MSK and its Application to Wireless Data Transmission

1. Introduction

Bandwidth Efficient methods for transmission of data have been and continue to be of great importance to the data dependent community. A community that is experiencing exponential growth. Minimum Shift Keying (MSK) is a continuous phase modulation (CPM) technique that offers advantages in performance and ease of implementation. This paper will provide some insight as to how a simplified MSK modem can be implemented and modeled. The model presented is suitable for a system simulator such as *Ptolemy* from UC Berkeley.

1.1 Background

The academic field of "Data Transmission" is loaded with modulation schemes. Most involve translation of data bits or patterns into a unique combination of phase, frequency or amplitude. Some of the more notable techniques are listed in Table 1.

MODULATION TECHNIQUE	COMMON ACRONYM
Frequency Shift Keying	***FSK***
Multi-level Frequency Shift Keying	***MFSK***
Continuous Phase Frequency Shift Keying	***CPFSK***
Minimum Shift Keying	***MSK***
Gaussian Minimum Shift Keying	***GMSK***
Tamed Frequency Modulation	***TFM***
Phase Shift Keying	***PSK***
Quadrature Phase Shift Keying	***QPSK***
Differential Quadrature Phase Shift Keying	***DQPSK***
π/4 Differential Quadrature Phase Shift Keying	***π/4 DQPSK***
Quadrature Amplitude Modulation	***QAM***

Table 1: Modulation formats [6].

Each of the modulation formats listed in Table 1 is suited to a specific application. In general, schemes where two or more bits are represented by a symbol (e.g. QAM, QPSK) require better signal to noise ratios (SNR) than two-level (binary) schemes for similar bit error rate (BER) performance. Additionally, in a wireless system, schemes that have more than two levels (m-ary) generally require greater power amplifier linearity.

Most implementations of the modulation formats listed in Table 1 are synchronous. When data rates exceed 1200 bits/second or when the transmission medium is subject to non-ideal affects (e.g. fading or SNR < 25 dB) synchronous data transmission is preferred over asynchronous. Synchronous data transmission is characterized by the presence of a clock which is synchronous to the data. The clock is embedded in, and therefore recoverable from, the modulated signal. MSK is a synchronous modulation format.

Another important consideration in data transmission is bandwidth. Digitally modulated data, composed of sharp "one to zero" and "zero to one" transitions, results in a spectrum rich in harmonic content that is not well suited to RF transmission. Hence, digital modulation formats that minimize bandwidth (BW) consumption are in vogue. As implied earlier, digital modulation involves the mapping of changes in data states to changes in amplitude, frequency, phase, or some combination of the three. After smoothing the transitions (discontinuities) in phase, frequency or amplitude, we can see, through Fourier analysis, BW consumption is reduced. An entire family of modulation formats, categorized as

continuous phase modulation (CPM) minimize BW consumption by eliminating phase discontinuities. CPM state changes are represented by non-abrupt changes in phase and frequency while the amplitude of the carrier envelope remains constant (i.e. phase modulation or frequency modulation).

1.2 Theory of Continuous Phase Modulation

The simplest form of constant envelope modulation is binary FSK. This type of modulation is composed of two sinusoids

$$s_1(t) = \cos \omega_1 t \tag{1}$$

and

$$s_2(t) = \cos \omega_2 t \, . \tag{2}$$

Where $s_1(t)$ is selected for a logic zero and $s_2(t)$ is selected for a logic one. An implementation of the this type of FSK system would appear as shown in Figure 1.

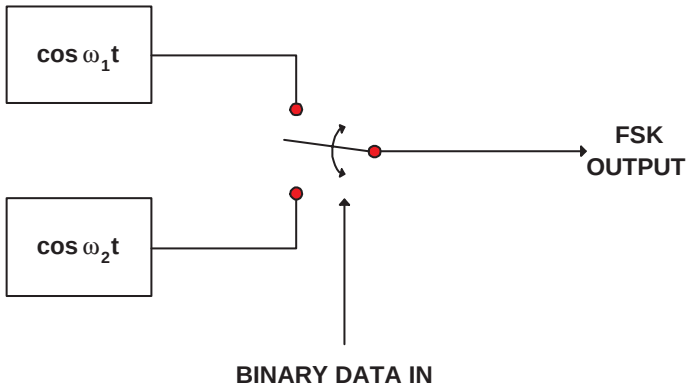

Figure 1:Discontinuous phase FSK modulator.

This type of FSK is referred to as “discontinuous phase” or “non-coherent” FSK because there is no coherence between the phase of ω_1, ω_2 and the modulating signal (BINARY DATA IN). CCITT standard V.21 (300 baud) is an example of non-coherent FSK (f_1 = 1080 Hz and f_2 = 1750 Hz) it has also been applied to other low data rate systems where bandwidth efficiency is not an issue.

An alternative method, to that depicted in Figure 1, for generating FSK is to use a voltage controlled oscillator (VCO) or frequency modulator. The key aspect of the VCO is, for instantaneous frequency deviation proportional to the modulating signal *m(t)*, the phase must be proportional to the integral of the modulating signal. In mathematical terms

$$e(t) = \mathrm{Re}\{A_C \exp[j\phi(t)]\} \tag{3}$$

where, $\phi(t) = \omega_C t + \phi_C + \psi(t)$

$e(t)$ = modulated carrier (FM is a type of angle modulation)

A_C = amplitude of unmodulated carrier

ω_C = angular frequency of unmodulated carrier

ϕ_C = initial carrier phase angle

$\psi(t)$ = instantaneous phase angle due to the modulating signal *m(t)*.

Rewriting equation (3) applying Euler's equation for exponential relations,

$$e(t) = A_C \cos[\omega_C t + \phi_C + \psi(t)] \tag{4}$$

where,

$$\psi(t) = K \int_{-\infty}^{t} m(\tau) d\tau \,. \tag{5}$$

Here, *m(t)* is the modulating signal and *K* is a constant of proportionality. Substituting (5) into (4) yields the equation for continuous phase FSK (CPFSK).

$$e(t) = A_C \cos[\omega_C t + \phi_C + K \int_{-\infty}^{t} m(\tau) d\tau] \,. \tag{6}$$

In the case where *m(t)* is bipolar (logic 1 = +1, logic 0 = -1) the integration in equation (5) can be carried out as follows. First we must have an expression which represents the bipolar data signal as shown below.

$$m(t) = \sum_{n=-\infty}^{+\infty} S_n r(t - nT) \tag{7}$$

where, S_n = ±1, according to bipolar data polarity

r(t) = rectangular pulse with amplitude *1/(2T)* and duration *T*.

Substituting (7) into (5) yields

$$\psi(t) = K \int_{-\infty}^{t} \sum_{n=-\infty}^{+\infty} S_n r(\tau - nT) d\tau \tag{8}$$

$$\psi(t) = K \sum_{n=-\infty}^{+\infty} S_n \int_{-\infty}^{t} r(\tau - nT) d\tau \,. \tag{9}$$

For the interval ***0 ≤ t < T***

$$\psi(t) = K \sum_{n=-\infty}^{+\infty} S_n \int_{0}^{t} \frac{1}{2T} d\tau \tag{10}$$

$$\psi(t) = K \sum_{n=-\infty}^{+\infty} S_n \frac{1}{2T} t \,. \tag{11}$$

Using the result from equation (11), equation (6) can be written as

$$e(t) = A_C \sum_{n=-\infty}^{+\infty} \cos[\omega_C t + K S_n \frac{1}{2T} t + \phi_n] \,. \tag{12}$$

Where, $nT \leq t < (n+1)T$ and ϕ_n is adjusted according to the phase of the previous symbol ϕ_{n-1}. As a result, if the modulating signal (NRZ binary data) has no impulses, the modulated carrier *e(t)* will not have phase discontinuities. A general representation for CPFSK modulator is shown in Figure 2.

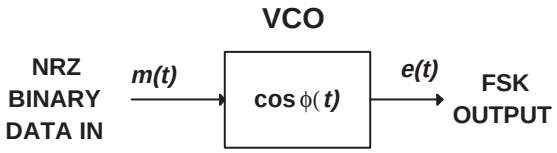

Figure 2:CPFSK generation.

Looking at the term *K* in equation (12)—referred to as a constant of proportionality earlier—for constant envelope modulation systems (e.g. FM) *K* is proportional to the peak frequency deviation of the VCO. For CPFSK, the modulation *m(t)* is NRZ data and

$$K = 2\pi f_d T . \tag{13}$$

Where, $f_d = |f_2 - f_1|$

T = symbol or bit period

f_1 = logic zero (S_n=-1)

f_2 = logic one (S_n=+1).

The term f_dT is commonly known as the modulation index *h*. Substituting $2\pi h$ for *K* in (12) yields

$$e(t) = A_C \sum_{n=-\infty}^{+\infty} \cos[2\pi f_C t + \frac{\pi h S_n}{T} t + \phi_n] \tag{14}$$

for, $nT \leq t < (n+1)T$.

For MSK, h = 0.5 so (14) can be written as

$$e(t) = A_C \sum_{n=-\infty}^{+\infty} \cos[2\pi \left(f_C t + \frac{S_n}{4T} \right) t + \phi_n] \tag{15}$$

for, $nT \leq t < (n+1)T$.

Consequently, if S_n is +1, the frequency of the carrier is shifted higher to

$$f_2 = f_C + \frac{1}{4T} . \tag{16}$$

When S_n is -1, the frequency of the carrier is shifted lower to

$$f_1 = f_C - \frac{1}{4T} . \tag{17}$$

Subtracting (17) from (16) yields,

$$f_2 - f_1 = \frac{1}{2T} \tag{18}$$

Equation (18) states, for *h = 0.5*, the frequency separation between f_1 and f_2 is equal to half the bit rate. Figure 3illustrates a set of four waveforms for a typical MSK modulator.

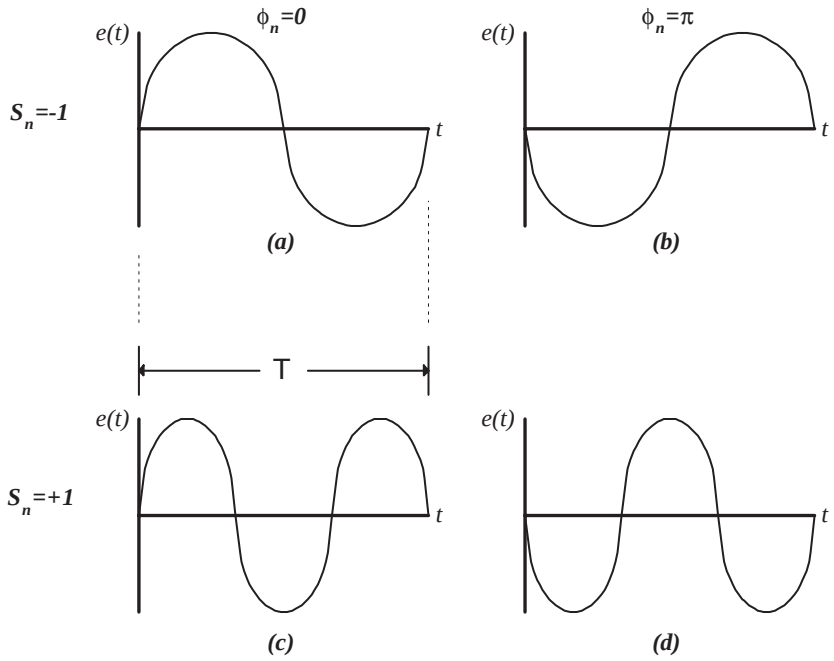

Figure 3: Typical MSK modulator outputs for (a) logic 0 and ϕ_n=0, (b) logic 0 and ϕ_n=π, (c) logic 1 and ϕ_n=0, and (b) logic 1 and ϕ_n=π.

In Figure 3 it should be noted that *e(t)* for a logic 1 (S_n= *+1*) has only one more zero crossing than a logic 0 (S_n= *-1*). In other words, zero crossing detector must discriminate down to the occurrence of a single zero crossing in order to accurately demodulate the data signal. According to [7], a non-coherent zero crossing detection scheme will perform nearly as well as a coherent detector if the FSK signals are orthogonal. For two signals to be orthogonal over an interval *T*, their inner product must be zero,

$$\int_0^T x(t)y^*(t)dt = 0 \tag{19}$$

where, $y(t) \neq x(t)$ and $y^*(t)$ is the complex conjugate of *y(t)*. If *x(t)* and *y(t)* are real then (19) becomes

$$\int_0^T x(t)y(t)dt = 0\,. \tag{20}$$

For any FSK system where $\omega_1=2\pi f_1$ and $\omega_2=2\pi f_2$, equation (20) can be written as

$$\int_0^T \cos(\omega_1 t)\cos(\omega_2 t)dt = 0 . \quad (21)$$

Performing the integration in (21) yields

$$\frac{\sin(\omega_1-\omega_2)T}{2(\omega_1-\omega_2)} + \frac{\sin(\omega_1+\omega_2)T}{2(\omega_1+\omega_2)} = 0 . \quad (22)$$

For MSK equation (18) can be rewritten in terms of radian frequency

$$\omega_1 - \omega_2 = \frac{\pi}{T} . \quad (23)$$

And the radian carrier frequency is

$$\omega_C = \frac{\omega_1+\omega_2}{2} . \quad (24)$$

Substituting (23) and (24) into (22) yields

$$\sin(2\omega_C T) = 0 . \quad (25)$$

Which has the solution

$$\omega_C = \frac{k\pi}{2T} . \quad (26)$$

Where *k* is an integer.

As a result, if equation (26) is satisfied, f_1 and f_2 will be orthogonal. In fact, equation (23) represents the “minimum” separation between f_1 and f_2. Hence, the name Minimum Shift Keying. This modulation scheme was first proposed by Doelz and Heald [3] which resulted in a US Patent, assigned to the Collins Radio Company. Figure 4 is a simplified block diagram for a MSK modulation system.

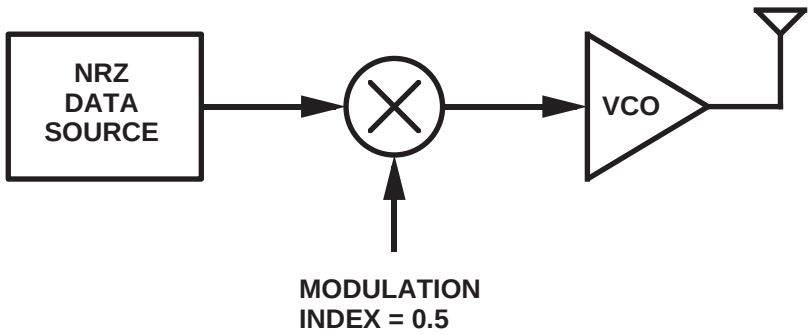

Figure 4: MSK modulator.

A modulation scheme is of little practical value if it can not be demodulated in an efficient manner. The definition of efficiency is somewhat application dependent—in general cost, size, and power consumption must be minimized while performance must be maximized. The performance of a MSK demodulator can be quantified by measurement of the

signal-to-noise ratio (SNR) versus BER. SNR is considered by some as an inaccurate method for comparison of performance because of its dependence on noise BW. In other words, the SNR can be significantly altered by varying the system BW. Unless all systems have identical BW it is difficult to make comparisons. A more accepted measure of performance is E_b/N_0 versus BER. E_b/N_0 is related to SNR as shown below,

$$\frac{E_b}{N_0} = \frac{S}{RN_0} = \left(\frac{S}{N}\right)\left(\frac{B_n}{R}\right). \tag{27}$$

Where, S = signal power

R = data rate in bits per second

N_0 = noise power spectral density (watts/Hz)

E_b = energy per bit

$B_nN_0 = N$ = noise power

B_n = noise BW of IF filter.

Figure 5 is a plot of typical performance characteristics for ideal coherent and non-coherent MSK detectors.

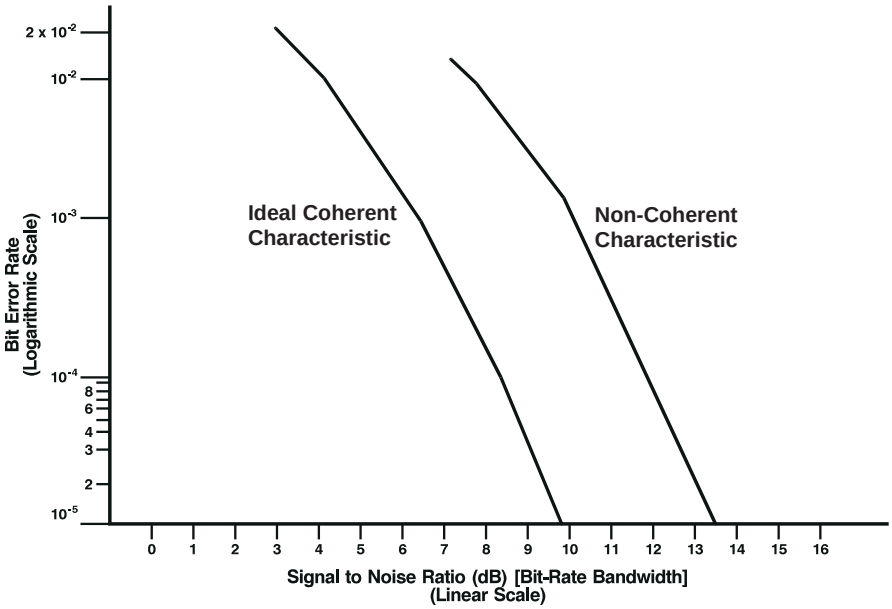

Figure 5: BER versus SNR where noise BW equals the bit rate.

2. Implementation

In this section we describe the implementation of a 1200/2400 bit/second MSK modem. That is, a MSK modulator and demodulator including carrier detection and data synchronization.

2.1 MSK modulator

Depending on selected bit rate and digital data input, the modulator generates one of the 1200 Hz, 1800 Hz, or 2400 Hz frequencies. The frequencies for different data rates and bit polarity (logic one or zero) are given in Table 2.

Baud rate (BPS)	TX data in	MSK tone frequency
1200	1	1200 Hz
1200	0	1800 Hz
2400	1	2400 Hz
2400	0	1200 HZ

Table 2: Tone frequencies for 1200/2400 bps MSK modem.

The modulator generates tones that are phase continuous and change frequency at the zero crossings as shown in Figure 6.

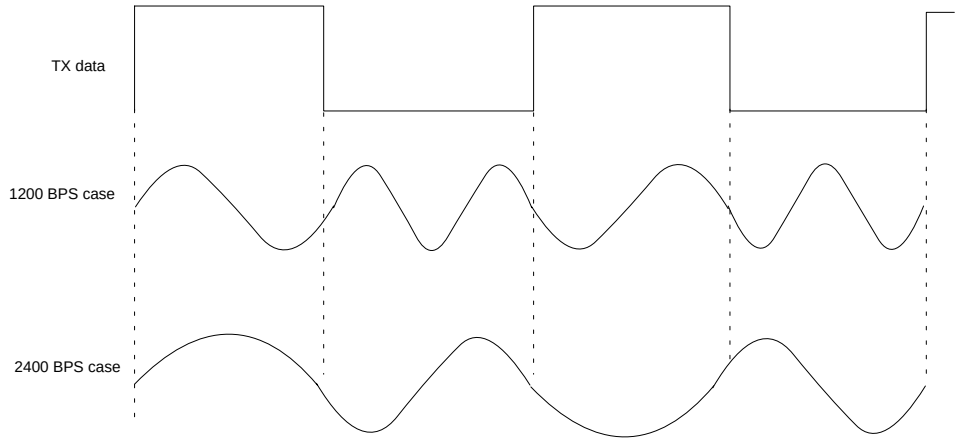

Figure 6: MSK modulator output waveforms.

The modulator block diagram is shown in Figure 7. Primarily, the modulator consists of a digitally controlled oscillator and a low pass filter. The modulation index is 0.5 and the carrier frequency is 1500Hz for 1200 baud and 1800Hz for 2400 baud case. This way of implementing the modulator is sometimes referred to as serial modulation.

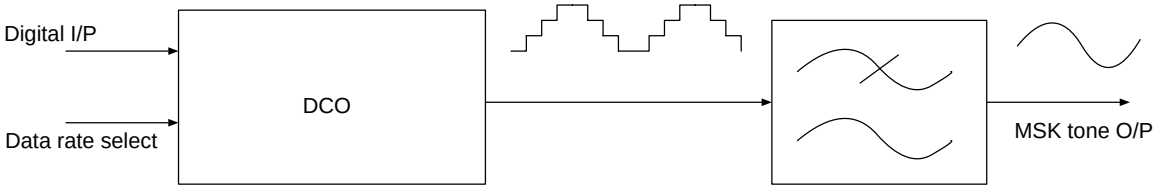

Figure 7: MSK modulator block diagram.

2.2 MSK demodulator

Demodulation is the process of converting the MSK sinusoidal signals to digital ones and zeros. Figure 8 below shows the block diagram of a non-coherent detector. The front end of this detector consists of a band pass filter whose frequency response plot looks similar to Figure 15 for 1200 bps and Figure 16 for 2400 bps. The filter output is sliced to produce a digitized form of the MSK signal, with positive half of the sine wave transformed into high going pulse and the negative half into low going pulse. The monostable is a zero crossing indicator, it produces a pulse every time the input signal changes polarity. The frequency discriminator then differentiates between the two MSK tones based on the zero crossing

signal frequency. This detection scheme is non-coherent because the receiver is neither phase nor time synchronized to the transmitter.

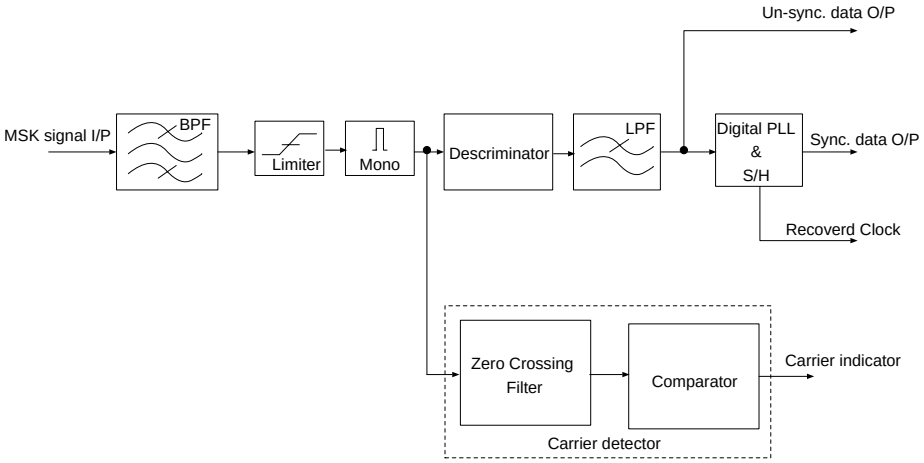

Figure 8: Non-coherent MSK demodulator.

The carrier detector output indicates if MSK signal is being received. The carrier detector works off of the zero crossing indicating signal. The zero crossing filter discriminates between good and bad occurrences of the zero crossings and counts the bad zero crossing occurrences over a pre-defined period. The comparator compares the bad zero crossing count with a pre-set limit to make a decision on the carrier detector output.

The phase locked loop generates a reference clock running at the selected baud rate. The unclocked data output is sampled on the rising edge of this clock to provide synchronized data output. The period of this synchronous clock is adjusted every once per cycle if required to line up the rising edge of the clock with the middle of the unclocked data output.

2.3 Transmit and Receive Synchronization

The modulator provides a transmit reference clock to the user, this clock needs to be used to feed the digital input data to the device.

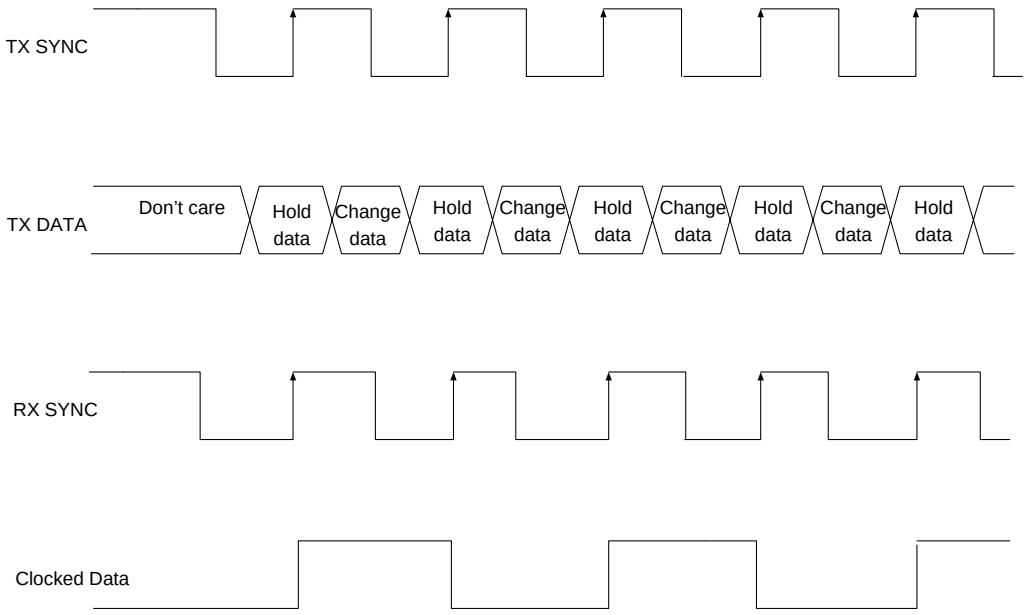

Figure 9: Transmit and receive timing diagrams.

The de-modulator provides the recovered clock and data in reference to this clock. The timing diagrams of the TX and RX sync clocks are shown above. These two clocks can be used to enable synchronous communication between the modem and data terminal equipment.

Figure 10below illustrates an example of how the modem can communicate synchronously with any instrument that has synchronous RS-232C interface.

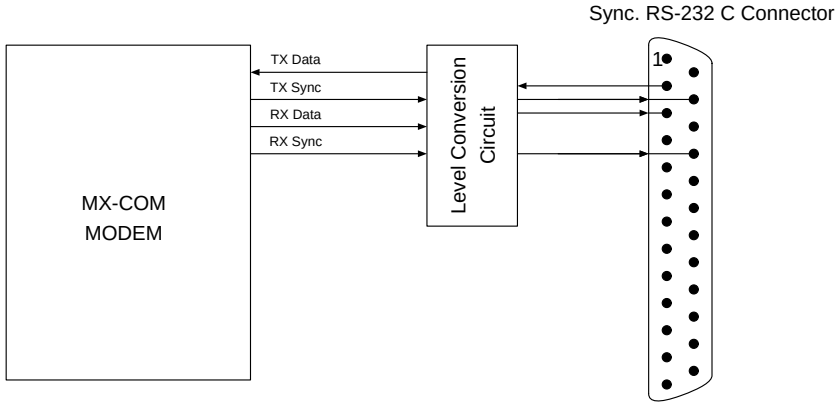

Figure 10: Synchronous RS-232C interface.

2.4 Carrier Detector

The carrier detector is an important feature of the de-modulator. The carrier detect indicator may be used to wake-up or shut down some circuitry in the system to reduce the power consumption.

The carrier detector response time is user controllable via an external input. By altering the state of the external input the user is able to trade sensitivity for response time. In other words, the faster the response the more likely the carrier detect will false on noise and if falsing must be minimized then response time must be compromised.

3. Modeling

With increasing demand to move products quickly to market and to keep costs down, verifying a design through simulation before building a breadboard saves time and money. The rapid growth of PC based simulation tools brings modeling capability to a larger number of design engineers. The verification process can be broken down into three steps:

1. **Creating a model of the design** - Software modeling is a process of emulating hardware blocks of a design. Models can be designed on a hierarchical building block approach. The lowest level models may emulate simple hardware functions such as filters, amplifiers, or digital gates. Several lower level models can be joined together to create more complex models such as modulators or demodulators. These higher level models can be joined together to create even higher level models such as a complete communication system.

2. **Simulating the function of the model** - Simulation is the process of stimulating a model with various inputs and observing the models response. The response of the model can be recorded in plots and data files for analysis.

3. **Analyzing the plots and data collected from the simulation** - Analysis is the process of verifying the plots and data against design goals. This process helps the designer ensure correctness of or see how changes may improve the design.

The models, simulation, plots and pseudo code included in this section were derived using a modeling tool called Ptolemy. Ptolemy is a UNIX based modeling tool that has been ported to run on a PC based UNIX system called Linux. Linux and Ptolemy are freeware software packages and are available to be downloaded from several sites on the internet. One such site is *sunsite.unc.edu*. The Ptolemy home page is http://ptolemy.eecs.berkeley.edu. Other archives sites can be found using a web browser search engine. There are several other PC based modeling tools available from various software companies.

The scope of this document is to model an MSK communications scheme. Each topics will be broken up by function. The topics will include modeling MSK generation and detection (including a description of bit synchronization and carrier detection) and modeling MSK at the system level. The conclusion will review the topics and results presented in this document.

Discrete Time Models

All the models presented below are in the discrete time domain with successive samples uniformly spaced in time. The complete transmitter->channel->receiver model is a multirate system, meaning that different sampling rates are used at various stages in the model. For example the input data bits to be transmitted are sampled at the baud rate either 1200 or 2400 samples per second. The transmitter model's output sample rate is N times the baud rate. The majority of the receiver also runs at this N times rate. We typically used N=60 for the simulation results presented below. The emphasis of this section is on modeling the basic functions included in a typical serial MSK modem offered by MX-COM. RF interfaces and channel models are discussed to aid the reader in developing a complete system model however they are not fully developed. In order to simulate frequency offsets between the transmitter and receiver an interpolating/decimating block that changes the sample rate by M/N can be inserted between the transmitter and receiver. For a 0.1% frequency offset let M=1000 and N=999. For in-depth coverage of multirate processing see [2].

In the following sections, pseudo-code is provided to show the algorithm or function performed by each of the blocks in the model. The authors hope the pseudo-code will aid the readers in modeling and performing simulations of their system designs. The pseudo-code is presented in a simple manner ignoring certain programming considerations such as memory allocation, minimal buffering, type declaration, and scheduling of execution sequence. The models below focus on the core algorithm not the implementation details. Various system simulation environments such as Ptolemy have their own unique ways of dealing with these considerations which are beyond the scope of this paper.

The following conventions are used in the pseudo-code.

// starts a comment which ends at the end of the line

= assignment or comparison depending on context

+ , - , $\times$, / all have their normal meaning. Division is also shown as $\frac{x}{y}$ and multiplication as $x \cdot y$

Y[x] x is the index of array Y, x and Y are used here as an example - they may have any name. Inputs and outputs of each model are treated as an arrays. When referring to an input or output, increasing values of x progress forward in time. Thus Input[x-1] would refer to the previous input to the model at sample x. If a block is multirate, then it consumes more or fewer input samples than it creates output samples. For example the MSK modulator model creates N output samples for every input bit. That is upon processing Input[i], it will create Output$[N \cdot i]$, Output$[N \cdot i + 1]$, Output$[N \cdot i + 2]$, up to Output$[N \cdot (i+1) - 1]$

F(x) function F( ) evaluated with parameter x. For example:

sin(x) is the sinusoidal function of trigonometry

cos(x) is sin(x + π/2)

sinc(x) is sin(x)/x and has the value 1 at x equal 0

sign(x) returns 1 for 0 and positive x and -1 for negative x

for i=0 to N-1 { command sequence } iterates command sequence N times with i increasing by 1 each iteration.

while (expression) { command sequence } repeats command sequence as long as expression remains true.

if (expression) { seq. 1 } else {seq. 2} executes seq. 1 if expression is true otherwise executes seq. 2.

3.1 MSK Modulation

MSK modulation is the process of converting digital data to sinusoids that represent digital data values. In MSK modulation digital data must be converted to a bipolar representation of the data. The bipolar data is then modulated with a modulation index of 0.5. The resulting frequencies are equal to the carrier frequency plus and minus one forth of the baud rate. The initial phase constant of the VCO determines whether data transitions occur at the signal peak (i.e. cos MSK) or the zero crossing (i.e. sin MSK).

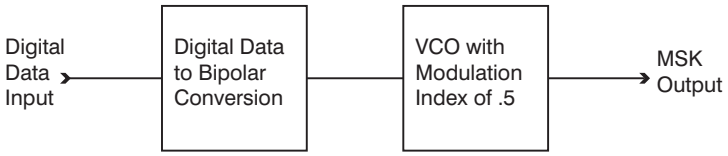

Figure 11: Typical MSK Modulator.

3.1.1 1200 Baud MSK Signals

Figure 12 shows 1200 baud MSK data modulated with a 1500 hertz carrier. MSK sine waves are continuous phase and can only start at zero for a phase constant of zero and pi radians. The first four traces show the sin MSK symbol shapes for data 1 and 0, while the fifth one shows the eye diagram. See Figure 19 for a sequence of bits at 1200 baud.

3.1.2 2400 Baud MSK Signals

Figure 13 shows 2400 baud MSK data modulated with a 1800 hertz carrier. All four symbol shapes are shown followed by the corresponding eye diagram. See Figure 20 for a sequence of bits at 2400 baud.

Pseudo-Code Listing 1 MSK Modulation Model

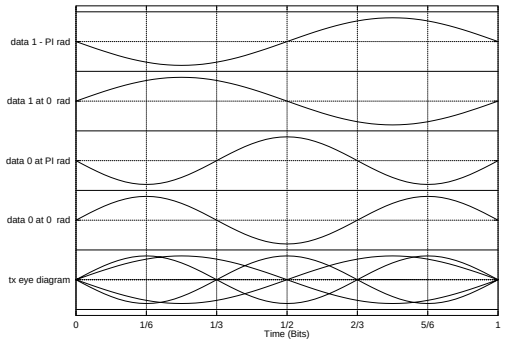

Figure 12: 1200 Baud sin MSK.

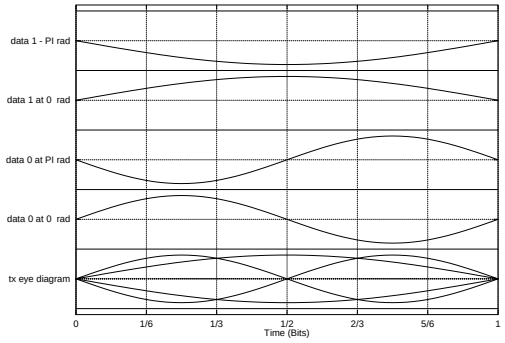

Figure 13: 2400 Baud sin MSK

I/O, Variables, and Constants

```
input[ ]      // transmit data bit array
output[ ]     // output array
N             // interpolation rate -> output sample rate = N x baud rate
2πp[i]        // phase constant in radians  0,π for sin MSK  π/2, 3π/2 for cos MSK
p[i]          // phase constant at sample i has value of  0 or  0.5
Map[0] = 1    // input data 0 mapped to bipolar data 1
Map[1] = -1   // input data 1 mapped to bipolar data -1
c             // carrier normalized to the baud rate
              // 1200 Baud Value = 1.25 = 1500(Carrier)/1200(Baud Rate)
              // 2400 Baud Value = 0.75 = 1800(Carrier)/2400(Baud Rate)
```

Pseudo-Code for MSK modulator

```
dk = Map[input[i]];                  // Map input data value (dk) for this symbol.
for n=0 to N-1 {                     // Generate N output samples per input bit
```

$$\text{output}[N \cdot i + n] = \sin\left(2\pi\left[\left(c + \frac{dk}{4.0}\right) \cdot \frac{n}{N} + p[i]\right]\right);$$

}

p[i+1] = fractional remainder of $\left(p[i] + c + \frac{dk}{4.0}\right)$; // calculate phase constant for next input bit

---

3.2 MSK Demodulation

MSK Demodulation is the process of converting Sinusoids to digital data. MSK demodulation looks for specific frequencies that represent digital data. Shown in Figure 14is a block diagram for a typical MSK demodulator. MSK demodulation can be broken down in to three sections, signal demodulation, bit synchronization and MSK carrier detection.

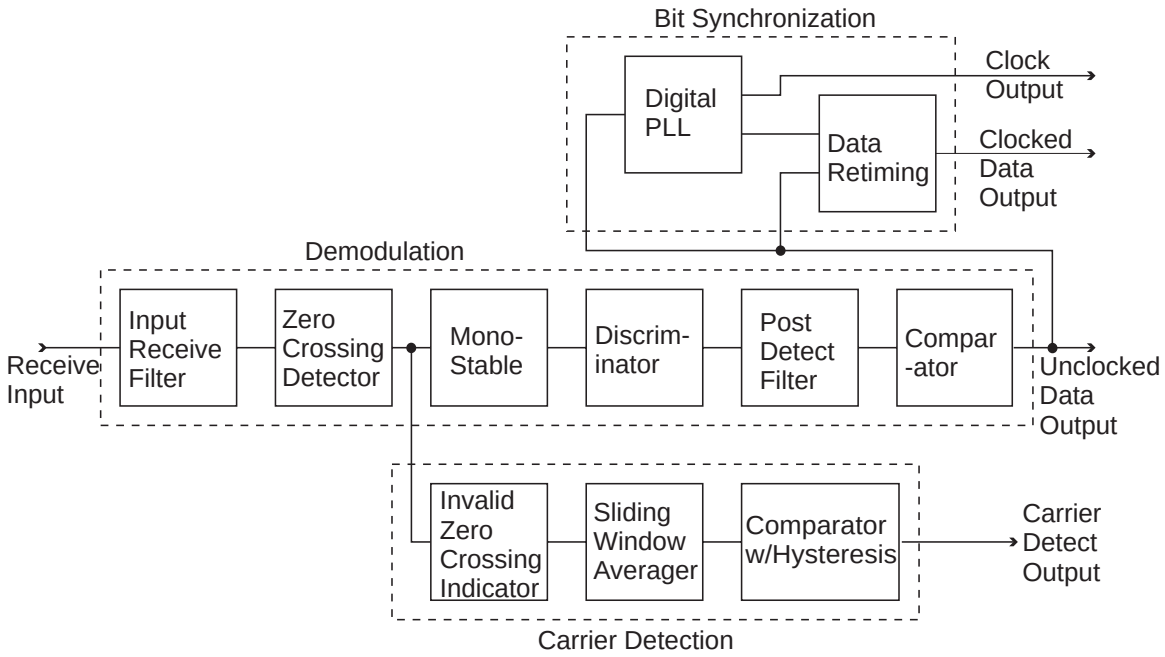

Figure 14: MSK Demodulation Block Diagram.

3.2.1 Demodulator

The MSK detector demodulates the digital information from the received analog signal. There are six blocks that make up the detector. Each block processes the output of the previous block. The output of the sixth block is recovered digital data. The blocks are listed below and will be discussed in order of signal progression.

(a) Input Receive Filter

(b) Zero Crossing Detector

(c) Monostable

(d) Discriminator

(e) Post Discriminator Low Pass Filter

(f) Data Slicer/Comparator

3.2.1.1 RX Filter

The receiver's input filter is a band pass filter centered around the main lobe of the MSK spectrum. It is designed to have a passband width equal to the bit rate (i.e. 2100-900 = 1200Hz and 3000 - 600 = 2400Hz). The frequency domain

impulse responses for both baud rates are shown below in Figure 15 and Figure 16. These graphs show the magnitude responses of linear phase finite impulse response (FIR) filters. The filter weights are based on a windowed difference of sinc( ) functions such as:

$$\left[\operatorname{sinc}\left(\frac{2\pi \cdot f_H \cdot t}{f_S}\right) - \operatorname{sinc}\left(\frac{2\pi \cdot f_L \cdot t}{f_S}\right)\right] \cdot \text{window\_function}(t) \qquad (28)$$

where:

f_H is the upper cutoff frequency of the bandpass filter

f_L is the lower cutoff frequency of the bandpass filter

f_S is the sampling frequency of the bandpass filter

Most system simulation packages include tools for generating filter coefficients based on spectral templates, thus coefficients are not tabulated. If an infinite impulse response (IIR) filter is used then it should be group delay equalized over the passband. See section 3.4 for a simple model of an FIR filter.

For thorough coverage of digital filters refer to a digital signal processing text book such as [2],[5] or [8].

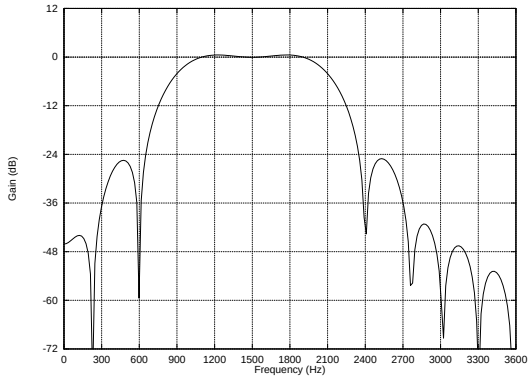

Figure 15: Rx 1200 Baud Filter.

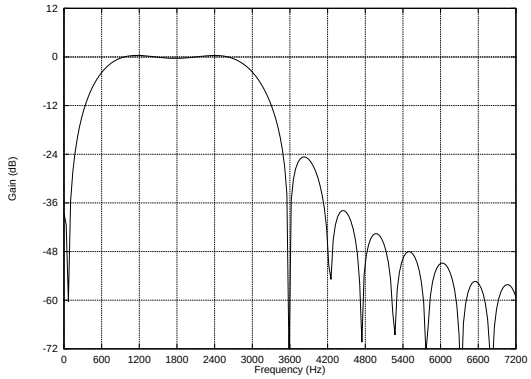

Figure 16: Rx 2400 Baud Filter.

3.2.1.2 Zero Crossing Detector

The zero crossing detector monitors the incoming sinusoid signal from the RX input filter at a rate of N samples per symbol. When the signal passes through zero the output is pulsed high for one sample period. The zero crossing detector then monitors for the next zero crossing.

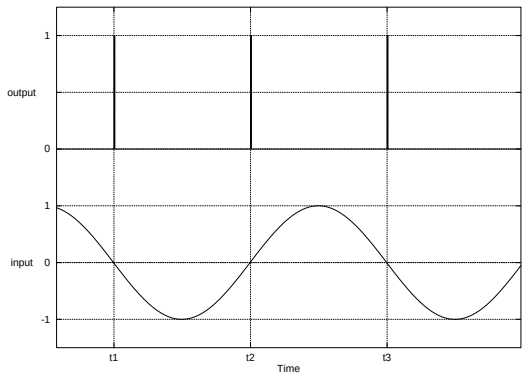

Figure 17:Plot of Zero Crossing Function.

Pseudo-Code Listing 2 Zero Crossing Detector

I/O

input[i] = current input sample

output[i] = current output sample

Pseudo Code

```
if sign(input[i]) != sign(input[i-1])
    output[i] = 1;
else
    output[i] = 0;
```

3.2.1.3 Monostable

The monostable monitors the pulsed output signal from the zero crossing detector. When a pulse is detected the output of the monostable is set high and the timer is restarted. The value of the timer is set to 1/3 a symbol time for 1200 baud and 1/2 symbol for 2400 baud. When the time-out period of the monostable expires the output is set low. If a new pulse is detected on the input before the timer has expired, the timer is restarted and the output remains high. Figure 18 shows the function of the monostable.

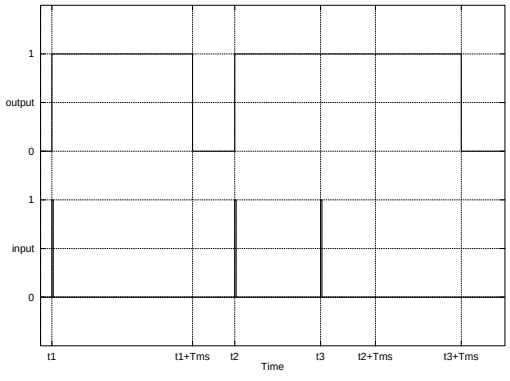

Figure 18: Plot of Monostable Function.

Pseudo-Code Listing 3 Monostable Model

I/O, Variables, and Constants

```
input[i]          // Current input sample
output[i]         // Current output sample
T_ms = N/3        // 1/3 Bit time @ 1200 Baud
T_ms = N/2        // 1/2 Bit time @ 2400 Baud
timer = 0         // initialize timer first time
```

Pseudo Code for Monostable

```
if input[i] = 1     then    timer = T_ms;
if  timer > 0       then    output[i] = 1;
                    else    output[i] = 0;
if timer > 0        then    decrement timer;
```

3.2.1.4 Discriminator

The discriminator uses the output signal of the monostable to reconstruct the digital information. In Figure 19 and Figure 20, the operation of the discriminator is displayed. The signals in these figures are under perfect conditions without passing through the receive filter. Under more realistic conditions the output the discriminator looks much less perfect and hence needs to pass through the Post Discriminator Filter to look more like the transmitted bits. See the discriminator output trace in Figure 31 to understand the need for the low pass filter.

3.2.1.4.1 1200 Baud Case

In Figure 19, the TX data in signal is included as a reference to the digital data being decoded. The TX MSK out signal is the modulated signal that is applied to the zero crossing detector. As seen in this signal, a data bit one has 2 zero crossings while a data bit zero has 3 zero crossings per bit time. The monostable output signal is the input to the discriminator. (Refer to the monostable for an explanation of it's output signal). The input signal to the discriminator is time shifted to create two additional signals. The first is shifted by one sixth of a bit time and the second is shifted by one third. These three signal are logically NANDed together to produce the discriminator's output signal. For a logic one the zero crossings are spaced such that the output of the monostable is low for one sixth of a bit time before the middle of the bit and again before the end of the bit. The logical NANDing of the three signals creates a pulse width equal to one half of the bit period for each of the two zero crossings in a logic one. This creates a digital one with the appropriate bit timing. For the zero bit, the zero crossings keep the monostable output high so the NAND output stays low.

3.2.1.4.2 2400 Baud Case

In Figure 20, the TX data in signal is included as a reference to the digital data being decoded. The TX MSK out signal is the transmitted signal for which the discriminator is decoding. As seen in this signal, a data bit with the value of one has 1 zero crossings and a data bit with the value of zero has 2 zero crossings per bit time. The monostable output signal is the input to the discriminator. The input signal to the discriminator is time shifted by one half of a bit period and is logically NANDed with the input. The logical NANDing of the two signals creates the proper bit period. For a zero, the zero crossings keep the monostable output high so the output of the NAND stays low.

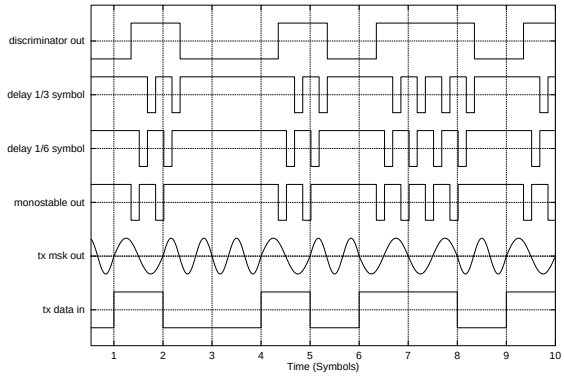

Figure 19:1200 Baud Discriminator Signals.

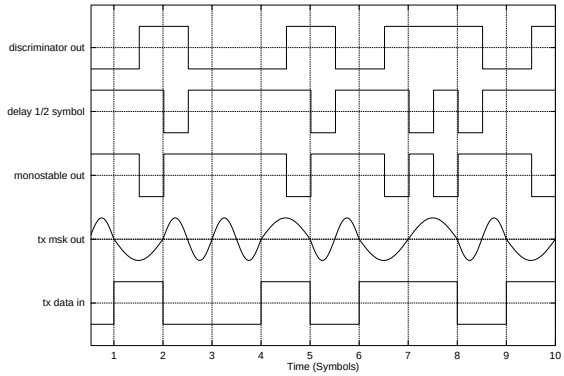

Figure 20: 2400 Baud Discriminator Signals.

Pseudo-Code Listing 4 Discriminator Model

```
I/O, Variables and Constants

    input[i]                        // current input sample
    output[i]                       // current output sample
    Taps[]      = {0, N/6, N/3}     //1200 Baud
                = {0, N/2}          // 2400 Baud
    state                           // variable used to compute logical And of delayed inputs

Pseudo Code for Discriminator       // runs at N x baud rate

    state = 1                       // initialize to true for each input sample.
    for x = 0 to size of Taps-1
        state = state & input[i - Taps[x]]       // compute logical And of each delayed input

    output[0] =  complemented value of state     // logical Invert
```

3.2.1.5 Discriminator Low Pass Filter

The discriminator low pass filter is designed to pass frequencies up to half the bit rate. This corresponds to the highest frequency needed to pass an NRZ data sequence 101010... As seen in spectral plots in Figure 21 and Figure 22, the discriminator low pass filter rolls off sharply beyond its cutoff to remove high frequency noise out of the discriminator. These plots depict a finite impulse response filter with weights of a windowed sinc( ) function. See section 3.4 for a finite impulse response filter model.

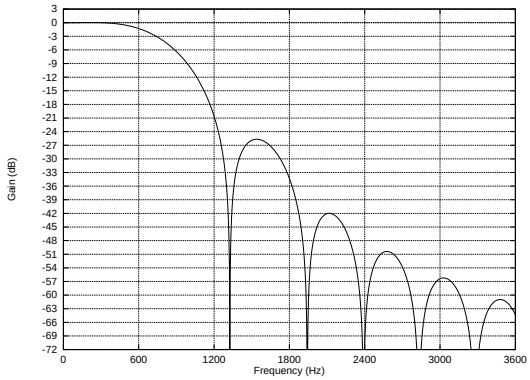

Figure 21: 1200 Baud LPF.

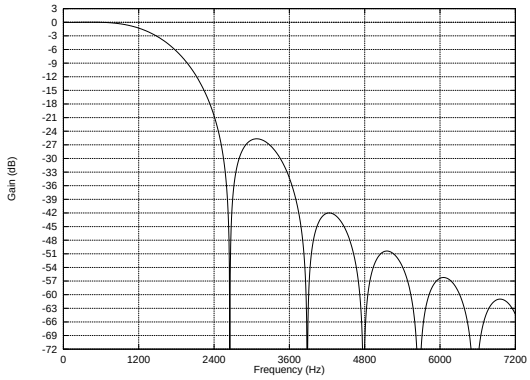

Figure 22: 2400 Baud LPF.

3.2.1.6 Bit Slicer

The bit slicer is simply a comparator with a threshold half way between a one and a zero. That is it outputs 1 for inputs greater than 0.5 and output 0 for inputs less than 0.5. Section 3.4 shows pseudo-code for a comparator. That model provides for hysterisis; however for use as a bit slicer both the upper and lower threshold can be set to 0.5 to essentially eliminate the hysterisis.

3.2.2 MSK Carrier Detection

The carrier detector is made up of three blocks, the bad zero crossing detector, sliding window filter and Comparator with hysterisis. The carrier detect section monitors the information out of the zero crossing detector to determine whether the zero crossing are timed properly for a 1200 or 2400 baud MSK signal. If the zero crossings are properly spaced a logic level is output to indicate that a valid MSK signal is being received.

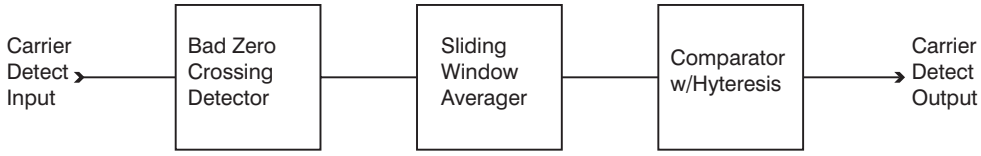

Figure 23: Carrier Detect Block Diagram.

3.2.2.1 Bad Zero Crossing Detector

The Bad Zero Crossing Detector monitors the zero crossings of the filtered input signal. The time between zero crossings is compared against a referenced. If the Bad Zero Crossing Detector determines that a zero crossing is not properly time spaced the output pulses high, otherwise the output remains low. Figure 24 and Figure 26 show the timing windows used for 1200 and 2400 baud. Figure 25 and Figure 27 show that with a properly timed input signal (i.e. zero crossings within the window) the output is low. With an improperly timed input (i.e. zero crossing outside the window or end of window reached) the output pulses high. The windows are based on the ideal zero crossing spacings +/- 5% of the bit time. Thus each baud rate would have two valid windows, however the receive input filter introduces a slight intersymbol interference at symbol boundaries and thus the windows need to widen towards each other to account for this. Since the windows of the 1200 Baud case become very close, a single window is used.

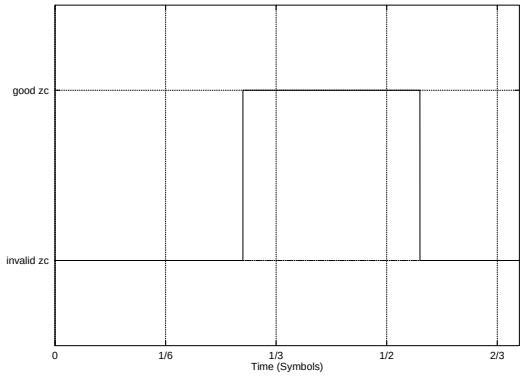

Figure 24: 1200 Baud Valid Zero Crossing Window

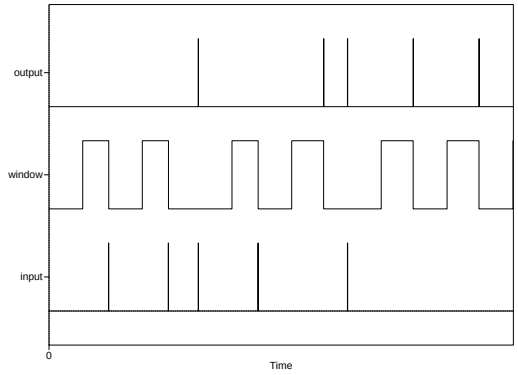

Figure 25: 1200 Baud Bad Zero Crossing Detector.

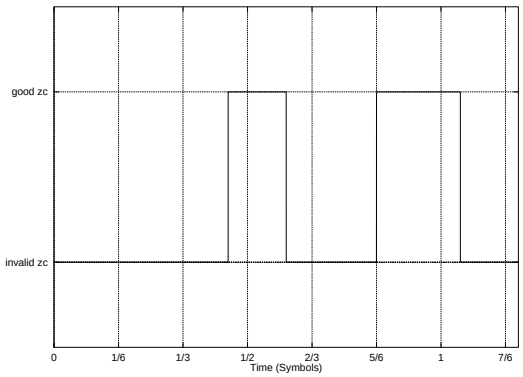

Figure 26: 2400 Baud Valid Zero Crossing Window.

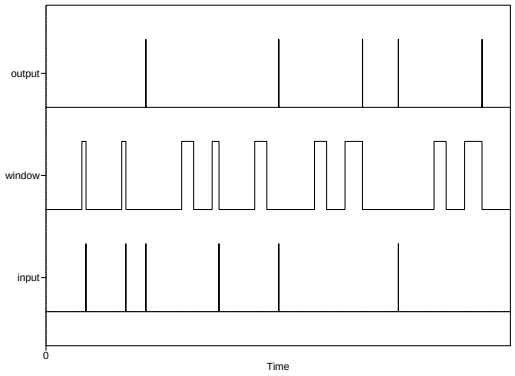

Figure 27: 2400 Baud Bad Zero Crossing Detector.

Pseudo-Code Listing 5 Bad Zero Crossing Detector

I/O, Variables and Constants

```
input[i]            // current input sample
output[i]           // current output sample
START               // Array of valid-window START times
    {Nx0.2833}          // 1200 baud - one window
    {Nx0.45, Nx0.8}     // 2400 baud - two windows
STOP                // Array of valid-window STOP times
    {Nx0.55}            // 1200 baud - one window
    {Nx0.6, Nx1.05}     // 2400 baud - two windows
counter             // used to keep track of time since last zero crossing occurred
OK                  // flag used to indicate properly timed zero crossing
```

Pseudo-Code for Bad Zero Crossing Detector

```
OK = TRUE;                          // flag defaults TRUE for no zero crossing input
if (input[i] = 1) {                 // if zero crossing occurs this sample
    OK = FALSE;                     // then flag defaults FALSE and window is checked
    for j=0 to size of STOP and START arrays        // loop over all valid-windows
        if  ((counter >= START[i])  & (counter <= STOP[i]))  // check if counter is in window
            then   OK = TRUE;                       // set flag TRUE if in window
    reset counter;                  // counter is reset at every zero crossing
    }
if (counter > max. value in STOP array)
    then    { OK = FALSE; reset counter; }    // window expired -> spacing is too large for MSK
    else    increment counter;
output[i] =  logic inverse of  OK;  // output pulses high for bad zero crossings
```

3.2.2.2 Carrier Detect Sliding Window Filter

The sliding window filter maintains a count of the number of bad zero crossings within the last L samples. It can be modeled as a finite impulse response filter of length L with all coefficients set to 1. See section 3.4 for pseudo-code of a finite impulse response filter. To mimic response time of MX-COM's serial MSK modems the length of the window should be eight symbols so L=N x 8 where N is the number of samples per symbol.

3.2.2.3 Carrier Detect Comparator

The carrier detect comparator monitors the output of the sliding window filter to determine if the zero crossing spacings are consistent enough to look like MSK. It has hysterisis to prevent chattering in noisy conditions. The comparator's upper threshold is set to five while its lower threshold is set to one. Thus, it trips to the detect state when there are no bad zero crossings in the last L samples and trips to the non-detect state when there are more than five. Section 3.4 shows pseudo-code for a comparator with hysterisis. That models output should be inverted to have an active high MSK carrier detect.

3.2.3 Bit Synchronization

The Bit Synchronization section re-times the data from the demodulation section. There are two blocks in this section, the Phase Lock Loop block (PLL) and the re-timing block. The PLL creates the clock that is used to re-time the data.

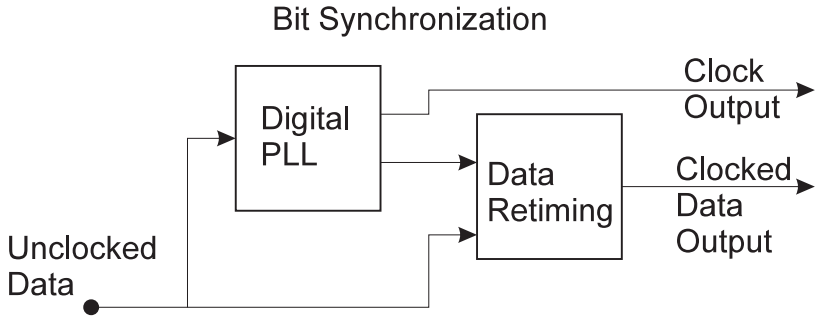

Figure 28: Bit Synchronization Block Diagram.

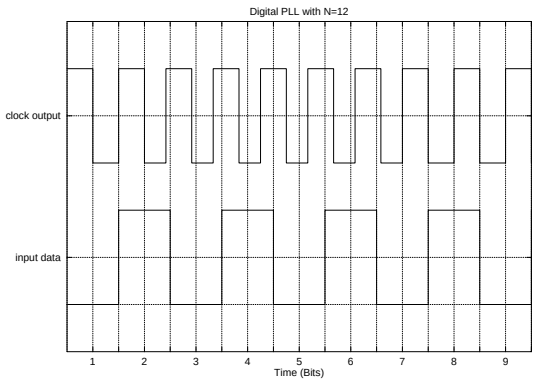

Figure 29: Bit Synchronization Plot.

3.2.3.1 Digital Phase Lock Loop

The PLL aligns the output clock rising edge to the middle of the input NRZ data bit. When the PLL is not aligned with the incoming data a correction factor is applied to the PLL clock period until it becomes synchronized with the incoming data. As seen in Figure 29 the output clock starts out of phase with the data and it becomes synchronized by the end of the plot. In Figure 29 the correction factor used is one 12^{th} of a bit period to more clearly show the locking process

Pseudo-Code Listing 6 Digital PLL

I/O, Variables and Constants

```
input[i]        // current input sample; input is NRZ data
output[i]       // current output sample; output is clock which locks to NRZ data changes
X               // the correction factor   (Note: for MX-COM MSK modems X = 5% bit time)
N               // number of samples per symbol
early           // flag set TRUE if data transition is early
late            // flag set TRUE is data transition is late
counter         // a general counter
reset           // flag set TRUE when algorithm determines it is time to reset counter
```

Pseudo Code for First Order Digital PLL

```
reset = 0;
if  counter<N/2-(0.5X) and input[0] not equal to input[1]        // set early flag
      then      early = 1;
if counter>N/2+(0.5X ) and input[0] not equal to input[1]        // set late flag
      then      late = 1;
if counter is equal to N - 1 - X and early and NOT late         // apply correction factor
      then      reset = 1;                                      //       by resetting early
if counter is N-1 and NOT early and NOT late                    // apply no correction
      then      reset = 1;                                      //       by resetting on time
if counter is N-1 and early and late                            // reset on time since flags conflict
      then      reset = 1;
if counter is equal to N - 1+ X                                 // apply correction factor
      then      reset = 1;                                      //       by resetting late
if counter > N/2                        // set output clock phase based on counter state
      then      output = 0;
      else      output = 1;
```

```
if reset is equal to 1  then {          // reset flag is true then reset counter and early and late flags
      counter = 0;
      late = 0;
      early = 0;
      }
      else  increment counter;
```

3.2.3.2 Data Re-timing

The date re-timing block aligns the data to the clock recovered by the PLL. The most common solution is a D flip-flop. The data is applied to the D input of the flip-flop and the PLL output clocks the data through the flip-flop. A model of a D flip-flop is shown in section 3.4.

3.3 System Simulations

Figure 30 shows an MSK system consisting of a transmitter, noisy channel, and a receiver. By simulating a complete system with the models defined above, various signals are plotted in Figure 31. The transmit data bits have a 16 bit preamble for bit synchronization. The receive filter output shows how noise degrades the band limited MSK signal. This degradation is more clearly seen in the receive filter eye diagrams, Figure 32 and Figure 33. The discriminator output clearly shows the need for the post discriminator low pass filter. The unclocked output exhibits jitter (i.e. variable length bit duration) and shows the need for sampling it in the middle of the bit to reliably recover it. The clocked data output shows relatively little jitter in the re-timed data. Additionally the carrier detect output and internal signals are shown to help the reader understand their function. While the transmitter is not enabled many bad zero crossings are indicated. These keep the sliding window filter output from going below the carrier detect threshold. After the start of transmission few bad zero crossings are seen and the carrier detect output trips high when the sliding window filter output goes below the threshold.

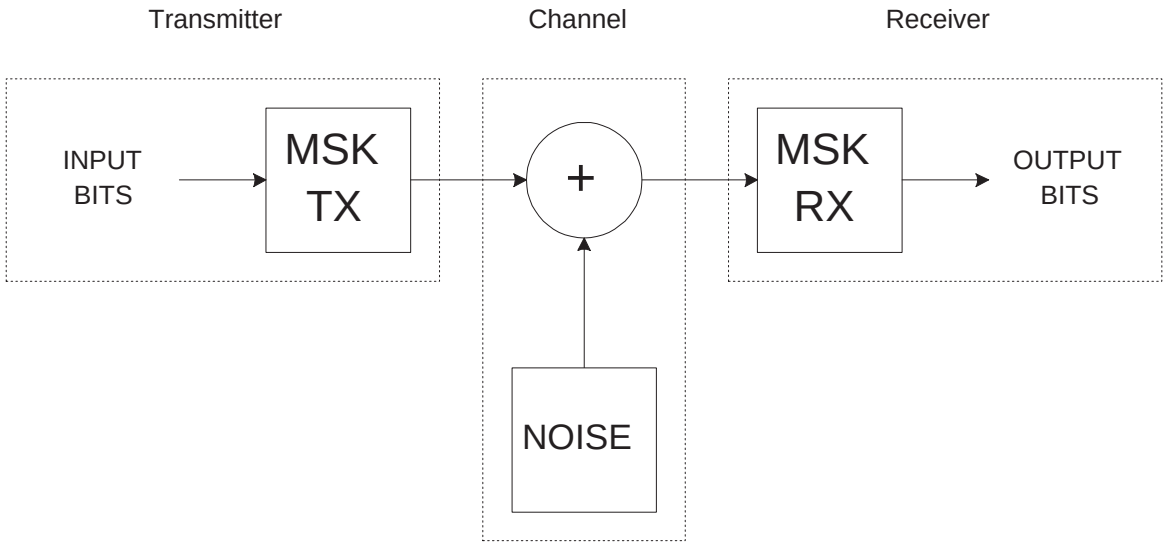

Figure 30: MSK Communications System with Noisy Channel.

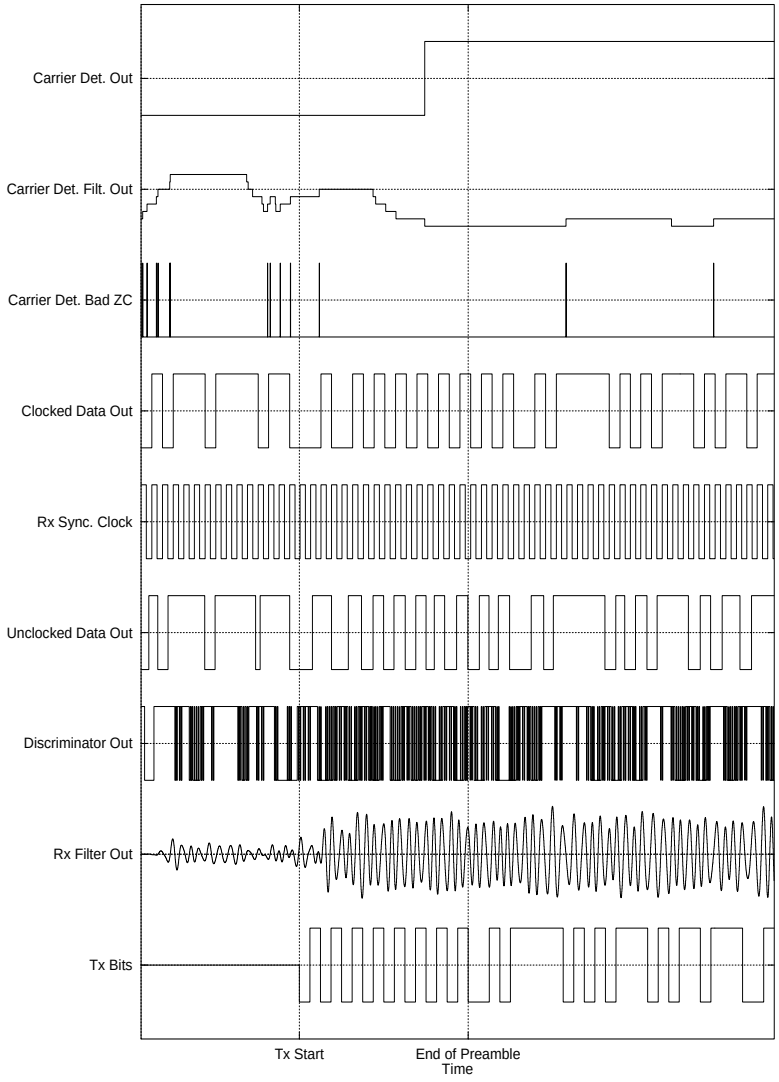

Figure 31: System Simulation Signals for 1200 baud at 12dB SNR

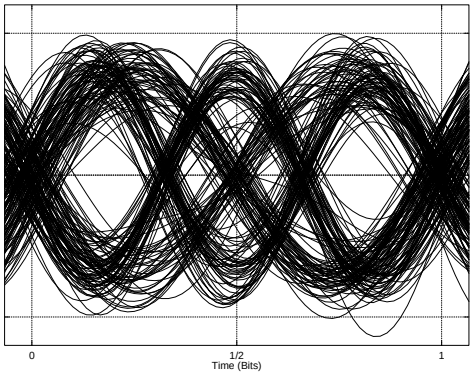

Figure 32: RX Eye Diagram for 1200 Baud at 12dB SNR.

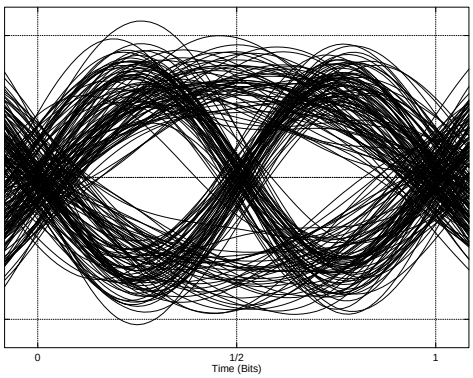

Figure 33: RX Eye Diagram for 2400 Baud at 12dB SNR.

3.3.1 MSK Simulations with various SNR

Figure 34 shows the simulated bit error rate performance of the system depicted in Figure 30. The channel is assumed to be additive white gaussian noise (AWGN). In the simulations the noise source produces white noise from DC to half the sampling rate. Thus the total noise power is much higher than the noise in the bit rate bandwidth. Equation (29) computes an effective SNR from a measured SNR assuming white noise over the measurement bandwidth.

$$\mathrm{SNR}_{\mathrm{Eff}} = \mathrm{SNR}_{\mathrm{Meas}} + 10\log\frac{\mathrm{BW}_{\mathrm{Meas}}}{\mathrm{BW}_{\mathrm{BitRate}}} \tag{29}$$

which can be rearranged as:

$$\mathrm{SNR}_{\mathrm{Meas}} = \mathrm{SNR}_{\mathrm{Eff}} - 10\log\frac{\mathrm{BW}_{\mathrm{Meas}}}{\mathrm{BW}_{\mathrm{BitRate}}} \tag{30}$$

To adjust the effective SNR to 12dB assuming:

AWGN source is a random variable with zero mean and variance = σ^2 = 1 i.e. dB power = 0dB.

MSK peak signal level = 1 i.e. signal power = 20log(RMS voltage level) = $20\log\left(\frac{1}{\sqrt{2}}\right)$ = -3dB

and a sampling rate 60 times the baud rate i.e. $\mathrm{BW}_{\mathrm{Meas}}$ = 30 times the baud rate

$$\text{Correction Factor} = 10\log\frac{\mathrm{BW}_{\mathrm{Meas}}}{\mathrm{BW}_{\mathrm{BitRate}}} = 10\log\frac{30}{1} = 14.77\text{dB}$$

thus $\mathrm{SNR}_{\mathrm{Meas}}$ = 12dB - 14.77dB = -2.77dB

The SNR without adjusting noise power level is:

$$\mathrm{SNR}_{\mathrm{Meas}} = 10\log\frac{\text{Signal Power}}{\text{Noise Power}} = \text{-3dB}$$

so it is necessary to decrease the noise power by 0.23 dB so:

$$\text{noise variance} = 10^{\frac{-0.23}{10}} = 0.9484$$

to arrive at other effective SNR we can simply adjust the noise variance by a scale factor:

$$\text{scale factor} = 10^{\frac{12-SNR}{10}}$$

The table below shows the required noise variance for SNR near 12dB with the system sampling rate equal to 60 x baud rate.

Effective SNR (dB)	Noise Variance	Scale factor from reference
13	0.753	0.794 = 1/1.259
12	0.948	1 i.e. 12dB is reference
11	1.194	1.259
10	1.503	1.585 = $(1.259)^2$

Table 3: Noise variance for various effective SNR.

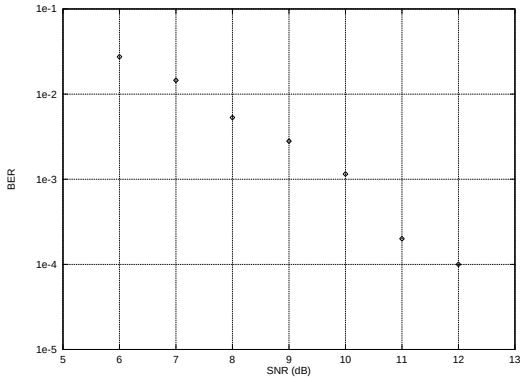

Figure 34: Simulated Bit Error Rate Performance.

3.3.2 FM System

Because of the relatively narrow bandwidth required by MSK, it can often be transmitted over existing analog radio links with little or no modification. Figure 35 is a block diagram of FM transmission of MSK. Pseudo-code models for the key blocks of this system are shown below.

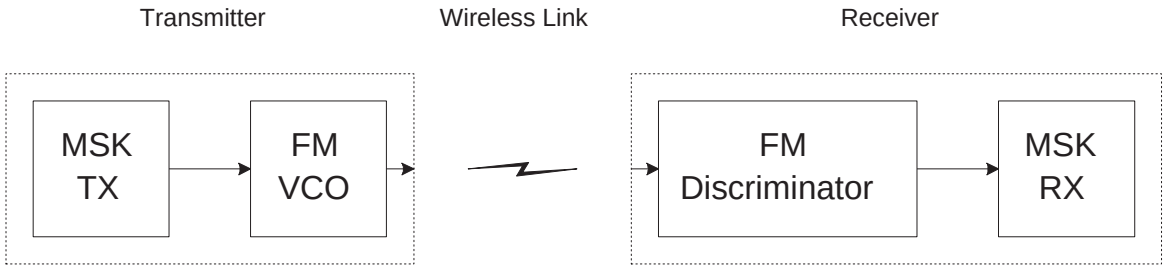

Figure 35: FM Block Diagram.

3.3.2.1 FM modulation model

Pseudo-Code Listing 7 VCO based FM modulator model

I/O, Variables and Constants

input[ ] // input sequence

output[ ] // output sequence

IntIn = 0 // variable used to store integral of input[] sequence up to current time.

step // difference between consecutive input samples normalized to interpolation rate M.

carrier // FM carrier frequency normalized to interpolation rate M

deviation // FM deviation frequency normalized to carrier frequency.

$\omega_c = \frac{2\pi \cdot \text{carrier}}{\text{M}}$ // carrier frequency constant

$\omega_d = \frac{2\pi \cdot \text{carrier} \cdot \text{deviation}}{\text{M}}$ // deviation frequency constant

---

PSEUDO-CODE // run for each sample, output sample rate is
// M times input sample rate
// use linear interpolation between input samples

$\text{step} = \frac{\text{input[i]} - \text{input[i - 1]}}{\text{M}}$;

for j=0 to M-1 {

IntIn = IntIn + input[i-1] + $\text{step} \cdot \text{j}$; // simple integrator

output[M · i + j] = $\sin(\omega_c \cdot (\text{i} \cdot \text{M} + \text{j}) + \omega_d \cdot \text{IntIn})$;

}

3.3.2.2 FM Discriminator Model

A simple heterodyne discriminator model based on Figure 35 can be built with multipliers, delays and filters. Pseudo-code is shown for the mixer and the discriminator.

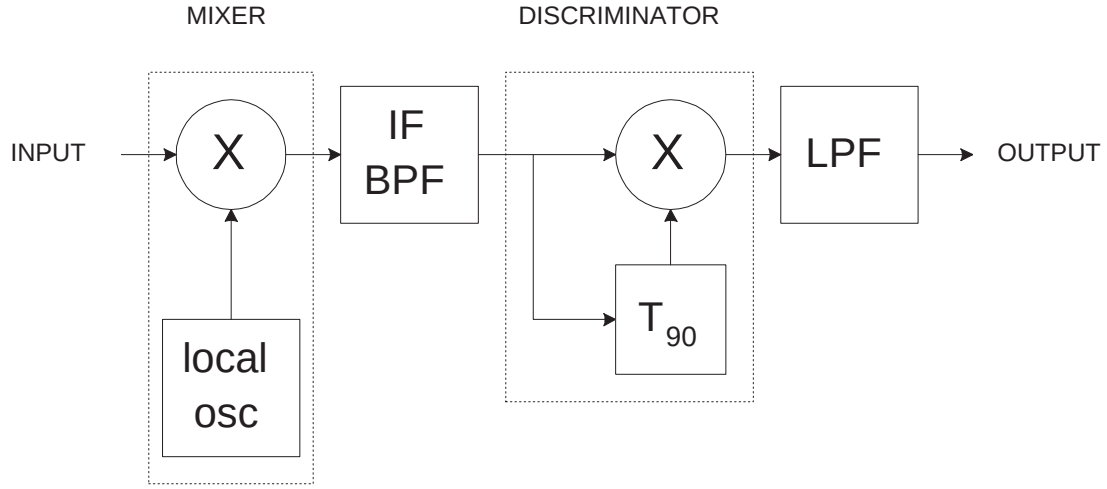

Figure 36: Heterodyne Discriminator Block Diagram.

Pseudo-Code Listing 8 Mixer

I/O, VARIABLES and CONSTANTS

RFinput[i] // RF input
IFoutput[i] // IF output
// local oscillator frequency normalized to sampling rate of block
$\omega_c = \frac{2\pi \cdot f_c}{f_s}$ // note f_c is offset from the RF carrier by the IF frequency.

PSEUDO-CODE for MIXER

IFoutput[i] = RFinput[i] $\times$ sin($\omega_c \times$ i);

Pseudo-Code Listing 9 FM Discriminator

I/O, VARIABLES and CONSTANTS

IFinput[i] // input to discriminator
DSoutput[i] // discriminator output
T_{90} // Delay equal to one quarter of the IF center frequency

PSEUDO-CODE for DISCRIMINATOR

DSoutput[i] = IFinput[i] $\times$ IFinput[i - T_{90}];

3.4 General Purpose Models

This section list various general purpose models which the authors hope are self explanatory.

Pseudo-Code Listing 10 Finite Impulse Response Filter Model

I/O, VARIABLES and CONSTANTS

input[i] // current input sample
output[i] // current output sample
L // FIR filter order
j // index to previous samples
sum // used to accumulate tap x weight products
weights[0..L-1] // array of filter coefficients

PSEUDO-CODE // run for each sample, input and output sample rates are equal

sum = 0

```
j = 0
while j<L {
    sum = sum + (input[i-j] x weights[j]);
    increment j;
    }
output[i] = sum;
```

Pseudo-Code Listing 11 Comparator with Hysterisis

I/O, VARIABLES and CONSTANTS

```
input[i]        // current input sample
output[i]       // current output sample
Hth             // upper threshold
Lth             // lower threshold
```

PSEUDO-CODE // run for each sample, input and output sample rates are equal

```
if (input[i] > Hth)                         // output trips high
    then output[i] = 1
    else if (input[i] < Lth)                // output trips low
            then output[i] = 0
            else output[i] = output[i-1]// output stays same as last output
```

Pseudo-Code Listing 12 D type flip-flop

I/O, Variables, and Constants

```
D[i]        // current input sample
C[i]        //  clock input
Q[i]        // current output sample
state       // used to store sampled
```

Pseudo-Code

```
if (C[i] = 1) & (C[i-1] = 0)        // rising edge of clock input
    then state = D[i];              // sample input
Q[i] = state;                       // output state
```

3.5 References

[1] J.B. Anderson, T. Aulin and Carl-Erik Sundberg, *Digital Phase Modulation*, Plenum Press, New York, NY, 1986.

[2] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*, Prentice Hall, Inc., Englewood Cliffs, NJ, 1983.

[3] M.L. Doelz and E.H. Heald, "Minimum Shift Data Communications," US Patent No. 2,977,417, March 28, 1961. Assigned to the Collins Radio Company.

[4] W. D. Greeg, *Analog and Digital Communication*, John Wiley and Sons, Inc. New York, NY, 1977.

[5] R. A. Haddad and T. W. Parsons, *Digital Signal Processing - Theory, Applications, and Hardware,* W. H. Freeman and Company, 41 Madison Avenue, New York, NY, 1991.

[6] Kostedt, F. and Kemerling, J., "Practical GMSK Data Transmission," Wireless Design and Development, vol. 3, No. 1, pp. 21-25, January 1995.

[7] E. A. Lee and D.G. Messerschmitt, *Digital Communications*, Kluwer Academic Publishers, Boston, MA, 1988.

[8] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice Hall, Inc., Englewood Cliffs, NJ, 1975.

[9] J.G. Proakis, *Digital Communications*, 3rd Ed., McGraw-Hill, Inc., New York, NY 1995.

[10] T. S. Rappaport, *Wireless Communications - Principles and Practice*, ISBN 0-13-375536-3, Prentice Hall PTR, Upper Saddle River, NJ, 1996

[11] Sklar, B., *Digital Communications: Fundametals and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

