A Software Implementation for Federal Standard 1052 (Mil. Std. 188-110A HF Modems)

Robert McGwier, N4HY
64 Brooktree Road
East Windsor, NJ 08520-2438
Email: rwmcgwier@home.com

Abstract

Federal Standard 1052 is a modem designed for use on HF. It is specifically designed to overcome the effects of propagation on HF to a certain degree. This paper describes the approach taken by the author in a software implementation for the personal computer. The author has chosen to do the initial implementation on Pentium and Dee-Alpha based computers running Linux.

Keywords: Serial HF Modem Federal Standard 1052 Linux PC

Introduction

In the late 1980's, the author was one of several people who pushed digital processing and its myriad applications and possibilities (McGwier, et. al. l-3). During the early years, our personal computers did not have sufficient power to do the myriad tasks one needed to do in order to implement an extremely sophisticated modem, such as Mil. Std.188-110A. We were users of digital signal processing chips, and while these DSP chips have gotten faster, our general purpose computer microchips have sky rocketed in their power to accomplish difficult tasks.

During the 1990's, we have seen the publication and distribution of many software packages implementing increasingly complex signal processing software for a variety of applications, including a demodulator for the Advanced Composition Explorer (Karn, 4), an HF demodulator designed by Peter Martinez (Jacob, 5), and Forward Error Correction software (Kam, 6). The quality of the sound equipment in computers has steadily improved, as has the drivers needed to efficiently utilize them. The ubiquitous Soundblaster™ by Creative, Inc. has enabled much to be done with our computers.

Almost from the outset of our DSP efforts, there has always been the desire to produce a really high quality HF modem for use by amateur radio operators. It has also always been clear that it was going to take a really serious effort by a signal processing professionals to implement this in the short term. There have been several commercial modems, many of them really good designs. We will not list them here for fear of leaving one or more out. These being proprietary designs, none of them have been as good a learning experience as they could have been. Since no serious signal processing professional has come forward, and the long term has arrived, we will demonstrate an implementation of the Mil. Std. 188-110-A1 demodulator. It is written entirely in C, since I can't force myself to learn C++. It uses standard sound cards for the PC to deliver digital samples to the processor from your audio source and to render audio for your radio. In this paper we will give a cursory overview of the design of the demodulator. After receiving all of the necessary approvals from my employer, the source code will be released under a Gnu Public License. It is the hope of the author that this will increase serious experimentation on HF in a way that we have seen fostered by the release of the latest software design by Martinez (6). Most of this paper will be about the actual standard with the exception of the highlights from the demodulator. Another paper, will be done elsewhere with all of the gory details.

Federal Standard 1052 (formerly Military Standard 188-110-A1)

During the 1980's, the US adopted a standard design for communications using the HF. Its design considerations were primarily to allow more reliable communications on HF that were more reliable than standard FSK based RTTY signals and to make use of the rapidly advancing digital signal processing chips. A block diagram of the design is a little complicated to follow but is included for completeness.

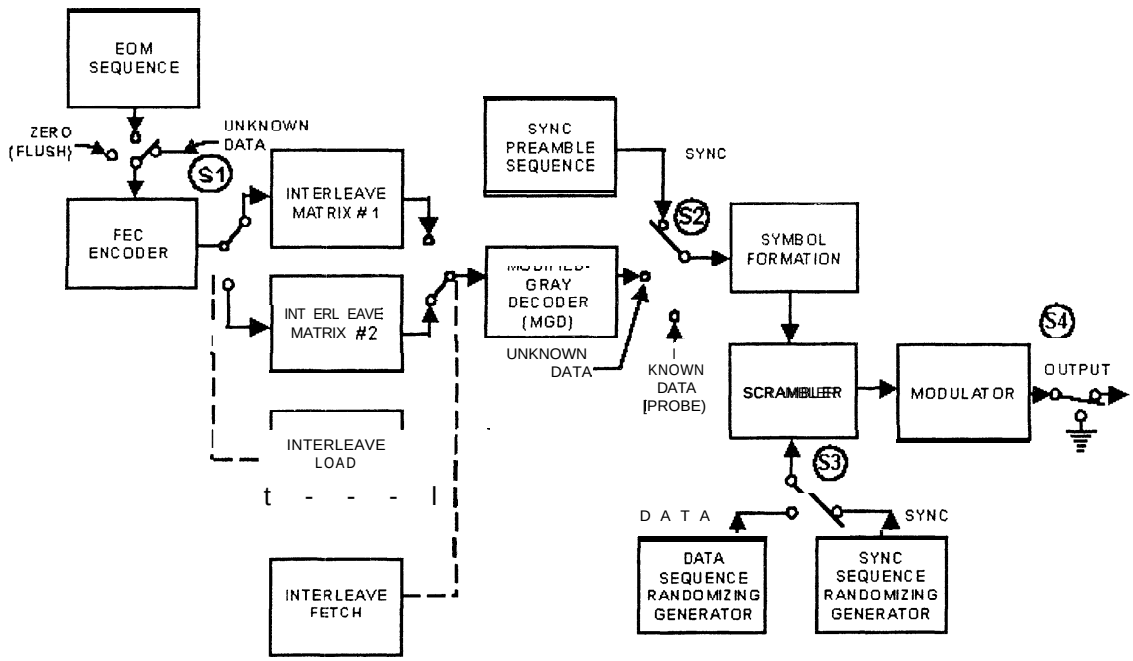


Figure 1. A block diagram of the serial modems operations

The design adopted an 8 PSK signal at 2400 baud as bearer of information symbols. Since it is 8 PSK, every baud carries three bits of data. Data is encoded by changing the phase by one of eight values: 0, pi/4, pi/2, 3pi/4, and so on,, up to 7 pi/8. Since there are eight different angles we could change the phase by, this allows us to transmit three bits of information. So theoretically, we could be sending 7200 bits per second of data with this signal. This would not work very well on HF to say the least.

The first thing that we would find out is that we have no idea where in the data we are, and if we did, which of the eight possible rotations of the data could be the correct one for our communications purpose. That is, we would not know which of eight possible phases to use to decode the data. In order to overcome this, we will transmit training data. This training data in the HF serial modem world and elsewhere is called a preamble. We will send the following numbers, remembering that we send three bits at a time. Thus we will transmit numbers between O-7. When we are sending SYNC, the switch S2 in figure one is in the SYNC position.

The synchronization pattern to transmit is

0, 1, 3, 0, 1, 3, 1, 2, 0, D1, D2, Cl, C2, C3, 0.

The three-bit values of D1 and D2 are very important to our communications channel. They choose at what speed we will transmit data and simultaneously, how much protection we will enforce to help out our data. When we are attempting to find a signal like this, we look for the pattern 0,1,3,0,1,3,1,2,0 in the data being received and then assume we are getting a valid sync and then listen carefully to the rest of the preamble. The rest of the preamble consists of several copies of everything but C1,C2, and C3 being repeated. Cl, C2, and C3 are counters and count down to zero in a special way. After we get to zero, we know that we are to immediately begin to decode what follows as the actual message. Why do we do this? This allows us several chances at detecting that the signal is present. Once we do detect, we are given several pieces of data that are known to us. When you are transmitting known data, it is much easier to measure what effects the HF channel is having on your data, what frequency offset exists between you and the transmitting station, and finally to determine the timing needed to synchronize the receiver and the transmitter; This is the hardest aspect of all such demodulators to get right. Harris RF though their way of doing the channel mitigation job was so nice that the patented it. We do things differently here and is the real impetus for doing this work. We'll return to that later.

While it is straightforward to understand 10,9,8... . 1,0 blast off for the Cl, C2, and C3, the purpose of D1, and D2 require more explanation. It is quite straightforward to understand one aspect of the function of D1 and D2. The worse the channel conditions are, clearly, the slower we must transmit data in order to be successful in getting our message across. But wait, I already said that we were always going to send data at 2400 baud. How can we reconcile these two different views? In order to transmit data at a lower effective rate, even though we are transmitting at 2400 baud, we will use redundancy to help us conquer the noise and interference on the channel. An easy to understand (but dumb) kind of redundancy comes when you just repeat what you are saying and voting on what is the most likely message given the many copies. Here the redundancy comes from forward error correction and in some cases additional redundancy from multiple copies, but we ***never*** majority vote. The first layer of redundancy is done by a standard IBS rate ½ constraint length 7 convolutional code. See Figure 2.

Phil Karn, KA9Q, has done a very nice decoder for this exact code(6). We have gladly stolen that code under the GPL. Convolutional codes are extremely powerful tools for providing for noisy channels in the unlikely (ever) case we have additive Gaussian noise. This is not a good model for HF noise. The noise has a whitish component, but it is very impulsive. This causes burst errors. Convolutional codes do not handle burst errors well in general. A clever device has been introduced to alleviate the fact that the noise is highly correlated. One way would be to decorrelate the effects of the noise by spreading its effects out in time. This is extremely effective and much of the new hot topic in coding, Turbo codes can attribute their success in large part due to the very large decor-relators. These are called interleavers. In Fed. Std. 1052, you take a block of data and apply the FEC to it. You then take the encoded bits and permute their order in time. This has been studied in depth by mathematicians and engineers as to how to do this "optimally." Such a pemutation is used here. You then transmit the encoded+permuted data. If burst errors occur now, their effects are spread out on the receiving end by applying the inverse of the permutation or deinterleaver. This does a great job at decorrelating the noise. The faster you send data, the more time you need to spread the bits out so larger permutations are used for the faster data. Each interleaver takes exactly the same amount of time, irrespective of how fast the data is transmitted. On good channels given a data rate, one uses a short interleaver. Given the same data rate, as the channel degrades, one can go to a long interleaver to help. The disadvantage of the interleaver is the group delay through it. Short interleaver are 0.6 seconds long and long interleavers are 4.8 seconds long!

And still, this is not enough. The channel is just too erratic or *nonstationary* for the estimate we made about the channel during the preamble to be true or effective enough to help us for long. For this reason, ***probes*** are sent interspersed amongst the encoded and interleaved data. These probes are known data. Everyone has agreed ahead of time (in the standard) what the probes will be and when they will be sent. This allows you to relearn the channel to some degree even in the midst of sending unknown data (the message) to the receiver. This allows you to adapt much more quickly to known data in the channel and to remove frequency offset., timing drift, as well as other channel effects. How much known data is transmitted with the unknown data is a function of the D I, D2 values. Table 7 gives the values for the "channel symbols" used for Dl and D2. Table 8 gives the actual bit values to these channel svmbols. The thing to keep in mind when trying to understand why things are done this way, it is to allow for a lousy channel where possible. D1,D2 are streams of bauds. One using correlation against all of the known values for Dl ,D2 in order to decide what is the most likely actual values of D 1 and D2 when you are receiving them. In most of the cases amateurs will use, there will be 20 probes sent for every 20 data bauds. This will give a pretty reliable 1200 bps. Only in the case of 2400 bps do you use 32 unknown data bauds to 16 probes. Data rate, 2400 bps, is not used frequently by the commercial and government users of this standard.

75 bps is handled very differently and is a miserable thing to code but worth every bit of the pain involved in getting it right. There are no known data probes sent. Long sequences of data are sent for each bit and cross correlation is used to determine which values are being sent. Since these have such tight correlation peaks, they are very good data in and of themselves for learning about the channel and attempting to produce good data . This mode will copy data that simply cannot be heard by the ear on an interference free open band channel.

For Phil's decoder to work, soft symbols (probabilities) have to be computed for each bit. One more signal processing trick has made the demodulator better than it otherwise would have been. As you can see from Figure 2., the data is not sent out in the order O,l,. . . as we move around the points on the circle for the constellation values. In all cases but one, any two constellation values decode to a pattern of bits that differ in at most one bit. Intuitively, this is done because the most likely error in a noisy environment, full of clutter, is to mistake one neighbor for another. This is helped tremendously by giving neighbors to the extent possible, very similar addresses. Let's take an example. Suppose I compute a soft symbol to be at 75 degrees with a magnitude of 0.75 and I am transmitting at 1200 bps so I am using the (XY) or dibit symbols. Notice that I am above and to the right of the line from 135 degrees, through the origin, down to 315 degrees. Notice that both symbols here have the same first bit. Thus when I compute a probability to send to Phil's decoder, I am going to be pretty certain that the first bit is a zero. I will be less certain about the second bit but I will claim it looks more like a one than a zero because it is closer to 90 degree line. This type of division of the circle in ways that allow you compute probabilities where some of the bits are really certain is called modified-Gray decoders. Given the way we compute symbols, the probabilities fall right out.

Some miscellaneous information is that the following scrambling sequence for the **sync** preamble shall repeat every 32 transmitted symbols:

7 4 3 0 5 1 5 0 2 2 1 1 5 7 4 3 5 0 2 6 2 1 6 2 0 0 5 0 5 2 6 6

This is added symbol by symbol to the phase of the transmitted data just before the preamble is sent out on the air. In addition to this a randomizer is added to the data symbols and the probes (all zeros) just before they are transmitted. This is a 480 long bit pattern, which is 160, 3 bits symbols, which are added just before transmission. They are derived from the shift in Figure 3 with initial fill 101110101101.

The Software Implementation

We will concentrate here on what is new in the demodulator since everything else is just implementation details. In all demodulators for serial modems on HF, something must be done to overcome the effects of the channel on the data. The performance requirements placed in Table 10 show that you must overcome channel defects that are at times severe. In addition to this, we would also like to be more robust in the face of in band, narrow interferers, where we have chosen to put an adaptive notch filter. All demodulators to date, known to the author, attempt to solve the following problem. Suppose we are giving a stream of data samples coming from an A/D. Let's call this stream Y(t), where t = the integers and correspond to sample times. Let X(t) be zero stuffed data, that is we will assume that the Y(t) are being issued at a rate that is an integer multiple of the baud rate. We will put zeroes in the X(t) where it is not time for a data baud. We assume the following model for the observed data:

$$Y(t) = \mathrm{Sum}_{i=1}^{L} h(l)X(t-l) + W(t).$$

Here h(l) are the filter coefficients which we assume will model the channel operating on the data and W(t) is noise and we assume it is additive.

It is typical to assume (whether or not it is true) that the filter h is approximately invertible. A filter is then applied to the Y(t) that produces a soft symbol. Sometimes there is added to this feedback of the previous decisions $X\hat{}(t)$. An adaption procedure is implemented to minimize the error between the new soft symbol or block of symbols and the hard decisions for X. In some cases, no feedback is done at all. In this case, not all of the intersymbol interference is removed that it is possible to remove. If Harris uses in their radios, exactly what they have patented, then they are not doing decision feedback in the traditional sense but are working their way in from the probes on either side of the unknown data.

Very bad things happen to the equalizer approach and other inversion approaches when the filter defined by h has zeroes in its spectrum. All sorts of tricks have to played to fix the numerical instabilities induced. Trying to invert the channel through an equalizer where the channel has a bunch of zeroes in its spectrum is the mathematical equivalent of dividing by zero.

A better approach would be to take the point of view that we wish to guess putative data symbols, and then find the best filter given this data estimate that will produce the observations Y(t). Given this new filter estimate, one can then find better data. This sounds like an iterative procedure. It is. The first step, given a filter find the data, is called Expectation. The next step of re-estimation of the filter coefficients is called Maximization and the algorithm which implements it is commonly called the EM algorithm as a result. On its face, it is too expensive. If one knew the channel response, *a priori,* one could do the viterbi algorithm and this would indeed outperform the equalizer methods normally chosen but it is very expensive. There is also no theoretically clean way to derive better filter coefficients in an iterative fashion from the viterbi algorithm output other than least squares given the output data. This

coupled with the cost of producing the data makes it prohibitive. Lets take a different approach. For completeness we give the following very dry mathematical rendering of what we do here approximately. It is taken directly from the original paper by Dempster, et. al. (8). If you study the model we have given for the A/D samples, Y(t), it is clear that the data values X(t) are hidden from our view by the filter h, and the additive noise W(t). Since X are assumed to be independent in a certain way, they have nice properties and are a type of mathematical stochastic process called a Markov Chain. This is an extremely powerful concept and allows the analysis and algorithm which follow.

The EM Algorithm in general

Let X be the set of hidden variables, Y the set of observable variables, and Omega_X and Omega_Y the set of values or sample space for these variables. The product of Omega_X and Omega_Y can be thought of as the space of complete samples and Omega-Y the set of incomplete samples. If (x,y) is a complete sample, then y is the observable part.

Let f(x, y I theta) specify a family of functions one of which governs the generation of complete samples and let g(y I theta) specify a family of functions one of which governs the generation of incomplete samples. The functions f and g are related by the following equation.

```
g(y | theta) = Integral_{x in Omega_X} f(x,y | theta) dx
```

The EM algorithm attempts to find a "value for theta which maximizes g(y I theta) given an observed y, but it does so by making essential use of the associated family f(x, y I theta)" [Dempster et al., 1977](7).

EM can be thought of as a deterministic version of Gibbs sampling. EM operates on the means or modes of unknown variables using expected sufficient statistics instead of sampling unknown variables as does Gibbs sampling. Both EM and Gibbs sampling are used for approximation with incomplete data.

Suppose that x is the missing data and y is the observed data. It may be straightforward to calculate p(x, y I theta) but not so easy to calculate p(y I theta).

Define Q(theta I theta') as follows:

```
Q(theta | theta') = E[log p(x, y | theta) | theta', y]
```

where x is the random variable and we are taking the expectation with respect to the probability density p(x I theta',y).

The EM algorithm works as follows

1. Set i to 0 and choose theta-i arbitrarily.
2. Compute Q(theta I theta-i)
3. Choose theta i+l to maximize Q(theta I theta i)
4. If theta_i 1- theta_i+1, then set i to i+l and return to Step 2.

where Step 2 is often referred to as the expectation step and Step 3 is called the maximization step.

The generalized EM (GEM) algorithm is the same except that instead of requiring maximization in Step 3 it only requires that the estimate be improved.

Here is a proof sketch showing that each iteration of EM actually improves the current estimate theta.

We start with the following simple identity (see the Appendix at the end of this document):

```
log p(y | theta) = log p(x, y, | theta) - log p(x | theta, y)
```

and take expectations of both sides treating x as a random variable with distribution p(x Itheta',y).

```
log p(y | theta) = E[log p(x, y | theta) | theta', y] -
                   E[log p(x | theta, y) | theta', y]
```

(1)

We note without proof that the second term in (1) is minimized when theta = theta' (see Exercise 1 below).

Now consider any value theta" such that

```
E[log p(x, y | theta") | theta', y] >
E[log p(x, y | theta' ) | theta', y]
```

and note that if we replace theta' by theta" we increase the first term in (1) while not increasing the second term so that

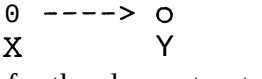
```
p(y | theta") > p(y | theta')
```

thereby improving our current estimate.

In general, computations required for EM can be quite difficult requiring complex integrations to compute the required expectations and perform the maximization. In the remainder of this section, we consider a special case of EM well suited to distributions in the exponential family.

Consider the problem of learning a model for the unsupervised learning problem in which X and Y are boolean variables, Y depends on X, and X is hidden. In this case, complete samples (x,y) are drawn from { 0,1 } ^2, incomplete samples are drawn from { 0,1 }.

The hypothesis space is defined by the following network structure.

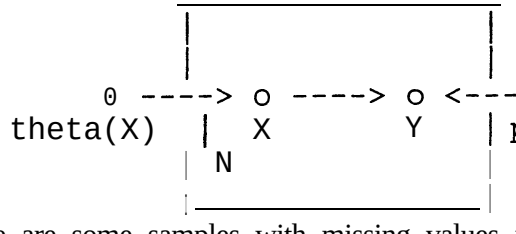


Here are the parameters for the above structure.

```
{theta(x) = Pr(X=x) : x in {0,1}}

{phi(x,y) = Pr(Y=y | x=x) : (x,y) in {0,1}^2}
```

Here is a graphical model representing the learning problem.



The boxed part of the model indicates a plate (see [Buntine, 1994]) representing N samples.

Here are some samples with missing values indicated by underscores.

```
X       Y
---------
_       1       Note that in the general case the
_       0       values for X could be either partially
_       1       or completely missing in the data.
_       0
_       0
   ...
```

We represent the parameter distributions using beta distributions. Let's assume that Beta[n,m] is the distribution that corresponds to out having observed n O's and m 1's. For each parameter distribution, the associated counts correspond to sufficient statistics. In the case of the theta parameters, we have

```
n(x) = Sum-{s in S} (1_{Value(s,X) = x})
```

where S is the set of samples comprising the data, s is a variable ranging over samples, Value(s,V) is the value assigned to the variable V in sample s, 1_{Value(s,V) = v}= 1 if Value(s,V) = v, 1_{ Value(s,V) = v} = 0 if Value(s,V) != v. In the case in which there is no missing data, we have

```
xi(theta(1) | S) = Beta(n(0),n(1))
```

Similarly, for the phi parameters, we have

```
n(x,y) = Sum_{s in S} (1_{Value(s,Y) = y} 1_{Value(s,X) = x})
```

and, again in the case of no missing data, we have

```
xi(phi(1,1) | S) = Beta(n(1,0),n(1,1))
```

In the case of missing data, instead of using the sufficient statistics which we don't have, we use the expected sufficient statistics to iteratively assign estimates to theta and phi. This iterative process proceeds in two steps.

In the first step, called the expectation step, we compute the expected sufficient statistics as follows.

```
E(n(x) | S,theta,phi) =
    Sum-{s in S} Pr(X = x | Y = Value(s,Y), theta, phi)

E(n(x,y) | S,theta,phi) =
Sum-{s in S} (Pr(X = x|Y = Value(s,Y),theta,phi)1_{Value(s,Y) = y})
```

The required computations can easily be carried out for distributions in the exponential family.

In the second step, called the maximization step, we set theta and phi to their mode conditioned on the expected sufficient statistics. In this case, finding the mode is trivial. For example, if the prior on phi(1,1) is Beta[alpha_1,alpha_2], then we're looking for the mode of

```
Beta[alpha_1+E(n(1,0)|S,theta,phi),alpha_2+E(n(1,1)|S,theta,phi)]
```

which is just

```
                                alpha_2+E(n(1,1)|S,theta,phi)
phi(1,1)' = ---------------------------------------------------------
------
                                [alpha_1+E(n(1,0)|S,theta,phi) +
                  alpha_2+E(n(1,1)|S,theta,phi)]
```

The formulation of EM given here is perfectly suited to working with distributions in the exponential families, including the Bernoulli, Poisson, normal, gamma, beta, multinomial, and Dirichlet distributions.

Our Implementation

Chrystomos Nikias and Minn Shao (9) gave nice implementation in a demodulation problem which is almost like ours. They assumed real data (ours is complex), they assumed all unknown data, we make use of the fact that our data is surrounded by known data in all the modes except 75 bps. There, the problem is more like theirs but we still have several things going for us. They make several simplifying assumptions that do cost you performance but allow the algorithm to be adapted to real time processing. Rather than assume that all moments of the distributions are necessary above, they assume that only first and second moments and pairwise conditional expectations are needed in a clever way.

This simple but powerful assumption allows one to approximate the total likelihood problem given above in a meaningful and implementable way. The viterbi algorithm mentioned above, given N observations, L length filter, and M symbols in our constellation costs $O(NM^L)$ operations and $O(M^L)$ memory. As you can see, it pays to allow filter length's L, which are inadequate in order to gain the power of the dynamic program. This is prohibitively expensive. While the full EM algorithm greatly reduces the memory requirement, the cost is still $O(NM^L)$. Under the assumptions in (9), which we have experimentally tested, this cost drops to $O(NM^2)+ O(LN^2)$ operations and $O(N^2)$ memory. This is completely feasible and is similar in cost to the Harris patented equalization algorithm while in no way resembling. It uses approximate total likelihood instead and reestimates filters and data together an algorithm derived from appropriate use of Bayes rule and a Markov model of the data.

Results so far

The demodulator works on the air. We will demonstrate it at the conference. The 75 bps version should outperform all known amateur radio modems with a cost. The occupied bandwidth is a full SSB bandwidth but greatly reduced powers are required for adequate communications.

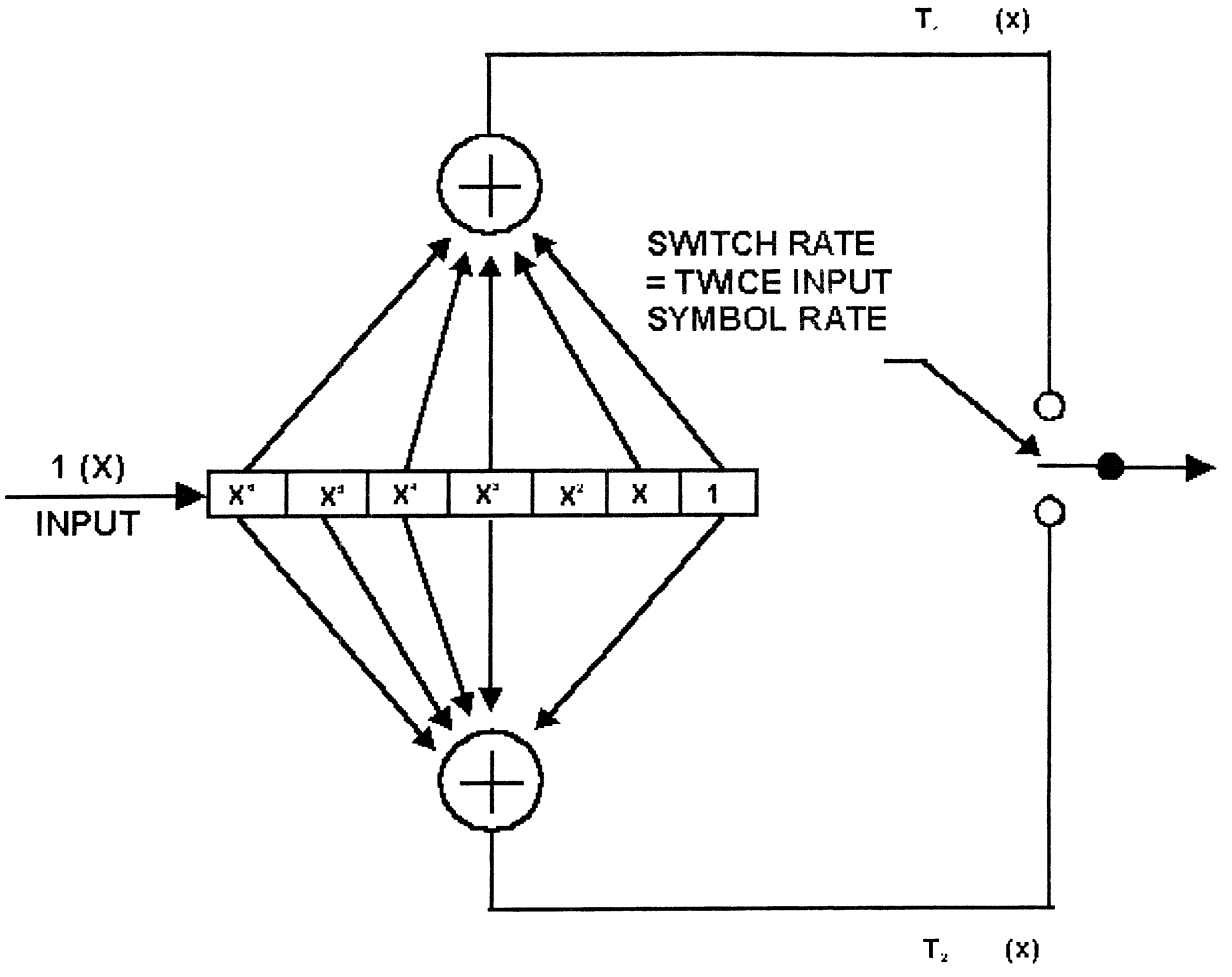


Figure 1. Rate ½ K=7 convolutional code

$T_1(x)=x^6+x^4+x^3+x+1$

$T_2(x)=x^6+x^5+x^4+x^3+1$

Table 1. Error-correcting coding

DATA RATE (b/s)	EFFECTIVE CODE RATE	METHOD FOR ACHIEVING THE CODE RATE
4800	(no coding)	(no coding)
2400	1/2	Rate 1/2
1200	1/2	Rate 1/2 code
600	1/2	Rate 1/2 code
300	114	Rate 1/2 code, repeated 2 times
150	1/8	Rate 1/2 code, repeated 4 times
75	1/2	Rate 1/2

Table 2. Interleaver matrix dimensions

	LONG INTERLEAVER		SHORT INTERLEAVER	
Bit rate (b/s)	Number of rows	Number of columns	Number of rows	Number of columns
2400	40	576	40	72
1200	40	288	40	36
600	40	144	40	18
300	40	144	40	18
150	40	144	40	18
75	20	36	10	9

Table 3. Bits-per-channel symbol

Data rate (b/s)	Number of bits fetched per channel symbol
2400	3
1200	2
600	1
300	1
150	1
75	2

Table 4. Modified-Gray decoding at 4800 b/s and 2400 b/s

INPUT BITS			
First bit	Middle bit	Last bit	Modified-Gray decoded value
0	0	0	000
0	0	1	001
0	1	0	011
0	1	1	010
1	0	0	111
1	0	1	110
1	1	0	100
1	1	1	101

Table 5. Modified-Gray decoding at 1200 b/s and 75 b/s

INPUT BITS		
First bit	Last bit	Modified-Gray decoded value
0	0	00
0	1	01
1	0	11
1	1	10

Table 6. Bits-p-channel symbol

Data rate (b/s)	Number of bits fetched per channel symbol
2400	3
1200	2
600	1
300	1
150	1
75	2

Table 7. Assignment of designation symbols Dl and D2

	SHORT INTERLEAVE		LONG INTERLEAVE	
BIT RATE	Dl	D2	D1	D2
4800	7	6	-	-
2400 (secure voice)	7	7		
2400 (data)	6	4	4	4
1200	6	5	4	5
600	6	6	4	6
300	6	7	4	7
150	7	4	5	4
75	7	5	5	5

Table 8. Channel symbol mappingfor sync preamble

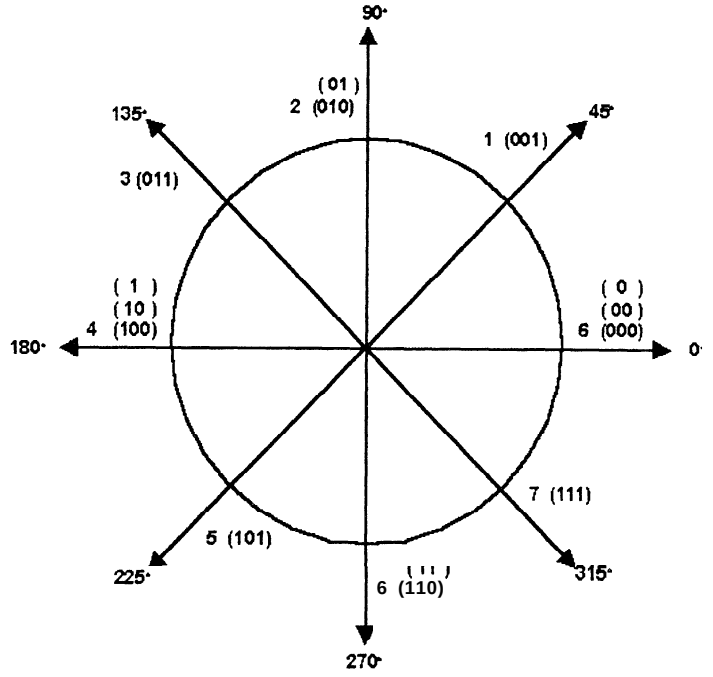
CHANNEL SYMBOL	TRIBIT NUMBERS
000	(0000 0000) repeated 4 times
001	(0404 0404) repeated 4 times
010	(0044 0044) repeated 4 times
011	(0440 0440) repeated 4 times
100	(0000 4444) repeated 4 times
101	(0404 4040) repeated 4 times
110	(0044 4400) repeated 4 times
111	(0440 4004) repeated 4 times

Table 9. Data phase waveform characteristics

Information rate	Coding rate	Channel rate	Bits/channel symbol	8-phase symbols/ channel symbol	No. of unknown 8-phase symbols	No. of known 8-phase symbols
4800	(no coding)	4800	3	1	32	16
2400	1/2	4800	3	1	32	16
1200	1/2	2400	2	1	20	20
600	1/2	1200	1	1	20	20
300	1/4	1200	1	1	20	20
150	1/8	1200	1	1	20	20
75	1/2	150	2	32	all	0

Table 10. Fed Std. Serial (single-tone) mode minimum performance.

User bit rate	Channel paths	Multipath (ms)	Fading BW (Hz) [1]	SNR (dB) [2]	Coded BER
4800	1 Fixed			17	1.0×10^{-3}
4800	2 Fading	2	05.	27	1.0×10^{-3}
2400	1 Fixed			10	1.0×10^{-5}
2400	2 Fading	2	1	18	1.0×10^{-5}
2400	2 Fading	2	5	30	1.0×10^{-3}
2400	2 Fading	5	1	30	1.0×10^{-5}
1200	2 Fading	2	1	11	1.0×10^{-5}
600	2 Fading	2	1	7	1.0×10^{-5}
300	2 Fading	5	5	7	1.0×10^{-5}
150	2 Fading	5	5	5	1.0×10^{-5}
75	2 Fading	5	5	2	1.0×10^{-5}



LEGEND:

0°...315° = PHASE (DEGREES)

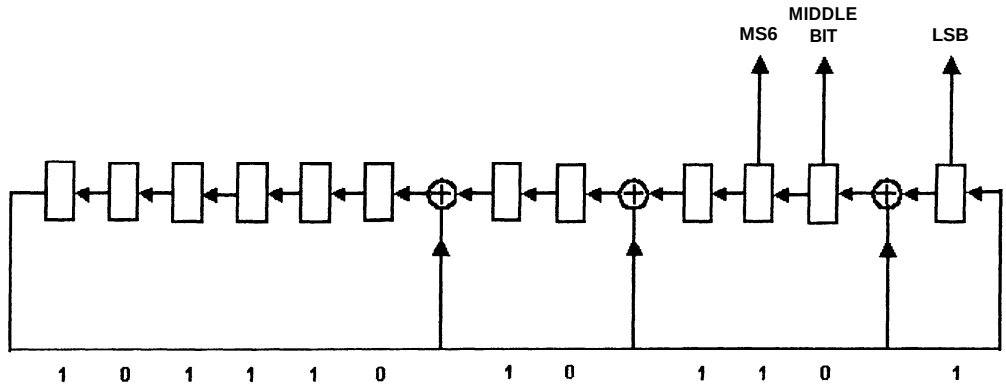
0...7 = TRIBIT NUMBERS

(000)...(111) = THREE-BIT CHANNEL SYMBOLS

(00)...(11) = TWO-BIT CHANNEL SYMBOLS

(s)...(l) = ONE-BIT CHANNEL SYMBOLS

Figure 2. Symbol Values



NOTES:

1. INITIAL SETTING SHOWN
2. SHIFTED 8 TIMES BETWEEN OUTPUTS

Figure 3. Linear Feedback Shift Register for scrambling

Reference

(1) *Digital Signal Processing and Amateur Radio*, Thomas A. Clark, W3IWI, and Robert W. McGwier, N4HY Sixth Computer Networking Conference Proceedings, 1987.

(2) *DSP Modems: It's Only Software,* Robert W. McGwier, N4HY, Sixth Computer Networking Conference Proceedings, 1987.

(3) *The DSP Project Update*, by Thomas Clark, W3IWI and Robert McGwier, N4HY, Seventh Computer Networking Conference, 1988.

(4) *Ace Demodulator,* **http://people.qualcomm.com/karn/code/acedemod**, Phil Kam, KA9Q

(5) *PSK31 Modem*, **http://aintel.bi.ehu.es/psk31.html**, Eduardo Jacob, EA2BAJ

(6) *Forward Error Correction Codes*, **http://people.qualcomm.com/karn/code/fec/**, Phil Kam, KA9Q

(7) *Federal Standard 1052*, **http://ntia.its.bldrdoc.gov/~bing/fs-1052/htm** (formerly MS-188-110A1).

(8) "Maximum Likelihood from incomplete data via the EM algorithm", A.P. Dempster, N.M. Laird, D.B. Rubin, Journal of the Royal Statistical Society, Vol. B-39, pp. l-37, 1977

(9) "An ML/MMSE Estimation Approach to Blind Equalization", Shao, M., Nikias, C., IEEE Transactions on Communications., IV, 1994, ISBN 0-7803-1775-0/94, pp. 569-572.

Biography

Robert McGwier is N4HY. He has published numerous articles in ham radio journals and participated in many AMSAT and TAPR projects. He wrote the demodulators in the AEA DSP2232 and Quiktrak tracking software for AMSAT. He participate in the Microsat project and was a member of the board of directors for AMSAT. While having been mostly absent from ham radio for the last 8 years, he has concentrated on family, Boy Scouts, Golf, and work. Bob resides with his wife, Shann, N2HPE and three children at 64 Brooktree Road, East Windsor, NJ. He is rwmcgwier@home.com.