio frames containing IP packets over an asynchronous serial link, for use with the KA9Q NOS program.[1]

Description

The KISS protocol is designed to be easy to implement in simple embedded devices, which are capable of asynchronous serial communications. While it allows arbitrary data to be transferred, there is no support for flow control or error handling.

KISS uses Serial Line Internet Protocol framing, but defines its own simple protocol instead of encapsulating raw IP packets, as SLIP does. The KISS protocol messages provide for setting common TNC control variables as well as sending arbitrary message packets to be forwarded by the TNC. While the packets may be arbitrary, protocols commonly used with KISS include AX.25 and IPv4.

Special characters

Hex value	Abbreviation	Description
0xC0	FEND	Frame End
0xDB	FESC	Frame Escape
0xDC	TFEND	Transposed Frame End
0xDD	TFESC	Transposed Frame Escape

Similar to SLIP, back-to-back FEND codes should not be interpreted as empty frames. Instead, all but the last FEND code should be discarded. This can be used for synchronization, and can be used to give the receiver's AGC time to stabilize.

If the FEND or FESC codes appear in the data to be transferred, they need to be underlined escaped. The FEND code is then sent as FESC, TFEND and the FESC is then sent as FESC, TFESC. Two FESCs in a row are a protocol violation and can be used to signal an aborted transmission. This allows the recipient (usually the TNC) to avoid misinterpreting subsequent data as part of a valid frame. Any data received before the next FEND would be properly discarded.

Command codes

Any of these codes may be sent from the host to the TNC, but only the "Data frame" code should be sent from the TNC to the host. "In multi-port TNCs, the upper 4 bits of the type indicator byte can specify one of up to sixteen ports."[1]

Command bytes	Name	Argument length	Description
0x?0 [...]	Data frame: port X	Varies	The following bytes should be transmitted by the TNC. The maximum number of bytes, thus the size of the encapsulated packet, is determined by the amount of memory in the TNC.
0x?1, 0x??	TX DELAY	1	The amount of time to wait between keying the transmitter and beginning to send data (in 10 ms units).
0x?2, 0x??	P	1	The persistence parameter. Persistence=Data*256-1. Used for CSMA.
0x?3, 0x??	SlotTime	1	Slot time in 10 ms units. Used for CSMA.
0x?4, 0x??	TXtail	1	The length of time to keep the transmitter keyed after sending the data (in 10 ms units).
0x?5, 0x??	FullDuplex	1	0 means half duplex, anything else means full duplex.
0x?6, [...]	SetHardware	Varies	Device dependent.
0xFF	Return	0	Exit KISS mode. The TNC stops processing KISS protocol and returns to its vendor-specific behavior.

The command bytes are shown here in Hex, but are sent as bytes, not hex strings. In all cases except the Return command, the high nibble indicates which port (on a multi-port TNC) the command applies to.

A TNC may support other nonstandard commands, at the vendor's discretion.

Packet format

Begin	Command	Data0..DataN	End
FEND	High nibble – Port Index Low nibble – Command	Data	FEND

Send the characters "TEST" out of TNC port 0

C0 - FEND	00 - DATA FRAME	54 - "T"	45 - "E"	53 - "S"	54 -"T"	C0 - FEND

Send the characters "Hello" out of TNC port 5

C0 - FEND	50 - DATA FRAME: port 5	48 - "H"	65 - "e"	6C - "l"	6C - "l"	6F -"o"	C0 - FEND

(hex byte 50 is binary 0101 0000, which a multi-port TNC would interpret as "DATA FRAME, PORT 5" because 0101 -> 5)

Send the bytes 0xC0, 0xDB out of TNC port 0

C0 - FEND	00 - DATA FRAME: port 0	DB - FESC	DC - TFEND	DB - FESC	DD - TFESC	C0 - FEND

Both bytes in the example message require escaping, so they are each sent as a two-byte sequence. The TNC will unescape them prior to transmission.

Exit KISS mode

C0 - FEND	FF - RETURN	C0 - FEND

The TNC's behavior after returning from KISS mode is vendor-specific. Some provide a line-oriented "node client" or "mailbox" command client.

Common uses

- AX.25
- APRS devices

Alternatives

- 6PACK[2][3]

See also

- Serial Line Internet Protocol (SLIP)

References

1. Mike Chepponis, K3MC; Phil Karn, KA9Q (1987). *The KISS TNC: A simple Host-to-TNC communications protocol* (https://web.archive.org/web/20220607102917/http://www.ka9q.ne t/papers/kiss.html). ARRL 6th Computer Networking Conference. Redondo Beach CA. Archived from the original (http://www.ka9q.net/papers/kiss.html) on 2022-06-07.
2. "6PACK a "real time" PC to TNC protocol" (https://web.archive.org/web/20221227153923/htt p://private.freepage.de/cgi-bin/feets/freepage_ext/41030x030A/rewrite/alexs/xfr/flexnet/6pac k_en/6pack.htm). Archived from the original (http://private.freepage.de/cgi-bin/feets/freepag e_ext/41030x030A/rewrite/alexs/xfr/flexnet/6pack_en/6pack.htm) on 2022-12-27. Retrieved 2012-04-08.

3. "6pack Protocol — The Linux Kernel documentation" (https://docs.kernel.org/networking/6pack.html). *docs.kernel.org*.

External links

- Device::TNC::KISS (https://search.cpan.org/~rbdavison/Device-TNC-0.03/lib/Device/TNC/KISS.pm) Perl interface to a KISS mode TNC, by R Bernard Davison on CPAN (2007)

- TNC-2 KISS source code (Z-80 assembly language): https://github.com/cheponis/KISS-TNC2