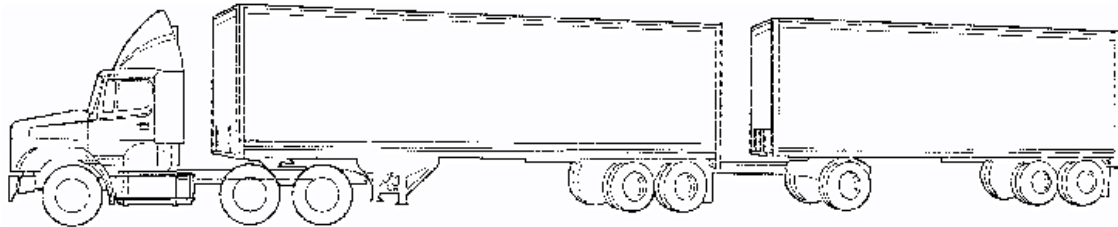


# WireLinks on the FSKNet

2/23/98



## **Summary:**

In this discussion, we'll focus on the WireLink. We expose packet types, data structures, and trailer registration methods. As a prerequisite, you should have a fair understanding of what FSKNet is and what tools are available to monitor network traffic.

## ON FIRM GROUND

The WireLink product utilizes the FSKNet foundation to relay switching and warning information between tractors and trailers. The low cost of implementing FSKNet with parts readily available made it an easy choice.

In disclosing the packet types and structures used to perform operations associated with the WireLink, we are all able to develop similar products that communicate and interact in beneficial ways. It is counterintuitive to disclose the culmination of years of effort, but history has shown that the industries who build on open architectures succeed; companies who covet their proprietary methods, who limit their customers' choices with a "buy ours or bust" attitude, fail.

## THE BASICS

By the time you read this article, you should be familiar with FSKNet and have experimented a little with *FSKMON* or some other FSKNet monitoring device. This being the case, you're ready to dissect the WireLink messaging scheme. Otherwise, see the articles "Understanding FSKNet" and "WireLinks on the FSKNet."

As you'll recall, almost all WireLink packets 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.

The WireLink tractor module has been assigned DeviceID number 50. The WireLink tractor has active monitor capability, meaning it's capable of distributing permission tokens and registering devices onto the net. Interleaved with the permission tokens, a tractor module continually updates the status of trailer modules to reflect the status of switches in the cab.

The WireLink trailer modules are all assigned the Device ID number 25. You may think this odd because an obvious problem arises when we have multiple trailer units all installed – how do they differentiate amongst themselves?

### **SAME BUT DIFFERENT**

Though every WireLink trailer has been assigned the same device ID, each has been assigned a unique 32-bit serial number. This allows for four billion different serial numbers. A unique number is key to the success of WireLink communications.

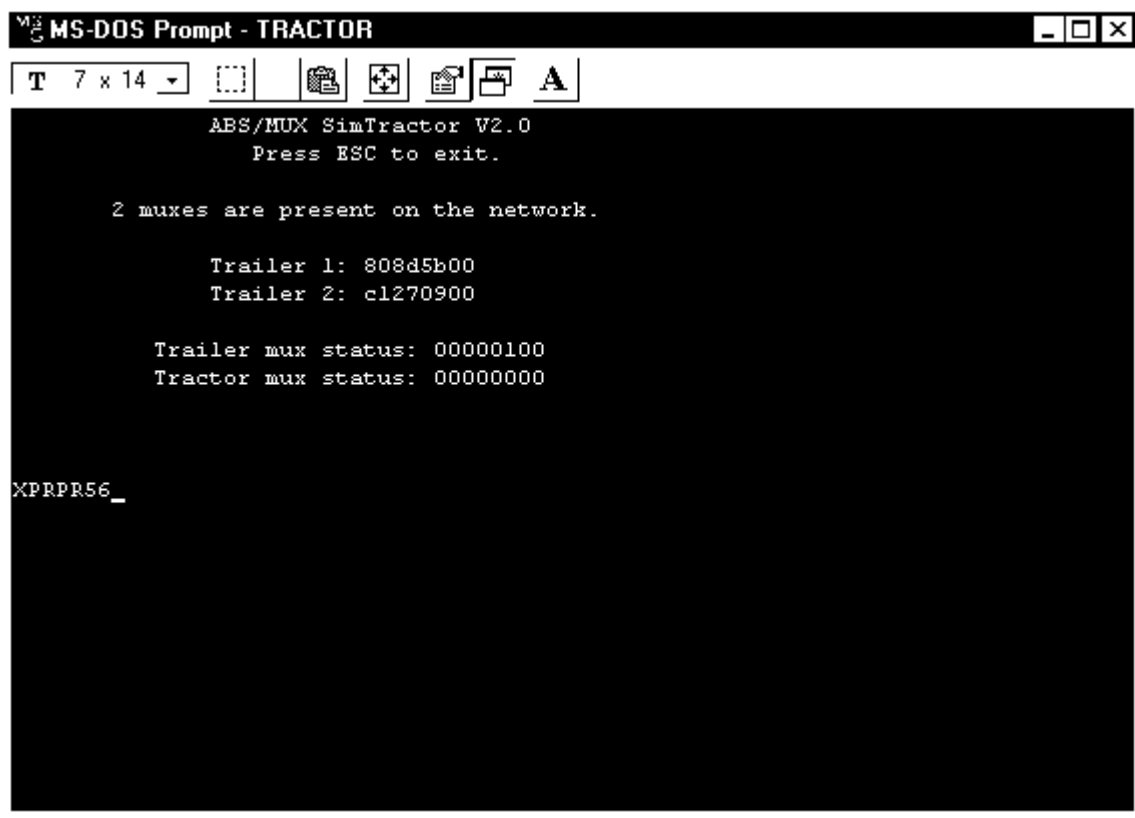
The WireLink tractor sends a message to an individual trailer by embedding that trailer's serial number in the data portion of the packet.

**This is a big idea:** Although the actual DEST field can be only 0-255, devices can freely utilize the DATA field to further differentiate amongst similar devices.

Now that you know Device IDs associated with WireLink trailers and tractors and how they address each other, you're ready to examine the WireLink packets.

### **SIMTRACT, SIMTRAIL**

We've also made available two other software packages that may be of interest: SIMTRACT.EXE and SIMTRAIL.EXE. These two programs run on a PC in DOS and simulate a WireLink equipped tractor or trailer. They are functionally equivalent in terms of product development.



```
MS-DOS Prompt - TRACTOR
ABS/MUX SimTractor V2.0
Press ESC to exit.

2 muxes are present on the network.

Trailer 1: 808d5b00
Trailer 2: c1270900

Trailer mux status: 00000100
Tractor mux status: 00000000

XPRPR56_
```

**Picture 1:** SIMTRACT in action. This simulated tractor has found two trailers and has identified their serial numbers. We can also tell that switch two on a trailer has been closed.

We used these simulation tools during our product development, and learned they are an indispensable debugging tool.

SIMTRACT, and SIMTRAIL are available on our website at [www.air-weigh.com](http://www.air-weigh.com)

## ON WITH THE SHOW

The rest of this article is organized in groups of similarly related network functions. Throughout the project lifecycle, it will prove to be a valuable reference.

## **SOUNDING OFF FOR AUTO REGISTRATION**

The WireLink trailers differentiate amongst themselves by their unique 32-bit serial number. This is the method that a tractor uses to find out what serial numbers are present.

When the WireLink Trailer module wakes up, it is responsive to the **Sound Off** data packet sent by the tractor. The **Sound Off** data packet contains an 8-bit key. The 8-bit key is used to seed the standard FSKNet CRC engine. The trailer then runs all the bytes of its serial number through the CRC engine, which yields a seemingly random number between 0 and 255. Values above 200 are rounded to 200. This number is used to count down the number of milliseconds before the **Sound Off Echo** is broadcast onto the net.

When the WireLink Tractor module receives the echo, it instructs the trailer module to not respond to future **Sound Off** commands. This way when one device is found, it will no longer compete for bus space the next time the **Sound Off** command is issued.

The **Sound Off / Echo** process is repeated several times to ensure that all WireLink equipped trailers are found.

If your company intends to create WireLink equipped trailers, we can reserve for you a range of serial numbers so as not to collide with other manufacturers. This technique allows us to dynamically register which and how many WireLink trailers are on the network.

### **Sound Off -- Commands all listening units to echo.**

**Message Type:** 0

**Data:** A single byte used to seed the sound-off procedure

### **Sound Off Echo (NOTE: This is not in the standard FSKNet format!)**

This is not a true FSKNet packet, but a simple six byte stream of data described as follows. Again – The Sound Off Echo consists of these six bytes ONLY! No headers or checksums are sent. Only these six bytes are placed on the bus.

A common question is, “Why the nonstandard packet? Why not use a ‘normal’ packet?” The answer has to do with time. The 6-byte echo as defined takes only 24mS to transmit, where even the smallest ‘normal’ packet takes 44mS to transmit. The response window for Sound Off Echo is a scant 200mS. With every trailer WireLink attempting to echo at random intervals during this period, it is beneficial to minimize the possibility of a given trailer stepping on another’s echo. Collisions do happen, but since the Sound Off / Echo process is repeated several times we can be assured that all WireLinks will quickly be found and accounted for.

BYTE	0xAA
DWORD	Little-endian serial number of trailer WireLink.
BYTE	8-bit FSKNet standard CRC of the preceding 5 bytes.

**Ignore Sound Off -- Instructs a unit to ignore the Sound Off message.**

**Message Type:** 1

**Data:** A four byte record described as follows:

DWORD	Little-endian serial number of target device.
-------	---

**Heed Sound Off -- Instructs a unit to echo the next time it receives the Sound Off message.**

**Message Type:** 2

**Data:** None

## **WireLink Information**

You can determine the WireLink state of a vehicle with only a few different packet types. The current generation of software allows for 4 channels from tractor to the trailer (backward) and 4 channels from trailers to tractor (forward.) The channels are packed into the high nybble of a byte, and each bit maps to a switch -- if the switch is ON (active) the bit is '1' and if the switch is OFF (inactive) the bit is '0.' Future versions may allow for even more channels, but this will simply be a matter of changing the packet length to 2,3,4, or more bytes. As we increase our number of channels (it is inevitable) we will always be backward compatible if byte 0 is used for channels 0-7, byte 1 for channels 8-15, etc.

### **Get Tractor WireLink State -- Request the tractor switch position information.**

**Message Type:** 7

**Data:** None.

### **Tractor WireLink State – This is the reply to “Get Tractor WireLink State”**

**Message Type:** 5

**Data:** A single byte indicating the WIRELINK state of the given tractor.

If a switch is active, the bit will be '1'. If inactive, the bit will be '0'.

### **Get Trailer WireLink State -- Request the switch information from a trailer unit.**

**Message Type:** 3

**Data:** A five byte record described below.

<u>FIELD</u>	<u>SIZE</u>	<u>DESCRIPTION</u>
Serial #	[DWORD]	The little-endian serial number of the target WireLink.
Status Flags	[4 BITS]	The status of the tractor WireLink making the query.
WireLink State	[4 BITS]	The caller's WireLink state.

It is assumed that the caller of a trailer will always be a WireLink equipped tractor, so embedding the caller's WireLink state in this request allows us to update the trailers at the same time we request data from them.

Bit 0 of the Status Flags will be set if the tractor is having trouble communicating with the target WireLink. This indicates that the target WireLink's transmitter has probably gone

bad. All other bits in the low nybble are reserved. Trailer WireLink modules are always device 25. They differentiate amongst themselves by their unique serial number. When querying a trailer WireLink, you must specify which serial number you are addressing. All trailer WireLinks will respond if serial number 0xFFFFFFFF is used.

If you were to query the Tractor WireLink for the Trailer WireLink state, it would return the collective state of all the trailer WireLinks. The individual WireLinks states are logically ORed together, so if any given WireLink has a channel active, it will show up as a 1 in the collective state. The tractor WireLink doesn't do any serial number checking when it gets this request -- it will respond no matter what serial number you throw at it. The tractor WireLink will also ignore the "Caller WireLink State" field.

### **Trailer WireLink State -- This is the reply to "Get Trailer WireLink State."**

**Message Type:** 4

**Data:** A single byte indicating the WireLink state of the given trailer.

If a switch is active, the bit will be '1'. If inactive, the bit will be '0'. Bit 4 reflects the ABS status. 0=Fail, 1=Ok.

## **Device Information**

To identify the revision, date of manufacture, and serial number of a given “black box”, use this pair of network messages.

### **Get Device Information -- Requests the generic information from a given device.**

**Message Type:** 10

**Data:** A little-endian DWORD serial number of target device, if applicable.

NOTE: All devices of the same ID will respond if queried as serial 0xFFFFFFFF

### **Device Information -- The information associated with a given device.**

**Message Type:** 11

**Data:** A multibyte record as described below.

<u>FIELD</u>	<u>SIZE</u>	<u>DESCRIPTION</u>
Serial #	[DWORD]	The little-endian serial number of the device.
Version Major	[BYTE]	The major version number of device.
Version Minor	[BYTE]	The minor version number of device.
Production Day	[BYTE]	Day of month device was programmed (set at factory)
Production Mon	[BYTE]	Month device was programmed (set at factory)
Production Year	[BYTE]	Year device was programmed (set at factory)
Descriptor	[16 BYTES]	Nomenclature of the device.

<<END>>