

**Understanding the
FSKNet
Power Line Modem
Tractor-Trailer Communications Protocol**

2/23/98

Summary:

This article introduces the reader to the FSKNet protocol. It explains why it came about and how it can be utilized for a variety of applications. Hardware schematics and pseudocode examples are also given. The network, as explained, can accommodate up to 3000 devices distributed across 30 individual systems.

1.0 Introduction

For too long the number of signaling devices on a trailer have been restricted by the number of connections of the seven-pin J560 tractor/trailer interconnect. This limitation was fine in the distant past, but the needs of the modern trucking industry demand more than seven wires to operate reefers, sliders, drop-axles, door sensors, ABS sensors, and lighting.

To handle the increasing signal requirements, some manufacturers have proposed installing a second J560 connector or abandoning the J560 altogether in favor of a 13-pin interconnect. The major problem with these types of solutions is that you will not be able to freely mix old and new trailers in double and triple configurations. In the long run you'll end up consuming all the additional wires anyway. Will you install yet another J560 or a 24-pin interconnect?

Instead of adding more and more physical wires to a tractor/trailer system, we at Air-Weigh have developed power-line-carrier technology for use in multiplexing signals across the existing J560. To encourage the adoption of this technology by other manufacturers, we are freely disclosing the hardware, the network protocols, and the packet types that we use to communicate between the tractor and trailers.

From the bottom up, this article will describe the elements necessary to create a simple FSKNet compliant device. We'll start with the physical layer, move up to the protocol, and finish off with a sample application.

2.0 Hardware

FSKNet utilizes power-line-carrier technology to transmit and receive data across the existing wiring of the J560 interconnect. We do not employ spread-spectrum, frequency hopping, or anything else so clever. Our ambition was to create a simple, inexpensive interface that almost anyone could design for with a minimum of development outlay. Holding with this ideal, we chose to utilize a simple frequency-shift-key technique.

The frequencies used are centered around 132.45Khz where 131.85Khz is MARK (1) and 133.05Khz is SPACE (0). The output of a standard UART is used to drive the frequencies to MARK or SPACE as required. A receiver need only differentiate between these two frequencies and present a receiving UART with MARK or SPACE as necessary -- that's all there is to it.

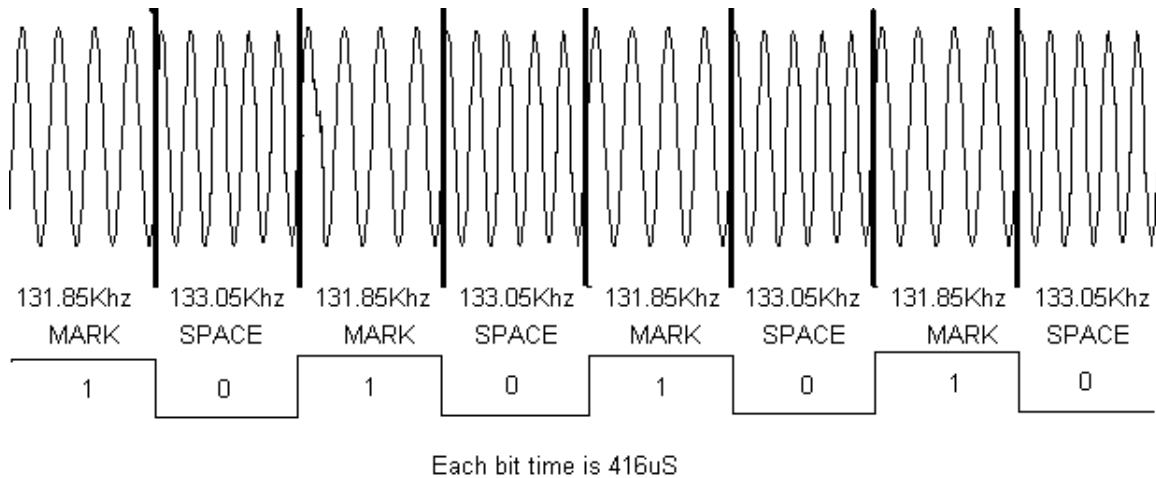


Figure 1: The FSK Technique sends binary data by sending one of two given frequencies.

You can build the necessary modem from discrete components or ICs from any number of manufacturers, but it's easiest to utilize the SGS-Thompson ST7537HS1 Home Automation Modem. This device has proven to be highly reliable and is a mere \$7.09 at 5000 units, or \$5.21 at 50,000 unit quantities.

Data sheets for the ST7537HS1 modem can be retrieved from the World Wide Web at www.st.com

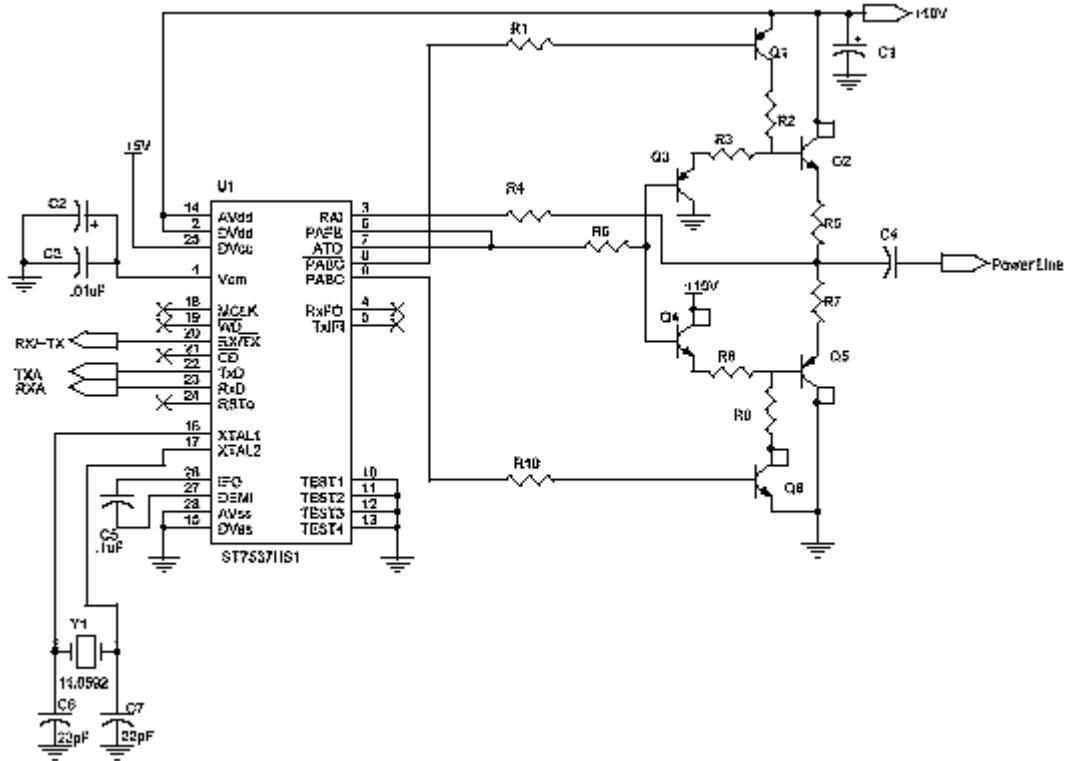


Figure 2: This circuit is very similar to what we use at Air-Weigh. You'll probably need to design your own AGC and impedance matching stages for best performance, but this circuit as-is should satisfy most requirements. See Appendix A for detail.

3.0 Network Protocol

Needless to say, no two devices can transmit at the same time or else they would squash each other's message. We have created a system, or protocol, to arbitrate which device can transmit, when, and for how long.

The protocol we have created for tractor/trailer communications is called FSKNet. It is designed to be very reliable and highly noise-immune. It operates with a standard UART at 2400N81 over the hardware described in the previous section.

FSKNet has what are called active monitors and standby monitors. There may be only one active monitor at any given time, and its job is to dictate who has access to the bus by delivering a special permission token (described later). There may be multiple standby monitors, and their job is to make sure the active monitor doesn't fail -- if the active monitor fails, a

standby monitor will assume the role of active monitor.

At power up, a device pauses for one second to see if an active monitor has already been declared. It does this by looking for the token preamble (described later.) If no active monitor is detected, the device waits an additional period of time before sending five tokens to itself and assumes the role of active monitor. Time to wait is measured in milliseconds and is calculated by taking the Device ID, subtracting 50, then multiplying by 100. When other devices see the token preamble, they will know that an active monitor has been declared.

The active monitor then calls each FSKNet node in pseudo-round-robin fashion and gives them a permission token to originate a transaction. If the target node requires use of the bus, it will "ACK" (described later) the permission token within 20ms and will be free to use the bus undisturbed for 250ms. If the target node has no need for the bus when given the permission token, it will "NAK" (described later) the token within 20ms and remain silent.

To avoid the situation where two devices declare themselves to be active monitors on the same system, an active monitor must revert to a standby monitor whenever it detects a token preamble.

If the standby monitor doesn't see the token preamble for ten seconds, it assumes the active monitor has failed and resets -- thereby restarting the active monitor arbitration process.

Permission tokens, ACK, and NAK

The permission token is the active monitor's way of giving a node permission to use the bus. The complete permission token is the following 11 byte stream:

```
[0xAA][0xA6][0xF5][0xAA][DEST][CSUM1][0xA5][SRC][0][CRC][CSUM2]
```

Each bracketed element is exactly one byte in length. The first three bytes are referred to as the token preamble, and is used by standby monitors to ensure the active monitor is still operating. The last eight bytes are

described below:

[0xAA]	Every FSKNet message, in this case a token, begins with 0xAA
[DEST]	The Device ID that the permission token is destined for.
[CSUM1]	Mod-256 checksum of the 0xAA and destination fields.
[0xA5]	This is a TYPE field, and type 0xA5 is a permission token.
[SRC]	The originator of the token.
[0x00]	This is always zero.
[CRC]	This is the 8-bit CRC of the previous six bytes.
[CSUM2]	This is the Mod-256 checksum of the previous seven bytes.

The "ACK" and "NAK" responses are each three bytes in length and are described as follows.

```
ACK [0xAA][0xFA][0xA4]
NAK [0xAA][0xF5][0x9F]
```

CRC In Detail

The function (in 'C') we use for computing the Cyclic Redundancy Check, or CRC, is as follows:

```
static BYTE CRC_Byte(BYTE Seed,BYTE Data)
{
    int j;

    for (j=0;j<8;j++)
    {
        if (((Data^Seed)&1)!=0)
        {
            Seed^=0x18;
            Seed>>=1;
            Seed|=0x80;
        }
        else Seed>>=1;
        Data>>=1;
    }
    return(Seed);
}
```

The CRC is always seeded with 0xFF prior to computing the CRC on a given number of bytes. As an example of how to use the above function, here is a program that computes the CRC of a five byte array:

```
void main (void)
{
  BYTE Arr[5]={1,2,3,4,5};
  BYTE j,Crc;

  Crc=CRC_Byte(0xFF,Arr[0]);
  for (j=1;j<5;j++)
  {
    Crc=CRC_Byte(Crc,Arr[j]);
  }
  printf("CRC=%x\n\r",Crc);
}
```

Device ID assignments

Device IDs that are granted permission tokens are always in the range of 50 to 79. FSKNet can service thousands of individual “black boxes” but given the bandwidth limitations of the 2400 baud nature of FSKNet, we do not anticipate the need for more than 30 individual systems.

To get a unique Device ID assigned to your company's FSKNet compliant system, contact Air-Weigh engineering at (541) 343-7884, or you can FAX us at (541) 431-3121. We will also be happy to give your company advice on how to get the most use out of the assignment.

You may assign Device ID #78 to your system during development -- this way you won't need to contact Air-Weigh until your product is ready for release. Remember to change your systems' ID to the Air-Weigh assigned number before distribution.

Implications

The real power behind FSKNet is that you needn't comply with any specific protocol during the 250ms after ACKing the permission token. This means you may devise your own protocol and packet structure to operate during this time

slice. In this way FSKNet is more like a scheduler than a complete network protocol.

The advantage of FSKNet is clear. When your system receives the permission token, you may transmit data in J1587, J1939, or any other format you wish. Just make sure you aren't using the bus for longer than the 250mS allotment!

If you know your device will be installed in an existing system, you may want to forego any attempt at being an active monitor. This will greatly simplify your development, as the device need only wait until it receives the permission token before transmitting -- no other overhead needs to be coded. Keep in mind, however, that if the active monitor is ever removed or disabled then your system will never receive a permission token.

Exceptions

If device 79 ACKs the permission token, the active monitor must automatically give the device an extended time slice of 750ms. Device ID 79 is the AW5200 scale system, and the extended time slice is necessary to give the scale a sufficient period to communicate with all of its trailer sensors.

Access Time

Access time would be slow indeed if an active monitor distributed permission tokens in a truly blind round-robin fashion. This would mean that, present or not, the monitor would hand out a token to potentially 30 devices before circling back around to devices that are known to be connected! This could result in access delays of more than ten seconds – unacceptable by anyone's standards.

Fortunately, the preferred method for an active monitor to hand out tokens is to give permission to known units first, then attempt to “bring on” a unit that may or may not be present on the bus.

For example, let's say that devices 50, 55, and 56 were present on the network. The order of permission would look something like this:

```
50,55,56, <--Known to be present
60, <--Unknown
50,55,56, <--Known
61, <--Unknown
55,55,56, <--Known
62, <--Unknown
Etc...
```

Now, let's just say that device number 63 gets turned on and connected to the network. The permission sequence would look like this:

```
50,55,56, <--Known to be present
63, <--Unknown.. But ACKs the permission token – Now it is
      known!
50,55,56,63, <--Known
64, <--Unknown
50,55,56,63, <--Known
65, <--Unknown
Etc...
```

Another interesting byproduct of this method of token distribution is that you can easily handle the situation when a node has been removed. As a general rule, if a node doesn't ACK or NAK the token for five consecutive rotations, it has probably been removed and can be reverted to an "unknown" status. In this way it allows for hot plug ins and drop-outs without affecting normal network traffic.

By employing the pseudo-round-robin method, bus access time is greatly improved and can be calculated by using the following formula:

$$\text{Access time in milliseconds} = (nA-1*294) + (nN*64) + 64$$

Where: nA = Known nodes that ACK the token.
 nN = Known nodes that NAK the token.
 64 = This is how long it takes to test if an unknown node
 has been attached.

Though there is some overhead, this bus arbitration technique

guarantees each node an opportunity to send data within a reasonable time. Unlike many other proposed protocols, there is no need for collision detection or further means of bus arbitration.

4.0 Sample application

Sensing whether the trailer door is open or closed is just the sort of simple project good examples are made from. The knowledge you gain here will be enough to prepare you for most FSKNet applications.

To simplify our pseudocode, let's assume that this system will be installed on a preexisting FSKNet network, which means a device capable of being an active monitor is already in place. Of course, we can't count on this assumption when deploying products into the field, but for now let's make that assumption.

The simple door sensor system must have two parts -- a trailer module and a tractor module. The tractor module will wait for the permission token. When it receives this token, it will send an ACK and then ask the trailer module if the door is open or not. The trailer module will wait for the tractor to ask if the door is open or not, at which time the trailer will reply with either a YES or NO message.

The pseudocode for this system is simple:

```
/******  
//Pseudocode for the tractor module querying one trailer  
/******  
#define DEVICE_ID 78  
  
void main (void)  
{  
  STRING String;  
  BYTE Timer;  
  
  while (TRUE)  
  {  
    WaitForFSKNetToken(DEVICE_ID);  
    SendFSKNetACK();  
    SendStringOutUART("OPEN?");  
    Timer=100; //Only wait 100 ms for a reply.  
    while (Timer>0)  
    {  
      String=GetIncomingStringIfAvail()
```

```

        if (String=="YES!") print("Door is open!");
        if (String=="NO!") print("Door is closed!");
        Delay1ms();
        Timer--;
    }
}
}

//*****
//Pseudocode for the trailer module.
//*****
void main (void)
{
    STRING String;

    while (TRUE)
    {
        String=GetIncomingStringIfAvail();
        if (String=="OPEN?")
        {
            printf("Somebody asked me a question!");
            if (DoorIsOpen()) SendStringOutUART("YES!");
            else SendStringOutUART("NO!");
        }
    }
}
}

```

In a situation where there would be multiple trailers being towed by one tractor, you could very easily modify the above example to address different trailers, in turn, during the same time slice. The tractor code would be modified to look something like this:

```

//*****
// Pseudocode for the tractor module querying multiple
// trailers
//*****
#define DEVICE_ID 78

void main (void)
{
    STRING String;
    BYTE Counter,Timer;

    while (TRUE)
    {
        WaitForFSKNetToken(DEVICE_ID);
        SendFSKNetACK();

        SendStringOutUART("1 OPEN?");
        Timer=33; //Only wait 33 ms for a reply.
    }
}

```

```

while (Timer>0)
{
String=GetIncomingStringIfAvail()
if (String=="YES!") print("Door 1 is open!");
if (String=="NO!") print("Door 1 is closed!");
Delay1ms();
Timer--;
}

SendStringOutUART("2 OPEN?");
Timer=33; //Only wait 33 ms for a reply.
while (Timer>0)
{
String=GetIncomingStringIfAvail()
if (String=="YES!") print("Door 2 is open!");
if (String=="NO!") print("Door 2 is closed!");
Delay1ms();
Timer--;
}

SendStringOutUART("3 OPEN?");
Timer=33; //Only wait 33 ms for a reply.
while (Timer>0)
{
String=GetIncomingStringIfAvail()
if (String=="YES!") print("Door 3 is open!");
if (String=="NO!") print("Door 3 is closed!");
Delay1ms();
Timer--;
}
}
}

```

To modify the trailer code to reply only when its associated trailer door is being queried--Simply modify the line *if (String=="OPEN?")* and you've got it!

Before shipping your product, be sure to contact Air-Weigh to receive a unique Device ID number. Remember to change your code to reflect this new unique ID -- two devices with the same ID on the same net can lead to unpredictable behavior .

5.0 Conclusion

All in all, FSKNet is a great solution for allowing multiple vendors to use the same technology for tractor/trailer communications. It allows us to develop proprietary protocols within individual systems if needed, while at

the same time taking steps to ensure that nobody walks on anybody else's transmission medium.

Whether your niche in onboard electronics is load sensing, obstacle warning, lighting, or failure indication, we all stand to gain with power line modem technology. We stand to gain a lot more if our systems can cooperate.

If you have any questions or need further assistance, please do not hesitate to call us at Air-Weigh.

For further reading about networking concepts in general, we suggest buying a copy of "Understanding Data Communications" from Sams Publishing. You can reach them at (800) 428-SAMS. Ask for ISBN# 0-672-09343

Appendix A

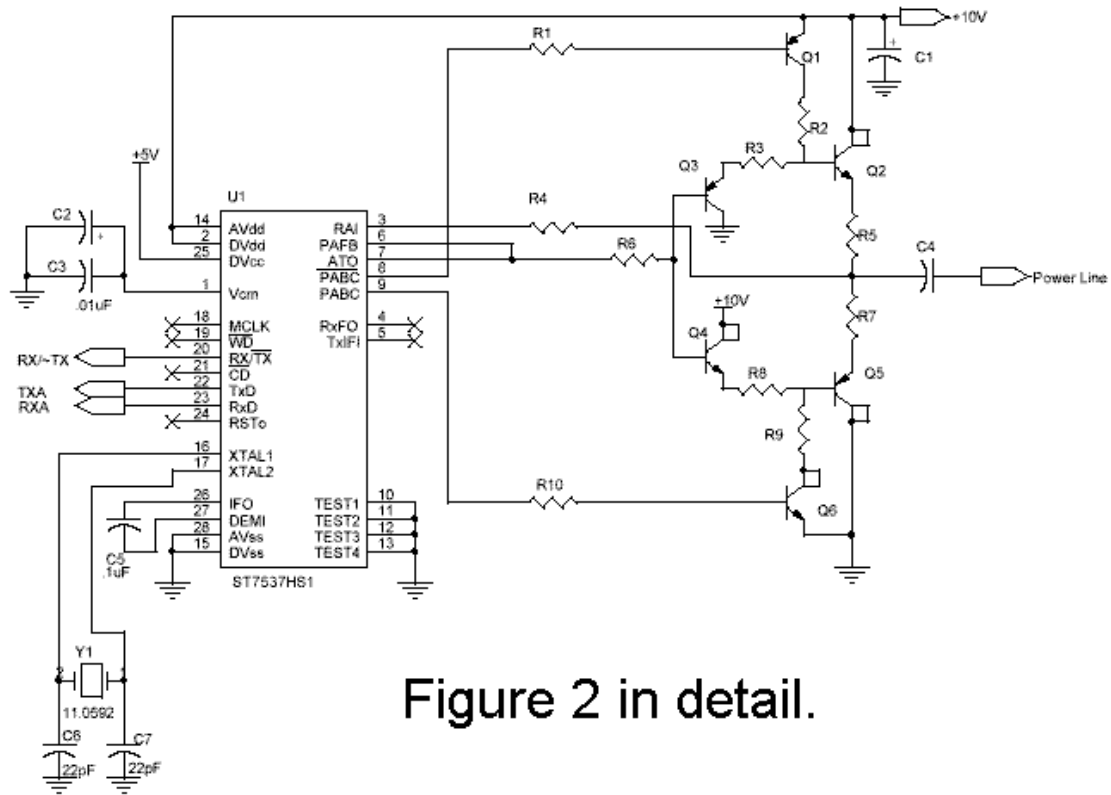


Figure 2 in detail.