

# Untangling the FSKNet

2/23/98

**Summary:**

Like all other network protocols, the workings of FSKNet remain a mystery until directly observed during operation. This article introduces the reader to our FSK to RS232 adapter and monitoring software, *FSKMON*. *FSKMON* is a program that monitors all FSKNet throughput and presents data in a human-readable format. The source code for *FSKMON* is available in electronic form from the Air-Weigh web site at [www.air-weigh.com](http://www.air-weigh.com)

## **SEEING IS BELIEVING**

In the protocol specification for FSKNet, it was explained in general terms how to create your own minimally compliant FSKNet device. We set the foundation for what Tokens, ACKs, NAKs and preambles are supposed to look like. Interface circuitry and simple pseudocode applications were also presented.

Well, if you're like us, any new protocol specification is in doubt until you're able to experiment with it on your own terms. After all, a spec is usually a dream founded more on marketing hype than reality.

Fortunately, there's nothing like a little hands-on to separate an empty promise from a genuine solution that works.

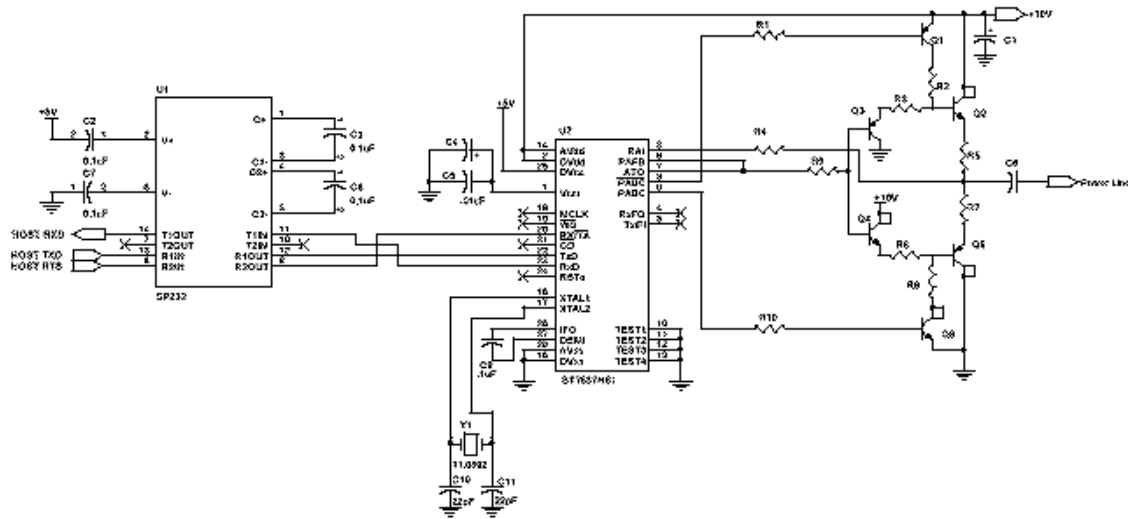
Responding to this belief, we at Air Weigh are making available an RS232-to-FSK adapter, some source code, and a working DOS executable that will allow you to observe an operating FSKNet.

## **HARDWARE**

You'll need access to a live network before any observations can begin. At a minimum, you'll want a 12V/2A Power supply, a WireLink Tractor module, a WireLink Trailer module, a 486+ PC, and an RS-232 to FSK adapter.

The WireLink Tractor/Trailer combination will not be discussed at length in this article. Their involvement with our discussion is merely to provide a subject for observation. Future articles will delve into the specifics of WireLink packets and structures, but for now just apply 12V and let them talk to each other.

If your power supply has an ammeter, you should see it pulsating between 250 and 500 milliamps when the WireLinks are attached. The ambient current draw of the WireLinks are around 250mA, and the 500mA pulsations indicate that one of them is transmitting. If you don't see the pulsations, check your setup.



**Figure 1:** This circuit is very similar to what we use at Air-Weigh. Note how the RTS line is used to switch between transmit and receive modes.

An RS-232 to FSK adapter is required to allow a PC access to the FSKNet. Encapsulated RS232-to-FSK adapters will be available from Air-Weigh to the public for around \$100.00 on or about March 15, 1998. However, if you are under tight time constraints, you may choose to build your own. The schematic shown in Figure 1 is very similar to what we'll be producing, and should fulfill your short-term development needs.

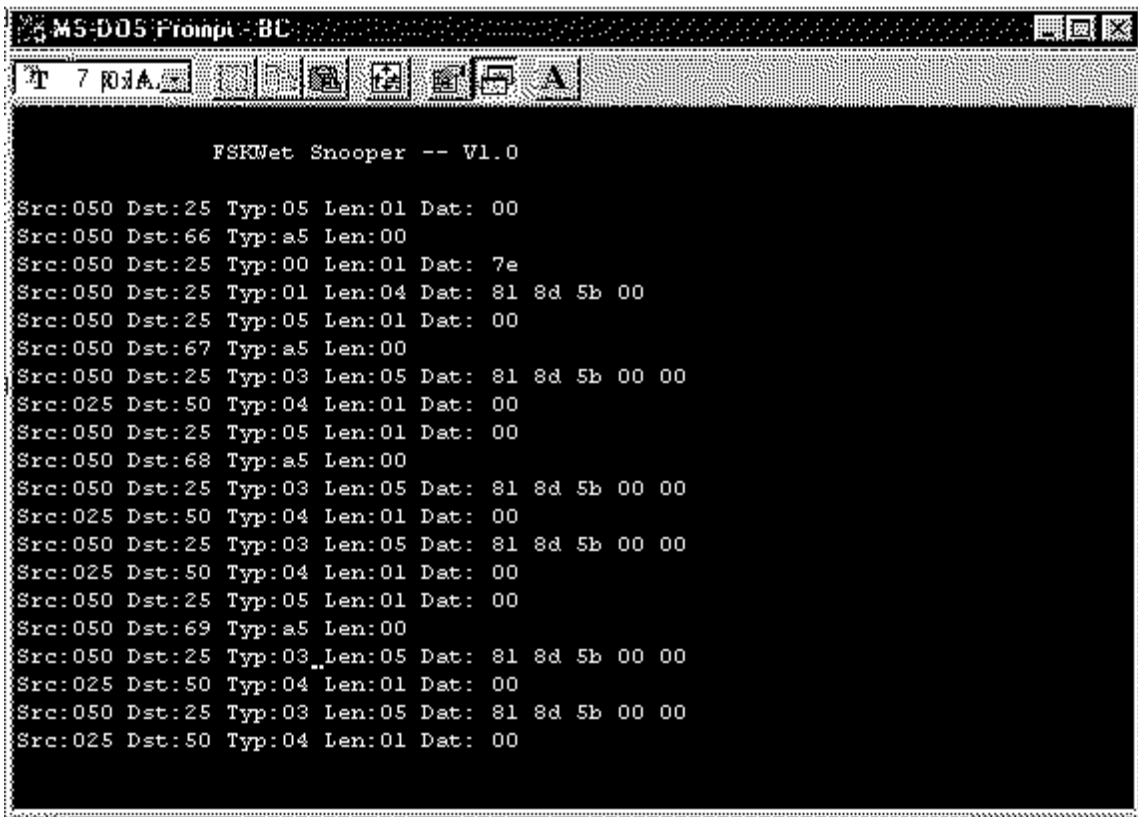
Whether you've built one yourself or are using the Air-Weigh adapter, you'll need to connect it to the same 12V that powers the WireLinks. The operation of the WireLinks should be unaffected at this time; if either of them give an error indication, something must be wrong with the RS-232 to FSK adapter.

Finally, connect the adapter to COM1 of your PC.

**SOFTWARE:**

Once the physical hardware is installed, observing the network is just a matter of software. To this end, we've created an FSKNet monitor (*FSKMON*) that runs in DOS on your PC. You may freely download the source code and executables for *FSKMON* from our web site at [www.air-weigh.com](http://www.air-weigh.com), or you can call us at (541) 431-3121 and we'll ship it to you on a floppy for \$25.00.

*FSKMON* displays the data in a format familiar to any network engineer: Destination, Source, Packet Type, Data Length, and Data are all shown in a straightforward manner. If your hardware configuration is correct when you execute *FSKMON*, you should see something similar to that in Picture 1.



**Picture 1:** FSKMON in action. Here we are watching the dialog between WireLink Tractor and Trailer modules.

## LOOK WHO'S TALKING

Now network throughput is visible, but what are we looking at?

First, realize that the WireLink Tractor module has been assigned an FSKNet Device ID of 50 and the Trailer module is Device ID 25.

Now we can understand that the first packet is a message being sent from device 50 (Tractor) to device 25 (Trailer.)

The next packet is being sent from device 50 (Tractor) to device 66 (Unknown.) This may seem odd, but if you look at the message type (0xA5) you'll see it is a permission token. If device 66 were on the network, it could ACK the token and utilize its 250mS timeslice.

The remaining packets can be deciphered in exactly the same way. The banter between the WireLink trailer and Tractor modules continue, and permission tokens are delivered in turn to devices 67, 68, and 69.

## SELECTIVE SIGHT

FSKMON won't show you everything that happens on the network. Specifically, FSKMON is written to acknowledge only those packets that adhere to the following format:

[0xAA][DEST][CSUM1][TYPE][SRC][LEN][DATA][CRC][CSUM2]

Field	# Of Bytes	Description
[0xAA]	1	Every FSKNet message begins with 0xAA
[DEST]	1	The Device ID that this message is destined for.
[CSUM1]	1	Mod-256 checksum of the 0xAA and destination fields.
[TYPE]	1	This describes what type of message it is.
[SRC]	1	The originator of the packet.
[LEN]	1	The length of the data portion of this message (0-54).
[DATA]	0-54	The data bytes comprising the message.
[CRC]	1	This is the 8-bit CRC of the message up to this point.
[CSUM2]	1	This is the Mod-256 checksum of all the previous bytes in this message.

As you will recall from the protocol spec, Permission Tokens fit this format exactly. Most of the WireLink data packets also adhere to this format.

A good question would be, "Why CSUM1 and CSUM2 when a CRC is sufficient?" The answer is that we are making allowances for even the slowest microprocessor. Computing a CRC on the fly is a rather intensive operation for a 4-bit micro, so for them it is acceptable to validate packet integrity based on the checksums alone. Also, CSUM1 is useful for any

micro in quickly determining if a potential incoming message is merely random noise.

Remember that a given device that you create needn't comply with this format during its time slice. For the 250mS that your device owns the bus, you can transmit data in any format you wish. Just bear in mind that the data will not be acknowledged by FSKMON. However you choose to packetize your data, **do not utilize the bus for more than 250 milliseconds!** You will squash other device's transmissions and compromise the network!

## BEHIND THE SCENES

Now that you know what FSKMON is and what it does, it's time to peel back the curtain and examine the source code. FSKMON is built from five 'C' modules, each briefly described below.

### iomgr.c –

IO Manager contains all the functions that perform IO to access the hardware. By utilizing services of the IO manager rather than accessing the hardware directly, other modules maintain a high degree of platform independence.

### delays.c –

Delays contain the various delay routines used by other modules. By utilizing the services of Delays, other modules maintain a high degree of platform independence.

### datalink.c –

Datalink utilizes the IOManager and Delays to send or receive individual bytes of data on the network. This module has knowledge of intercharacter spacing and other matters pertaining to the network.

### fsknet.c –

This module utilizes Datalink and Delays to send and receive entire FSKNet packets. It exports a complete API to give higher level application modules access to the network. It understands how to process tokens, token preambles, and packets that comply with the standard FSKNet format.

main.c –

Main is what's referred to as the "applications layer" and coordinates the behavior of the other modules. Its job is to print the information on the screen and to process keyboard input.

FSKMON was built with Borland C 3.1, but with only minor changes you could use Microsoft Visual C.

Now that you know what modules comprise FSKMON, a complete dissection should be reasonably straightforward. We suggest that you start with *main.c* and follow it through – it's worth your while even if you're intended target is not a PC. Since FSKMON is written in 'C', and the IO is abstracted by the IO Manager, it should be a very speedy operation to port the network API to the embedded application of your choice.

## CONCLUSION

By this time you have everything necessary to start monitoring existing FSKNet implementations. The net should seem more 'real' and not just a vague promise. Also, by dissecting FSKMon, you'll be well qualified to start developing your own FSKNet compliant devices.

---

**Borland C**  
Borland International  
100 Borland Way  
Scotts Valley, CA 95066  
[www.borland.com](http://www.borland.com)

**Microsoft Visual C**  
Microsoft Corp.  
One Microsoft Way  
Redmond, WA 98052  
[www.microsoft.com](http://www.microsoft.com)

