

Application Note AN-401

“Channel Coding and Protocol Considerations For Embedded Wireless Modules”

Document Control

Created By	Steve Montgomery	8/12/01
Engineering Review		
Marketing Review		
Approved - Engineering		
Approved - Marketing		

Revision History

Revision	Author	Date	Description
1.0	SJM	8/12/01	Document Created
1.1	SJM	11/30/03	Updated with current information

1.0 Introduction

When designing a wireless product, it is important for the designer to understand that communications via an RF link have different considerations than communications over wires. For example, a wire is capable of communicating a DC voltage; an RF link is not.

There are other significant differences that involve the design of RF communications circuitry. By using RF modules, however, these differences are hidden from the designer.

For all intensive purposes, the RF module is a “black box” that accepts data at one location and regenerates the data at another location. As long as the data meets certain requirements, the receiving end will regenerate the data to be an exact match of the data present at the transmitting end.

In this application note, we will make several references to the data “channel”. The channel that we are referring to is the path that data takes from the transmitter to the receiver, including all RF circuitry, all base-band circuitry, and the RF propagation path.

Specifically, this application note will describe the channel requirements and limitations for our EWM-900-FDTC modules, though the concepts can be applied to all of our embedded wireless modules. Furthermore, it will introduce a channel coding technique that can be implemented in software to meet the channel requirements. The source code for this software will be presented at the end of this applications note and is available on our software and documentation CD.

In addition to channel coding, the designer must consider the protocol used to communicate information between the transmitter and receiver. The designer should never assume an error-free channel between the transmitter and receiver. Therefore, some protocol must be in place to identify valid transmission at the receiving end and to ignore garbled or corrupted transmissions.

This applications note will also review basic protocol considerations for sending and receiving data using a UART.

2.0 EWM-900-FDTC

This applications note uses the EWM-900-FDTC to demonstrate the data transmission channel characteristics of an embedded wireless module. However, we do not recommend the EWM-900-FDTC for data applications, because it is optimized for voice applications. It is used in this applications note for convenience only.

In order to fully grasp the channel requirements and limitations imposed by the EWM-900-FDTC, it is important to have a fundamental understanding of the module's operation.

Figure 1 shows a block diagram of the module.

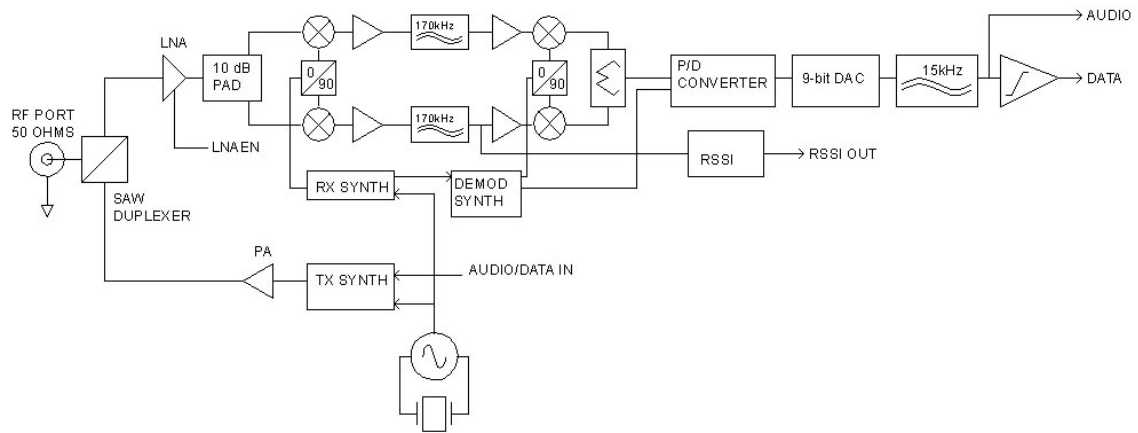


Figure 1: Block Diagram of the EWM-900-FDTC

2.1 FSK Modulation

There are two ways to FSK modulate the carrier of the transmit section: direction modulation of the VCO (voltage controlled oscillator) and modulation of the crystal reference.

We cannot modulate the crystal reference because it is used for the receiver synthesizer. Therefore, we must modulate the VCO directly.

The transmit VCO is part of the transmitter synthesizer. The other components in that system are the PLL, the loop filter, and the reference oscillator (crystal).

The PLL continuously compares the phase of the VCO to the phase of the crystal oscillator through a series of programmable dividers. It outputs an error current, which is filtered using a low-pass filter (loop filter) and then applied to the VCO to adjust the frequency. Any frequency drift in the VCO will be corrected by the PLL as long as the rate of the drift is within the loop filter bandwidth. Therefore, to modulate the VCO directly, the frequency components of the modulation waveform must be above the loop-filter bandwidth. Any frequency component that is within the loop filter bandwidth will be tracked out by the PLL.

The loop-filter bandwidth is 150Hz for the EWM-900-FDTC.

As a rule of thumb, we say that minimum frequency component of the data stream should be three times of the loop filter bandwidth. The minimum frequency component is determined by the longest time that the data stream stays at a 1 or a 0.

This, in-effect, gives the transmit modulation circuitry a high-pass characteristic. In other words, it will act as though the data is AC coupled into the transmitter.

Figure 2 shows this characteristic.

In this figure, the top trace shows the raw data used to modulate the transmit carrier. The data rate is 19.2kbits/second. The data pattern is:

[0x55][0x55][0xFF][0x00][0x00][0x00][0x00][0x00]

The bottom trace is the demodulated audio output from a HP8920A test set. The AC coupling characteristic is apparent. As the 1-0-1-0 combination of the 0x55 bytes are sent, the center point of the waveform begins downward. The rate at which it falls is determined by the loop filter bandwidth. A higher bandwidth will result in a faster decay. A lower bandwidth will result in a slower decay.

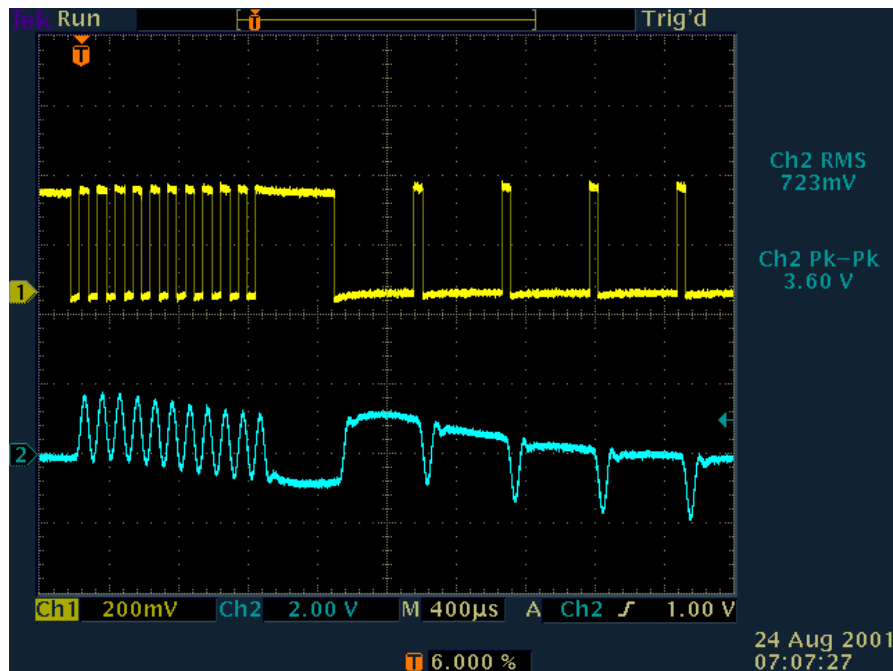


Figure 2: TX Data VS. FSK modulation

The #2 marker at the left of the second waveform marks the demodulator output voltage when the carrier is not being modulated. If the 0x55 bytes were sent continuously, the average voltage of the demodulated waveform would be equal to the voltage at marker #2.

The high-pass characteristic of the transmitter will always try to maintain the average voltage equal to that marker.

In order for the data to be properly recovered at the receiver, the average voltage of the waveform must be close to one half of the peak voltage of the waveform. Otherwise, data distortion will occur.

This limitation affects how the data must be arranged if it is to be recovered properly. If the data to be sent has long strings of 1's and 0's, it must be encoded to remove those low frequency components. A good coding scheme will maintain DC balance, which means that the data will have the same number of 1's and 0's.

2.2 Zero IF Reception

The EWM-900-FDTC receiver section is based on a zero-IF architecture. The received carrier is converted directly to base-band and then demodulated to recover the modulation waveform.

DC offsets in the receiver chain limit the performance of the receiver. Therefore, a DC offset correction circuit is used to compensate the DC offsets. This circuit will act as a high pass filter to the demodulated audio with low-frequency cut-off of about 50Hz.

This high-pass characteristic is less limiting than the transmit characteristic and will not affect the coding requirements of the channel. However, it should be considered when using the EWM-900-FDTC module to receive a signal transmitted from another source.

2.2 Data Slicer

The data slicer is used to regenerate the data waveform from the demodulated audio signal.

A low-pass filter that is used to limit the noise bandwidth of the audio signal precedes the data slicer. This filter's bandwidth is set to 20kHz, which limits the maximum data rate to 20kbits/second (or 10kHz).

3.0 Channel Coding

The purpose of channel coding is to maintain the frequency components in the data stream inside the bandwidth determined by the TX loop filter and receive audio filter.

The technique we are going to discuss here is 2-byte to 3-byte encoding.

Bytes are encoded on a nibble-by-nibble basis, with each nibble being converted to 6 bits. The 6 bit result is then stuffed into the proper position of a three byte word.

Each 6 bit result is selected such that there are the same number of 1's and 0's. In other words, it is DC balanced. There are 20 possible six bit combinations with three 1's and three 0's. Sixteen are used. Encoding is accomplished using the lookup table in table 1.

For every 6 bits received, there are only 16 valid bit combinations. Any other bit combination is illegal and indicates an error. This provides a very simple error detection mechanism.

Decoding is accomplished by reversing the encoding process. The valid 6-bit combination is used to reference a decoding table that contains 4 bit results. Invalid table entries can be populated with a -1 to indicate a decoding error (i.e. a bit or multiple bits were corrupted).

4 bit	6 bit
0000	000111
0001	001011
0010	001101
0011	001110
0100	010011
0101	010101
0110	010110
0111	011001
1000	011010
1001	011100
1010	100011
1011	100101
1100	100110
1101	101001
1110	101010
1111	101100

Table 1: 4-to-6 bit encoding

Figure 3 shows the results of encoding the [0x55][0x55][0xFF][0x00][0x00][0x00][0x00][0x00] byte pattern used earlier. Only the 0x00 bytes are encoded. The 0x55 and 0xFF bytes are explained later in this applications note.

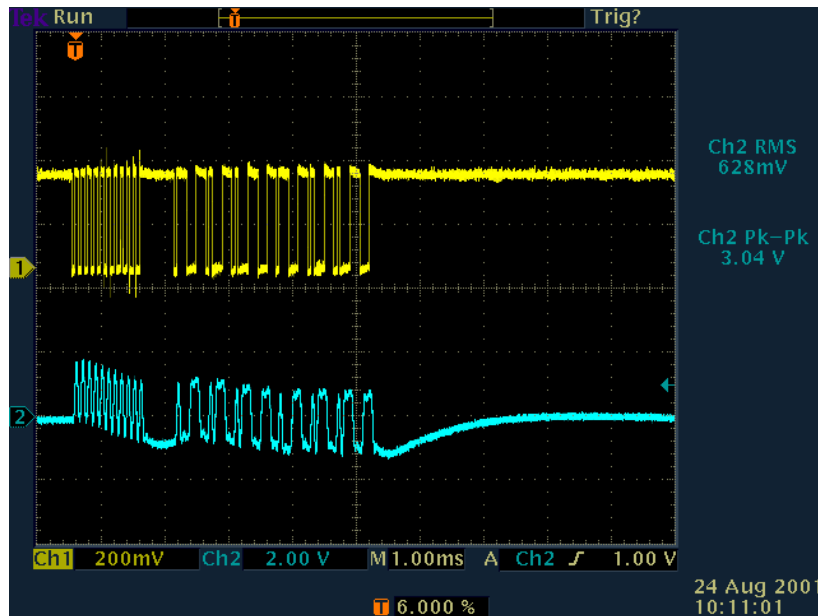


Figure 3: TX Data VS. FSK modulation after channel coding

4.0 Pre-amble

A pre-amble is required to set the slicing threshold of the data slicer.

For the EWM-900-FDTC, this pre-amble is simply two 0x55 bytes.

5.0 UART Synchronization byte

It is possible for the data slicer to incorrectly miss the start bit of the first byte of the pre-amble. If this happens, the subsequent 1-0-1-0 pattern will confuse the UART. The UART will not be able to synchronize on the start bit of subsequent bytes.

The purpose of the 0xFF is to allow the UART to synchronize on subsequent bytes. This will happen because of the lack of 1-0 transitions in the byte.

6.0 Packet Start Codes

So far, we have discussed all of the coding requirements needed due to transmitter and receiver design restrictions. Although the channel coding technique presented here has met the requirements for the transmitter and receiver, we still have no way to discriminate an actual data packet from background noise.

Some modules, such as the Linx SC series transceiver will hold a continuous high level when an un-modulated carrier is present. This is an apparent advantage when using a UART because the first bit transition can be used to mark the start of a packet. In reality, however, these types of modules can hold this high level only with a high receiver SNR. As the SNR degrades, the output of the data slicer begins to hash with digital noise. The input signal level at which this happens is the DC modulation sensitivity. Typically, the DC modulation sensitivity is 5-7dB stronger than the minimum detectable signal. This means that any design that requires that DC level in the presence of an un-modulated carrier will suffer a 5-7dB penalty in receiver sensitivity, which can cut the effective range of the link in half.

In order to fully exploit the sensitivity of a receiver, the design must account for digital noise being present in between data packets.

Figure 4 shows the digital noise that precedes a data packet.

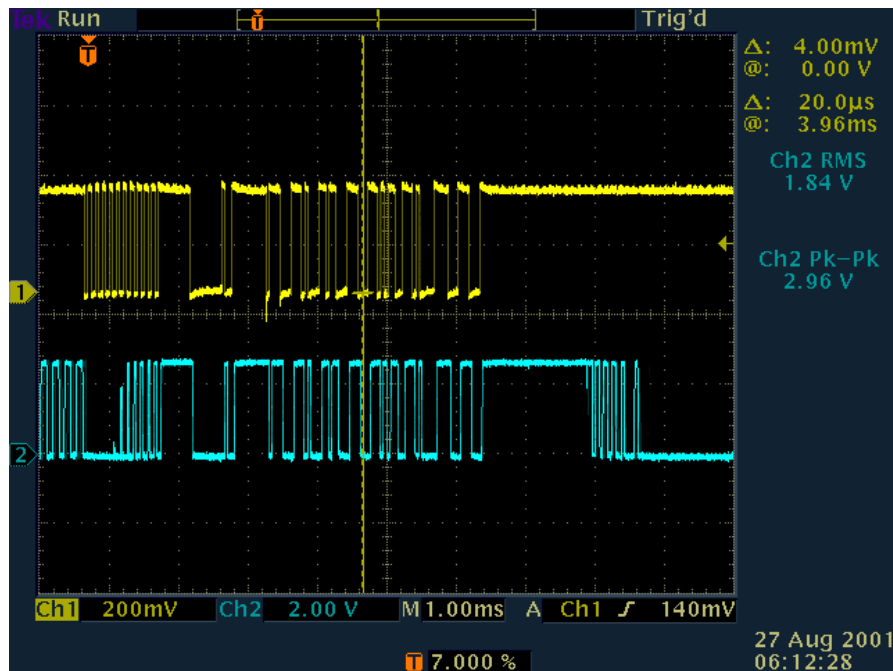


Figure 4: Transmit data vs. receive data

The top trace is the original transmitted data; complete with preamble, UART sync byte, start codes, and data. The bottom trace is the receive data output. Notice the digital noise to the left and to the right of the actual packet.

The purpose of the packet start codes is to provide a unique identification of the start of a packet. These codes are selected so that they will not be found in the digital noise or in the data portion of the packet.

For the EWM-900-FDTC, the start codes are [0x00][0xFE]. Both are codes will generate 6-bit combinations that will be illegal in the encoded data of the packet. Therefore, it should be impossible to synchronize on sequential bytes in the data portion.

7.0 Packet Structure

Putting all of the pieces of the packet together, the structure looks like this:

[0x55][0x55][0xFF][0x00][0xFE][encoded data]

The encoded data portion is the payload-bearing portion of the packet. This can include a packet header, byte count, data, and/or CRC.

8.0 Example Software

Software for encoding and decoding this packet structure is provided in ***an401.c***, which can be found on the software and documentation CD available from Radiotronix.

This software was written in C for the PIC16F877 and was tested using our evaluation boards.

8.1 Packet Generation

The following is a source code snippet from the *an401.c* file. It shows how a packet is generated:

```
while(1)
{
    delay_ms(500); // two packets per second

    /* send the pre-amble */
    putc(0x55);
    putc(0x55);

    /* send the uart sync byte */
    putc(0xff);

    /* send the start code */
    putc(0x00);
    putc(0xfe);

    /* send the data */
    putc_coded(0,1);
    putc_coded(2,3);
}
```

8.1 Data encoding

The packet generation loop calls the function *putc_coded* to encode the data and send it via the UART. That function in-turn calls *encode* to encode the actual data bytes.

The function takes two byte (four 4-bit nibble) and converts them to three bytes (four 6-bit nibbles).

8.2 Packet Reception

The packet reception loop continuously waits for the start codes. Once they are received, the loop will read in 6 bytes and call the decoder.

```
while(1)
{
    while(getch()!=0x00);    // wait for the start byte
    if(getch()==0xfe)
    {
        for(a=0;a<=5;a++)
            inbyte[a]=getch();
        decoder(inbyte[0],inbyte[1],inbyte[2]);
        putc(decbyte[0]);
        putc(decbyte[1]);
        decoder(inbyte[3],inbyte[4],inbyte[5]);
        putc(decbyte[0]);
        putc(decbyte[1]);
    }
}
```

8.3 Data Decoding

The data is decoded by reversing the encoding operation. If there is sufficient RAM to support a 64 byte table, this could be done using a look-up table. To conserve RAM in this example, we used a function called *dectrans* that iterates through the encoding table.

If the *dectrans* returns a decimal 16 then the received code is illegal, indicating a bit error.

9.0 Conclusion

In this application note, a technique for channel coding was introduced that maintains DC balance in the data stream with a minimal overhead. This coding technique also provides a robust method for detecting errors. Protocol considerations were also discussed and source code for the entire implementation of protocol and channel coding was presented.