Wireless Digital Communications: Design and Theory

Tom McDermott, N5EG

TAPR™

Includes software disk

Published by:
Tucson Amateur Packet Radio Corporation

Wireless Digital Communications: Design and Theory

By: Tom McDermott, N5EG
n5eg@tapr.org

Published by: Tucson Amateur Packet Radio Corporation

ISBN: 0-9644707-2-1

First Edition
First Printing, 1996
Second Printing, 1998

TAPR Production Editor: Greg Jones, WD5IVD
Proof Reader: Laura Amill

Trademark Notices

Excel is a trademark of Microsoft Corporation
Windows is a trademark of Microsoft Corporation

TAPR™

Tucson Amateur Packet Radio
8987-309 E. Tanque Verde Rd #337
Tucson, Arizona • 85749-9399

Office: (940) 383-0000 • Fax: (940) 566-2544
Internet: tapr@tapr.org • www.tapr.org
Non-Profit Research and Development Corporation

Contents

TAPR on the Internet

World-Wide-Web!

TAPR's Home Page can be reached at: **http://www.tapr.org**

FTP

The TAPR Software Library is available via anonymous FTP. You can access the library by ftp to '`ftp.tapr.org`'. Login as 'anonymous', with a password of 'your_account@internet_address'.

TAPR Special Interest Groups

TAPR maintains several SIGs in various areas of interest (Spread Spectrum, DSP, HF Digital, Networking, and more). Full details on SIGs can be found at **http://www.tapr.org/tapr/html/sigs.html** along with a web interface for subscribing to mail lists.

You can also send an e-mail message to **`listserv@tapr.org`** with the subject line "`Request`" and the following text in the body of the message:

`lists`	(for list of mail groups on TAPR.ORG)
`information listname`	(where listname is the name of the mail group)

When you are ready to subscribe, send e-mail to **`listserv@tapr.org`** with the following command in the message text:

```
subscribe list YourFirstName YourLastName Callsign
```

http://www.tapr.org/tapr/html/sigs.html

Membership in TAPR

TAPR was founded in 1982 as a national organization with interests in the areas of packet and digital communications. Today, TAPR continues as a membership supported non-profit amateur research and development organization. TAPR continues to develop kits for the amateur radio community and is working actively on publications and communications standards. Membership in TAPR includes a subscription to the quarterly published *Packet Status Register* newsletter. *PSR* has been published since 1982 and is recognized as an authoritative source for up-to-date user and technical information on packet radio. Much of the material in *PSR* is timeless. Back issues may be obtained from the TAPR office in four year volumes. Membership in TAPR is $20 per year for the US and possessions, Canada, and Mexico. Outside North America is $25 per year.

Foreword

Digital transmissions are rapidly supplanting analog transmissions in both amateur and commercial radio communications. Digital radio transmissions offer many inherent advantages. Digital techniques allow virtually noise-free transmission of data, messages, control commands, sounds and images to be achieved. Digital transmissions allow information to flow from multiple sources to multiple destinations on the same radio channel in an organized and efficient way. Information is communicated in a form that is easy to store and access. And when needed, information security can be readily added to a digital transmission.

Amateur Radio can be proud of its pioneering contributions to digital radio communications techniques, and especially the Tucson Amateur Packet Radio Corporation (TAPR), which developed the first widely used terminal node controllers for amateur packet radio.

I believe one of the most important contributions TAPR has made to digital radio communications recently is the publishing of this book. Tom McDermott, who has an extensive professional background in digital telecommunications, covers the topic of digital radio transmissions in a remarkably lucid and accessible way. Tom has held the complex mathematics typically found in most books on this subject to a minimum. Instead, Tom uses an easy to understand graphical approach to explain digital communications concepts. Tom has also included a collection of Excel™ spreadsheets with this book that allows the reader to explore a number of digital communications principles.

Amateur radio enthusiasts, engineering students and radio communications professionals will find this book both an excellent introduction to the digital radio communications and a useful day-to-day reference.

Frank H. Perkins, Jr. WB5IPM
TAPR/AMSAT DSP-93 Development Team Member

Excel™ is a trademark of Microsoft Corporation.

Preface

Amateur radio communication has progressed in many ways since it's beginnings in the early 1900's. General communications progressing from spark to CW, and voice from AM to FM and SSB. Similarly, data communications as a mode of amateur communications has progressed from it's beginnings using on-off keying (OOK) to FSK, and from RTTY to more modern modes of communications (synchronous and error-correcting). There has been a lack of good technical background material in the amateur radio literature on the principles and design of synchronous digital modems.

The wealth and quality of literature in the professional world in the subject area is astounding. However much of it may not be readily accessible to the radio amateur, whether for reasons of advanced mathematics, or simple lack of availability.

In writing this book, the aim has been to bring a concise group of topics covering a broad spectrum of amateur synchronous digital communications subjects to print in one place, and to make it readily accessible to the radio amateur. This text aims to present the information in a clear and straight-forward manner, with the maximum use of graphical and computer-assisted aids, and with a minimum of rigorous mathematical theory. However, digital communications deals with the application and solution of statistical phenomenon, and a minimum background is necessary. Where practical, the appendices provide short summaries of some of the important mathematical concepts that will be needed in understanding certain areas.

Overall, the field of digital communications could be generally broken down into two categories: bandwidth-limited communications, and power-limited communications. Much of the more recent professional literature on wireline modems focuses on the former, while in practice the amateur is many times more concerned with the latter. This text focuses more on the subject of power-limited communications, and emphasizes through examples the circuits and problems of the latter category of applications.

With time and the increasingly more crowded HF bands, however, the radio amateur will adopt more sophisticated data modems, offering higher throughput and narrower bandwidth operation under the demanding propagation conditions of the HF medium. This trend has already started, and should accelerate as the cost of technology, particularly Digital Signal Processing, continues to decrease. So, this text includes information on the subject areas of DSP-based modem filters, and forward-error-correcting codes, whose use by the radio amateur will become dominant within a few short years. While the data rate of VHF and UHF communications will increase, it is expected that for the radio amateur these will remain power-limited applications for some time. When interference-limited communications pose a significant problem, spread-spectrum techniques may prove valuable in reducing channel co-residency issues. These techniques are not covered in this book, however.

In the preparation of this text, I have relied on the study of a number of exceptionally well written textbooks, and on the IEEE literature in the area. These resources should be consulted whenever more depth or broader interest is desired. I would like to thank the reviewers of the text for many helpful comments, related both to the readability of the material and the inclusion of additional material of interest. Especially, I would like to thank Frank Perkins, WB5IPM, for his thorough and careful reading of the text, and his many useful comments, to Johan Forrer, KC7WW, for his broad and clear view of the material, his suggestions on additional interesting topics, and his excellent references to the literature on those topics, and to Phil Karn, KA9Q, for his helpful comments and suggestions.

Finally, I would like to thank Greg Jones, WD5IVD, who originally suggested that I write this book, and who then made arrangements for Tucson Amateur Packet Radio Corporation (TAPR) to publish the material, assuring it's availability to many radio amateurs.

Thomas C. McDermott, N5EG

Dallas, Texas
August 25, 1995

Chapter 1 Introduction

This section briefly introduces the concepts of data communications and modem design for the radio amateur. The design and use of data modems for amateur radio usage differ somewhat from those found for microwave radio line-of-sight, and for telephone modem usage, due mainly to different conditions encountered in the amateur radio service.

Some of the basic assumptions for amateur radio usage of data modems are:

1. The amateur operator is frequently more concerned with unscheduled or uncoordinated communication. This means that the channel may not necessarily be assigned full-time to the particular transmit/receive pair.

2. Most of the operation will occur in an area of the radio spectrum where noise and interference can be significant concerns. The signal to noise ratio (SNR) and the signal to interference ratio can not be assumed to be good.

3. Many times, the bandwidth of the emission is not absolutely important, or is not constrained as much as with commercial equipment. So, somewhat less bandwidth-efficient modulation systems can be utilized, with the advantage of good operational characteristics in poorer SNR conditions. Additionally, these simple modulation formats are forgiving as to frequency offset errors between the transmitter and receiver, and are forgiving of some frequency and phase response errors on the channel. Thus, the modems may be relatively inexpensive. These types of modems are typically used on VHF/UHF transmission systems.

4. Some other amateur usage may require sophisticated techniques, such as HF skywave propagation, where bandwidth is a concern and the channel is difficult and time-varying. More expensive modems are required to achieve acceptable throughput within FCC-specified channel widths. However, the SNR is still usually relatively poor, so adaptive modems show great promise.

Common modems designed for telephone operation, for example, assume that a high SNR exists, but that the bandwidth is fixed and narrow. Therefore, telephone modems optimize throughput at the expense of requiring high SNR. This is done via multi-level modulation methods which preserve bandwidth. Terrestrial microwave communications systems enjoy relatively good SNR, with some occasional fades, but are subject to the restriction of high channel efficiency requirements. Thus, these system must use narrow bandwidth channels (compared to the huge amount of data carried) which are imposed by the FCC since the channel is fully dedicated to the transmit/receive pair within a geographical area.

On the other hand, amateur modems frequently are not as restricted in terms of bandwidth, but usually are operated under poor SNR conditions. Therefore, amateur radio modems may have fundamentally different design configurations and requirements than other more commonly known modems. In this text, we will concentrate mostly on modems and modulation methods that are generally more suited to communications where the SNR is poor, but where the channel bandwidth is sufficiently wide.

Definition of Information

Assume that at any instant one of a set of messages may be transmitted. Then the amount of information transmitted by a particular message is related to the probability of that particular message. The probability of occurrence of the sum of all the messages is, of course, unity. The amount of information is then:

$$I = \log_2 \frac{1}{p_m} \qquad (1\text{-}1)$$

where p_m is the probability of one particular message, out of m messages. Let's assume that there are only two messages, say a binary one and a binary zero, and each has equal probability (of 1/2), then the amount of information conveyed by each symbol is:

$$I = \log_2 \frac{1}{0.5} = \log_2 2 = 1\ bit \qquad (1\text{-}2)$$

Assume that there are 4 possible messages, and that each has an equal probability of being sent, 1/4. Then the amount of information in any one message is:

$$I = \log_2 \frac{1}{0.25} = \log_2 4 = 2\ bits$$

That is, each message conveys 2 bits of information. If two messages are sent, one after the other, then the total information is the sum of the information in each message. So, for example, if 6 messages each individually containing 2 bits of information are sent, then the total information is 6*2 = 12 bits.

Source Entropy

If we divide the total information sent by an information source by summing the information for all possible messages, *m*, and we weight each message with its probability of being sent, then we define the entropy of the source as H:

$$H = p_1 I_1 + p_2 I_2 + \ldots + p_m I_m \qquad (1\text{-}3)$$

Suppose that the source could emit one of 4 messages, but they are not all equally probable. For example, message 1 has a probability of 0.5, message 2 has a probability of 0.25, message 3 has a probability of 0.15, and message 4 has a probability of 0.1. The total probability is (0.5+0.25+0.15+0.1) = 1.0 which is required in the definition. What is the entropy of this source? The information in each message is:

$$I_1 = \log_2 \frac{1}{0.5} = 1 \textit{ bit}$$

$$I_2 = \log_2 \frac{1}{0.25} = 2 \textit{ bits}$$

$$I_3 = \log_2 \frac{1}{0.15} = 2.74 \textit{ bits}$$

$$I_4 = \log_2 \frac{1}{0.1} = 3.32 \textit{ bits}$$

So, the entropy of this source is the sum of data in each message times each message's probability of occurrence:

$$1 \bullet (0.5) + 2 \bullet (0.25) + 2.74 \bullet (0.15) + 3.32 \bullet (0.1) = 1.743 \textit{ bits / message}$$

It can be shown that the entropy of the source is maximized when each of the events is equally probable. In the above case, if each of the 4 messages has the same probability of occurrence, 0.25, then the entropy of the source would be:

$$2 \bullet (0.25) + 2 \bullet (0.25) + 2 \bullet (0.25) + 2 \bullet (0.25) = 2.0 \, bits \ / \, message$$

In another example, if the source could send only one message, then p would be 1.0 (it's the only message, so it's probability of being sent is unity). The information in the message is log(1) = 0 bits, so the entropy would be 1 * 0 = 0 bits/message. A communications system is not necessary if we know beforehand that the transmitter can continuously send only one message!

If messages do not have an equal probability to be sent, then the efficiency of the transmitter can be enhanced by coding the data into a smaller set of equally-probable messages.

Shannon's theorem shows that if the source of equally-likely messages, M (with M being a large number) generates information at a rate, R, and given that we have a channel of capacity C, then there must exist a way to code the messages so that if R is less than or equal to C, then the probability of an error at the receiver can be made arbitrarily small (i.e. approaches zero). Conversely, if the rate exceeds the capacity of the channel (R > C), then the probability of error at the receiver for every possible message approaches 1. More advanced discussions on these topics can be found in Taub and Schilling (1971), Proakis (1983), and Viterbi and Omura (1979).

Fundamental Data Capacity

In his pioneering papers on the fundamentals of data communications, Shannon showed that the error-free capacity of a white Gaussian noise band-limited communication channel cannot exceed a certain bound (Shannon, 1948; Shannon, 1949; Slepian, 1973). This bound is known as the Shannon-Hartley theorem.

$$C = B\log_2\left(1 + \frac{S}{N}\right) \tag{1-4}$$

This capacity limit assumes Additive White Gaussian Noise (AWGN is discussed in a later section) and no interference, fading, etc., where C is the channel capacity in bits/second, B is the bandwidth of the channel in Hertz, S is the signal power in Watts, and N is the noise power in Watts. As shown in the formula, doubling the bandwidth of a channel potentially doubles the capacity, assuming that the signal power is increased in order to compensate for the increase in noise power that coincides with the increase of channel bandwidth. As can also be seen, if the S/N ratio is very large, (lots of transmit power, little noise) then the capacity of the channel is also large. Shannon's limit does not specify how to build a system to meet the capacity limit, only that no system can exceed the limit. All practical modems are worse than the capacity bounds in (4). The key conclusion from Shannon's theorem is: *noise does not inherently limit the error rate of the channel, it only limits the capacity of the channel.*

Signal to Noise ratio - Eb/No

A common terminology for describing the signal-to-noise ratio of a digital system is to specify the energy of a single data bit to the noise power density. These two quantities are Eb, expressed as the energy per bit in joules (watt-seconds), and No, the noise spectral density, expressed as watts/Hertz. Then noise within a given bandwidth is N, equal to the product of the density, No, and the bandwidth, B. Thus the units of noise, N, are watts. Therefore, Eb/No is a dimensionless quantity, and it represents the signal-to-noise ratio per bit. However, it is commonly expressed, relative to unity, in decibels. Thus, when the ratio Eb = No is one, and the quantity is expressed as Eb/No = zero dB. Shannon's theorem can then be re-expressed in terms of Eb/No, and in terms of the channel capacity, C/B, which is expressed in bits/second per Hz of bandwidth:

$$\frac{C}{B} = \log_2\left(1 + \frac{E_b}{N_0}\frac{C}{B}\right) \qquad (1\text{-}5)$$

This can be solved for the required signal to noise per bit, Eb/No, versus the channel capacity. Rearranging and solving for Eb/No results in:

$$\frac{E_b}{N_0} = \frac{2^{\frac{C}{B}} - 1}{\frac{C}{B}} \qquad (1\text{-}6)$$

This is plotted as figure 1-1.

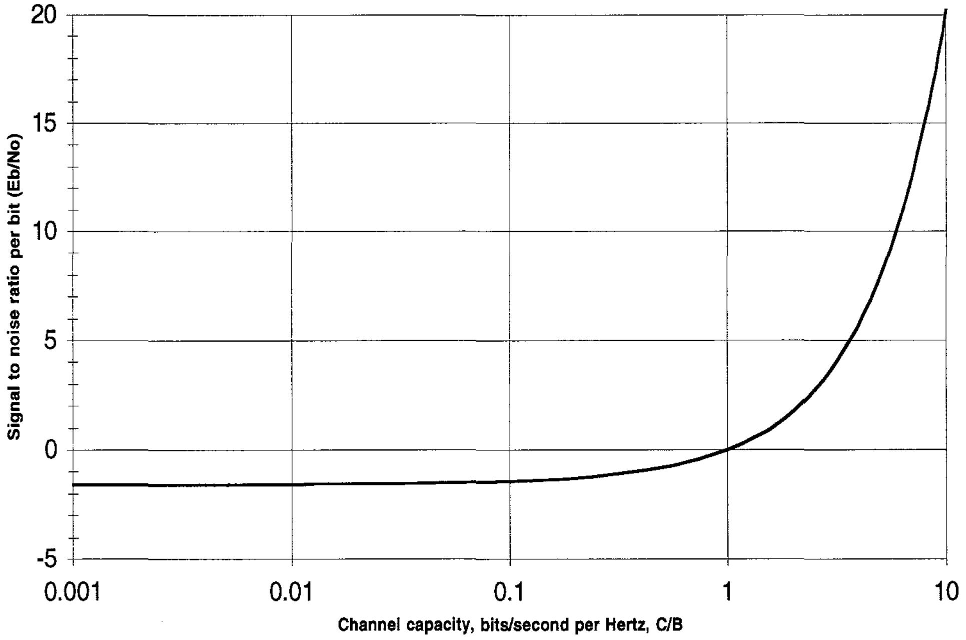


Figure 1-1 *- Required signal-to-noise ratio for a given channel capacity based on Shannon's limit.*

In examining figure 1-1, note that when the signal to noise ratio per bit reaches -1.6 dB., the capacity of the channel drops to zero. This is a fundamental limit in data communications. At a signal-to-noise ratio per bit of zero (the signal and the noise are equal) the maximum capacity of the channel is one bit per second per hertz of bandwidth. However, most uncoded modems, are on the order of 10 dB worse than this theoretical limit. All systems must lie above and to the left of the line.

Figure 1-1 indicates that it is not possible to build a modem that operates below or to the right of the curve and has an arbitrarily low error rate — even with error correction, retries, fancy protocols, or other means we have yet imagined. If the point of operation is above or to the left of the curve, it is theoretically possible to build a modem, which might require some extremely clever means (such as forward error correction). Shannon does not tell us how to construct such a device, only that it is theoretically possible to do so.

In real systems, the operation above and to the left of the curve does not guarantee that any particular modem implementation will be error free. In future sections we will see that modems without forward error correction will certainly have errors, even well above and to the left of the above curve. The forward-error-correction only allows our modem design to come down closer to the theoretical curve for error-free operation.

The ambition of modem designers, then, is to devise schemes that achieve high capacity and zero error rate while approaching as close to the curve (from above) as is possible. We shall not be so ambitious in this book. The remainder of this book will examine techniques in modulation, coding, and circuitry to permit design of modems suited to amateur radio applications.

Block Diagram of the Communication System

A system for communicating information over a radio channel can be diagrammed, with each of the portions reflecting a particular problem or area of interest. These areas are addressed in subsequent chapters of the book. Figure 1-2 is a block diagram of the transmission process. The data to be transmitted may optionally be coded with forward-error-correction symbols. These symbols permit the decoding of a signal with some errors, and may enhance the throughput under some circumstances. Next, the data may optionally be coded for transmission over the channel, perhaps to have desirable characteristics, such as inversion insensitivity or good DC-balance. The symbols then may be filtered, either at baseband (prior to modulation) or at RF (after modulation). The modulation moves the baseband signal up to a radio frequency channel of interest, and also determines many of the characteristics of the emitted signals.

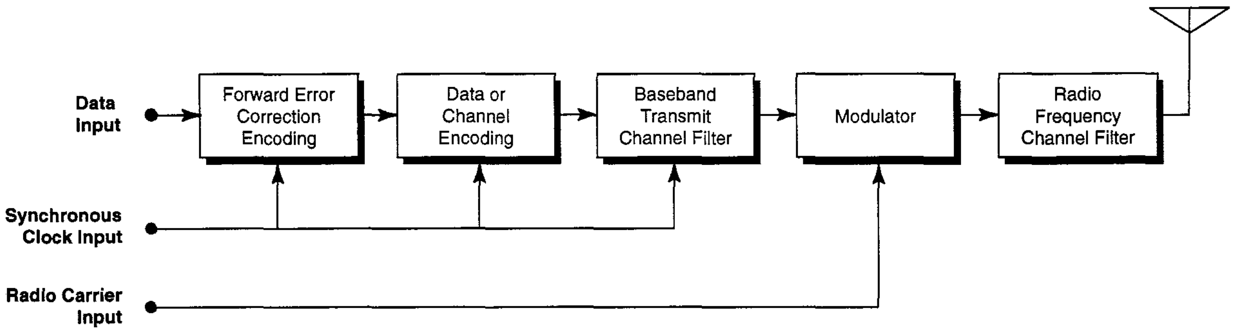

Data Input
Forward Error Correction Encoding
Data or Channel Encoding
Baseband Transmit Channel Filter
Modulator
Radio Frequency Channel Filter
Synchronous Clock Input
Radio Carrier Input

Figure 1-2 *- Block diagram of the data transmission process.*

The radio channel itself is an important part of the overall transmission system. Effects such as multipath, fading, and doppler shift are important to the characteristics of the modems, the modulation methods, and error control coding.

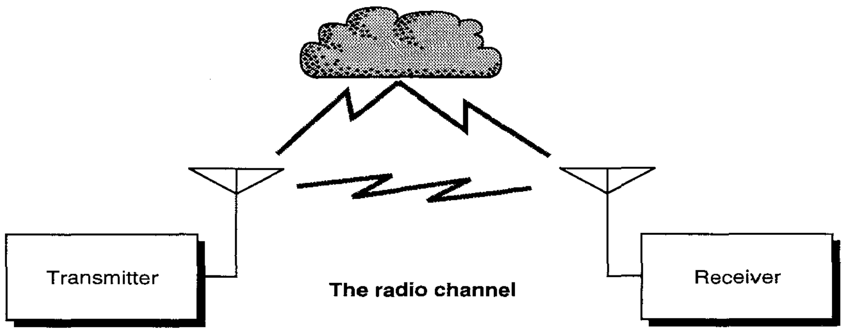


Figure 1-3 - *The radio transmission channel. Distortions due to the channel itself are significant.*

The reception process is shown in block diagram form in figures 1-4 and 1-5. The receiver is typically more complicated than the transmitter, and many issues that arise in the design of modems have to do with optimization and tradeoffs in the receiver.

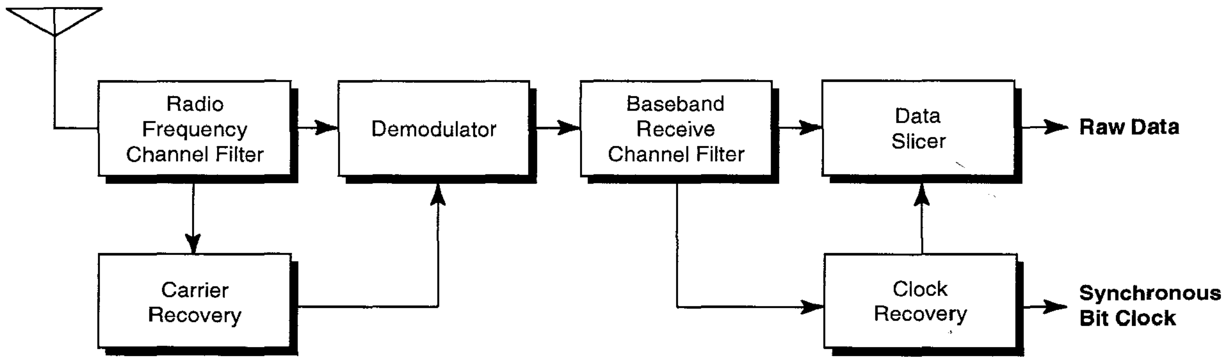


Figure 1-4 - *Block diagram of the reception process from RF input to sliced baseband data output.*

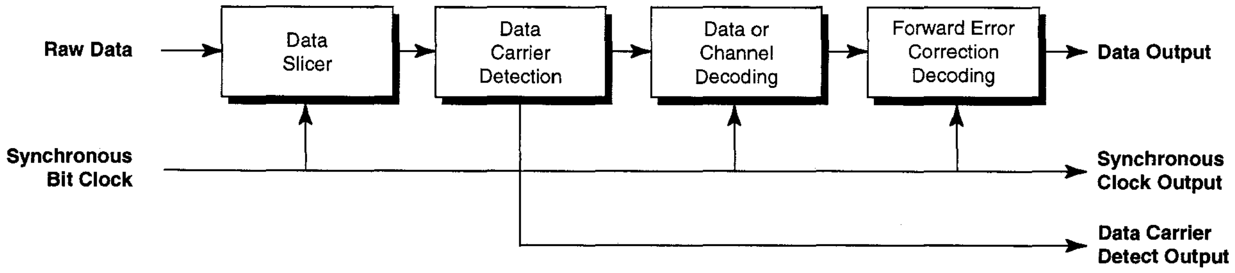


Figure 1-5 - *Block diagram of the reception process from sliced baseband data input to digital data, clock, and data carrier detect output.*

Review / Definition of Some Key Concepts

A few key concepts recur throughout the remainder of this book. This section provides a brief definition of some key terms, and reviews a few key concepts.

Digital Signal – A digital signal has one of a finite number of amplitudes, and has a periodicity measured as a fraction of the constant sampling interval. For example, a 4-level 1.000 kilosymbol per second signal is one that can have 4 different amplitude levels which can have a new value each 1 millisecond. A binary signal can have only two levels at each instant in time. In general, this text will refer to a digital signal as a signal that is quantized to certain levels and that is sampled at uniformly-spaced discrete points in time.

Digital Logic Signal – A digital logic signal traditionally has the values of either zero or one. Most of the time a binary signal will have the two distinct values -1 and +1. Sometimes these two levels will correspond to the zero and the one digital logic signal, but not always (depending on how the signal is coded). A 4-level signal would usually have the values -3, -1, +1, and +3.

Frequency Spectrum of a Digital Signal – The frequency spectrum of a digital signal of a square wave contains only the odd-harmonics plus the fundamental frequency of the square wave. If this is extended to the digital signal, and it is assumed that the digital signal is an alternating one/zero pattern, then the spectrum is that of a square wave. If the digital pattern is a pseudo-random data signal, then the spectrum of that signal is represented by a sin(x)/(x) roll-off, sometimes also called a sinc(x) roll-off, where x is the frequency in radians. Thus:

$$x = \pi B \tag{1-7}$$

where B is the bit rate in bits per second.

Figure 1-6 illustrates the time-domain and frequency-domain properties of the square wave, alternating one/zero data pattern, and a pseudo-random data pattern. Negative amplitude (in the case of the pseudo-random data stream) means a positive magnitude but a phase reversal compared to the fundamental signal.

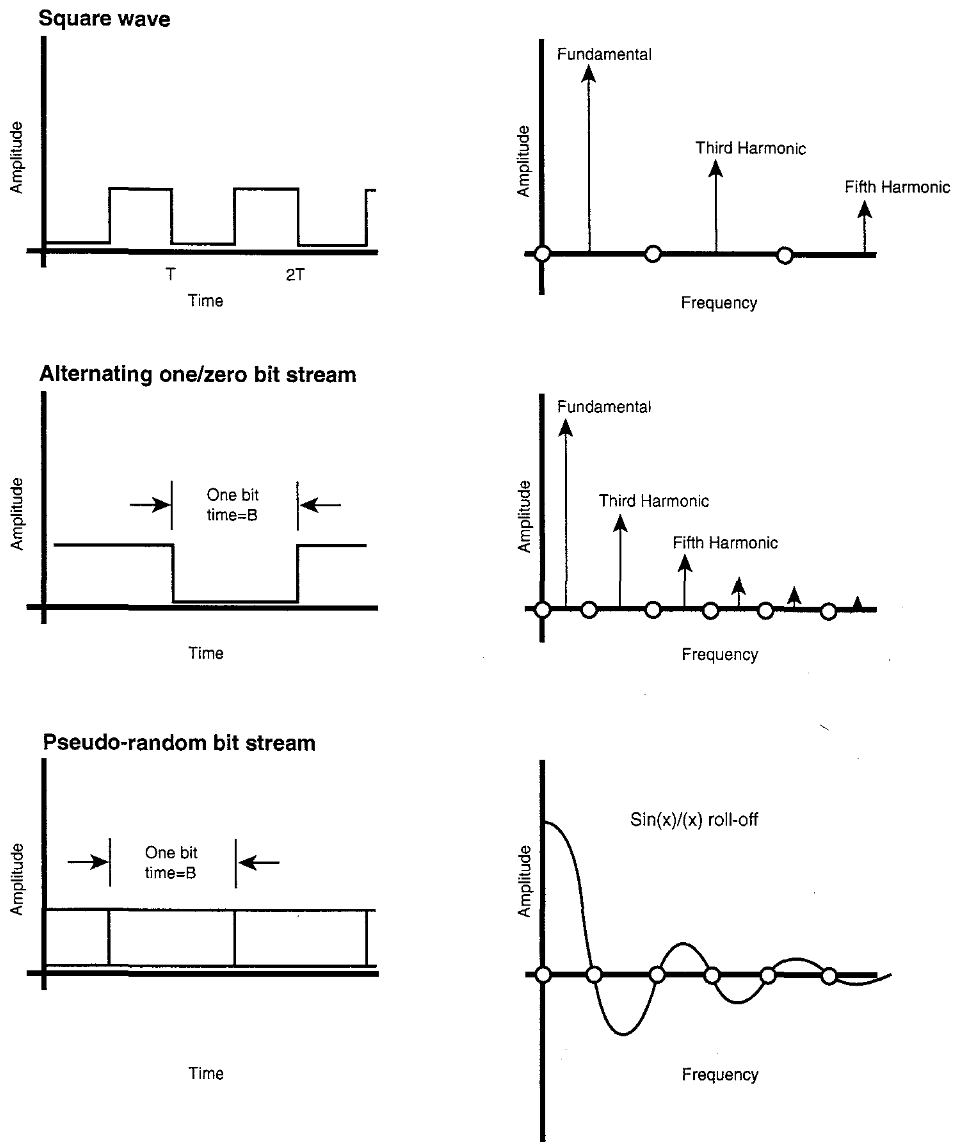


***Figure 1-6** - Time and frequency representations of square wave, alternating one-zero data pattern, and pseudo-random data pattern.*

Interestingly, note that the time response of a brick-wall filter (square, flat passband with an infinitely steep roll-off at 1/2 f) is also sin(x)/(x), but in time rather than frequency.

Eye Pattern

One important concept is that of an *eye pattern.* The eye pattern can be visualized as super-imposing a large number of consecutive oscilloscope traces of a signal on top of one another. Figure 1-7 shows the process of overlaying the traces, which are exactly time-aligned at a multiple of one data bit time. This can be accomplished by triggering an oscilloscope on a stable clock signal related to the data signal, but without triggering the oscilloscope on any part of the data signal itself (i.e.: the clock is connected to an external trigger input of the scope, and the data signal is connected to a vertical display input of the scope).

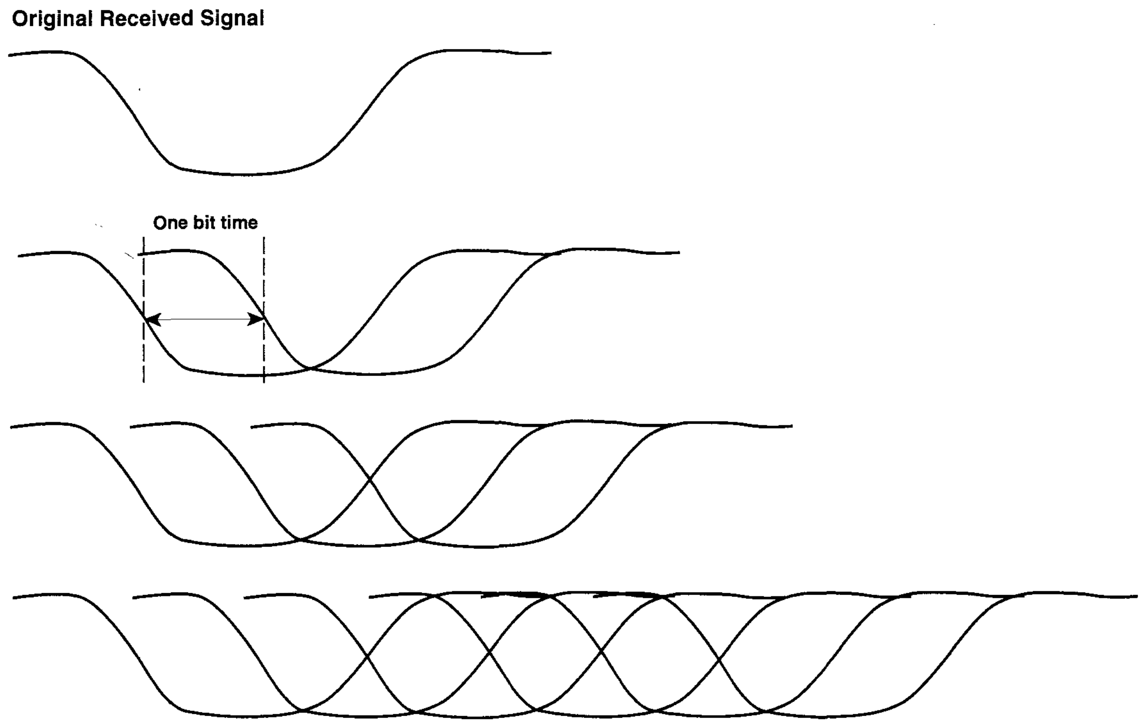


Figure 1-7 *- Generation of an eye pattern by super-imposing a signal on top of itself many times, but each one delayed by exactly one bit time.*

Signal Constellation Diagram

A *signal constellation* results when the above eye pattern is sampled once per bit time with a sample-and-hold circuit. If the sampling point is chosen correctly, (i.e. at a time when the eye is always maximum or when it is always minimum) then the constellation shows the signal at the time of maximum opening, perhaps corrupted by a little noise. Figure 1-8 shows a constellation diagram for the eye pattern in figure 1-7. If the eye pattern contains some noise, then the two points on the constellation diagram will become 'fuzzy'. If the noise becomes large enough, then a point may occasionally cross the dashed dividing line (halfway separating a one from a zero), and a bit error occurs.

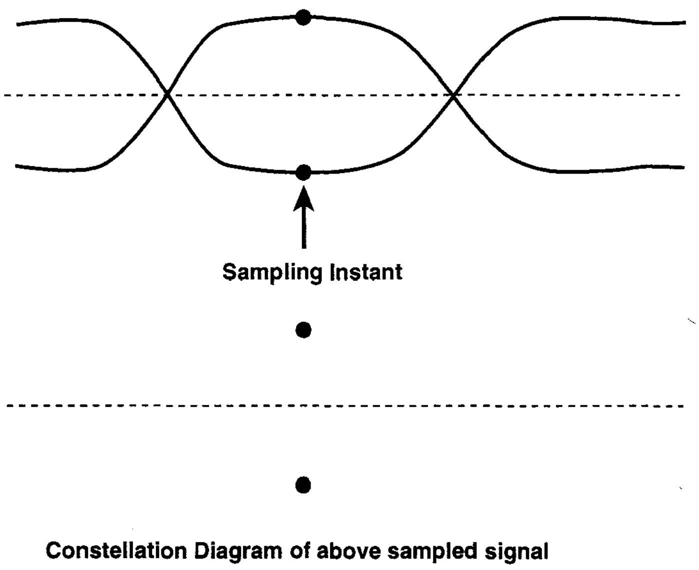
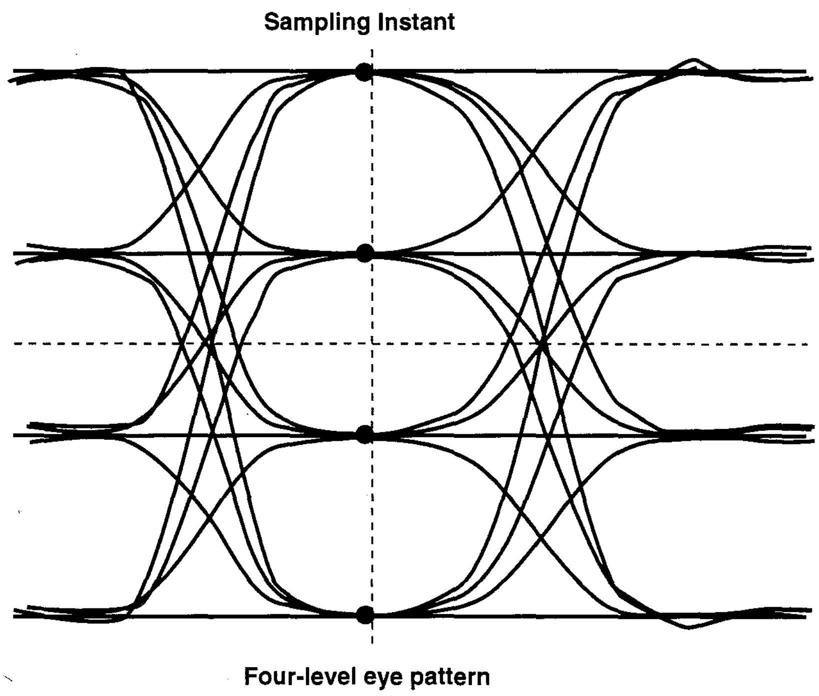


Figure 1-8 *- Constellation diagram. The eye pattern is sampled at the decision time, and those points displayed. If the eye pattern has noise, then the two points on the constellation diagram will become fuzzy.*

In the signal constellation shown above, each symbol conveys one bit of information (2 states). In discussion on digital data transmission, the measures used to discuss the information transfer rate are bits-per-second, for example 9600 bps, which is the actual data rate. The rate of symbols per second is called a *baud*, for example 9600 baud. When 2-level signaling is used, bps = baud. Alternatively, each symbol may convey more than a single bit. For example, instead of two levels +1 and -1 volts, we may choose to code 4 levels into one bit, say +3, +1, -1, and -3 volts. Then each symbol conveys 2 bits of information, and the baud rate is one-half the bit rate. The constellation diagram for a 4-level signal could look like figure 1-9. Note that the states are equidistant from one another (a distance of 2 volts between states).



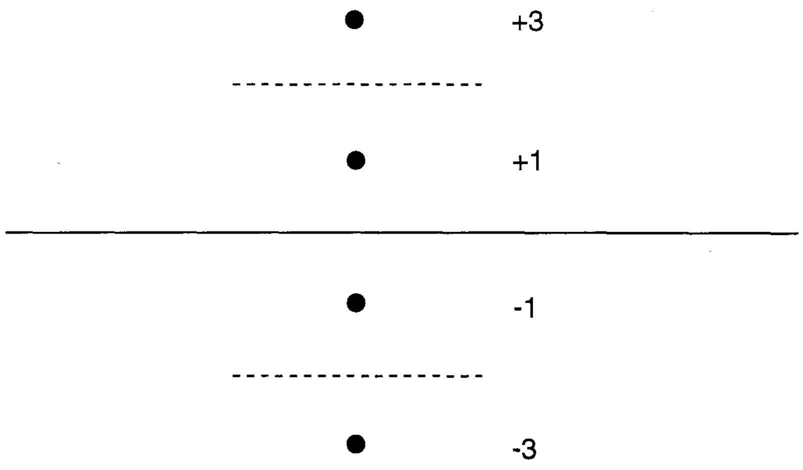


Figure 1-9 *- eye pattern and constellation diagram of a 4-level signal. Each symbol conveys 2 bits of information. Note that the states are equidistant from one another.*

Chapter 1 - References

Proakis, J. "Digital Communications." McGraw-Hill, Inc., 1983, Chapter 2.

Taub ,H. and D. Schilling. "Principles of Communications Systems." McGraw-Hill, Inc., 1971, Chapter 13.

Shannon, C. "A Mathematical Theory of Communications." Bell System Technical Journal (BSTJ), July 1948.

Shannon, C. "Communication in the Presence of Noise", Proc. IRE, pp 10-21, 1949.

Slepian D. (ed). "Key Papers in the Development of Information Theory." IEEE Press, 1973.

Viterbi, A. and J. Omura. "Principles of Digital Communication and Coding." McGraw-Hill, 1979, Chapter 1.

Chapter

2

Additive White Gaussian Noise (AWGN)

In the process of determining how well a particular modem design works, we typically will look at the bit-error-rate (BER) of the modem vs. the signal-to-noise ratio (SNR), usually expressed as Eb/No, the energy in a bit vs. the noise power density. These charts will be discussed extensively in future chapters. In this chapter, an intuitive introduction to the derivation of a bit-error-rate curve is given. Different types of noise will give different BER curves. It is important that the characteristics of the noise be understood, and agreed upon, for testing modems. The most commonly agreed type of noise to use is *Additive White Gaussian Noise*, abbreviated AWGN. It is the simplest to model, and it provides a fair basis for comparison of two different modems. Let's look at the characteristics of this noise and see why it is a good choice for testing and why it models the real-world fairly well.

Definition of AWGN Noise

The term "white" in AWGN refers to the spectral density of the noise — that is, the power of the noise within a given frequency range. White noise has the property that the energy density is constant regardless of frequency. For example, if we were to look at the noise at the IF frequency of a receiver, with say an IF of 10.7 MHz, and a bandwidth of 20 kHz, then the noise would be white if the average power of the noise within a small subbandwidth of frequencies was the same everywhere throughout the IF passband. Thus, the average power in a 100 Hz bandwidth at 10.700 MHz would be the same as the average energy in a 100 Hz bandwidth at 10.690 MHz, and also at 10.710 MHz. Figure 2-1 is a plot of the power spectral density of a typical white noise signal versus frequency.

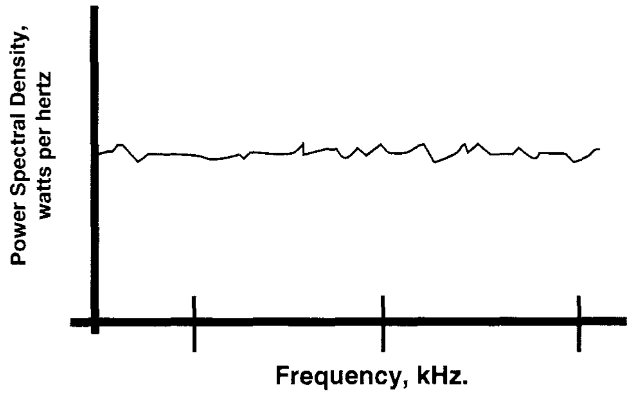


Figure 2-1 *- white noise has a power spectral density (watts per hertz) that is relatively constant vs. frequency.*

Other energy distributions are possible. One notable case is the use of an FM detector (such as a discriminator) where, after detection when we sample the baseband, the energy density is higher at higher frequencies, by 6 dB/octave. In this case, the energy per unit bandwidth at 2 kHz would be 6 dB (4 times) higher than at 1 kHz, which is not white noise. White noise due to external noise sources and thermal noise is a good assumption for the range of bandwidths and frequencies that we expect to use for our modem designs. White noise also has these same flat spectral density characteristics at baseband — that is if the energy from 1-11 Hz is the same as the noise energy from 20-30 Hz., then the noise is white.

In a real system, we will filter the data with a complex filter shape, and the received noise will be filtered with the same filter shape as the data. We will make the fairly gross assumption that the noise voltage that was white before being filtered is still white after being filtered by the data filter. In other words, we add the AWGN to the data, then filter the resultant signal at baseband. We will have to choose an equivalence factor which states that the noise energy of the filtered noise within the filter bandwidth, f, is approximately the same as white noise of some standard filter bandwidth, g. The equivalence factor would then be f/g. This will give us an approximation for generating BER curves. In a filter with a steep cut-off, the equivalence factor will be approximately one, and we'll assume a factor of one in the following computations. In the section on testing, a more precise calculation of noise will be made.

Gaussian Distribution

Next in AWGN is the term *Gaussian* which refers to the shape of the histogram (probability density curve) of a collection of sample noisevoltages. A Gaussian curve is also known as a 'normal' curve, and has a mean and a Standard Deviation. In testing our modems, we will want the mean of the noise voltage to be zero, and the standard deviation to be one. A mean of zero is the same as saying that there is no DC content to the noise — Gaussian-white noise always has a mean of zero. It turns out that the RMS voltage of a Gaussian distribution with mean equal to zero, and standard deviation equal to one is also equal to one, which is rather convenient. This implies that 63% of the time the noise voltage will be in the range -1 volt to +1 volt. It will be in the range -2 to +2 volts 95% of the time, and it will be in the range -3 to +3 volts 99.5% of the time. The noise voltage will only deviate out to -6 volts, or +6 volts twice in one billion events. The noise has an equal probability of being positive or negative (zero mean). Figure 2-2 shows the Gaussian probability density function (PDF) of a zero-mean, unit-standard deviation noise voltage signal, and a corresponding uniform probability density function for comparison.

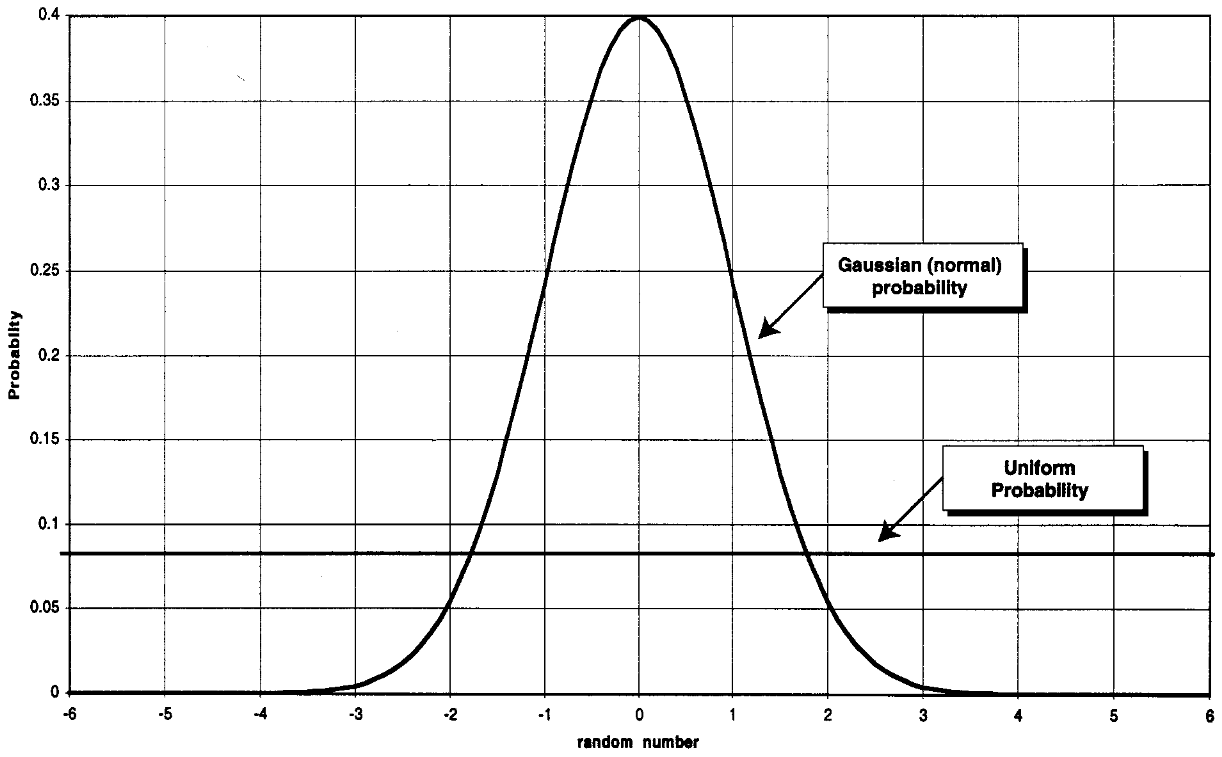


***Figure 2-2** - Gaussian and Uniform random probability distributions.*

Of the various types of noise, Gaussian white noise usually has the most detrimental impact on the performance of modems. Never the less, it also closely models the physical actions that occur in the generation of white noise itself, and what happens in the real world. This can be seen in one formula that is useful for generating a Gaussian-distributed random variable. If a number of random variables are generated, each of which has a uniform distribution, and they are added up, the result is a Gaussian-distributed random variable. Thus, Gaussian noise can be viewed as the sum of many different noise voltages processes, each of which has a uniform probability distribution. The following formula gives a Gaussian-distributed random variable with zero mean, and a standard deviation of one, where we sum 12 times to give good results:

$$\chi = \sum_{1}^{n=12} u_n - \frac{12}{2} \qquad \text{(2-1)}$$

where u is the uniform-distributed random number generator found in most software programming languages (i.e. BASIC, C, etc.) which produces some number between zero and one, each with equal probability.

The formula indicates that if 12 different uniform random numbers are added, then 6 is subtracted, the result yields a zero-mean Gaussian random number. Excel 5.0 has a Gaussian random number generator built-in (as well as a uniform random number generator).

Some random number generators do not do a good job in generating truly random numbers, usually with the result that a Gaussian number produced will tend to cluster towards zero (the mean) and will not have occasional values far removed from the mean. The uniform number generator would have to produce occasionally 11 or 12 consecutive numbers all around 1 (or all around zero). The common techniques using a seed and remainders may cause some cyclical tendencies that prevent a long stream of random numbers near one value (which should happen, but with small probability). So, some care has to be taken to validate the distribution. Usually this can be done by generating a large number of Gaussian random numbers and then plotting them to see if a few differ significantly from the mean (in accordance with how many standard deviations away from the mean they are). A better distribution of numbers can be obtained by the "rejection method" (Press, 1992).

A more thorough discussion of Gaussian white noise is described in Stearns and Hush (1990) and Press (1992).

Calculating the BER given AWGN

Appendix A gives more details on the Gaussian (normal) random distribution. If we know what a Gaussian-distributed number is and how it is distributed, then the effect that it has on the bit error rate of a signal can be determined. This is purely an exercise in statistics if an ideal eye pattern is corrupted by noise. Assume that we have an ideal eye pattern along with a perfect symbol-timing recovery circuit, and that the eye pattern is sampled at the best possible time (see Figure 2-3). At that best possible time, the value of the received data symbol will be either +1 volt or -1 volt, depending upon whether a one or a zero was transmitted. If AWGN noise is now added to the eye pattern, a data error can occur when the amplitude of the noise exceeds the amplitude of the eye (data signal) at the instant in time that the data signal is sampled.

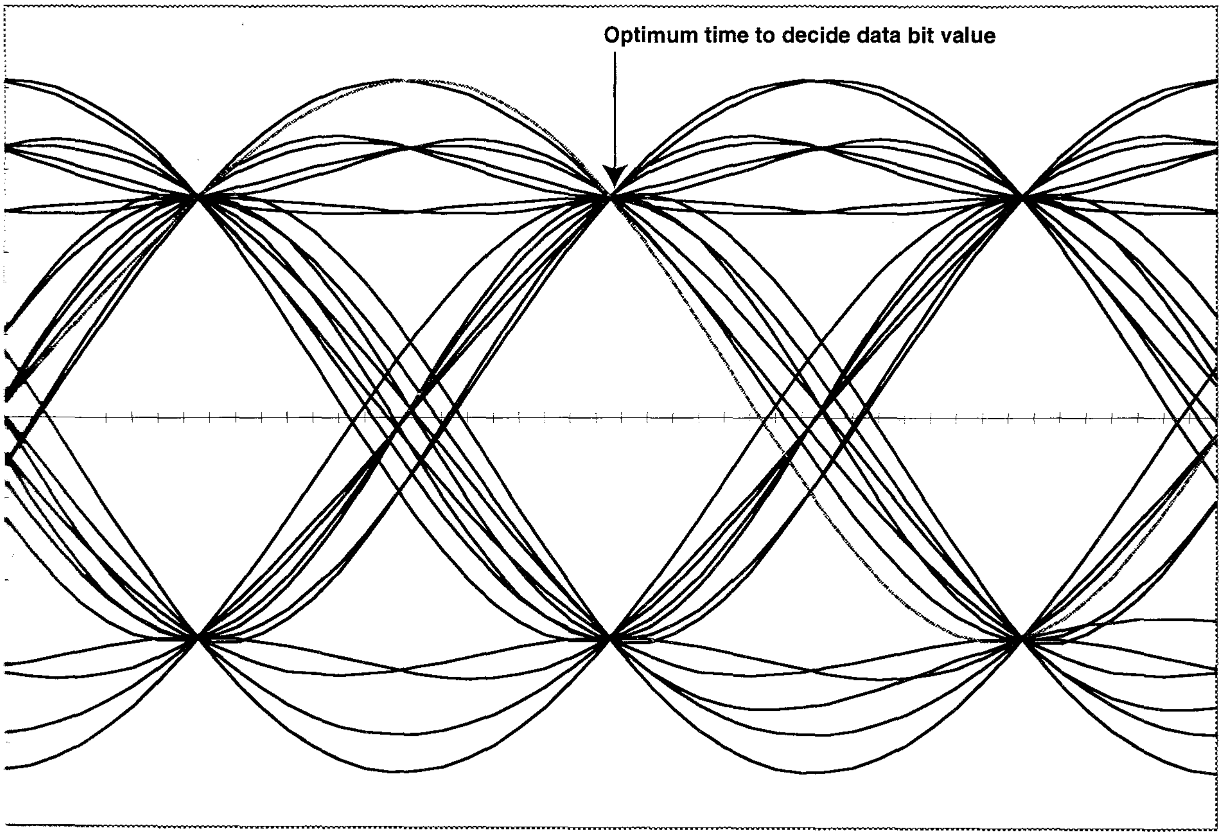


Figure 2-3 - Eye pattern showing optimum to sample the data for a value of +1 (logic one) or -1 (logic zero).

This happens half the time, since if the noise voltage is the same polarity as the data signal, then no error results. Only when the noise voltage is of opposite polarity and exceeds the amplitude of the data signal does an error result. So, the question to ask is, "How often does the noise voltage exceed a certain amplitude of the opposite polarity?" This is a difficult question to answer analytically, but a special mathematical function has been defined to answer exactly this question — it is called the *complementary-error-function*. This function tells us the area of one of the tails of the Gaussian-distribution function, see figure 2-4. There is no closed-form computational solution to this function. It either has to be looked up in a table, or it is available directly as a function in Excel 5.0. To show how to get the cumulative formula, we start by looking at the Gaussian random number distribution curve. This indicates that the probability density of a Gaussian random number (the noise voltage) at some value, x, given by the following formula:

$$P(x) = \frac{1}{\sqrt{2\pi}} \exp\left[- \frac{x^2}{2} \right] \tag{2-2}$$

which assumes that the mean is zero, and the standard deviation is one.

This formula gives us the density of probability of occurrence of a single value of voltage, x, while what is really needed is the entire area of the curve outside of n, which is the chance that x, the noise voltage, is equal-to-or-greater-than the desired data bit voltage, n. This is known as a *cumulative distribution*. To do this, equation (2) has to be integrated from n to plus infinity, thus finding that upper tail of the curve. (see figure 2-4, showing the area) This formula is expressed as:

$$Error(x) = \int_{n}^{+\infty} \frac{1}{\sqrt{2\pi}} \exp\left[- \frac{x^2}{2} \right] dx \tag{2-3}$$

Excel 5.0 has built-in functions to help us compute this integral, which is otherwise difficult to do. One function, NORMSDIST(), computes the above, except from minus infinity to $+n$. Fortunately, we recognize that this is just the other side of the probability curve. Since the total areas under any probability curve must be equal to one, and the curve is symmetrical, the error computed above can be restated as:

$$Error(x) = 1 - NORMSDIST(x) \tag{2-4}$$

Another very useful function to compute this probability is the complementary error function, abbreviated erfc(x), which is most commonly used in textbooks on the subject, and which is also available in Excel. The error function is really just some minor manipulations of the above error equation. ERFC is defined in terms of NORMDIST as:

$$1 - \mathit{NORMSDIST}(x) = \frac{1}{2}\mathit{ERFC}\left(\frac{x}{\sqrt{2}}\right) = Q(x) = Q\left(\sqrt{\frac{E_b}{N_o}}\right) \qquad (2\text{-}5)$$

This expression is difficult to compute easily. In future parts of this book the engineering notation will be used, and ERFC(x) will be used for the calculations. Given that we can either look up ERFC(n) in a textbook, or use Excel to compute the value, we are now ready to construct the bit-error-rate curve of the binary baseband signal when it is corrupted by AWGN. This function is commonly abbreviated as Q(x).

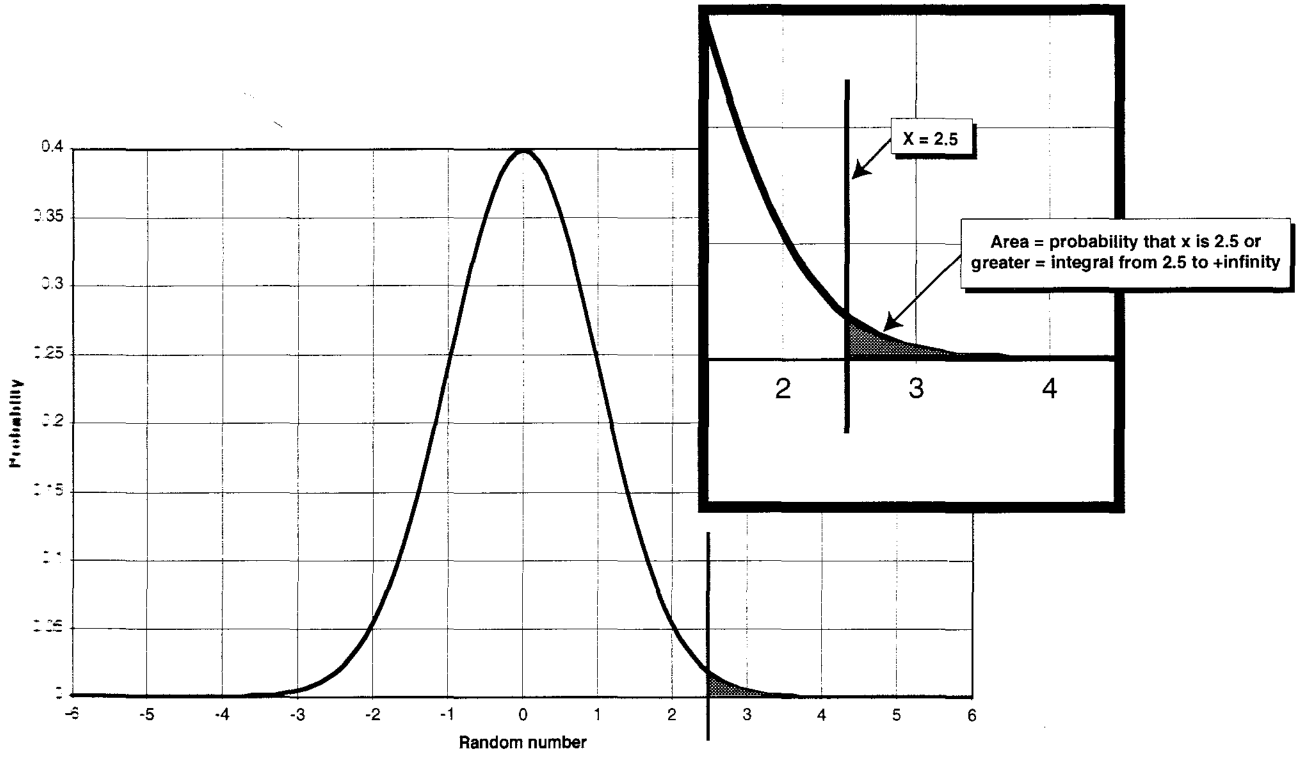


Figure 2-4 - The complementary error function is the area of the curve outside the X-value. In this example, the area is the probability that the noise voltage, x, is greater than +2.5 volts (RMS noise = 1 volt).

Generating a BER Curve

To generate the BER curve, it is necessary to plot at each desired signal-to-noise ratio the probability that the noise voltage is equal-to-or-greater-than the signal voltage. Figure 2-5 is a graph of this probability. Either the NORMSDIST or ERFC functions can be used, and the same curve will be generated when taking into account the factor of 1/2 and sqrt(2). This curve will be recognized quickly as the standard shape of bit-error-rate functions of digital data vs. noise. It shows that as the signal-to-noise ratio increases slightly, the bit error rate improves dramatically. This curve is sometimes called a *waterfall curve*, due to it's resemblance to a waterfall. The improvement is usually called the *threshold effect*, meaning that above some SNR there are almost no errors (lower-right part of figure 2-5).

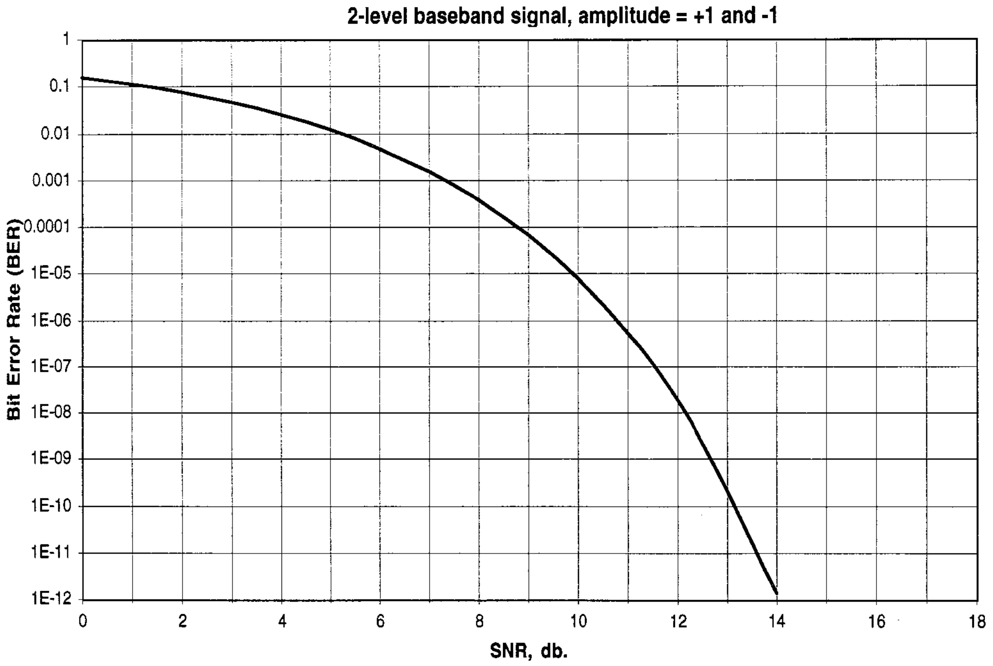


Figure 2-5 *- Probability that a given error voltage will exceed the received signal voltage vs. the signal to noise ratio, in dB.*

As will be seen in future chapters, this is a perfect curve and it assumes a perfect modem implementation. There are many defects in modem design and adjustment, as well as other effects, such as interference, fading, and multipath, which will prevent attaining error rates as good as those shown in this curve. This curve is the same as the theoretical error performance of coherent (orthogonal) 2-level PSK.

Relation of Power Spectrum and AWGN

In order to perform useful calculations with AWGN, it is necessary to describe the total power in a band-limited random white Gaussian signal. Measured in watts-per-hertz, the power spectral density describes the power of the noise is flat versus frequency for a white signal. Also, it was stated that the RMS voltage of a Gaussian random sequence is equal to the standard deviation, sigma. Thus, the power in the sequence is proportional to the RMS voltage squared, meaning that power equals sigma squared. Let's assume that the noise is band limited from DC to 15.9 Hz (100 radians/sec). Assume that the noise power in that bandwidth is 1 watt. Then sigma is 1 volt, and the signal has an amplitude of 1 volt RMS (assuming 1 ohm load impedance) in that bandwidth.

The power spectral density is just the power divided by the frequency spectrum (in hertz). If the double-sided spectrum goes from -15.9 hertz to +15.9 hertz, then the power spectral density is:

$$PSD = \frac{Power}{Frequency} = \frac{1}{31.8} = 0.0314\ watts\,/\,hertz \quad \text{(double-sided bandwidth)} \qquad (2\text{-}6)$$

Note that when we measure a real signal, we cannot distinguish the negative frequencies from the positive frequencies. This results in a measurement of the sum of the two. Thus, in this example, we measure 0.0628 watts per hertz in the real spectrum, 0.0314 of which occurs at positive frequencies and 0.0314 of which occurs at negative frequencies.

Converting our units from hertz to radians per second, we can alternatively formulate the same calculation as follows. The power in an AWGN random signal is equal to $1/2\pi$ times the area of the spectrum vs. frequency in radians/sec (which is flat for white noise of course). Let's assume that the noise is white from DC up to a frequency ω_m. Then the power spectral density is:

$$PSD = \frac{\pi P}{\omega_m} \qquad (2\text{-}7)$$

where P is the power in watts.

See figure 2-6 , for a diagram of the noise spectrum, and power spectral density.

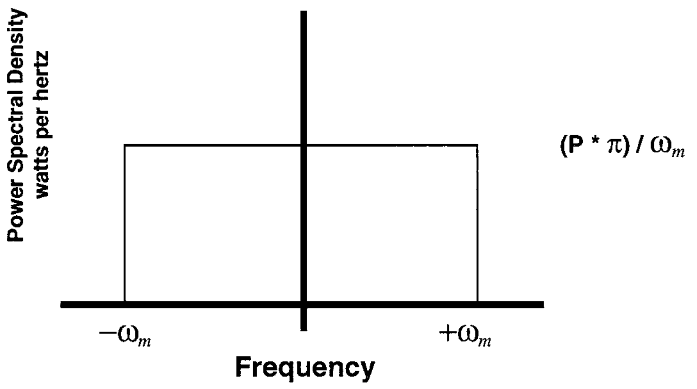


Figure 2-6 - *The power spectral density of a signal of bandwidth omega m includes both the negative and positive frequencies.*

Assume that the signal is flat from DC to 100 radians/second (15.9 hertz). Then the power spectral density is:

$$PSD = \frac{\pi P}{\omega_m} = \frac{3.14 \bullet 1}{100} = .0314 \ \mathit{watts / hertz} \qquad \text{(double-sided bandwidth)} \qquad (2\text{-}8)$$

Note that the units are watts per hertz (not radians) since the factor π cancels. The spectrum goes from -100 radians/second to +100 radians/second. Thus, the area is 200 * .031 = 6.28. Since the power in the signal is $1/2\pi$ times the area (in radians), the power is one watt, which agrees with the initial assumption. The double-sided spectrum is discussed and appropriate distributions are described in Stearns and Hush (1990), Taub and Schilling (1971), and Stremler (1982).

Normally when we describe power density we describe only the positive part of the bandwidth. We will add the power in the negative and positive parts of the spectrum. Then the single-sided bandwidth will be used to describe the PSD in watts/hertz, recognizing that it really consists of half that power in the positive spectrum and half that power in the negative spectrum.

Computing Noise and Signal Magnitudes

In the previous section we defined the energy in a bit, Eb, as the product of the signal power times the duration of that bit. Thus, if our baseband data signal is a random signal with amplitudes of +1 and -1 volt (and a 1-ohm load is assumed), then the average power is 1 watt. If the data rate is 1000 bits/second, then the time of a bit is 1 millisecond. Thus the energy of a bit is 1 watt * .001 second = .001 watt-second = 1 millijoule. Similarly, if a band-limited AWGN noise source has a power spectral density of .001 watts/hertz, then the Eb/No of this signal is:

$$\frac{.001\ watt - second}{.001\ \frac{watt}{hertz}} = 1 \qquad \text{(All dimensional units cancel)}$$

This is equivalent to a Eb/No of zero dB. If this noise signal is measured in a bandwidth from -1000 to +1000 hertz, the total power would be 1 watt (1000 hertz * .001 watt/hertz). Thus, the RMS noise voltage measured in a rectangular filter with unity response from DC to 1000 Hz. bandwidth would be one volt. The noise power increases directly with bandwidth — that is, if the bandwidth is doubled, the noise power doubles. *However, the noise power density is independent of the bandwidth.*

To compute the relative signal and noise magnitudes for different Eb/No ratios, it is merely necessary to scale accordingly. For example, if the Eb/No is +6 dB., then the Eb is 4 times greater than the No. Assuming the same Eb as before (1 millijoule), then the noise density would be one quarter as much as before or .00025 watts/hertz. The noise power in a 1000 hertz filter bandwidth (going from -1000 Hz. to +1000 Hz.) would then be .25 watts, corresponding to an RMS voltage (1-sigma voltage) of 0.5 volts in that bandwidth. *The measurement bandwidth of the noise must always be known, since white noise of infinite bandwidth has infinite power.*

Chapter 2 - References

Taub, H. and D. Schilling. "Principles of Communications Systems." McGraw-Hill, Inc., 1971, Chapter 7.2.

Stearns, S. and D. Hush. "Digital Signal Analysis," 2nd edition. Prentice-Hall, Inc., 1990, Chapter 13, and Chapter 15 section 2.

Stremler, F. "Introduction to Communications Systems," 2nd edition. Addison-Wesley Publishing Co., 1982, Chapter 4.

Press, ?, ? Tuekolsky, ? Veterling, and ? Flannery. "Numerical Recipies in C," 2nd edition. Cambridge Press., 1992.

Chapter

3

Antipodal, Orthogonal, and Non-orthogonal Signaling

In general, the transmission formats that are used will fall into one of three general types:

Antipodal -	the signaling states are exact opposites of one another.
Orthogonal -	the signaling states do not affect one another.
Non-orthogonal -	the signaling states affect one another (usually negatively) to some degree.

Let's look at these three types of signaling and then derive a general performance statement about them in terms of their bit-error-rate curves.

Antipodal Signaling

First, consider a two-level signaling system in which the transmission consists of either a *one* or a *zero* as the two states. A general demodulator could be described as being composed of two detection functions: one for the mark and one for the space. See figure 3-1 for the diagram of a two-level PSK demodulator for this case. In a phase detector, the output of the detector is an analog signal (after suitable filtering and gain scaling) with a value between -1 and +1 including zero. In carrier systems usually +1 and -1 are the two signal states being transmitted, although +1 may correspond to +5 volts (logical one) and -1 correspond to zero volts (logical zero). In this case, a zero transmission signal is an invalid signal state. The two transmitted states (PSK in this example) are 0-degrees and 180-degrees. Figure 3-2 shows the phase states schematically. Antipodal means that the two phase states are as far apart on the phase plane (Figure 3-2) as possible; the states are in this sense the 'opposite' of one another.

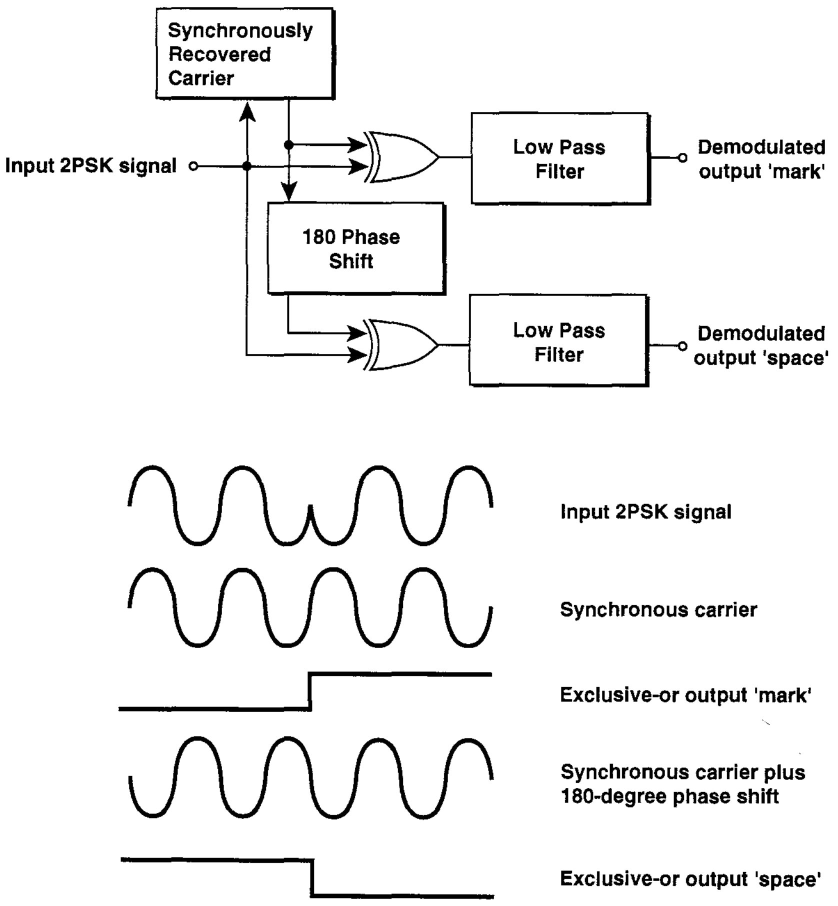

Synchronously Recovered Carrier
Input 2PSK signal
180 Phase Shift
Low Pass Filter
Demodulated output 'mark'
Low Pass Filter
Demodulated output 'space'
Input 2PSK signal
Synchronous carrier
Exclusive-or output 'mark'
Synchronous carrier plus 180-degree phase shift
Exclusive-or output 'space'

Figure 3-1 - Antipodal demodulation: the two detectors have opposite outputs for valid modulation states.

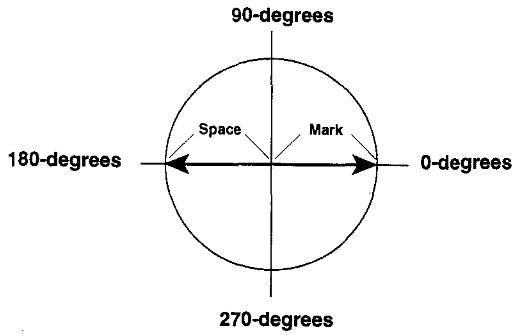


Figure 3-2 - the two phase states for antipodal modulation (PSK in this example).

In-phase signals (0-degrees to the reference) cause the mark detector output to be +1; 180-degree out-of-phase signals cause the output of the mark detector to be -1; and signals that differ by +90 or -90 degrees cause the output of the detector to be zero (halfway in between -1 and +1).

In this case, we just send the reference carrier to the first phase detector and a 180-degree offset reference carrier to the second phase detector. Now let's transmit the 0-degree signal representing a one data bit. The first phase detector will have its reference signal and the received signal in exact phase agreement, so its output will be +1. The second phase detector will have the received carrier 180-degrees out of phase with its offset reference carrier, so the output of that detector will be -1. The analog difference is +1 minus -1, which is +2, clearly a one data bit. When we transmit the other signaling state, the 180-degree carrier position, the first phase detector will produce a -1 output since the received carrier and the reference carrier are 180-degrees out of phase. The second phase detector will output a +1 since the received carrier and the offset reference carrier are exactly in-phase. Thus the output is -1 minus +1, which is -2, clearly a zero data bit.

In this case the two different phase detectors are always both participating in the decision about what the value of the output data bit is going to be. The two signaling states of 0- and 180- degrees are as far apart in the signaling space as they can possibly be, and each state affects both channels, so this is an antipodal signaling system.

Orthogonal Signaling

Now lets assume that we choose a transmission system where we have two phase states (representing the one and the zero, respectively). Also, let's ignore the problem of recovering a reference carrier for the time being, and assume that we have one at the receiver. In this second case of orthogonal signaling, we assign 0-degrees and +90-degrees as the two carrier phase positions representing the mark and the space states. In our generalized demodulator in figure 3-3, one of the demodulator blocks is the space demodulator, and the other is the mark demodulator. We then differentially sum the two in order to recover a one or a zero data bit. When we are sending a carrier at the 0-degrees phase position, the first phase detector has the reference carrier (which is at 0-degrees) exactly in-phase with the received signal. Thus, the output of the first phase detector will be +1. The second phase detector is fed with a reference carrier that has been 90-degrees offset from the reference carrier of the first phase detector. Its output will be zero, since the received carrier is at 0-degrees, and the offset-reference carrier is at 90-degrees. Clearly, the output of the system will be a 1 data bit, since the first detector output of +1 minus the second detector output of 0 yields an analog difference of +1.

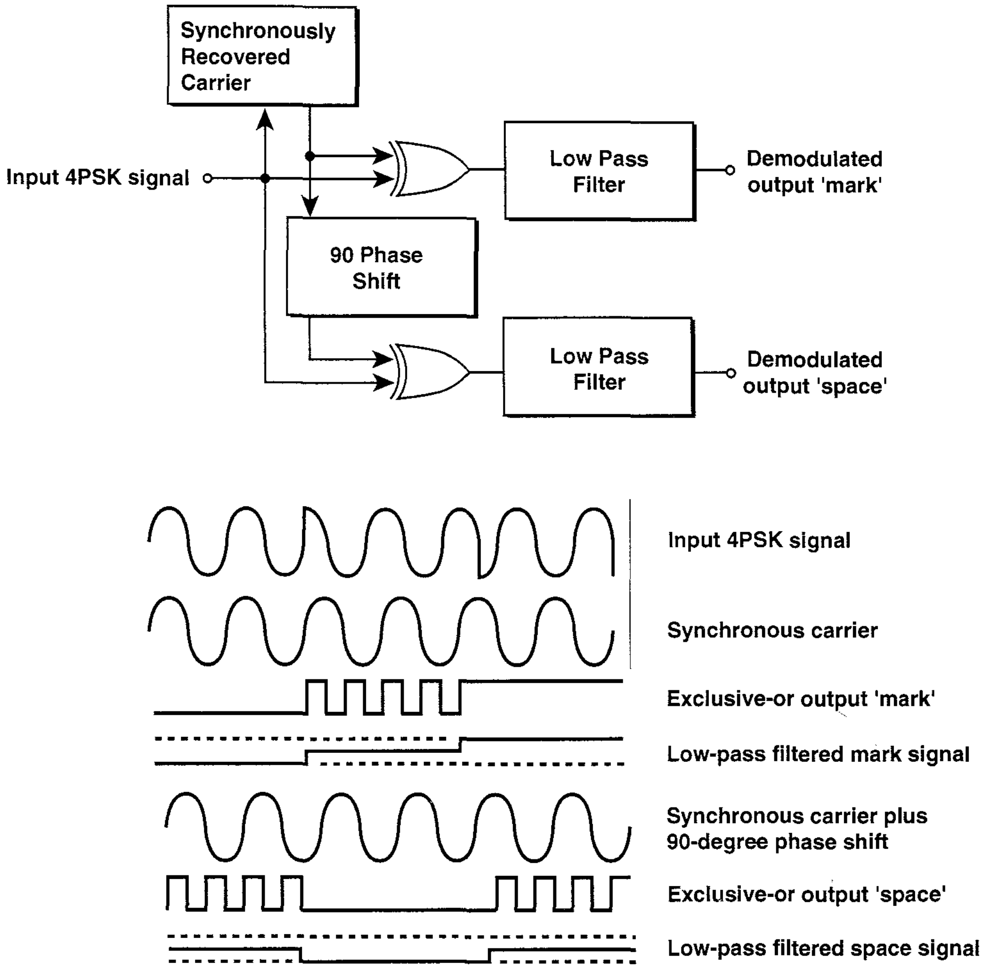

Synchronously Recovered Carrier
Input 4PSK signal
Low Pass Filter
Demodulated output 'mark'
90 Phase Shift
Low Pass Filter
Demodulated output 'space'
Input 4PSK signal
Synchronous carrier
Exclusive-or output 'mark'
Low-pass filtered mark signal
Synchronous carrier plus 90-degree phase shift
Exclusive-or output 'space'
Low-pass filtered space signal

Figure 3-3 *- Orthogonal signaling states (shown as 4PSK). The two detector have orthogonal outputs (one detector output is zero when the other is +1 or -1).*

If we now transmit the opposite signaling state, a carrier at the 90-degree position, we can see that the output of the first phase detector will be at zero, and the output of the second phase detector will be at +1, since the received 90-degree carrier is in the same phase as the offset-reference carrier. Thus, the output of the demodulator would be 0 minus +1, which is -1, clearly the correct result. Figure 3-4 shows the phase states for orthogonal modulation (4PSK in this example), and the two cases above are the Mark=+1 and Space=+1 phases.

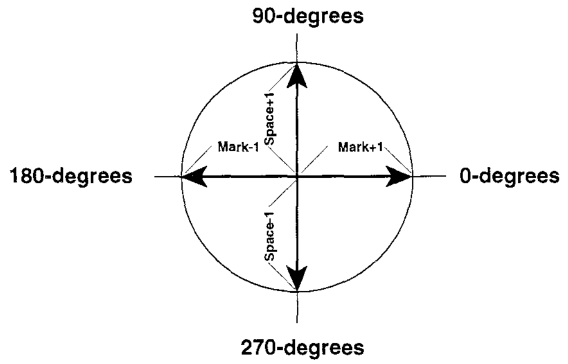


Figure 3-4 *- phase states for orthogonal modulation (4PSK).*

Something we notice quickly is that one of the two phase detectors has an output that is zero — which means the detector with the output of zero is not being influenced by the transmitted signal state. It really doesn't contribute to the results of the output. It is the output of the phase detector with a +1 state that has the influence on the output of the system. This is another way of saying that the signaling is orthogonal. *When discussing orthogonal signaling, we mean that the valid states of the signal only affect their particular detector; other detector or detectors are at the zero, or unaffected state.* So, in this system the two signal carrier states 0- and 90- degrees are orthogonal to one another (90-degrees is also orthogonal to 180-degrees, etc.).

Non-orthogonal Signaling

Now let's examine non-orthogonal signaling. Suppose that we take the preceding system, but set the two signaling states as 0-degrees and 45-degrees. We again have to re-arrange our two phase detectors to operate on the reference carrier and a 45-degree offset reference carrier. When the 0-degree state is transmitted, the output of the first phase detector will be +1, since the received carrier and the reference carrier will be in phase. The output of the second phase detector will be at +0.5, since the received signal will be 45-degrees different than the offset reference carrier. So, the output signal will be +1 minus +0.5, which is +0.5, clearly a one data bit. In the opposite signaling state, the first detector will have an output of +0.5, and the second detector will have an output of +1. Thus, the output signal will be +0.5 minus +1, which is -0.5, clearly a zero data bit. The phase detector that is out of phase with the received signal is still producing an output that is somewhat detrimental to optimum performance. The two signaling states affect both detectors, but not in the best way. Thus, the two signaling states are overlapped by one half.

Analysis of FSK signaling

Let's look at a case involving FSK. Referring to figure 3-5, we have a simple FSK demodulator. Again, there are two detection functions, one for the mark, and one for the space. However, contrary to the previous case, these are frequency detectors instead of phase detectors.

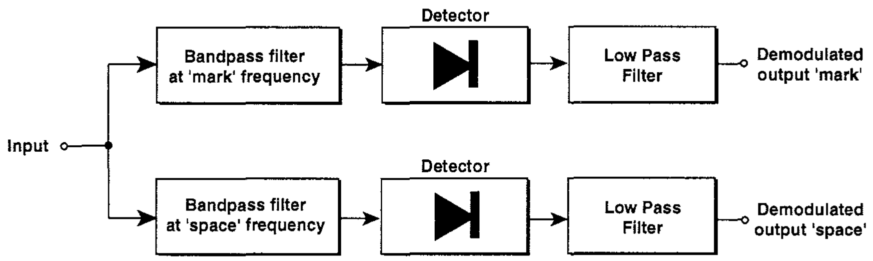

Input
Bandpass filter at 'mark' frequency
Detector
Low Pass Filter
Demodulated output 'mark'
Bandpass filter at 'space' frequency
Detector
Low Pass Filter
Demodulated output 'space'

Figure 3-5 *- non-coherent FSK detector.*
The demodulated outputs can only indicate presence or absence of energy through the respective bandpass filters, so the resultant demodulator is, at best, orthogonal, and in reality is non-orthogonal.

It's pretty easy to see that FSK can never be an antipodal signaling system, since the frequency detector can only determine if the received carrier matches its programmed frequency or doesn't. There is no frequency that is the opposite of some other frequency. Clearly, the best that we could do would be to make sure that the two frequency signaling states assured that, when we are transmitting a mark, the space filter detects nothing. Likewise, when we are transmitting a space the mark detector detects nothing, which would be orthogonal FSK signaling. In order to achieve this, two conditions would have to be met:

1. The two filters have to be selective enough that neither one responds at all to the other carrier, and
2. The frequency spectrum of each carrier must not have any spectral energy components at the frequency of the opposite filter.

In order to satisfy rule one, it is necessary to design good filters. In order to meet rule number two, it is necessary to carefully choose the frequency separation of the two carriers based on the data bit rate that is being sent (the baud rate). Recall that when the mark carrier is turned on and off with one and zero data bits it will generate a spectrum with a sin(x)/x roll-off characteristic centered about the mark carrier frequency. If the space filter is chosen to fall at one of the spectral notches of this roll-off characteristic, then no energy from the modulated mark carrier will fall onto the (infinitely narrow) space filter. Obviously the inverse is true also. The spectral notch in the modulated space carrier will fall onto the (infinitely narrow) mark filter. If this can be achieved, then we will have achieved orthogonal FSK signaling, since the mark and the space carriers will have no affect on the opposite channel. In other cases, where preventing modulated mark carrier or modulated space carrier from falling into the wrong filter is not possible, there will be non-orthogonal signaling. In reality, orthogonal FSK is defined when using a pair of phase detectors to demodulate the FSK signal (by mixing the received signal with the two carriers) since the spectral notch in each signal is infinitely narrow. Orthogonality means that the product of the signals, the mark and the space, are zero when integrated over some period of time (related to the signaling interval).

In order to achieve orthogonal FSK, it is necessary to set the tone spacing so the notch separation from the carrier falls on the other channel, or:

$$Frequency\ Spacing = \frac{m}{T} \tag{3-1}$$

where m is an integer, and T is the data bit period. But of course, we have to use synchronous detectors (phase detectors) to accomplish this.

In a previous section the bit error rate performance of a baseband signal in the presence of additive white Gaussian noise (AWGN) was computed. We will now see how the three different types of signaling will perform in the presence of AWGN. In that example it was assumed that the noise had standard deviation, and the data bit was either +1 or -1. This equates to the orthogonal case described above. So, the BER curve previously calculated is the curve that we use when looking at orthogonal FSK signaling, or at orthogonal PSK. Recall that the probability of a bit error was proportional to the signal to noise (SNR) ratio:

$$Prob\ Error = \frac{1}{2} ERFC\left(\frac{x}{\sqrt{2}}\right) = \mathrm{Q}\left(\sqrt{\frac{\mathrm{E}_b}{\mathrm{N}_o}}\right) \quad \text{for orthogonal 2FSK} \tag{3-2}$$

Also recall that this is the best case and assumes true orthogonality, which actually requires phase-coherent detection (not the two filter detector as shown above).

$$\frac{E_b}{N_o} = \text{power ratio, SNR per bit}$$

$$\sqrt{\frac{E_b}{N_o}} = \text{voltage ratio, SNR per bit}$$

Eb/No is the signal to noise ratio per bit (as a power ratio), thus the square-root of this quantity is the voltage ratio

Now, let's compare the SNR between orthogonal 2FSK and antipodal 2PSK signaling. In the case of orthogonal signaling, the output can be composed of four components: 1) noise from the mark filter, 2) signal from the mark filter, 3) noise from the space filter, and 4) signal from the space filter. Since the signaling is orthogonal, signal is received from one of the filters at any given time, but noise is always received from both filters. When the system is switched to antipodal signaling, there will still be noise from both phase detectors, but there also will be signal from both phase detectors. Thus, the SNR has improved by 3 dB. If this 3 dB. is converted to a voltage ratio, it is equal to the square-root of two. So, the received SNR from the previous case is multiplied by 1.414.

$$\mathit{Prob\ Error} = \frac{1}{2}\mathit{ERFC}\ (x) = Q\left(\sqrt{\frac{2E_b}{N_o}}\right) \quad \text{For antipodal 2PSK} \qquad (3\text{-}3)$$

For non-orthogonal signaling it is necessary to look at the width of the receive filters, as well as the amount of spectrum overlap due to the modulated carrier energy falling into the wrong filter. This means that it is not easy to compute a single performance number, since there are many variables. Figure 3-6 is a graph of the BER curves of coherent 2PSK (antipodal modulation), coherent 2FSK (orthogonal modulation), and non-coherent 2FSK (non-orthogonal modulation). The bit-error-rate functions for various modulation types are concisely summarized in Michelson and Levesque (1985) and are more thoroughly worked out in Proakis (1983).

For non-orthogonal binary FSK the probability of a bit-error is equal to:

$$P_e = \frac{1}{2}\, e^{\left(-\frac{E_b}{2N_o}\right)} \qquad (3\text{-}4)$$

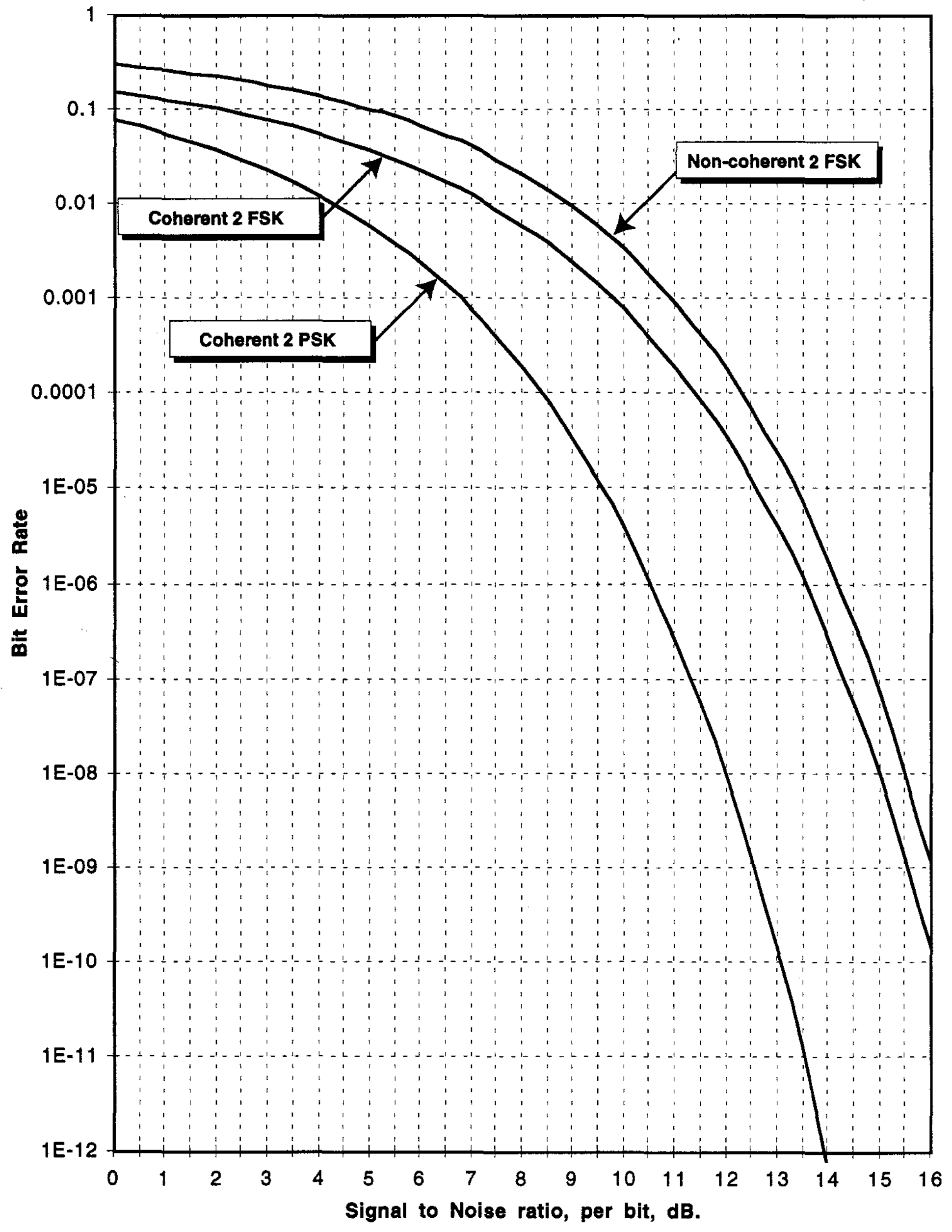


Figure 3-6 *- Theoretical Bit Error Rate vs. Signal-to-Noise ratio per bit, in dB. (Eb/No) for Antipodal, Orthogonal, and non-orthogonal 2-level signaling (coherent 2PSK, coherent 2FSK, and non-coherent 2FSK, respectively).*

Orthogonal Waveforms

A more precise definition of orthogonality can be shown mathematically, and can be readily visualized using multiplication (phase detection) of two waveforms. Two signals s_1 and s_2 are orthogonal over a time interval, T, when:

$$\int_0^T s_1(t)\ s_2(t)\ dt = 0 \qquad (3\text{-}5)$$

A physical interpretation of equation 4 is that the two signals s_1 and s_2 are multiplied (i.e. mixed) together and the resultant signal is integrated (low-pass filtered). If the output of the filter is zero at the instant in time T as compared to time zero (which can be arbitrarily chosen), then the two signals are orthogonal.

Figure 3-7 shows two signals, s_1 and s_2 that are orthogonal, and illustrates the result of equation 3-5. Note that the result has zero voltage delta between any two points separated by a time interval T, thus the two waveforms shown below are orthogonal. However, at a spacing of $T/2$, that there is no net difference in the voltages only if the starting and stopping points of the $T/2$ sample are precisely aligned with the zero crossings, otherwise there is a net difference between the voltages at the starting and stopping points of the $T/2$ sample (as illustrated in figure 3-7).

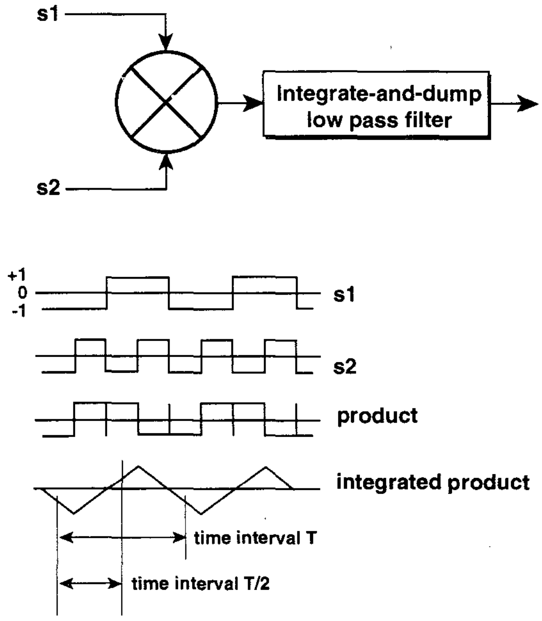


***Figure 3-7** - Interpretation of two orthogonal waveforms from equation 3-4. Waveforms* s$_1$ *and* s$_2$ *are orthogonal over the time interval* T *regardless of when the start and stop points are. However, they are orthogonal at a time interval of* T/2 *only when the starting and stopping points are carefully chosen (and they are not orthogonal with the start and stop points chosen in this figure).*

Thus, s_1 and s_2 are orthogonal at a sampling time difference of T regardless of the phase of the sampling time, but that they are orthogonal at a sampling time difference of $T/2$ only for carefully chosen sampling instants (sampling phase). s_1 and s_2 are said to be *coherently orthogonal* for $T/2$ (the sampling phase must be coherent to s_1 and s_2), and are *non-coherently orthogonal* for T (the sampling phase need not be coherent to either s_1 and s_2). A more rigorous treatment of orthogonal signaling is given by Proakis (1983, p. 148).

Orthogonal FSK Demodulators

A 2FSK detector can utilize this orthogonal principle to good effect. As a first example, assume that both s_1 and s_2 are signals, and that the phase of the two signals is known. Further, assume s_1 is the mark signal, and that s_2 is the space signal of a 2FSK modulation, and that s_1 is orthogonal to s_2 at the signaling rate, and also that s_1 and s_2 are carefully phase aligned to each other. Then a 2FSK demodulator can be described by the circuit in figure 3-8.

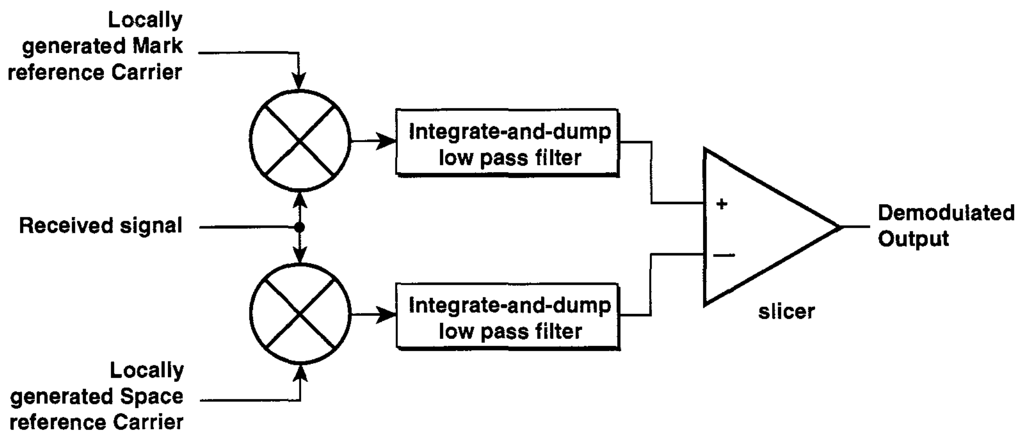

Locally generated Mark reference Carrier
Integrate-and-dump low pass filter
Received signal
+
–
Demodulated Output
Integrate-and-dump low pass filter
slicer
Locally generated Space reference Carrier

Figure 3-8 *- 2FSK demodulator based upon the mark and space waveforms being orthogonal. The locally generated reference carriers must be phase-aligned with the received signal, and the mark and space signals must be carefully phase aligned to each other at the demodulator.*

Since the mark and space signals are orthogonal to one another, then the output of the mark low-pass-filter will be maximum when a mark signal is received, and will be zero when a space signal is received. Alternately, the output of the space low-pass-filter will be a maximum when a space signal is received, and will be zero when a mark signal is received. Thus, this demodulator can attain near-optimal performance under the proper conditions.

In reality, generating the local mark and space reference carriers can be done, but we may not know the proper phase relationship of the local reference carrier to the received carrier. Or, s_1 and s_2 may have been sent without being phase aligned with respect to each other (thus making it impossible to generate local references that are properly aligned to both s_1 and s_2 simultaneously). In that case a more complicated demodulator is needed. Figure 3-9 illustrates such a circuit. In fact its performs the complex multiplication between each pair of signals, and then extracts the magnitude of the product. The squaring circuit is needed because if the locally generated carrier were 180-degrees out of phase with the received signal the result would be a -1 integrated output. The squaring circuit makes this a +1 — indicating complete correlation with the received signal. Additionally, the squaring circuit allows accurate generation of the magnitude, since the magnitude squared is equal to:

$$Mag^2 = \mathrm{Re}^2 + \mathrm{Im}^2 \qquad (3\text{-}6)$$

It is not necessary to extract the square-root of the sum, since the slicer is comparing two equivalent signals anyway. If we have a reference signal that is coherent to the received signal and aligned, we can dispense with the computation of both terms of equation 3-6. When the SNR is high, this yields a sufficiently good result.

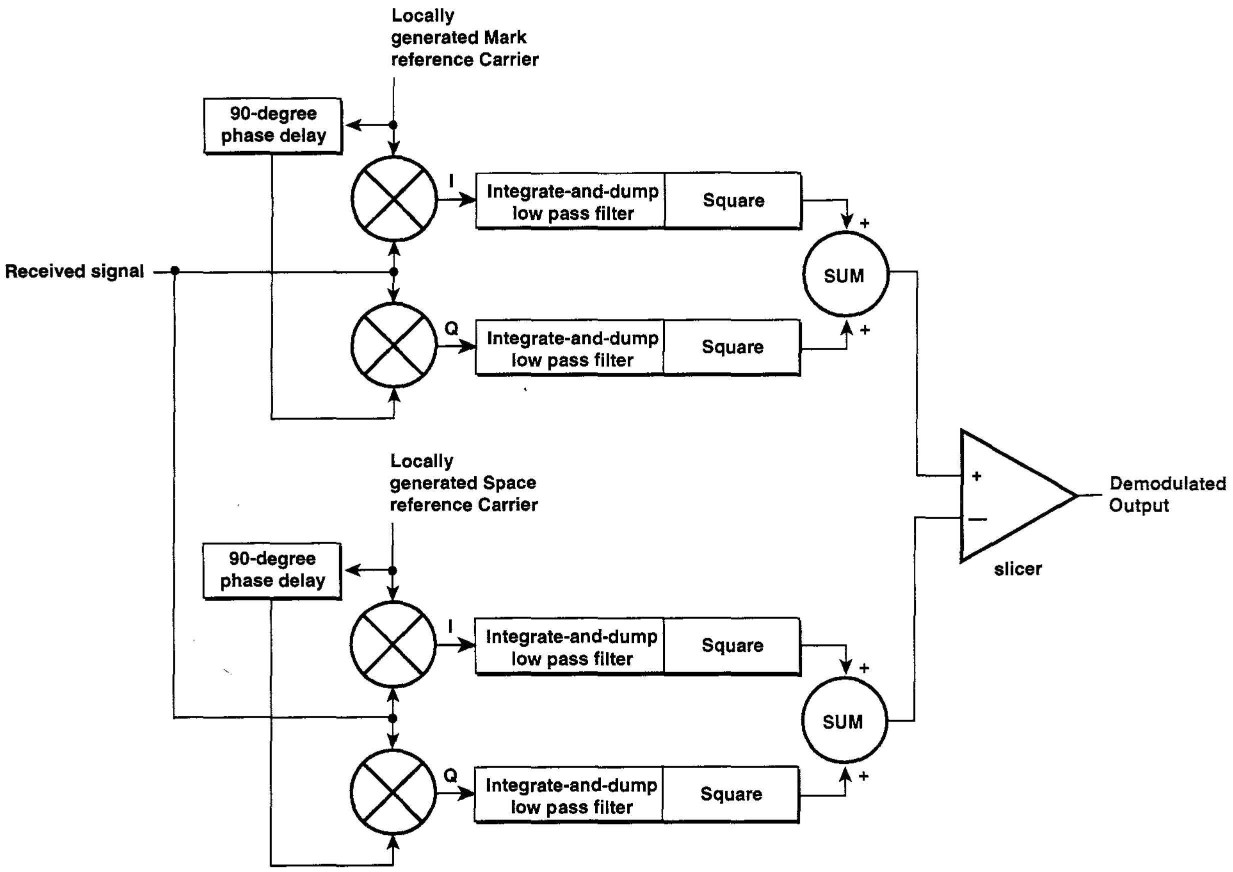

Locally generated Mark reference Carrier
90-degree phase delay
I
Integrate-and-dump low pass filter
Square
+
Received signal
SUM
Q
Integrate-and-dump low pass filter
Square
+
Locally generated Space reference Carrier
+
–
Demodulated Output
slicer
90-degree phase delay
I
Integrate-and-dump low pass filter
Square
+
SUM
Q
Integrate-and-dump low pass filter
Square
+

Figure 3-9 *- 2FSK demodulator based on orthogonal mark and space signals. The phase of the local reference carriers is arbitrary compared to the received signal, and the mark and space carriers do not have to be phase-aligned with respect to one another.*

The circuit of figure 3-9 allows more latitude in the generation of the signals s_1 and s_2. A technique similar to this is shown in Proakis (1983, fig 4.3.4). Although precisely generated, the exact phase relationships between s_1 and s_2 at the receiver may be lost if the transmission environment allows a slight frequency offset between the transmitter at one end and the receiver at the other end of the circuit (as may happen with a single sideband (SSB) circuit).

Chapter 3 - References

Michelson, Arnold M. and Allen H. Levesque. "Error-Control Techniques for Digital Communication." John Wiley & Sons, Inc., 1985, Chapter 1.5.

Proakis, John G. "Digital Communications." McGraw-Hill, Inc., 1983, Chapter 4.2.

Chapter

4 Carrier Transmission

Previous sections on the principles behind baseband transmission have dealt mostly with the transmitted and received signals prior to modulation, and after demodulation. This section deals with the modulation and demodulation processes, and the characteristics of the RF signal that results. In any real radio transmission system, we must modulate a carrier to effect the transmission of a data signal. The carrier can be modulated in many different ways. Some of those ways turn out to be equivalent to one another, but have been derived, and in fact modulated with different circuitry. Several characteristics of the modulated signal are quite important: 1) minimizing the emitted signal bandwidth, 2) minimizing the bit error rate at a given transmit power level, 3) resistance to interference, and finally, 4) resistance to the effects of propagation distortion (such as multipath transmission).

FSK - Frequency Shift Keying

One of the simplest modulation schemes, which has been used for many years, is to modulate the frequency of a carrier. A transmission system that sends a single binary bit of information per signaling interval (per baud) is a two-level transmission, where the transmitter sends two different frequencies. One frequency, the space frequency, represents one binary level while the other transmitted frequency, the mark, represents the other binary level. This is generally referred to as 2FSK, meaning 2-level FSK. Two-level FSK bears much similarity to amplitude modulation. For example, the binary signal turns one of the carriers on and off, while the inverse of the binary signal turns the other carrier on and off (obviously one carrier is on when the other is off). Thus, we can visualize the 2FSK system as two amplitude modulated carriers keyed differentially on and off. See figure 4-1 for an illustration of 2FSK as amplitude modulation. The spectrum of an amplitude modulated signal is easy to describe as being composed of the carrier and an upper and lower sideband. When we use unfiltered binary data to amplitude modulate a carrier, then the spectrum of the modulating signal is that of the baseband data signal, which for unfiltered binary data is a sin(x)/x spectrum shape. Thus a 2FSK signal is simply the sum of two such signals with the mark and the space carriers being superimposed on one another.

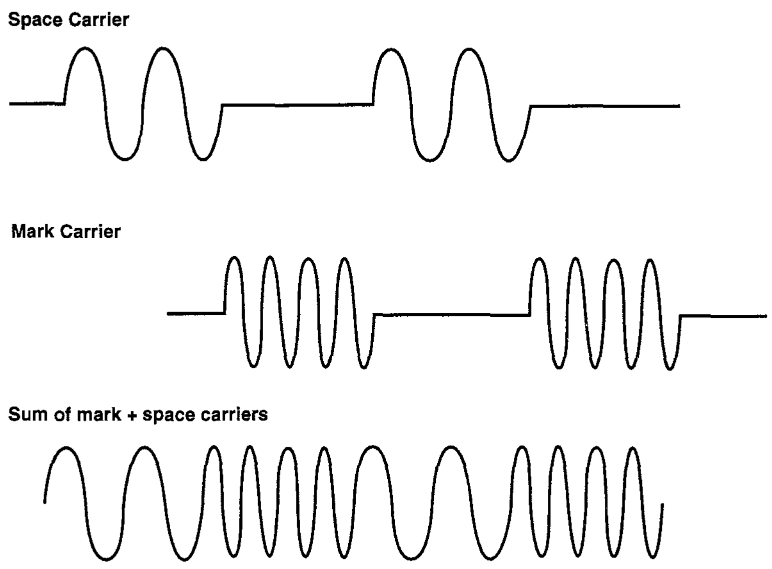


Figure 4-1 *- 2FSK can be viewed as the sum of an amplitude modulated space carrier plus an amplitude modulated mark carrier. A 0101 data pattern is illustrated.*

Figure 4-2 shows the resultant spectrum from a single tone amplitude modulating a carrier where an upper and a lower sideband are produced. Each is one-half the amplitude of the carrier at 100% modulation.

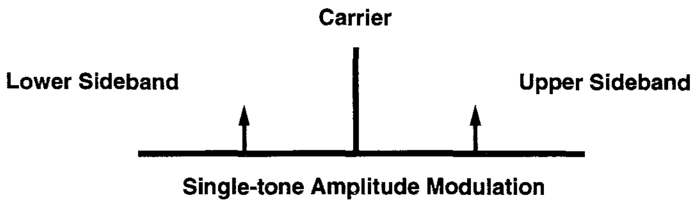


Figure 4-2 *- spectrum resulting from single-tone amplitude modulation of a carrier*

Figure 4-3 shows the spectrum resulting from an unfiltered binary data signal amplitude modulating a carrier. The data has a modulation response from near DC to a high frequency, and the upper and the lower sidebands are symmetrical around the carrier frequency. The width of the main lobe is twice the baud rate, expressed as a frequency. Thus, 1000 baud would produce a main lobe with a width of 2 kHz. between the first pair of spectral nulls.

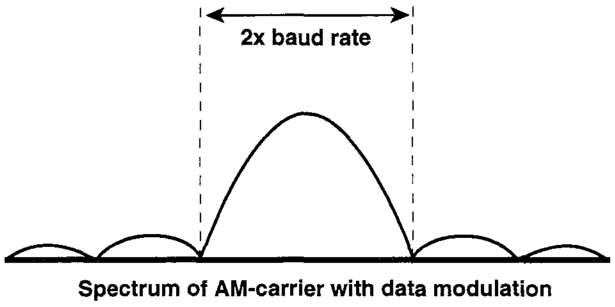


***Figure 4-3** - spectrum resulting from random-data amplitude modulation of a carrier. The first pair of nulls are separated by a frequency equal to 2 times the baud rate.*

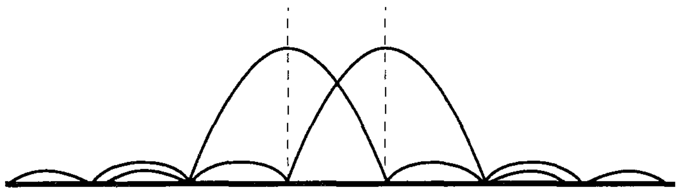


***Figure 4-4** - spectrum of FSK signal with 1/T separation (±1/2T deviation) unfiltered baseband data.*

The resultant spectrum of the FSK signal is the superposition of the two spectra, figure 4-4. Roughly speaking, two signals are orthogonal when there is no cross-talk between the signals — that is, there is no output on the space channel (due to the mark carrier) when a mark carrier is being transmitted, and vice versa. The computation of orthogonality involves complex numbers since the signals are sinusoids. It can be shown that the real part of the orthogonality is zero when the mark and the space carriers are spaced 1/2T apart from one another, where T is the baud rate. This is equivalent to a deviation of ± 1/4T. For example, if the data rate were 1000 baud, then the mark carrier must be 500 Hz separated from the space carrier for the real parts to be orthogonal; this can be restated to say that the space carrier is 250 Hz. below the channel center, while the mark carrier is 250 Hz above the channel center. The magnitude of the orthogonality (which includes both the real and the imaginary parts) is zero at a spacing of 1/T between the two carriers, or a deviation of ± 1/2T. For example, assuming a 1000 baud data stream, this equates to a separation of 1000 Hz between the carriers, or ± 500 Hz. deviation from the channel center frequency. The two carriers can be more widely spaced, but they will only be orthogonal at a regular spacing of n/T, where n is an integer. They can also be more closely spaced, but the resultant signal is non-orthogonal, meaning that a power penalty will occur upon demodulation as compared to the non-orthogonal case.

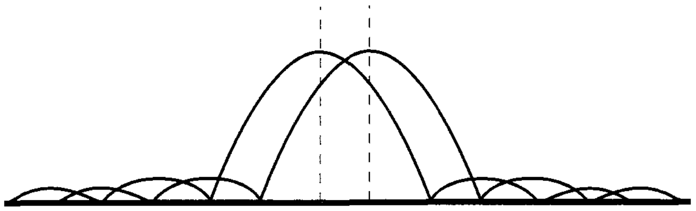


Figure 4-5 *- spectrum of FSK with 1/2T separation, (±1/4T deviation) unfiltered baseband data.*

If the phases of the mark and the space carriers are carefully controlled, then orthogonality can be achieved with a spacing of 1/2T between the mark and the space carriers, see figure 4-5. However, a coherent demodulation technique is needed to optimally demodulate such a signal; most common FSK demodulators (such as a discriminator) are not coherent. In this context, *coherent* means that the receiver is able to generate mark and space reference carriers that are exactly phase aligned with the received signal. Figure 4-6 is a block diagram of a typical non-coherent demodulator for 2FSK. If the phase continuity is controlled from one pulse to the next, then the modulation is called continuous-phase frequency-shift keyed modulation, usually abbreviated CPFSK. Controlling the continuous phase results in a signal spectrum that rolls off faster than if the phase is not controlled, and thus minimizes the bandwidth of the transmitted signal.

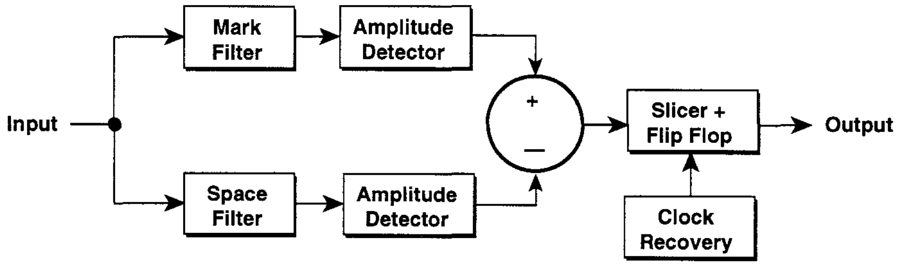


Figure 4-6 *- Non coherent 2FSK demodulator*

Another approach to non-coherent demodulation of 2FSK signals that is more amenable to fabrication in VLSI integrated circuits is based on two correlators, which can be implemented in switched-capacitor analog technology as well. Figure 4-7 is a block diagram of a correlator-type demodulator.

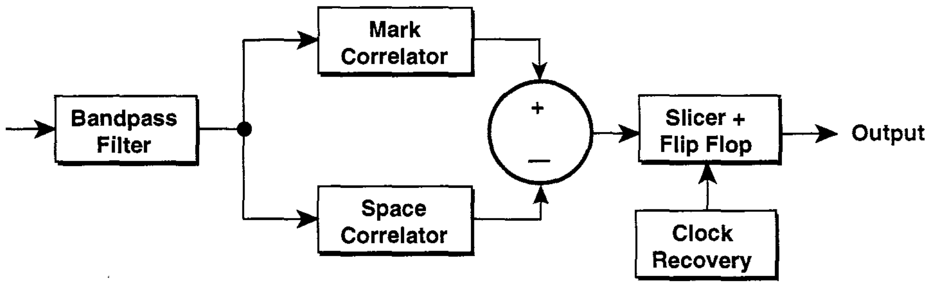


Figure 4-7 *- Non coherent 2FSK demodulator based on correlators*

The correlator is a device similar to a shift register that determines the degree of match between the received waveform and a stored reference waveform. The mark correlator stores a replica of the mark waveform, and the space correlator stores a replica of the space waveform. When the input signal has a good degree of match with the stored replica waveform, the output of the correlator is maximum. Correlators may tend to reject noise fairly well if they integrate the data over a significant time period. The correlators can be shift registers clocked at some convenient rate. It is also possible to coherently detect 2FSK, if reference carriers are available for the mark and space frequencies. Figure 4-8 is a block diagram of a coherent 2FSK demodulator. In theory a coherent 2FSK demodulator offers about 0.8 dB of improvement in SNR compared to a non-coherent demodulator at moderate E_b/N_0. In practice the improvement is usually less. The Mark and Space frequencies must be phase aligned with the received signal. Figure 3-9 (chapter 3) shows the schematic of a similar detector that does not require phase alignment between the locally-generated Mark and Space reference frequencies and the received frequencies (although this is then a non-coherent demodulator, and it suffers a 0.8 dB penalty as compared to figure 4-8).

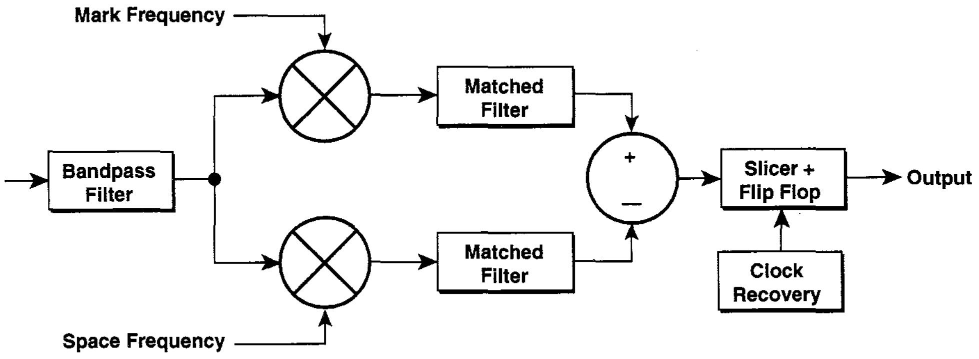


Figure 4-8 - *Coherent 2FSK demodulator*

An important characteristic of HF propagation is that phase coherence is not necessarily maintained over a path. Due to multi-path propagation, the mark and the space carriers may encounter different propagation delays, and thus the phase of each will be different. The phase will vary with time, and thus the use of 1/2T spacing, for which the signaling is orthogonal only with controlled phase, may result in carrier cross-talk. Thus 1/T spacing, where the carriers are orthogonal regardless of the phase, may be preferable in this case.

FFSK - Fast Frequency Shift Keying

A 2FSK signal that has the minimum possible orthogonal bandwidth is sometimes called Fast Frequency Shift Keying (FFSK) or also MSK, with a frequency separation of 1/2T. This requires generating the mark and the space signals coherently such that they are synchronized with one-half the bit rate. The mark signal must be phase coherent between different mark bits, and the space signal must be phase coherent between different space bits, all while maintaining ± 1/2T frequency shift. For example, assume that the data rate is 10,000 bits per second. Then the frequency shift for the mark is +2,500 Hz., and the frequency shift for the space is -2,500 Hz. from the carrier frequency. If the carrier frequency is made equal to a multiple of one fourth the bit rate, then the modulating wave can be switched during a zero-crossing, thus maintaining the continuous-phase criteria. In this example, the carrier frequency must be n * 2,500 Hz, where n=1,2,3,... Figure 4-9 shows the FFSK signal for a carrier frequency of 4 x 2,500 Hz (or 10,000 Hz.).

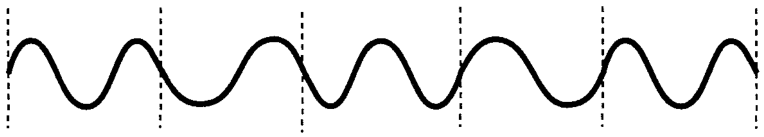


Figure 4-9 - FFSK carrier waveform for space = 7,500 Hz, mark =12,500 Hz, data rate is 10,000 baud. The figure represents the binary sequence 10101

Note that in the above waveform, the phase of one of the signals, such as the mark signal, has flipped between successive bits, and that the phase of the space signal has also flipped between successive bits. Successive one's or zero's do not have phase shifts. Thus, some phase modulation has been induced onto each of the two carriers.

mFSK - m-ary FSK

It is possible to increase the number of bits per symbol by having more than two distinct carrier frequencies. For example, if the input data stream were divided into 2-bit groups, making the baud rate half of the bit rate, then each di-bit group could be encoded as one of four carrier frequencies. The transmission then would consist of one of these four frequencies being sent during each bit time. This type of modulation is known as 4FSK. Similarly it would be possible to send one of 8 frequencies during each signaling interval representing 3 bits, or 16 frequencies representing 4 bits, etc. It must be kept in mind that the minimum spacing between the carriers must still obey the requirements listed above, namely 1/T tone spacing for non-coherent detection. If there are m tones, then the total bandwidth is on the order of m/T for non-coherent (and on the order m/2T if the tones are spaced 1/2T for coherent orthogonal detection). Figure 4-10 shows both the time-domain representation and the spectral representation of a 4-FSK signal.

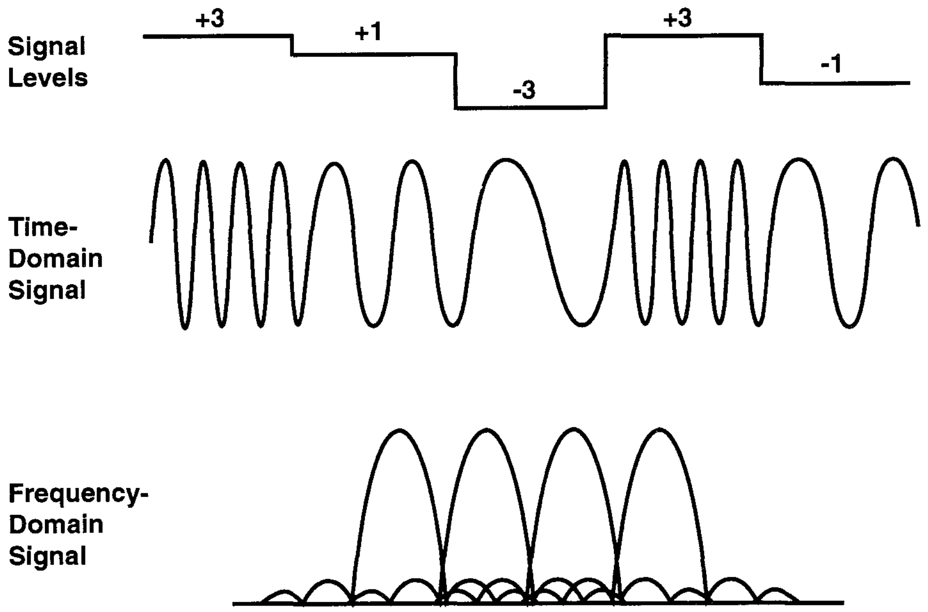


Figure 4-10 *- 4-ary FSK (4FSK) signals: signal levels, the time-domain representation of the signal, and the frequency spectrum of the signal.*

Interestingly the required SNR per bit *decreases* as the number of tones goes up. This is equivalent to saying that if one were to hold the data rate constant and decrease the baud rate by having multiple bits per symbol, then the performance of the system would improve. Additionally, if the reduction of the bit rate were continued indefinitely, and the number of tones in the symbol set were increased indefinitely, the bandwidth would increase to infinity, and the error rate would become arbitrarily small for Eb/No > ln2, which is equal to -1.6 dB. This is one limit of Shannon's formula. While infinite-bandwidth transmission systems are not useful, m-ary FSK is indeed useful for noisy channels. A general conclusion can be drawn: if we have noisy channels of large-enough bandwidth, m-ary FSK may be a good modulation choice, but the signal may be more susceptible to interference (due to the wider bandwidth). The fundamental limit of Shannon's theorem bears repeating: The smallest Eb/No for which it is possible to have an arbitrarily small error rate, given any possible modulation scheme is:

$$\frac{E_b}{N_0} \geq -1.6 \ dB \qquad \text{(Shannon's Limit)} \qquad (4\text{-}1)$$

Many modulation schemes will state that the error rate is small for a -5 dB or a -10 dB SNR. Such a measurement obviously cannot have been in terms of Eb/No, but has probably been measured as Signal-to-Noise (S/N) of the receiver in some bandwidth other than the Nyquist bandwidth. One method for measuring HF modems uses a 3 kHz BW for standardizing C/N (carrier power to noise power) measurements. Thus, some modem performance figures are stated based on this BW, and not based on the Eb/No. The C/N can be converted to Eb/No by the formula:

$$\frac{E_b}{N_0} = \frac{C}{N} B\ T_b \tag{4-2}$$

Where B is the noise bandwidth of the receiver in which the C/N was measured, and Tb is the duration of the bit time. BER measurements should always be based on Eb/No, and not on C/N, because if the noise bandwidth of the C/N measurement is not stated, *then the BER measurement is meaningless.*

PSK - Phase Shift Keying

In many systems, a more commonly used technique of modulation is based on changing the phase of the transmitted carrier, rather than the frequency. One good reason for using PSK is that a carrier can be more easily derived from the received signal. Then the received signal can be coherently demodulated by this received carrier, with 2PSK offering 3 dB. improvement in the signal-to-noise ratio (SNR) as compared to coherently demodulated 2FSK. 2PSK is an antipodal signaling scheme as opposed to the orthogonal 2FSK (see the section on orthogonal vs. antipodal signaling). In practice it would appear that the spectrum of a PSK-modulated signal is quite wide due to the sudden phase changes of the waveform. Figure 4-11 is the waveform of a 2PSK signal with unfiltered baseband data.

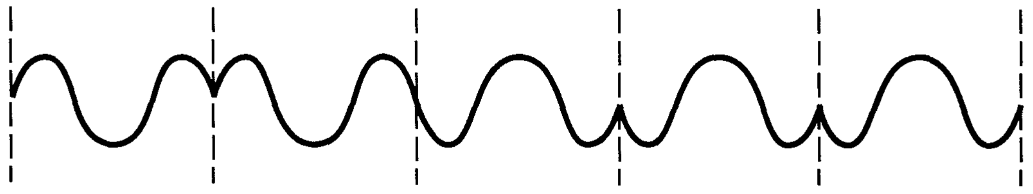


Figure 4-11 *- 2PSK carrier generated with unfiltered baseband data, corresponding to the sequence 1 1 0 0 0*

This waveform, while having constant output power, is excessively wide. If instead of transmitting an unfiltered baseband signal (which modulates the carrier), the baseband signal is first filtered with a raised-cosine filter response and then used to modulate (is multiplied by) the carrier, the waveform of figure 4-12 results. This waveform has a well controlled spectrum, but the filtering of the baseband data signal has imparted substantial amplitude changes onto the carrier. The signal for figure 4-12 could also have been generated by taking unfiltered (rectangular) baseband square data bits, using them to modulate a carrier, and then filtering the resultant signal at IF with a bandpass filter. In general, 2PSK cannot be both constant-amplitude and minimum bandwidth at the same time. If the minimum bandwidth carrier (as shown in figure 4-12) is passed through a class-C amplifier, the amplitude fluctuations will be reduced, but the transmit spectrum will be significantly widened by the process. If one were to then filter the wide transmit signal to minimum bandwidth, the filtering would re-introduce the amplitude variations.

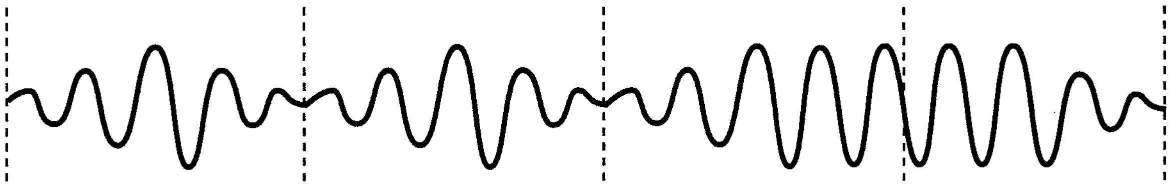


Figure 4-12 *- 2PSK carrier generated with filtered baseband data signal (such as raised cosine), corresponding to the sequence 1011.*

mPSK - m-ary Phase Shift Keying

There is no theoretical restriction to the size of the phase steps used in the phase modulation process (although there are some practical limits). In 2PSK, the two phases are placed as far apart as possible, 180 degrees. It is also possible instead to transmit 4 phase states, 0, 90, 180, and 270 degrees, which would convey two bits of information in each state. This type of modulation is called 4-PSK, or quadrature PSK, due to the 90-degree difference in phase positions. In the section on antipodal and orthogonal modulation, it was shown that 180-degree modulation is antipodal, and 90-degree modulation is orthogonal, so we can immediately deduce that there will be a 3-dB penalty in going from 2-PSK to 4-PSK. However, at the same time that the SNR is reduced, it must be realized that we can transmit twice as much information. So, with a 4-PSK signal, we could take our original data stream and produce two separate half-rate streams. We could then take these di-bits (which now occur at 1/2 the baud rate of the 2-PSK system) and use them to modulate the 4-PSK signal. Since the baud rate is half, the filter bandwidth can be halved, and we regain the 3 dB lost due to the smaller separation between phase positions. The BER vs. SNR for 2PSK and 4PSK (and OQPSK as well) are exactly the same.

It is possible to continue to decrease the phase spacing and incorporate more phase states. For example, we could use the phase states 0, 45, 90, 135, 180, 225, 270, 315 degrees. Then it is possible to transmit 3 data bits per state. This results in suffering another loss in SNR due to the closer phase states. However, the input rate is reduced from R to 2/3 R when going from 2 bits per baud to 3 bits per baud. Also, we do not gain back the same amount of SNR we lost by reducing the distance between the phase states. Thus, as m (the number of bits per baud) is increased, the required SNR for a constant bit error rate continues to increase. A good general conclusion is: m-PSK is well suited to narrow-bandwidth channels with good SNR ratios. For larger m, the modulation is poorly suited to channels where bandwidth is not so important and that have a poor SNR. Microwave point-to-point links can have good SNR's if engineered carefully. Multi-point HF, VHF, and UHF links probably are not well enough engineered to assure consistently high SNR. HF links usually do not have a good SNR.

OQPSK - Offset QPSK

One technique that works well to prevent the problem of transmitting a QPSK signal (with amplitude fluctuations) through a limiting amplifier (such as a class C) is to stagger the baseband data signals so that the two channels have data transitions that occur half-way apart in time from one another. This causes the phase transitions to occur twice as often, but with half of the phase change. Thus it prevents a 180-degree phase shift from occurring all at once, as happens in PSK. The 180-degree phase shift is what causes the amplitude to decay down to zero for the filtered signal. With offset-QPSK, there may be a phase change of 0 degrees or ±90 degrees at any one time. There is some amplitude droop during the 90-degree phase change. However, hard-limiting the signal does not radically alter the phase, so the spectral bandwidth is not appreciably increased if it is limited. This means that offset-QPSK will undergo substantially less bandwidth expansion when amplified in a class-C amplifier (a limiting amplifier) than QPSK undergoes. This offset-QPSK is usually referred to as OQPSK and has been used successfully in commercial satellites. Considering the common use of class-C amplifiers in amateur VHF and UHF transmitters, it may be a good modulation scheme for such equipment. Figure 4-13 shows the binary data streams for QPSK and OQPSK. Figure 14 shows the phase positions for both modulation formats. Figure 4-15 shows a QPSK carrier waveform and an OQPSK carrier waveform (but not corresponding to the data bits shown in figure 4-13).

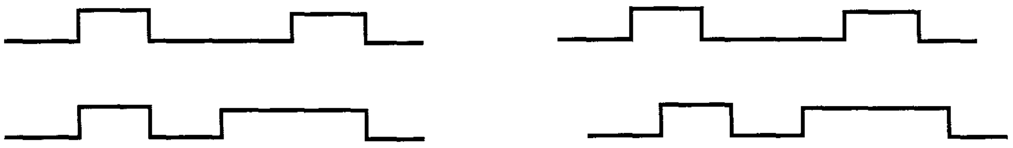


Figure 4-13 - *data streams are offset 1/2 bit time for OQPSK*

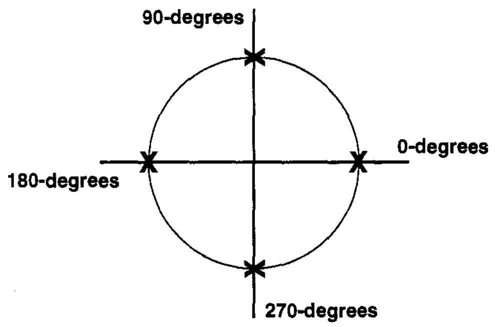


Figure 4-14 - *phase positions that QPSK and OQPSK can take,*
0, 90, 180, and 270 degrees. 270 degrees can also be considered as -90 degrees.
Four phase positions (states) means 2 bits per signaling state are conveyed.

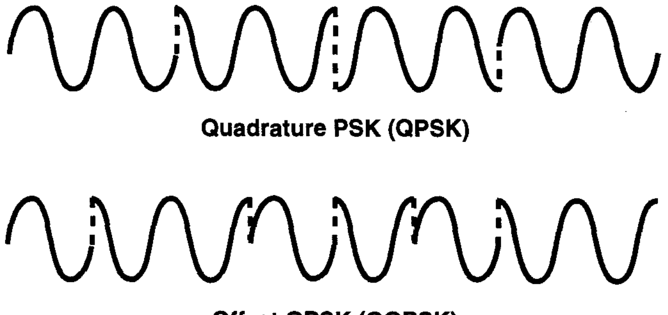


Figure 4-15 - *QPSK and OQPSK.*
OQPSK has phase transitions between every half-bit time, but they never exceed 90 degrees, resulting in much less amplitude variation of the bandwidth-limited carrier.
When applied to a class-C amplifier, QPSK results in significant bandwidth expansion, whereas OQPSK has much less bandwidth expansion.

Received Phase Ambiguity

In PSK receivers, and in many coherent receiver designs, it has been assumed that there is an ability to reconstruct a reference carrier for demodulation. This is not practical (or even possible) in many cases, and instead all that is possible is to recover a carrier that may match any of the several possible received states. Thus, the absolute phase of the received carrier is ambiguous with respect to the received state. A simple solution to this problem is to differentially encode the data to be transmitted, so that it is only necessary to examine the *phase difference* between two successive data symbols to resolve the actual value of the data. For example, in 2PSK transmission, it is not known which state is the 0-degree state and which state is the 180-degree state. It is possible to arbitrarily select one of them as the 0-degree state, but then there would be a 50% probability that the choice was wrong. Thus the data from the receiver would be inverted. It is shown in the section on coding that NRZI coded data is insensitive to inversion. The NRZI decoder can always reconstruct the original stream, whether the data were inverted or not. Thus, if NRZI coded data were used to modulate the transmitter, we do not need to concern ourselves with recovering the original carrier phase position. The receiver can choose either the 0-degree or 180-degree phases, and the correct data will ultimately be recovered. This function of including the coder and decoder is referred to as *differential PSK*, or *DPSK* for short, since in essence, the difference between the phase states is transmitted — an NRZI coder transmits the difference between successive data bits.

In DPSK, since the data are differentially coded, there will be error rate multiplication. This is due to the fact that if the first bit is wrong and the second bit correct, then after decoding by the NRZI decoder, both bits will be incorrect. Thus, DPSK will have poorer BER vs. SNR performance than PSK. Similarly, for QPSK, the received phase can be in any of 4 states, and the differential coding must cover the di-bit groups. In this case, if an error is made, there can be both a rotation of the carrier with respect to the correct position, or just a reversal of the phase. This results in either inverted data, or wrongly re-sequenced data. Thus, NRZI coding is not sufficient, but rather the differential coding must occur on the di-bit groups, so that both inversion and rotation can be resolved.

Performance of PSK modulation in the presence of AWGN

In the section on Additive White Gaussian Noise (AWGN), a computation of the BER vs. SNR for coherent antipodal 2-PSK was made. The result of that was that the SNR can be expressed as:

$$P_e = \frac{1}{2}\, erfc\left(\sqrt{\frac{E_b}{N_o}}\right) = Q\left(\sqrt{\frac{2E_b}{N_o}}\right) \qquad (4\text{-}3)$$

Where Eb/No is the signal to noise ratio per bit. In the section on Orthogonal signaling, we derived the probability of a bit error for orthogonal 2-PSK (or orthogonal 2-FSK) which is expressed as:

$$P_e = \frac{1}{2}\, erfc\left(\sqrt{\frac{E_b}{2N_o}}\right) = Q\left(\sqrt{\frac{E_b}{N_o}}\right) \qquad \text{(coherently demodulated)} \qquad (4\text{-}4)$$

For 4-PSK, it must be considered that each symbol conveys two bits, so we have to account for the fact that Eb/No = 0.5 * Es/No, the energy per bit is half the energy per symbol. Then the Symbol Error Rate vs. SNR can be expressed as:

$$P_s = erfc\left(\sqrt{\frac{E_b}{N_o}}\right)\left[1 - \frac{1}{4}\, erfc\left(\sqrt{\frac{E_b}{N_o}}\right)\right] = 2Q\left(\sqrt{\frac{2E_b}{N_o}}\right)\left[1 - \frac{Q}{2}\left(\sqrt{\frac{2E_b}{N_o}}\right)\right] \qquad (4\text{-}5)$$

Given that n = the number of bits/symbol, and m = the number phase states (i.e.: for n=3 then m=8) it can be shown for m-ary PSK that a reasonable approximation to the bit error rate vs. SNR can be described by:

$$P_e \approx erfc\left(\sqrt{\frac{nE_b}{N_o}}\,\sin\left(\frac{\pi}{m}\right)\right) = 2Q\left(\sqrt{\frac{2nE_b}{N_o}}\,\sin\left(\frac{\pi}{m}\right)\right) \qquad (4\text{-}6)$$

Figure 4-16 shows the BER vs. SNR performance of antipodal (coherent) 2PSK, orthogonal (coherent 2FSK) and non-coherent 2FSK demodulation. Figure 4-17 plots the BER vs. SNR for m-ary PSK, for several values of m. Note that as m grows, the required Eb/No increases (as shown previously), making m-PSK not as suitable for noisy channels. Figure 18 plots the BER vs. SNR for mFSK, for several values of m.

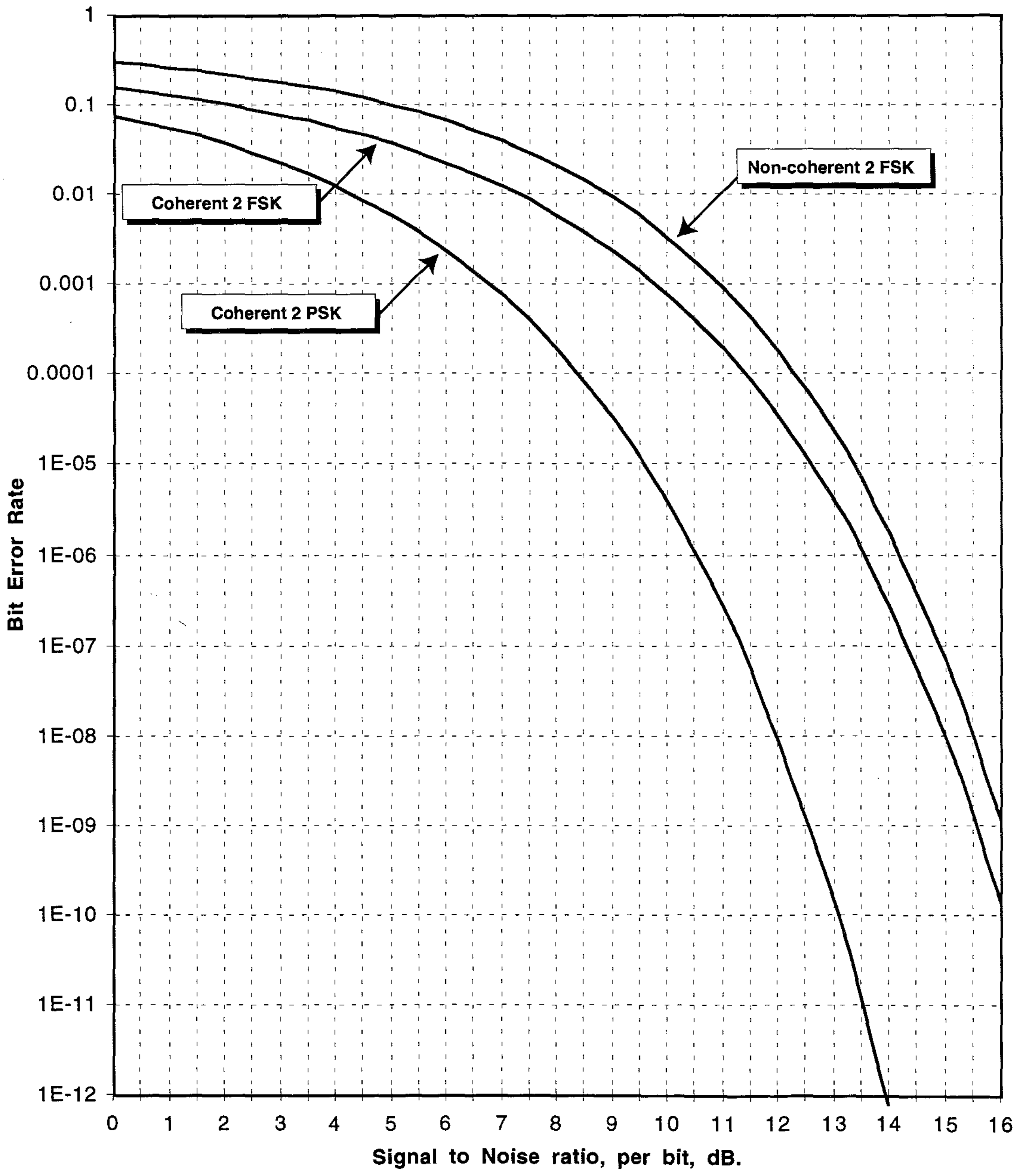


Figure 4-16 *- Bit Error Rate vs. Signal-to-Noise ratio for antipodal (coherent) 2PSK, orthogonal (coherent) 2FSK, and non-coherent 2FSK demodulation.*

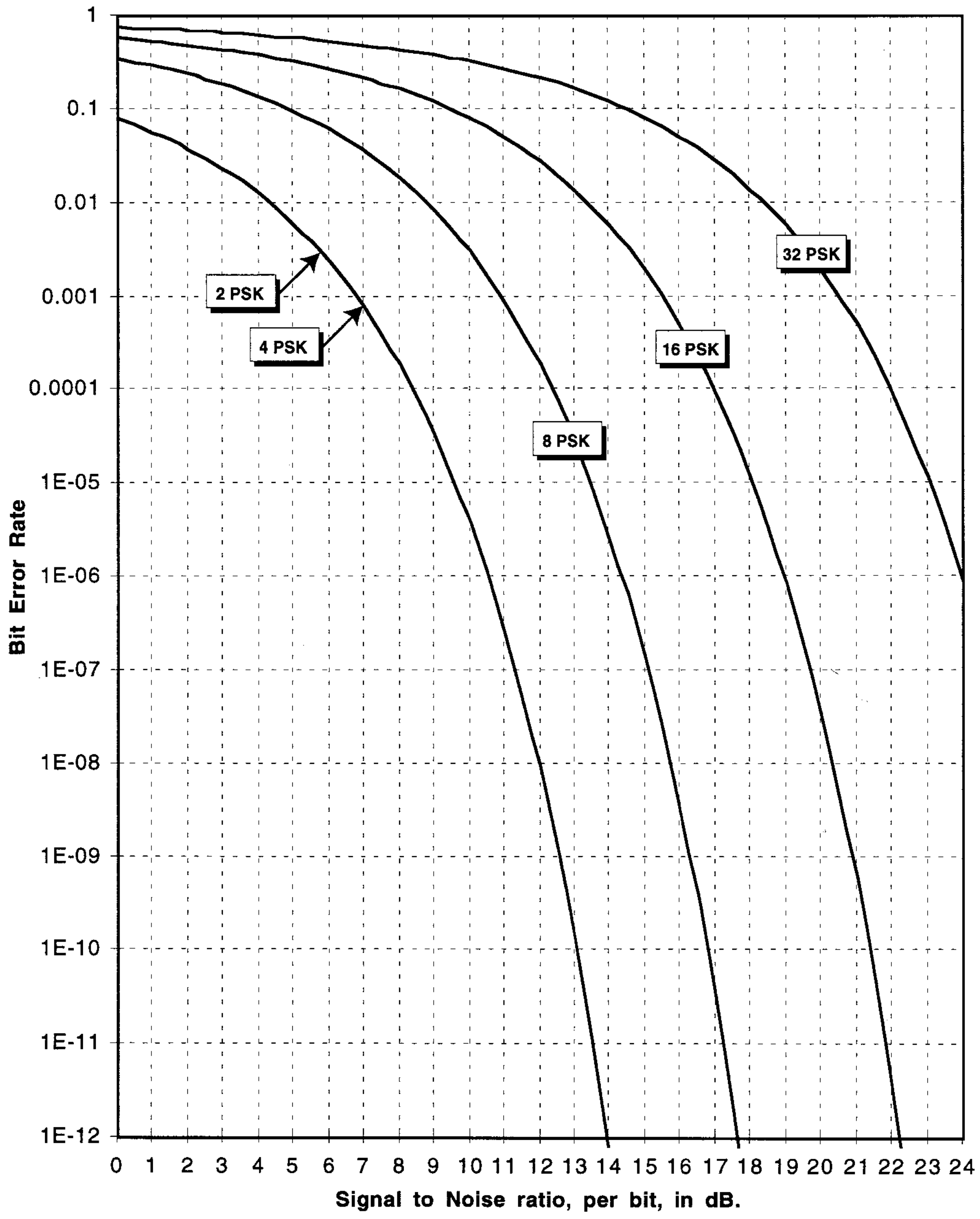


Figure 4-17 *- Bit Error Rate vs. Signal-to-Noise ratio for m-ary PSK, for several values of m. Note that as m grows, the required Eb/No increases (as shown previously), making m-PSK not as suitable for noisy channels.*

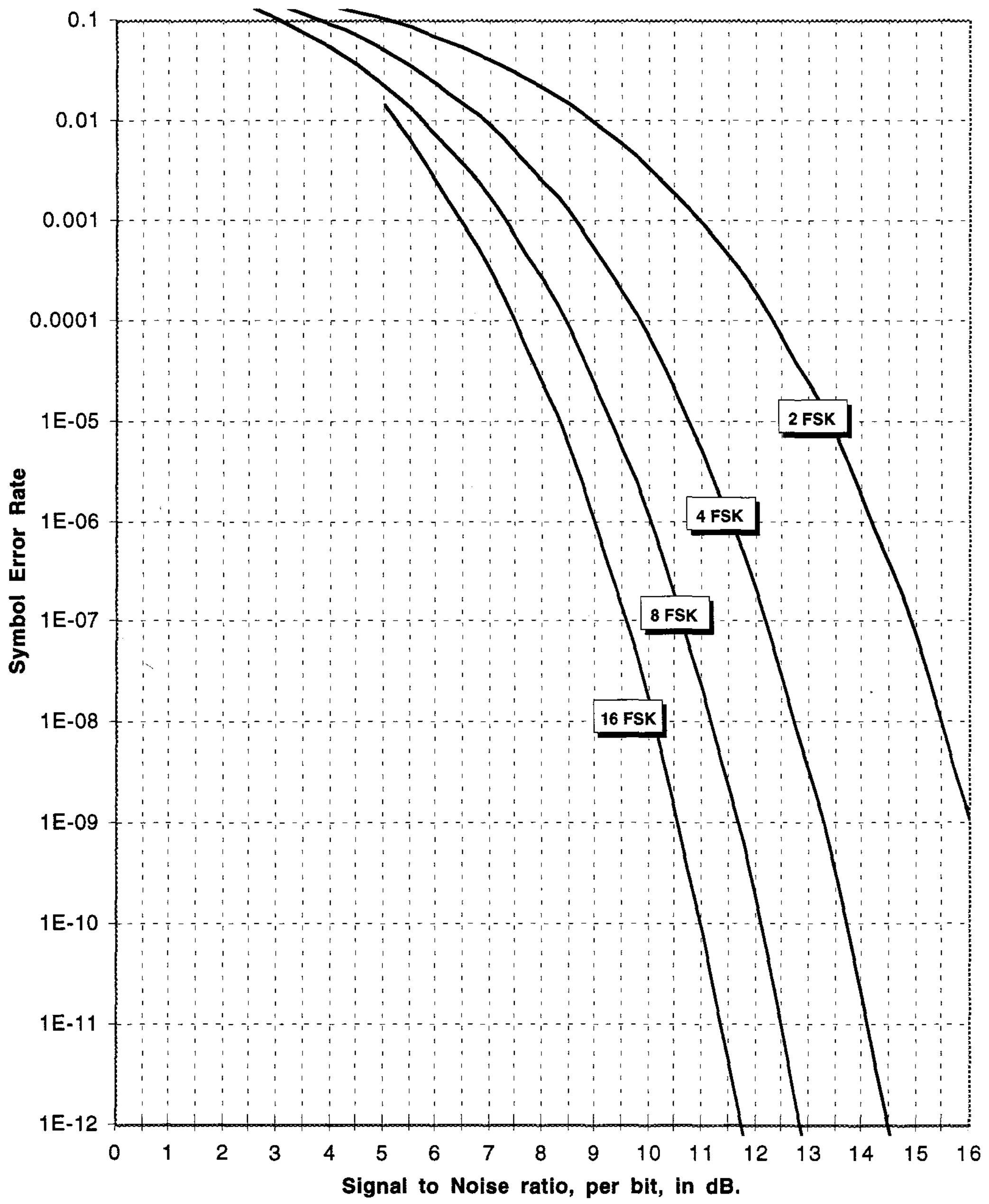


Figure 4-18 *- Symbol error rate vs. signal-to-noise ratio for non-coherent m-ary FSK for several values of m. Note that as m grows, the required Eb/No decreases, making m-FSK more suitable for AWGN noisy channels than m-PSK.*

ASK - Amplitude Shift Keying

Two level amplitude shift keying, or on-off keying (OOK) was originally used for amateur teleprinter work but rapidly discarded due to its susceptibility to interference in the off state. More generally, however, amplitude shift keying can be considered as the product of a baseband signal and a carrier signal in a mixer. If one assumes that the two binary logic states are +1 and -1 (rather than the one and zero that have been used in most of this text), then the operation is much more clearly seen. The +1 signal is a full-strength carrier, while -1 is a full strength carrier of the opposite phase. In this manner 2ASK (double sideband suppressed carrier amplitude modulation) and 2PSK are exactly equivalent modulations. In the case of four-level modulation, 4ASK is represented by four levels, +3, +1, -1, and -3. Then the numeric value indicates the amplitude of the signal, and the sign indicates the phase. Figure 4-19 illustrates a 4ASK signal.

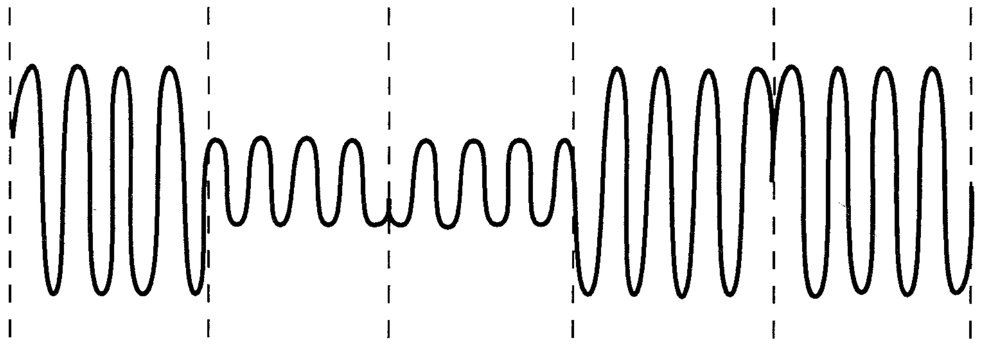


Figure 4-19 - *4 level amplitude modulated signal (4ASK), with signal states of +3, +1, -1, -3, +3 shown*

It is possible to transmit two amplitude carriers in quadrature to one another without any cross talk between them. Then each carrier can be independently modulated, in effect doubling the capacity within the same channel bandwidth. The two carriers are phased 90-degrees to each other so that they are orthogonal. This is called Quadrature Amplitude Modulation, or QAM for short. A 4QAM signal is comprised of two carriers, each with two signal states. It can thus be seen that 4QAM and QPSK are exactly the same modulation schemes. With more states, however, QAM becomes different from PSK, and much more efficient than PSK. An M=16 (16 state, or 4 bits per baud) QAM system is called 16QAM, and it exhibits a 4.14 dB improvement in BER vs. SNR as compared to a 16PSK system. As M increases, the performance advantage of QAM over PSK increases. Thus, QAM is the predominant modulation used in point-to-point high-density microwave communications systems. To compute the BER vs. SNR performance, QAM can be compared to PSK, for which the formulas were derived earlier. A good formula for comparing the power ratio needed for PSK to that needed for QAM (thus, the disadvantage of PSK) vs. the number of states, M is:

$$d = \frac{3M^2}{2(M-1)\pi^2} \tag{4-7}$$

Generally, ASK may not be good modulation choice on the HF bands due to the rapid, deep fading that is possible. The AGC circuitry needed to control the amplitude properly for demodulation may be difficult to implement, since it would have to track the fades carefully. Figure 4-20 is the block diagram of a QAM modulator.

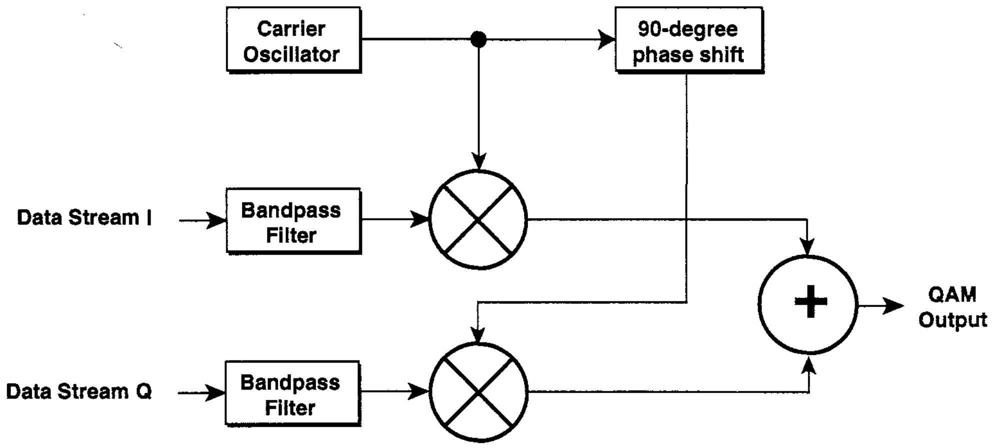


Figure 4-20 - Block diagram of Quadrature Amplitude Shift Keyed (QAM) modulator. The data streams I and Q may be 2-level (in which case QAM and QPSK are identical modulations), or they may be more than 2-level.

MSK - Minimum Shift Keying

In the literature, several definitions of MSK have been used. The one presented here is that defined by Pasupathy (1979). Earlier, it was shown that minimizing the phase transitions to 90 degrees (using OQPSK) resulted in less amplitude droop than a 180 degree phase change during QPSK (or 2PSK for that matter). It is also possible to eliminate the sudden phase changes during the modulation by gradually altering the phase, rather than changing it all at once. Let's assume that we start with OQPSK. If we used data bits with a sine-shape, rather than using rectangular data bits, then each data bit would have a mid-level value at the beginning and end of the bit time. See figure 4-21 for an illustration of the bits.

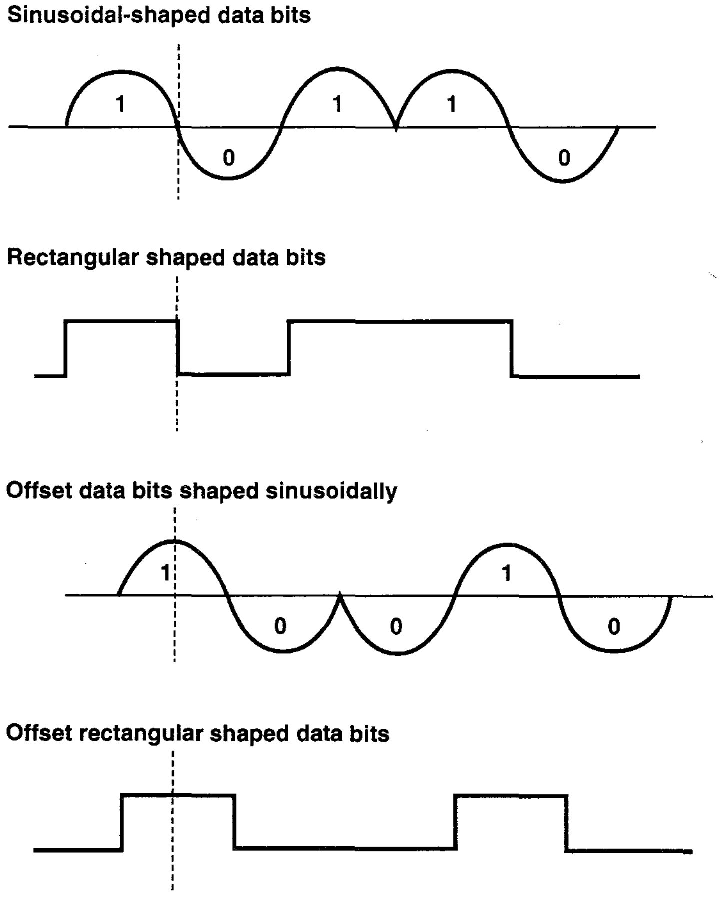

Sinusoidal-shaped data bits
1
0
1
1
0
Rectangular shaped data bits
Offset data bits shaped sinusoidally
1
0
0
1
0
Offset rectangular shaped data bits

Figure 4-21 *- sinusoidal-shaping of data bits for MSK transmission*

Then, the sinusoidal-shaped data bits are mixed with one carrier, and the offset sinusoidal-shaped data bits are mixed with the quadrature carrier. The resulting signals are essentially amplitude modulated. When we sum the power in the resultant signal, it is constant ($\sin^2 + \cos^2 = 1$). However, the phase gradually changes from that completely aligned in the center of one bit to that completely aligned in the center of the next bit. The result is a constant-envelope signal with gradual phase transitions. Interestingly, the output also looks exactly like an FSK signal with frequency shift equal to 1/2T (coherently orthogonal). Thus, it is possible to generate an MSK signal either with FSK or with PSK techniques. Generally some coding of the data is done to resolve phase ambiguity. This would have to be considered if FSK methods were used to generate the signal. The spectrum of the MSK signal, as contrasted to QPSK (and OQPSK, which has the same spectrum as QPSK) can be determined by the techniques that were discussed in the section on baseband filtering. The spectrum is derived by Fourier transform of the baseband pulse shape. Figure 4-22 illustrates the power spectral density of MSK and QPSK signals where baseband filtering has not been used on the signal. An advantage of MSK over PSK is the narrower transmit spectrum when using unfiltered baseband data (since MSK, in essence, uses shaped pulses).

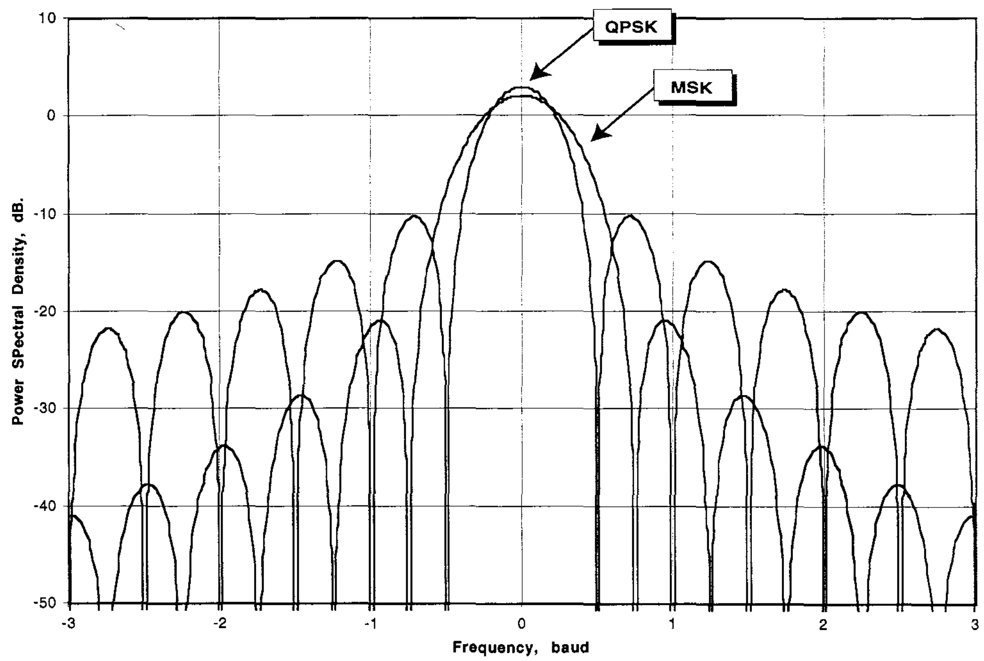


***Figure 4-22** - Power Spectral Density for MSK and QPSK, when there has been no baseband or RF filtering applied. MSK exhibits somewhat narrower spectrum than QPSK under these conditions.*

Suppose a linear transmitter (class A, or AB) is available. By eliminating the spectrum beyond the first lobes, an even narrower transmit spectrum can be generated with OQPSK modulation when appropriate raised-cosine baseband filters are used. So, MSK might not be as appropriate for use in crowded HF bands, unless it is also filtered carefully. Figure 4-23 shows the power spectrum of raised-cosine filtered QPSK for comparison with figure 4-22.

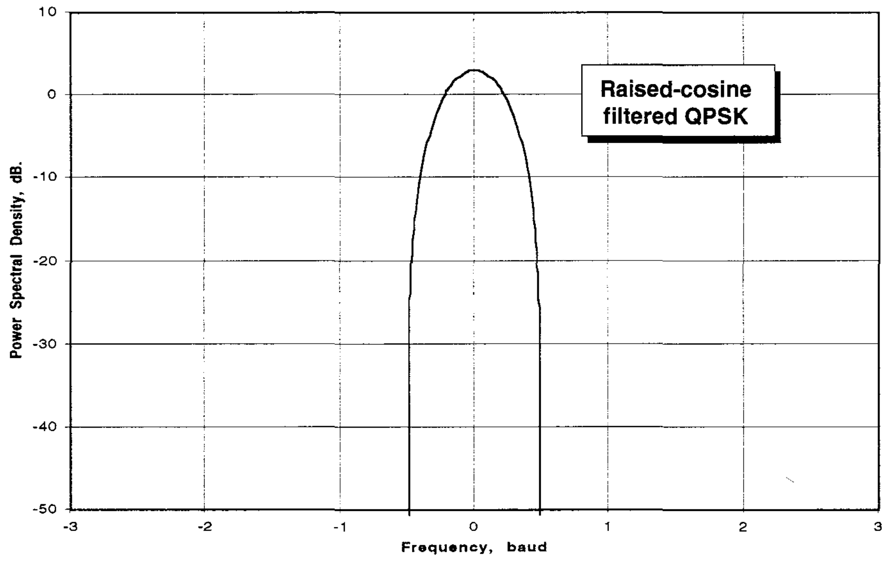


Figure 4-23 *- Power Spectral Density of Raised-cosine filtered QPSK modulation.*
Compare to unfiltered QPSK and MSK in figure 4-22. The cut-off is 0.5 of the data rate in bits per second.

Since the MSK signal could be generated with either PSK or FSK circuitry, it can also be demodulated with similar flexibility. The performance of the demodulator will vary depending on whether coherent or non-coherent techniques are used. If demodulated like a PSK receiver, then the MSK signal can achieve the same Eb/No performance as the QPSK receiver. If demodulated like an FSK receiver, then the performance will be poorer, just like the penalty paid to demodulate FSK as compared to PSK. Further more, non-coherent FSK demodulation performance will be poor due to the close carrier spacing — that is, the signaling will be non-orthogonal when demodulated non-coherently.

To recover carrier and bit timing from an MSK signal is relatively easy. If the signal is multiplied by itself (squared) then an equivalent FSK signal with 1/T spacing is produced. This signal possesses strong spectral lines at twice the mark and space frequencies, which can be bandpass filtered then divided by two to produce clean signals. The one half-baud clock is the product of the two filtered, divided signals. The two reference carriers are the sum and difference of the divided signals.

OFDM - Orthogonal Frequency Division Multiplexing

In cases where bandwidth is not an overriding concern, and where path propagation effects cause serious degradation of the performance of PSK transmissions, it may be desirable to lower the baud rate significantly. It has been shown that signaling at 300 baud sometimes exceeds the capability of a poor HF path. In this case, it may be necessary to generate a signal based on a number of carriers. This is called Orthogonal Frequency Division Multiplexing (OFDM) and is also sometimes called Discrete Multi-Tone (DMT) modulation. Actually, several different approaches are generally grouped under the label OFDM.

OFDM based on m-ary FSK

One approach to this was described previously, m-ary FSK. There are some modifications as to how much information can be transferred using mFSK; it is possible to code the FDM channels with FSK carriers in a number of different ways, and several will be discussed.

In the simplest method, only one of the m-carriers is turned on at any instant in time. This unique carrier conveys which symbol is being transmitted. The number of bits being conveyed, n, is equal to:

$$n = \log_2 m \tag{4-8}$$

That is, one of four carriers 'on' means one of four states is being transmitted, or conversely, 2 bits of information are being communicated. The previous discussion of m-ary FSK showed how this specific coding scheme is particularly efficient in terms of low BER vs. Eb/No for large m. The crest factor for this specific coding scheme is 1:1 — in other words, the peak power is equal to the average power (that of a single-tone sinewave). A older British military modem called 'Picolo' used this modulation method, with m = 32 (and thus n = 5).

Another way to code the carriers is to use all of them independently. For example, consider using m-carriers to convey m-bits of information. Then the possibility that all m-carriers are 'on' is finite (1 in 2^m). The possibility that all m-carriers are 'off' is also finite (1 in 2^m). In this case, the information rate, n, is the same as m, or n = m. However, the power is spread across all the carriers, it is not applied to just a single carrier at a time. If we assume that each individual carrier maintains a constant power, then the output power of a symbol depends on the number of carriers being transmitted at any instant in time. The bit error rate for each individual carrier should be constant and equal. The peak power is increased due to the crest-factor of summing m- different carrier sinewaves. The best possible summation yields a crest factor of 4.6 dB, while an uncontrolled summation (all carriers phase aligned) results in a crest factor of 14 dB (for m = 5 carriers). With careful phasing about 6 dB crest factor is achievable for even large values of m (see equation 4-9, below) In the average-power limited case the BER suffers a power penalty of roughly 10*log(1/m) since m-carriers are being transmitted. The power divides among the carriers equally, as opposed to one carrier being transmitted. For m=2, each carrier is -3 dB. compared to the one-carrier at a time (each is half the total power). Table 4-1 indicates the loss due to transmitters that are peak-power and those that are average-power limited vs. m, the number of carriers. In practice, many transmitters are not very linear unless operated under the peak power rating, and may overheat unless operated under the average power rating. Usually, the peak power limitation is more severe. Additionally, when operating near the maximum legal power level, current US amateur regulations limit the peak-envelope-power output.

Number of simultaneous carriers, m.	Power per carrier when average-power limited.	Power per carrier when peak-power limited.
1	0 dB.	0 dB.
2	-3 dB.	-6 dB.
4	-6 dB.	-12 dB.
5	-7 dB.	-13 dB.
8	-9 dB.	-15 dB.
16	-12 dB.	-18 dB.
32	-15 dB.	-21 dB.

Table 4-1 *- mFSK when all m-carriers simultaneously carry information. The loss of power per carrier compared to 1-carrier-at-a-time depends on whether the transmitter is peak- or average- power limited.*

The information rate for m-at-a-time as compared to 1-at-a-time is higher by the ratio m/log2(m). So, we have to choose equivalent information rates, n and m, between the two schemes to calculate the power penalty of the m-at a time as compared to the previous one-carrier-at-a-time scheme. At a given bit-rate, n, the bandwidth occupied by the m-at-a-time scheme is considerably less, but the cost is significantly poorer BER vs. SNR.

Combinations of the approaches described above are possible, with coded groups of carriers representing the symbol set, where only a subset of the carriers are turned on at any time. Such an approach is in a family of modulations called Coded OFDM, or COFDM for short. COFDM is discussed in Sari, Karam, and Jeanclaude (1995). Karn (1995) has proposed one such approach based on orthogonal symbol sets generated by Walsh functions, where one half of the carriers are on at any given time, and all symbols are orthogonal. The power penalty lost due to having many carriers on is partially compensated for by the coding gain of the orthogonal set. However, this modulation scheme may prove to be robust in the presence of CW interference, if the interferer can be marked as a bit erasure. Additional coding gain can be achieved when defective bits can be identified, even if their value is not known. More over, if the Hamming distance of the code is large enough, then the code will be resistant to having some of the individual carriers jammed.

Provided that bandwidth is of little concern, the one-carrier at a time approach should provide the best BER vs. SNR of these schemes due to optimum use of the transmitter power. However, this approach may not be optimal from an interference susceptibility point of view.

OFDM based on ASK

Another approach is to simultaneously transmit a number of carriers, each carrying a fraction of the data bits. Each carrier is modulated by a subset of the data bits. For example, if the data rate is 2400 baud, and the signaling rate is 150 baud, then we could generate 2400/150 = 16 channels, each of which would be an ASK or PSK channel. Thus, this signaling system would have slots for 16 carriers, all of which would be turned on at the same time. For HF propagation, the phase of the received signal is not well known, and varies, so the carriers should be spaced at least 1/T of the baud rate, or 150 Hz apart, to assure orthogonality. This system would occupy a spectral width of approximately 16 * 150 = 2400 Hz. Figure 4-24 shows the spectrum of such an OFDM system. The baseband filtering of each of the channels will determine the spectrum of each channel.

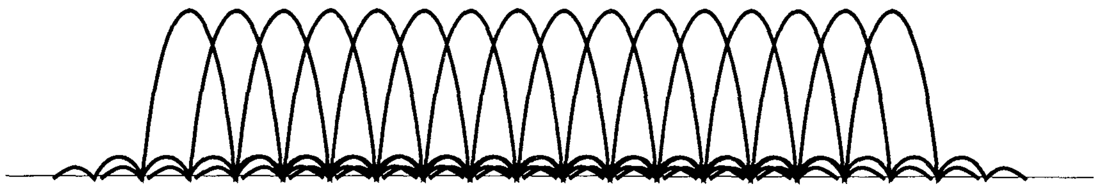


Figure 4-24 *- spectrum of 16-channel OFDM ASK system.*

The implementation of a OFDM system is really only practical with the use of digital signal processing (DSP). The 16 baseband channel filters can be implemented via a bank of FIR filters, but this is computationally slow, and so the use of unfiltered rectangular data bits may be considered for use. One difficulty faced with such a system is that although rectangular pulses don't interfere with other pulses in time, they have large spectral sidelobes, and so extend the spectral width when the number of carriers is small. Well filtered pulses on the other hand have ringing for many bit times, and that can be problematical with so many carriers. Recent work on computationally efficient optimum pulse shapes for OFDM has been reported where both spectrally-clean and temporally-limited properties of the pulse are good at the same time.

In the case where the channel phase does not vary, the general design approach is to construct quadrature ASK signals in each channel. If the phase is uncontrolled, we may have to halve the data rate (due to doubling the spacing of the carriers), and only non-coherent demodulation of the signal. The advantage of using ASK is that the transmit signal generation can be done using the Inverse Fast Fourier Transform (IFFT), and the receive signal demodulation can be done by using a fast Fourier transform (FFT) of the data set, thus avoiding a bank of filters. Since on HF there may be different (and varying) delay for each different channel, it may not be possible to reconstruct coherence between two carriers. Thus, use of this scheme on HF should probably consider non-coherent demodulation of the ASK signal (non-coherently orthogonal) which requires twice the carrier spacing of coherently-orthogonal signaling. Practically, the carriers may have to separated even further due to adjacent-channel or adjacent-bit-time interference between the carriers.

The phase of the individual transmit carriers should be carefully controlled or else the crest-factor of the resultant time-domain signal will be excessive. The crest-factor of a signal is the maximum voltage divided by the RMS voltage. If the crest-factor is high, then the transmitter must be very linear, which can be expensive. For example, if the crest factor of the signal were 18 dB., then the ratio of the peak signal power to average signal power (RMS voltage) would be 63.1 to 1. Thus, to transmit 100 watts average power, the peak power capability of the transmitter would have to be 6,310 watts. Staggering the phases properly can limit the crest factor to about 6 dB, or a peak to average power ratio of 4 to 1. Boyd (1986) shows that if the phases of N harmonically-related carriers are initially aligned, then the peak power is proportional to $\sqrt{2N}$. However, if the starting phases are given careful initial phase offsets, then the crest factor can be limited to about 4.6 dB. The offsets of these phases were suggested by Newman (1965) and are specified by the formula:

$$\varsigma_k = \frac{\pi(k-1)^2}{N} \tag{4-9}$$

Where k is the particular carrier (numbered from 1 to N), N is the total number of carriers, and delta-sub-k is the phase of the kth carrier. Thus the starting phase of the 1st carrier is zero, the 2nd carrier starts at pi/N radians, the 3rd carrier starts at 4pi/N radians, etc.

When rectangular pulses are transmitted, the matched-receive -baseband filter is an integrate-and-dump filter. It can be shown that a real-to-complex FFT in the receiver is equivalent to demodulating each sub band separately, and then performing an integrate-and-dump baseband filter. If the differential delay in the channels is large, then inter-symbol-interference will be introduced. This effect is minimized by making the symbol rate as slow as possible.

AFSK - Audio Frequency Shift Keying

Audio frequency shift keying is defined as FSK modulation of an audio-frequency carrier, and then the use of that carrier to modulate an FM or an AM transmitter. The introduction of an FSK modulated carrier to a single-sideband (SSB) transmitter is not AFSK modulation. This can be seen first by noting that SSB is a completely linear modulation scheme, and is in fact nothing more than heterodyning the audio carrier up to the channel frequency. Nothing about the characteristics of the audio carrier (except the frequency) are changed. However, when an audio FSK carrier modulates an FM (or an AM) transmitter, significant changes in the spectrum of the RF carrier occur. To illustrate this, figure 4-25 shows the spectrum from a low-deviation audio FSK carrier as applied to an SSB and an FM transmitter. The FM transmitter produces a carrier, and the audio tone becomes a subcarrier on the FM carrier. As a results many FM modulation sidebands are produced, depending on the deviation ratio. In the SSB case there is no carrier, so the FSK audio tone introduced does not become the subcarrier of anything, it becomes the carrier itself. The signal produced by a directly-FSK modulated transmitter, and audio-modulated SSB transmitter are identical (assuming a good quality SSB transmitter, and good quality audio source).

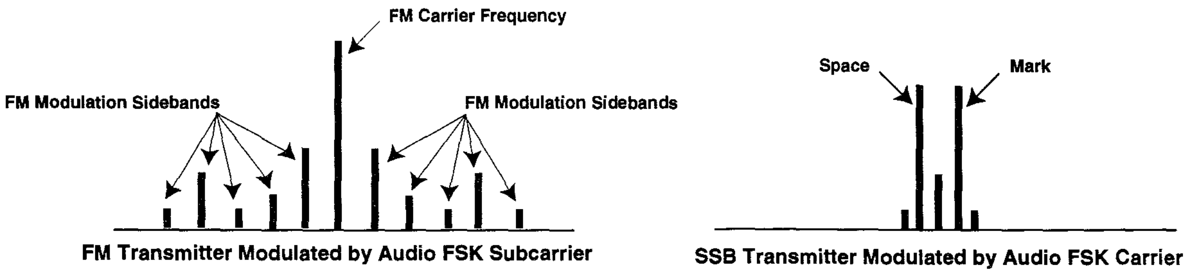

FM Carrier Frequency
FM Modulation Sidebands
FM Modulation Sidebands
FM Transmitter Modulated by Audio FSK Subcarrier
Space
Mark
SSB Transmitter Modulated by Audio FSK Carrier

Figure 4-25 - application of audio FSK tone to FM and SSB transmitters. The audio tone on an FM transmitter produces AFSK, while the audio tone on the SSB transmitter produces FSK.

To determine the performance of audio FSK on an FM transmitter, the signal-to-noise ratio of the demodulated FM signal must be determined. Then, the Eb/No of the resultant audio baseband signal can be calculated. In the case of FSK, the Eb/No calculation is performed directly on the RF channel frequency, not on the audio baseband signal. So, the performance gain/loss of the conversion to baseband process must be approximated.

For a sine-wave signal, and assuming that the carrier to noise ratio is greater than 10 dB., an FM system has an output signal to noise ratio (compared to an AM system) of:

$$\frac{S_o}{N_o} = 3\beta^2 \frac{C}{N} \qquad (10)$$

Where So/No is the signal to noise ratio, C/N (the carrier to noise ratio) = Sc/Nc, and beta is the modulation index (deviation / modulating frequency). $3\beta^2$ is the FM processing gain. This assumes no pre-emphasis, which may give perhaps another 1-2 dB improvement for this example. If the audio subcarriers are 1200 Hz, and 2200 Hz, and the data rate is 1200 baud, then the audio bandwidth extends up to about (2200+1200) = 3400 Hz. For 3.0 kHz. deviation, beta = 3000/3400 = 0.88, yielding an FM processing gain (vs. AM) of about 3.7 dB. Using a much-higher deviation ratio improves the S/N, but at the expense of a large increase of the bandwidth of the FM signal.

For an AM system with 100% modulation, the output S/N is the same as the carrier to noise ratio, C/N. Therefore, the performance of the AM-AFSK system should be worse than the FM-AFSK system by the processing gain, or about 3.7 dB. In reality, many common amateur AFSK demodulators based upon audio phase locked loops demonstrate much poorer performance than a direct baseband FSK demodulator.

Summary

Many modulation schemes have been developed to solve a number of different problems. Each method has advantages and disadvantages for different applications. Much of the research work has focused on maximizing the data rate while minimizing the bandwidth of channels with good signal to noise ratios. In the amateur radio service, this objective does not always align well with the channel characteristics, and some common modulation schemes are not very appropriate to this application. The following table (table 2) summarizes several amateur applications, and comments on some appropriate modulation formats. A concise tutorial on bandwidth, channel efficiency and coding can be found in Skar (1993).

Application	Modulation Format	Comments
VHF / UHF channelized point-to-point	FSK	Simple circuitry, compatible with class-C amplifiers, moderate performance.
	OQPSK	More complicated circuitry, compatible with class-C amplifiers, excellent performance with optimum demodulator.
	MSK	Most complicated circuitry. Fair performance with non-coherent FSK detection, excellent performance with optimum demodulator.
	FM-AFSK	Poor performance for narrowband FM, implementation dependent. Compatible with most available equipment, simple.
VHF/UHF channelized multipoint	FSK	Simple circuitry, compatible with class-C amplifiers, moderate performance. Forgiving of frequency and frequency shift errors among multiple stations.
	FM-ASFK	Poor performance for narrowband FM, implementation dependent. Compatible with most available equipment, simple. Forgiving of frequency and deviation errors among multiple stations.
HF non-channelized	CPFSK	Simple circuitry, adequate spectrum, poor to fair performance.
	mFSK	Possibly complex circuitry . Can exhibit best Eb/No under AWGN conditions. Wider bandwidth than other methods.
	PSK	Good for lower baud rates, poor for high baud rates due to multi-path.
	OFDM	Good for higher baud rates on HF, more complicated circuitry, probably requires DSP.
Satellite	PSK	Excellent performance for high baud rates, requires linear amplification to preserve narrow bandwidth.
	OQPSK	Compatible with class-C amplification, excellent performance.
	MSK	Most complicated circuitry. Compatible with class-C amplification. Excellent performance if optimum demodulator is used.

***Table 4-2** - Modulation Formats for some Amateur Radio Applications*

Chapter 4 - Reference

Boyd, Stephen. "Multitone Signals with Low Crest Factor." IEEE Transactions on Circuits and Systems, VOL. CAS-33, No. 10, October 1986, pg 1018-1022

Karn, Phil - KA9Q. TAPR HF Special Interest Group listserv (ftp:\\ftp.tapr.org\tapr\SIG\hfsig\mail_archive), 1995.

Newman, D.J. "An L1 extremal problem for polynomials." Proc. Amer. Math Soc., Vol. 16, pg 1287-1290, Dec. 1965

Pasupathy, Subbarayan. "Minimum Shift Keying: A Spectrally Efficient Modulation." IEEE Communications Magazine, July 1979, pg. 14-22.

Sari, Hikmet, Georges Karam, and Isabelle Jeanclaude. "Transmission Techniqes for Digital Terrestrial TV Broadcasting." IEEE Communicatoins Magazine, February 1995, pp 100-109.

Skar, Bernard. "Defining, Designing, and Evaluating Digital Communications Systems." IEEE Communications Magazine, Nov. 1993, pp 92-101.

Vahlin, Anders and Nils Holte. "Optimal Finite Duration Pulses for OFDM." IEEE Globecom 94, paper 7.7

Chapter

5

Frequency and Impulse Response of Optimum Modem Filters

A key question that arises in the design of a modem is how to determine the 'optimum' frequency response of the system. This frequency response has to take into account all of the frequency responses of the entire system, from the digital serial transmit data stream through to the digital receive digital data stream. There are a number of places where filtering of the analog data signal can occur. In the design of a modem, we hope to be able to control tightly the overall response. One way to do this is to make some parts that are not easily controlled negligible, and then carefully to shape the frequency response where control can be exercised. This is a very common sense approach to the problem.

The place where it is easiest to control the frequency response is at the baseband — that is the portion of the data stream that is completely at audio frequencies, and not containing any modulated subcarriers. This is because the relationship of the filter function to the overall channel response requirement is most obvious; in fact, it is the same function. Filtration at the Intermediate Frequency (IF) with a crystal filter is also possible, but is generally more difficult to control precisely. It is possible to purposely set the frequency response of the IF filter to be wider than necessary. In this case it may have an insignificant effect on the desired channel response. Some modulation schemes such as FSK result in a non-linear relationship between the IF response and the baseband response, but this can be small if the deviation index is low. More linear modulation schemes, such as PSK and ASK, result in a very straightforward mapping between the IF response and the baseband response. Thus, it is possible to optimally filter the received data signal prior to demodulation to baseband. However, it is normally easiest to do the filtering at baseband.

For the present time, it will be assumed that the frequency response of the IF filters, RF components, mixers, etc., and of the propagation path are all flat, with no phase delay. Given this, we can focus on the baseband response requirements.

How is the optimum frequency response determined? There are two basic requirements to satisfy when filtering the analog data stream:

1. The excess bandwidth should be minimized. This reduces the noise that is transmitted through the filter and improves the signal-to-noise ratio applied to the slicer circuit.

2. The inter-symbol-interference (ISI) should be minimized. ISI is essentially the ringing of a filter long past the time that an original data bit was applied to it. All filters ring (including digital FIR filters), but the ringing can be controlled to minimize deleterious effects. The ringing causes energy from previous data bits to add to the current data bit thus reducing the eye opening and degrading BER performance.

It is not possible to reduce the excess bandwidth to zero with a realizable filter, and we must be cautious in the design of the filter to reduce the ISI. To see why this is true, it is necessary to examine the properties of filters in both the time and the frequency domains. Prior to doing that, however, we will discuss an important relationship between the two — *Fourier Transform.* The Fourier Transform is a mathematical relationship that lets us convert, subject to some restrictions, a view of *frequency* into a view of *time*, and vice versa.

If we examine a digital data stream of some data rate, B, say 9600 bits per second (bps), it can be shown that the highest needed frequency for transferal of that data is 4800 Hz. This is obvious if starting with an alternating one-zero repeating pattern, and then filtering the resultant squarewave of all the odd-harmonics. What is left is a sinewave at 4800 Hz. So, the first approach to filtering the data signal might be to generate a so-called brick-wall filter. This is one that has a flat frequency response from DC to 4800 Hz., and then has zero response above 4800 Hz. Remember, this is a baseband filter. Although we know that this filter would have a minimum of excess noise bandwidth, how would this filter ring in the time domain?

One way to analyze the time-domain response of a filter is to look at its impulse-response. The impulse response is the time response of a filter to an infinitely-narrow impulse of unity energy magnitude. Now, it is not practical to generate such an impulse, but it does provide considerable convenience in analyzing filters. In fact, it is so convenient that we will come to rely on it to describe almost all useful modem filters.

Appendix C describes the Discrete Fourier transform, and some useful notations for complex numbers. Appendix D describes convolution and correlation, and the relationship between the impulse response and the frequency response. They may be useful to review prior to proceeding.

Optimum filter criteria

Much of the information needed to construct an optimal channel response for our system is based on the frequency and the time properties of filters. In many linear modulation systems, and in this example, all of the filtering can be done at baseband. The following criteria can be set for the overall optimum filter function:

1. The channel response must be symmetrical around 1/2 the data rate (in linear amplitude terms). This is known as Nyquist's second theorem, the vestigial symmetry theorem.
2. The response should be linear phase (we won't discuss this, except to say that there is a mathematical proof of this requirement).
3. The response should cross through the 1/2 amplitude point (symmetry) at one-half of the data symbol rate, in baud. This is known as Nyquist's first theorem, the minimum bandwidth theorem.

One other criteria not discussed is for spectrum compensation. Since we are transmitting a square data bit, and it is desired to have the received spectrum match the filter response shape, it is necessary to account for the fact that the spectrum of a square wave at the input to the transmitter filter has a roll off of sin(x)/x. Therefore, the channel response must be multiplied by the inverse of this roll-off, or x/sin(x), leading to rule 4:

4. The response must be additionally multiplied by:

$$\frac{\pi x}{\sin(\pi x)}$$

Where x is expressed as a fraction of the baud rate (from 0 to 1).

Another criteria can be set that will help minimize the amount and amplitude of ringing away from the data bit:

5. The filter transition from the passband to the stopband should be gentle, not abrupt. The gentler the transition, the lower the amplitude of the ringing.

One last criteria that is optional is to minimize the zero-crossing jitter of the waveform. This sometimes helps some clock recovery circuits to operate better. However, it is not necessary to have this minimal jitter criteria, since the average of all the zero-crossings are still at the right place. A correctly designed clock-recovery circuit will function properly even in the presence of significant zero-crossing jitter.

6. To minimize the zero-crossing jitter, the impulse response should cross 0.5 amplitude at the plus and minus one-half bit time.

There is a general class of filter transfer functions meeting the above requirements and used exhaustively in commercial digital microwave transmission systems. They are known as raised-cosine filters. We will investigate the properties of this class of filters through plotting the frequency response, the impulse response, and the eye patterns of data that have been passed through these filters. There are an infinite number of possible filters. They range from high-alpha gentle-sloped filters, having tame ringing but wide frequency response, to low-alpha filters which can have wild ringing but are close to as narrow a frequency response as it is possible to transmit the data. Other derivations of the filtering function are described in Feher (1987) and Bingham (1988).

Raised-Cosine Filter Responses

The raised-cosine family of filters meet the above five criteria (and the alpha=1.0 meets the sixth criteria). The name comes from the equation that generates the amplitude response shape. The formula for generating the amplitude vs. frequency is:

$$\begin{aligned}
&= 1 && \textit{from DC to } (1-\alpha)\frac{f_s}{2} \\
&= \cos^2\left(\frac{\pi}{4\alpha f_s}\left[2f - (1-\alpha)f_s\right]\right) && \textit{from } (1-\alpha)\frac{f_s}{2} \textit{ to } (1+\alpha)\frac{f_s}{2} \qquad (5\text{-}1) \\
&= 0 && \textit{from } (1+\alpha)\frac{f_s}{2} \textit{ to } f_s
\end{aligned}$$

where fs is the symbol rate of the data, expressed as hertz, and f is the particular frequency point that we are evaluating.

Note that for two-level modulation, the baud rate and the data rate inbits-per-second are the same. The factor alpha is known as the channel roll-off factor, and it determines how quickly the frequency response rolls off. An alpha of zero is the previously postulated brick-wall filter, where the roll-off occurs as abruptly as possible. In contrast, an alpha of one is the gentlest possible roll-off and the one with the widest spectrum. The range of values that alpha may take is from zero to one, but filters with an alpha value below about 0.25 are difficult to make in practice. Setting alpha=1.0 minimizes the zero-crossing jitter, but increases the spectral width of the transmitted signal compared to lower alpha values.

Figure 5-1 is a plot of the frequency response of several raised-cosine filters. They all pass through the 0.5 amplitude point at one half the bit rate, and all are symmetrical about the one half bit-rate frequency. Figure 5-2 is a plot of the raised cosine filters after adding compensation for the sin(x)/x roll-off of the transmit square wave signals. Notice the peaking that exists between 0.3 and 0.5 of the baud rate.

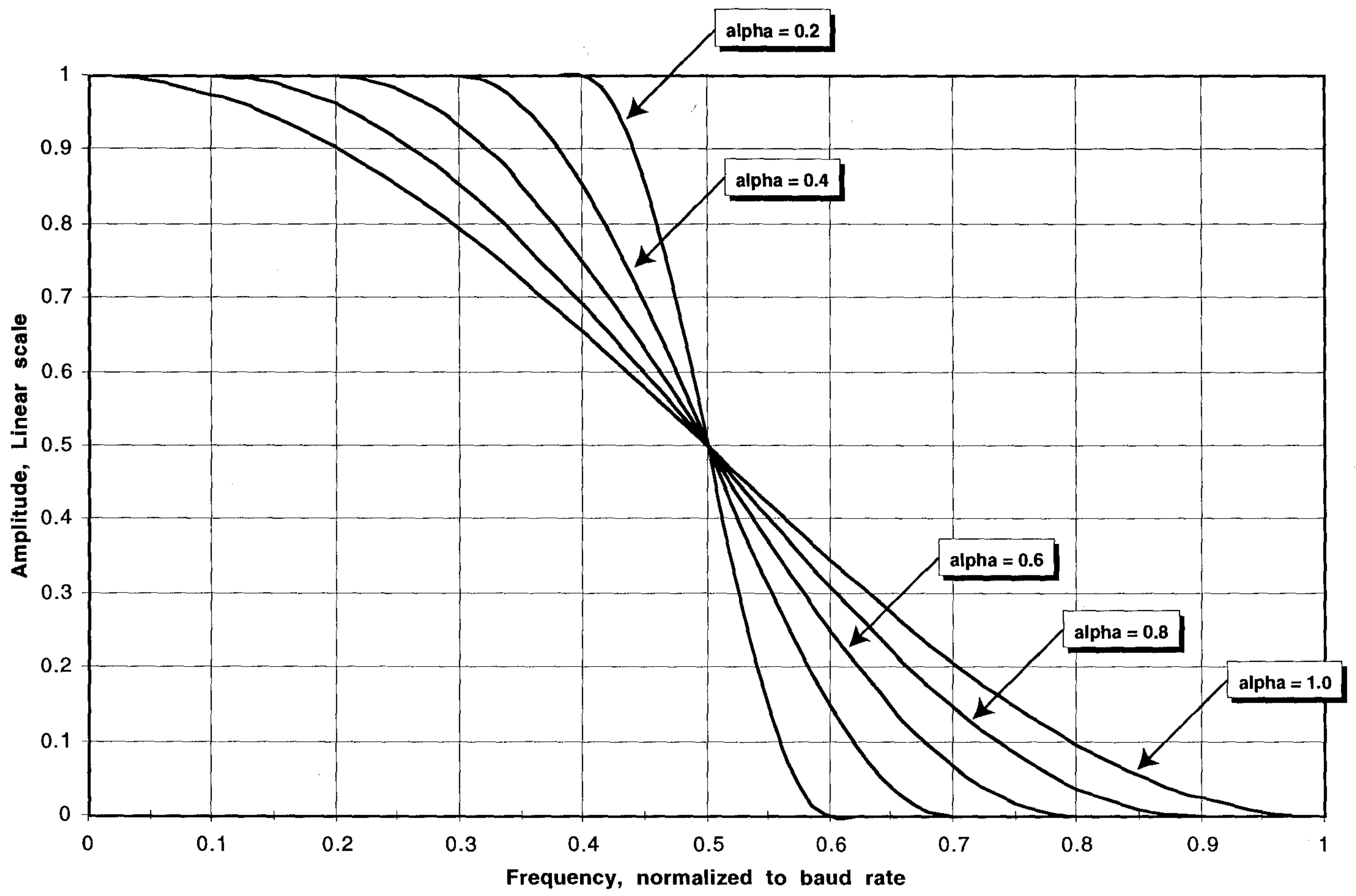


Figure 5-1 - Frequency response of Raised-Cosine Filters

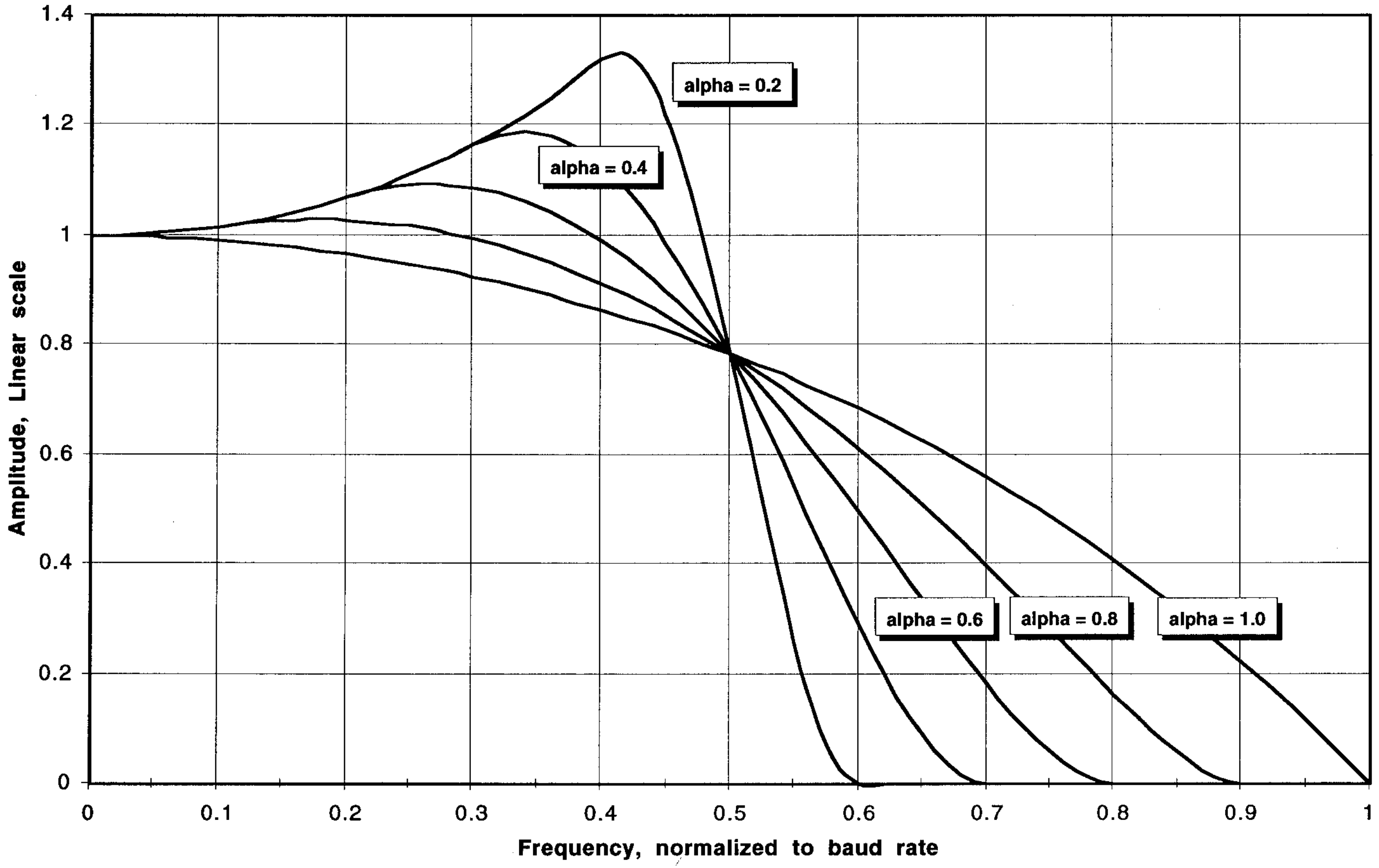


Figure 5-2 *- Frequency Response of x/Sin(x)-compensated Raised-Cosine Filters*

Filter Response Partitioning

So far, the shape of the spectrum needed at the receive slicer has been discussed. Obviously there is a need to partition the filtering between the receiver and the transmitter. One way to do this is to put half of the filtering at each place. Since the channel response will be the product of the two responses, then this involves taking the square root of the total response, and putting a square-root of raised cosine each at the receiver and transmitter. It has been shown that this partitioning of the filter function between receiver and transmitter can result in nearly optimal performance. Finally, there is a need to compensate the transmit filter for six(x)/x roll-off of the square-wave spectrum. Figures 5-3 and 5-4 are a block diagram of the baseband transmit and receive channels.

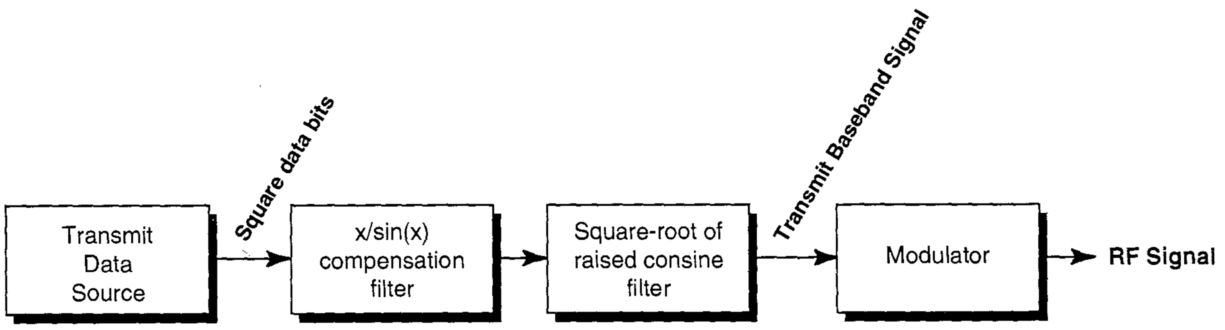


Figure 5-3 - *Transmit baseband filter model*

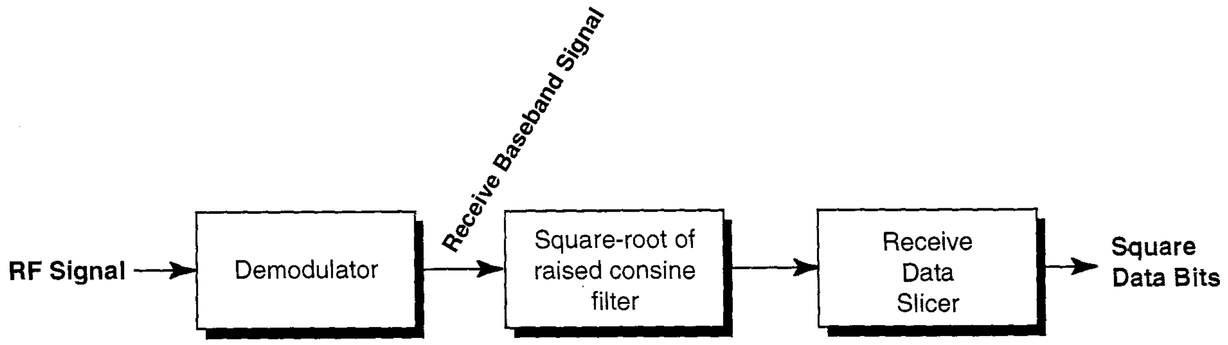


Figure 5-4 - *Receive baseband filter model*

Figure 5-5 is the square-root of raised cosine filter response. This is the filter shape that should be placed at the receiver. Figure 5-6 is the sin(x)/x compensated square-root of raised cosine response, and this is the filter shape that should be placed at the transmitter. Figure 5-7 is a logarithmic plot of the square-root of raised cosine filter, with the amplitude axis plotted in dB. Figure 5-8 is the corresponding filter with compensation for the sin(x)/x roll-off added. Notice that both filters have a steep roll-off between 0.5 and 1.0 of the baud rate.

The plots show only the amplitude of the filter transfer functions. It has been proven that the optimum phase condition is when the phase is linear across the total channel response. Any deviation from this linear phase condition requires additional transmitter power in order to achieve the equivalent performance of the linear phase case. Linear phase implies that the group delay of the filter is constant versus frequency. This can be seen by the definition of group delay:

$$Group\ delay = -\frac{d\phi}{d\omega} \qquad (5\text{-}2)$$

Where ϕ is the phase delay of the filter, in radians, measured at the frequency, ω , in radians/second. Thus, if the phase changes linearly with frequency, then the derivative is a constant as is group delay. This means that all frequencies of the spectrum will be delayed by the same amount of time. This linear phase condition can be achieved when the impulse response is completely symmetrical about the central peak. This is easiest to accomplish in a digital FIR filter, accounting for the popularity of this type of implementation.

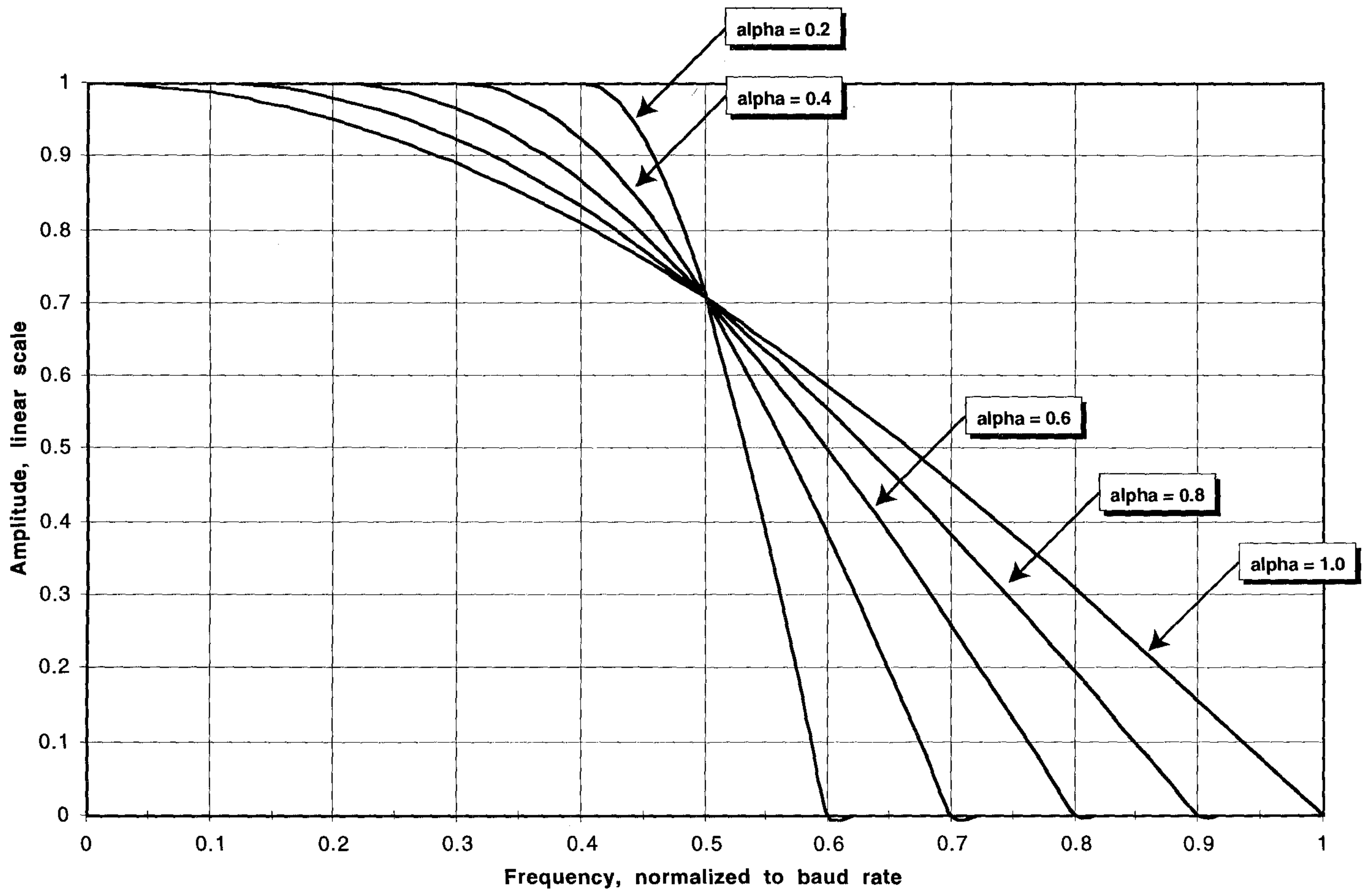


***Figure 5-5** - Square-root of Raised Cosine Frequency Response (Use as optimum receiver filter response)*

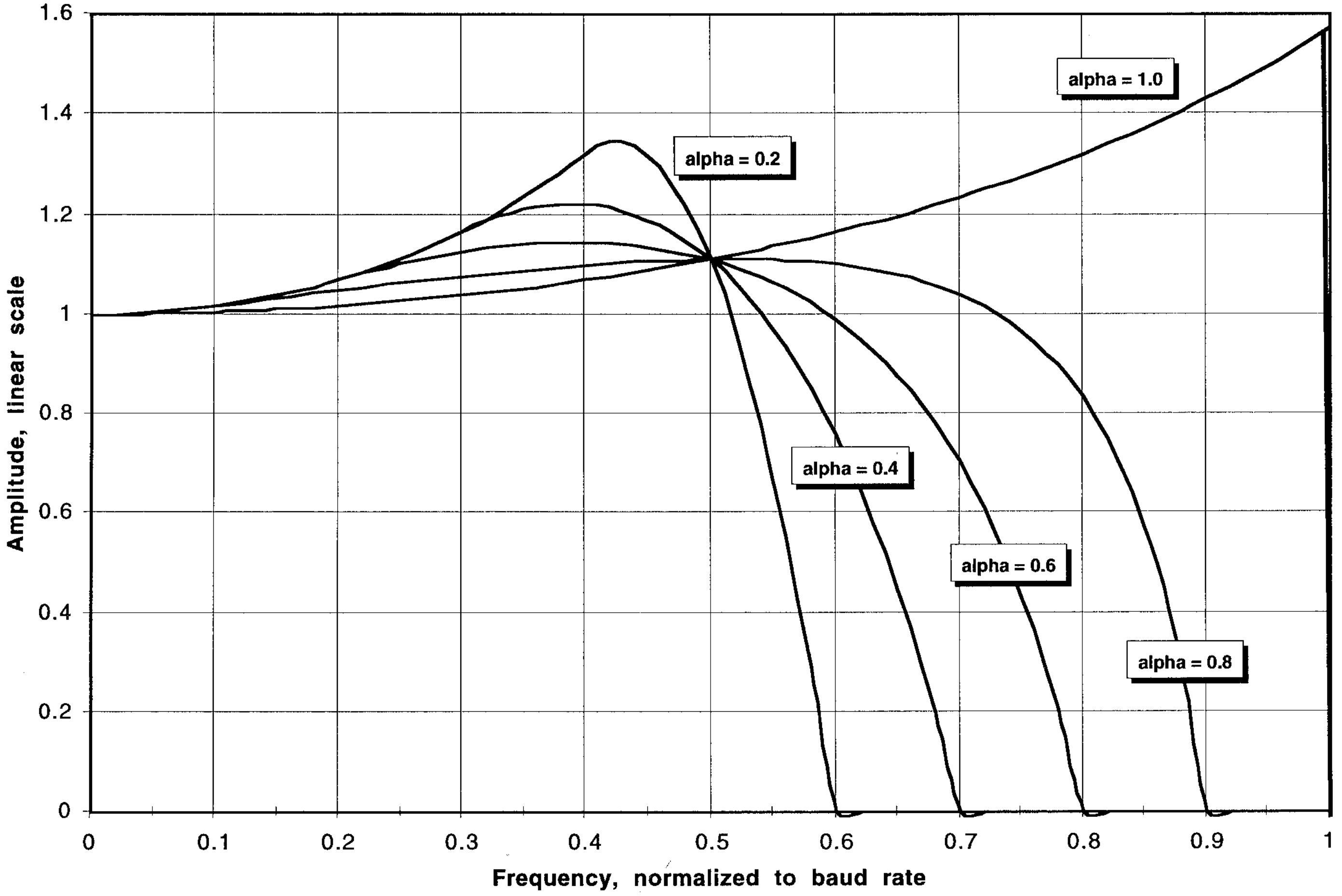


Figure 5-6 *- Frequency of Square-root of Raised Cosine plus Sinc Compensation (Use as optimum transmit filter response)*

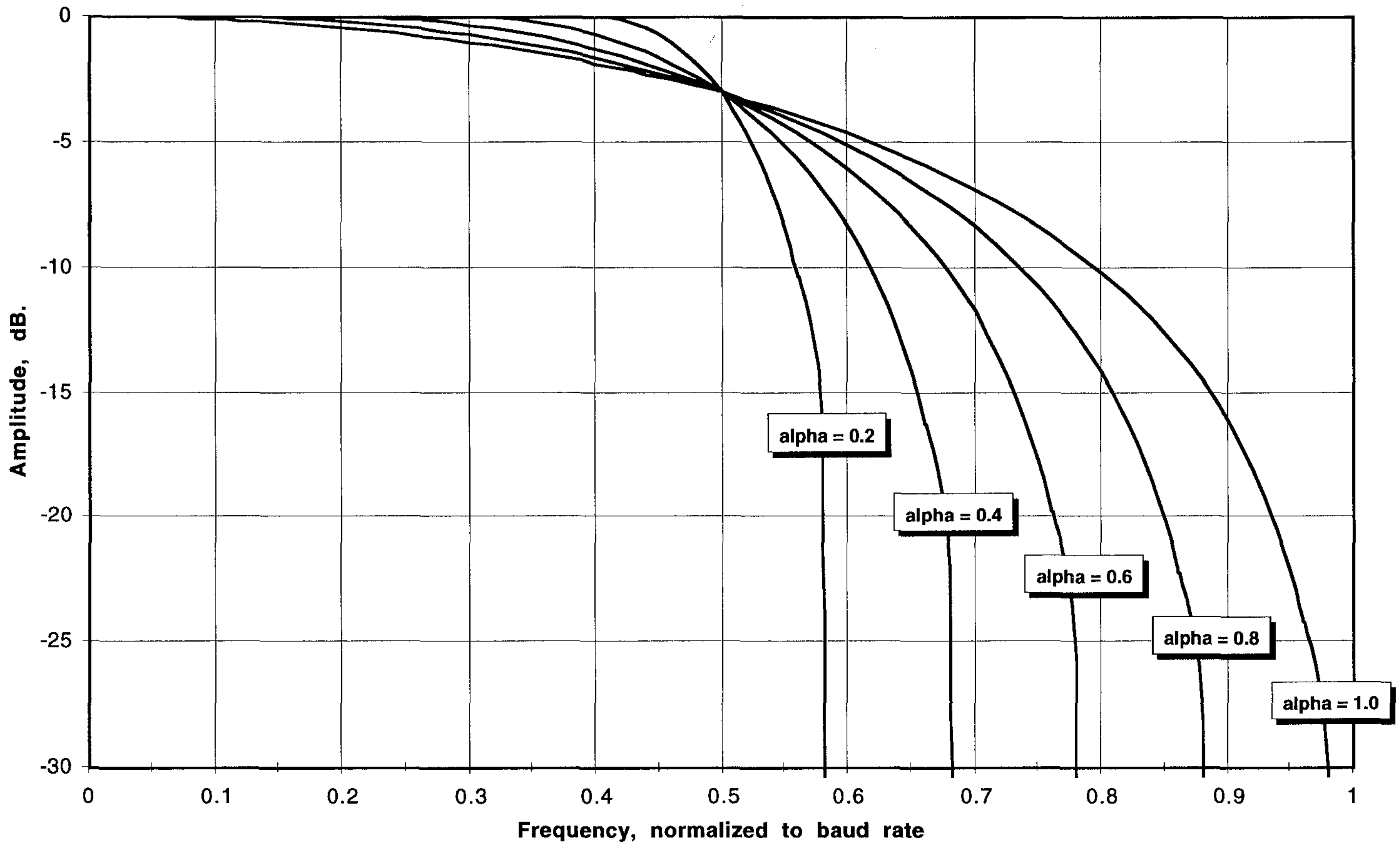


***Figure 5-7** - Square-root of Raised Cosine Frequency Response, Log*

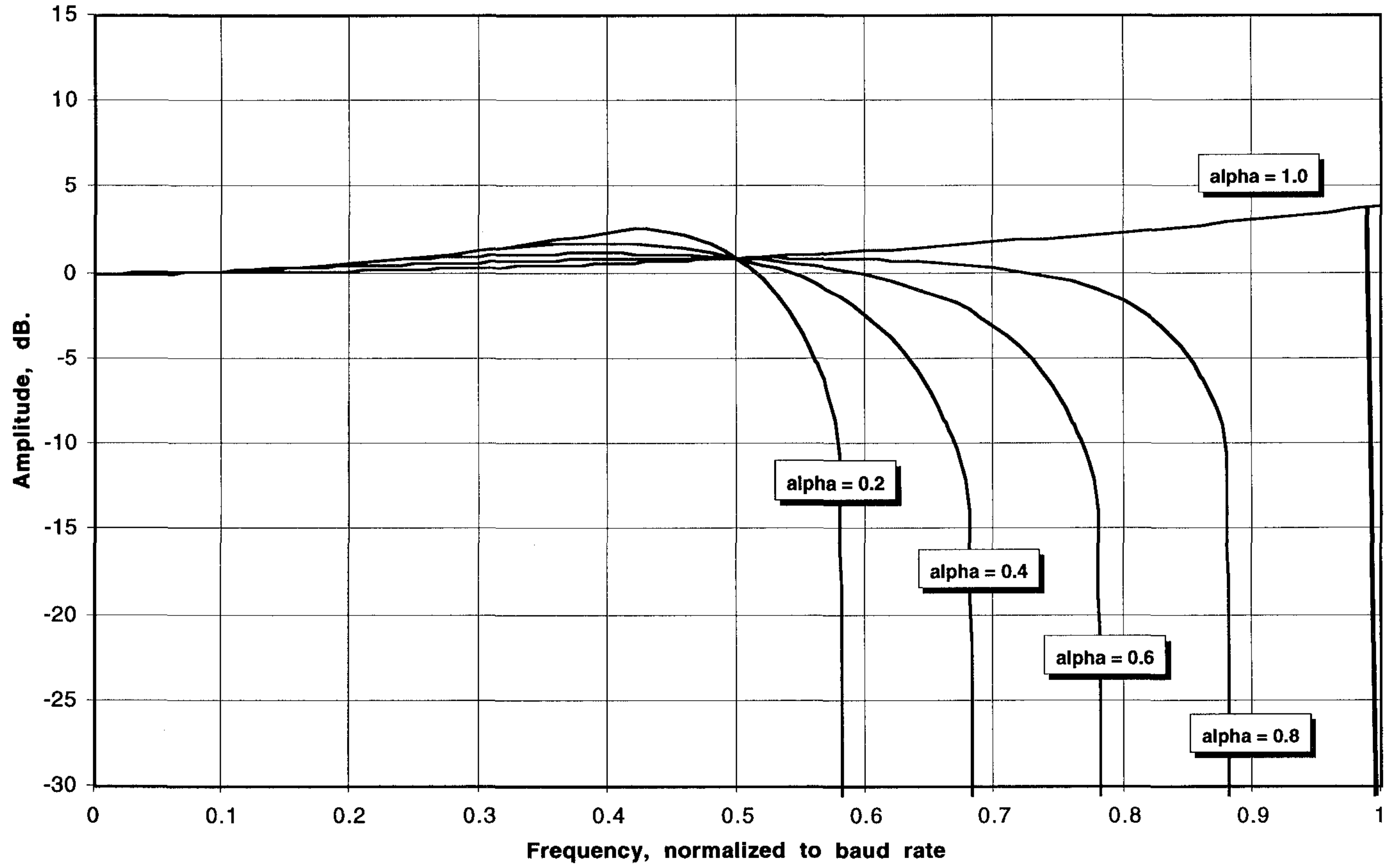


Figure 5-8 - *Frequency response of Square-root of Raised Cosine plus Sinc comensation, Log*

Filter Impulse Response

Next, it is necessary to derive the impulse response of these filters. Recall that the impulse response of the filter can be computed by taking the inverse Discrete Fourier transform of the frequency response. Fortunately, Excel includes a Fast Fourier transform and inverse Fast Fourier transform routine in the data analysis pack. The time samples will be shifted so that the impulse peak is in the center of the diagram, instead of being split with one-half at the beginning and one-half at the end of the plot. Figure 5-9 is a plot of the impulse response of the raised cosine filter. Figure 5-10 is a plot of the impulse response of the square-root of raised cosine filter, compensated for sin(x)/x roll-off.

All the information needed to compute the FIR filter coefficients for these optimum filters, both transmit and receive, has now been explained. A complete step-by-step design process will be detailed later. How is the impulse response used to tell what the shape of a square pulse looks like after it has passed through these filters? Recall from appendix C that there is a duality between convolution in the time domain and multiplication in the frequency domain. One method of finding the response of the filter to a square pulse that is the length of one data bit would be to compute the spectrum of the pulse, multiply it by the filter frequency response, and then to re-convert it back to the time domain to see it's response. However, since the impulse response of the filters has already been computed, it is much easier to convolve the square data bit in the time domain with the impulse response (in the time domain). This yields the time response of the output of the filter to excitation directly from the square pulse. This is exactly the same thing that happens in an FIR digital filter: convolution of two time sequences. From Appendix D, the formula for convolution is:

$$y(t) = \int_{-\infty}^{+\infty} h(\tau)\; x(t-\tau)\, d\tau \qquad (5\text{-}3)$$

Where $x(t)$ is the input to the filter at time t, $h(t)$ is the impulse response of the filter (which is assumed to be real) at time t, and $y(t)$ is the output of the filter at time t, while τ is the variable of integration. It turns out, of course, that this equation can be transformed to sampled-time for use in a DSP filter. This is in fact the heart of a digital FIR filter core, and is implemented as a loop. For each output time t, we scan across the entire impulse response of the filter and multiply the impulse response at each point with the signal that happens to be in that location. Notice that the data sequence is run backwards through the convolver due to the $(t-\tau)$ term. So, if the impulse response is 50 taps long, it is necessary to perform 50 multiplies and 50 additions for each output time sample. Many DSP chips have special hardware to perform this multiply-accumulate operation quickly.

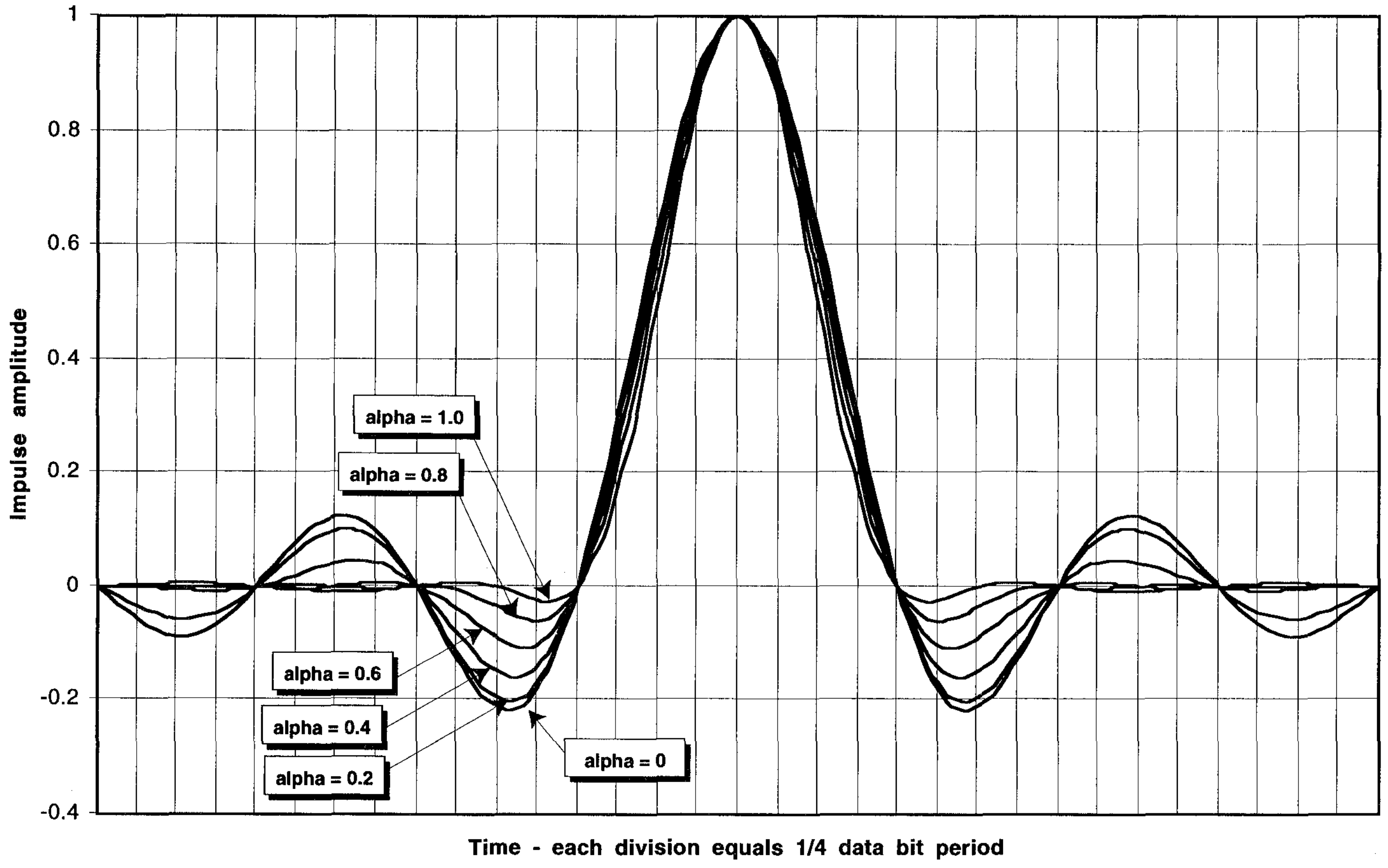


Figure 5-9 - *Impulse response of raised-cosine filters*

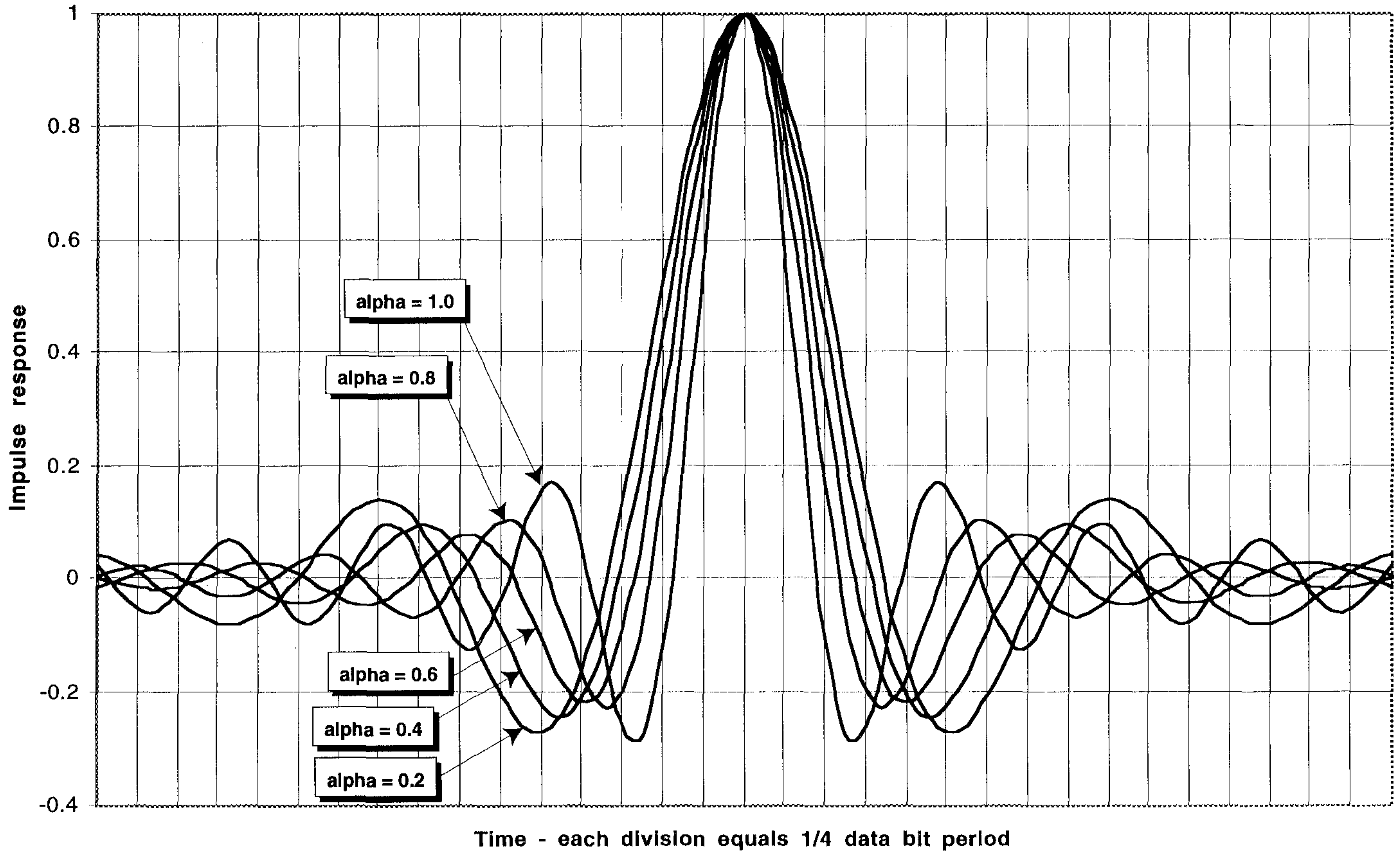


Figure 5-10 *- Impulse response of square-root of rasied cosine plus x/sin(x) compensation*

Filter Response to an Isolated Data Bit

Figure 5-11 is the convolution of a square pulse of one data bit length with the raised cosine filter of alpha = 0.4 that has been compensated for sin(x)/x roll off. Figure 5-12 is the same thing except for an alpha=1.0 filter. These figures are useful in that they tell us the pulse (not impulse) response of the transmitter to an isolated one bit. This can be used as a test requirement or a measurement of the transmit filter in the time domain. It's what would be observed if an oscilloscope were connected to the transmit filter, and an isolated data bit were sent through the filter. Measuring time domain response may be easier than trying to measure the frequency domain response.

While useful for verification of proper operation of the filter itself, the isolated data bit response, does not give a good feel for how well the system is going to perform. In order to get a qualitative indication of the proper operation of the entire system, we would like to verify that in fact the data bit has no inter-symbol interference (ISI) at exactly the sampling time. This is easiest to see on an eye diagram. The eye diagram is the overlapping of all possible trajectories of a voltage caused by data bits through the filter. When data is generated, we should assume that it is truly random in nature and cannot say what the sequence of bits is. This is because the raised-cosine filters have impulse responses that have energy (continued ringing response) removed from the central data bit by many bit times. It is true, though, that the length of the impulse response that is stored in a digital filter is finite, and thus the filter will only ring for a certain number of bit times. Thus, we only have to analyze all possible sequences of data that are as long as or shorter than the filter ringing time. The same statement could be made about an analog filter except that the truncation in time is due to noise overcoming the decreasing response of the filter away from the central data bit. It is not so easy to compute the exact equivalent 'length' of an analog filter however.

One way to generate all possible sequences of data of length *n* or less (with the exception of the length-*n* all zero's sequence) is to generate a data bit stream from a maximal-length pseudo-random shift register. This arrangement produces a pseudo-random binary sequence, abbreviated PRBS. The circuit and the arrangement of taps that produces these sequences are listed in Appendix B for some values of n. The length of the repeating PRBS pattern is:

$$Maximal\ Length = 2^n - 1$$

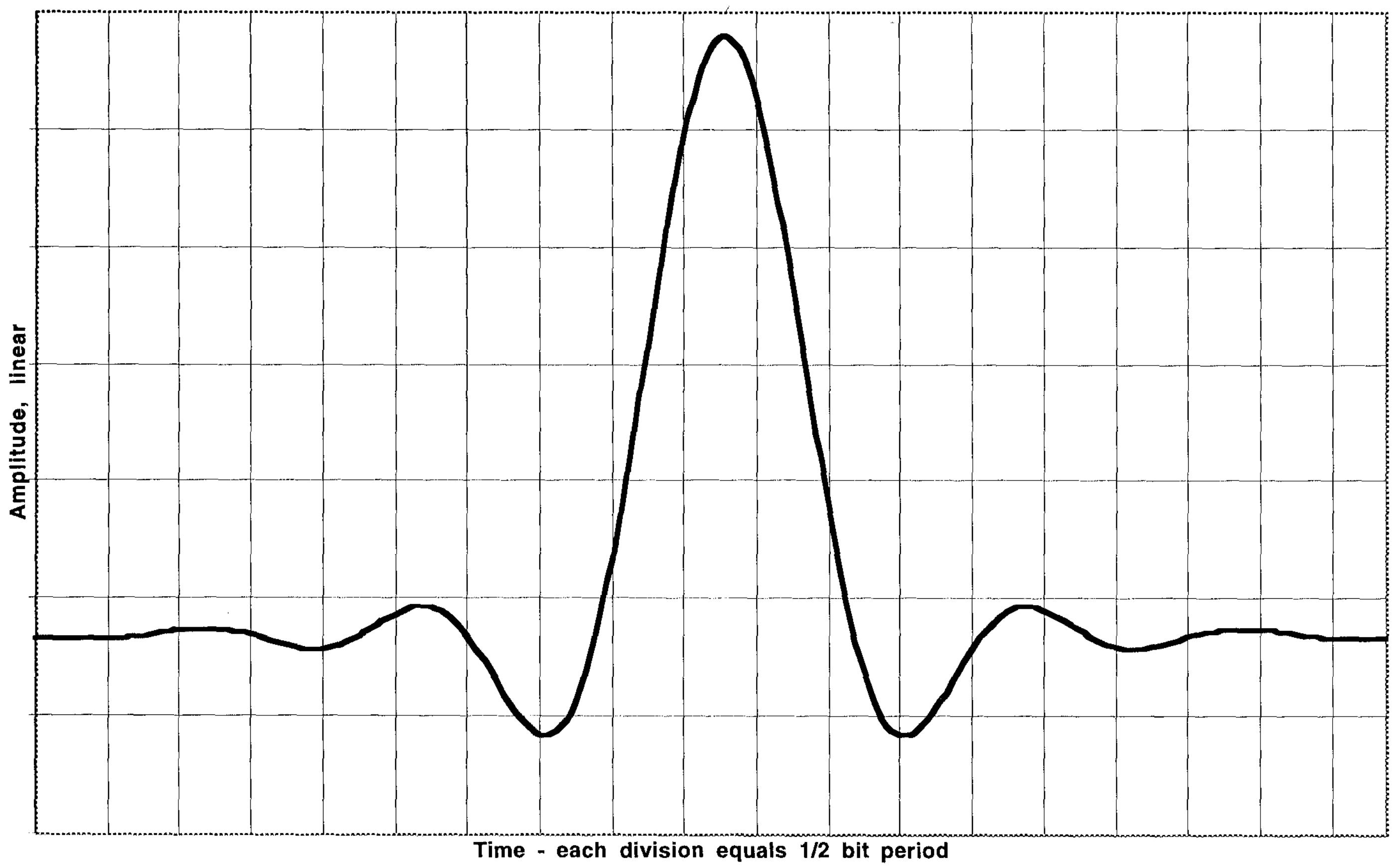


Figure 5-11 *- Transmit pulse shape for Square-root of raised cosine + x/sin(x) compensation for alpha = 0.4*

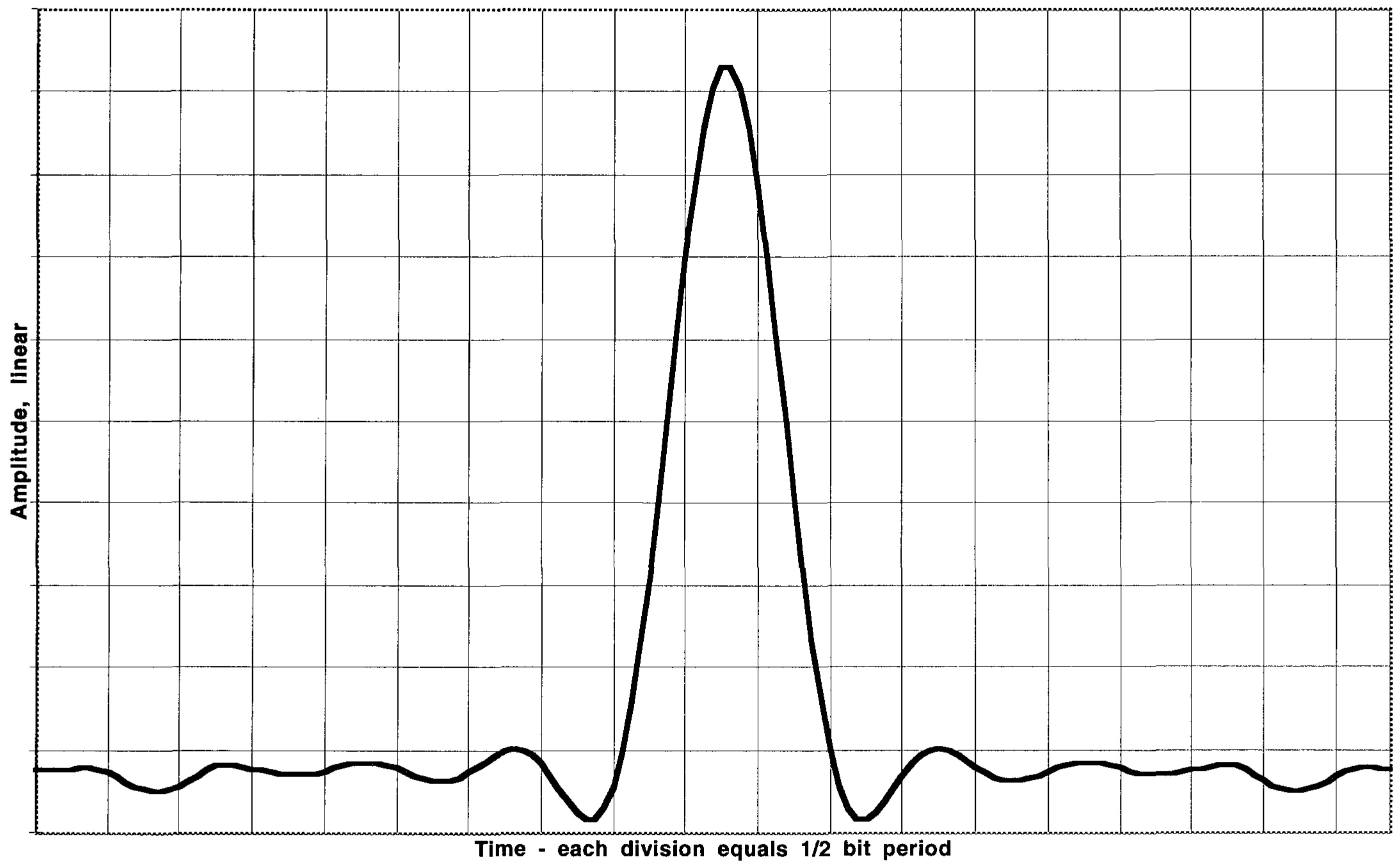


Figure 5-12 *- Transmit pulse shape for Square-root of raised cosine + x/sin(x) compensation for alpha = 1.0*

Where *n* is the number of stages in the shift register. The length of the longest bit stream for which all possible sequences are generated is *n*, with the exception that the all-zero's state is not generated. So, we should pick our PRBS generator length *n* to exceed (by 1) the number of bits contained within the filter's impulse response. This will accurately test all possible trajectories of signals through the filter. To do this, we generate the PRBS, and then convolve it with the filter impulse response. For practical purposes, a sequence of 2^5-1 = 31 is used. Otherwise, the calculations take a very long time in Excel, and some of the possible trajectories are in fact not shown in the eye-diagram figures. A good-enough idea of the eye-diagrams is achieved even with this limitation. however. Figure 5-13 is the time-domain response of an alpha=0.2 filter to a PRBS sequence. and figure 5-14 is the time-domain response of an alpha=0.8 filter to a PRBS sequence. The low-alpha (0.2) filter has much more overshoot (ringing) than the alpha=0.8 filter.

It is not at all obvious how the ringing and overshoot are related to the exact bit intervals when looking at figures 5-13 and 5-14. An *eye diagram* (constructed using these figures) makes the relationship very clear.

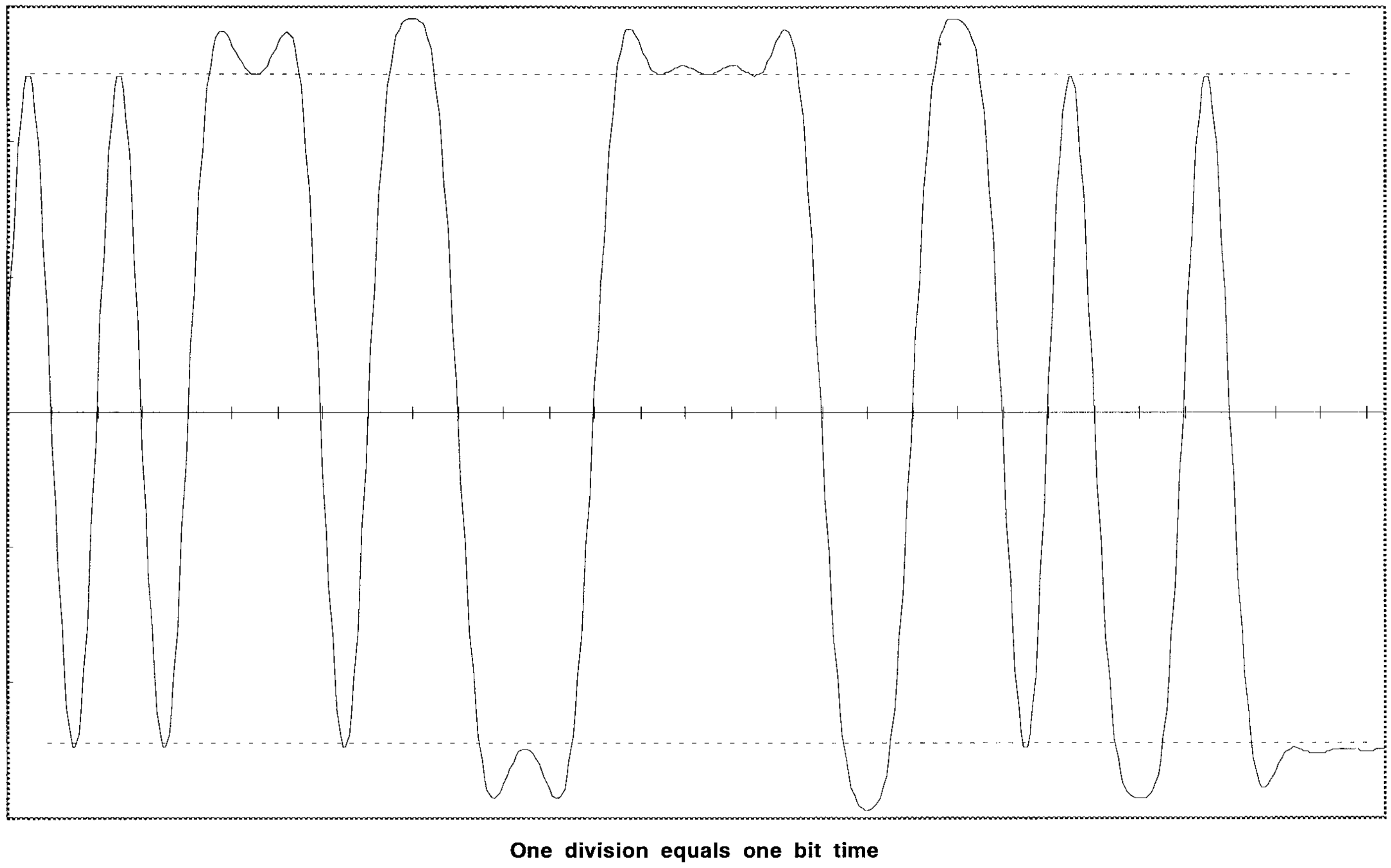


Figure 5-13 *- Time-domain response of x/sin(x) compensated raised-cosine filter alpha = 0.8, sequence =* $2^5 - 1$ *PRBS*

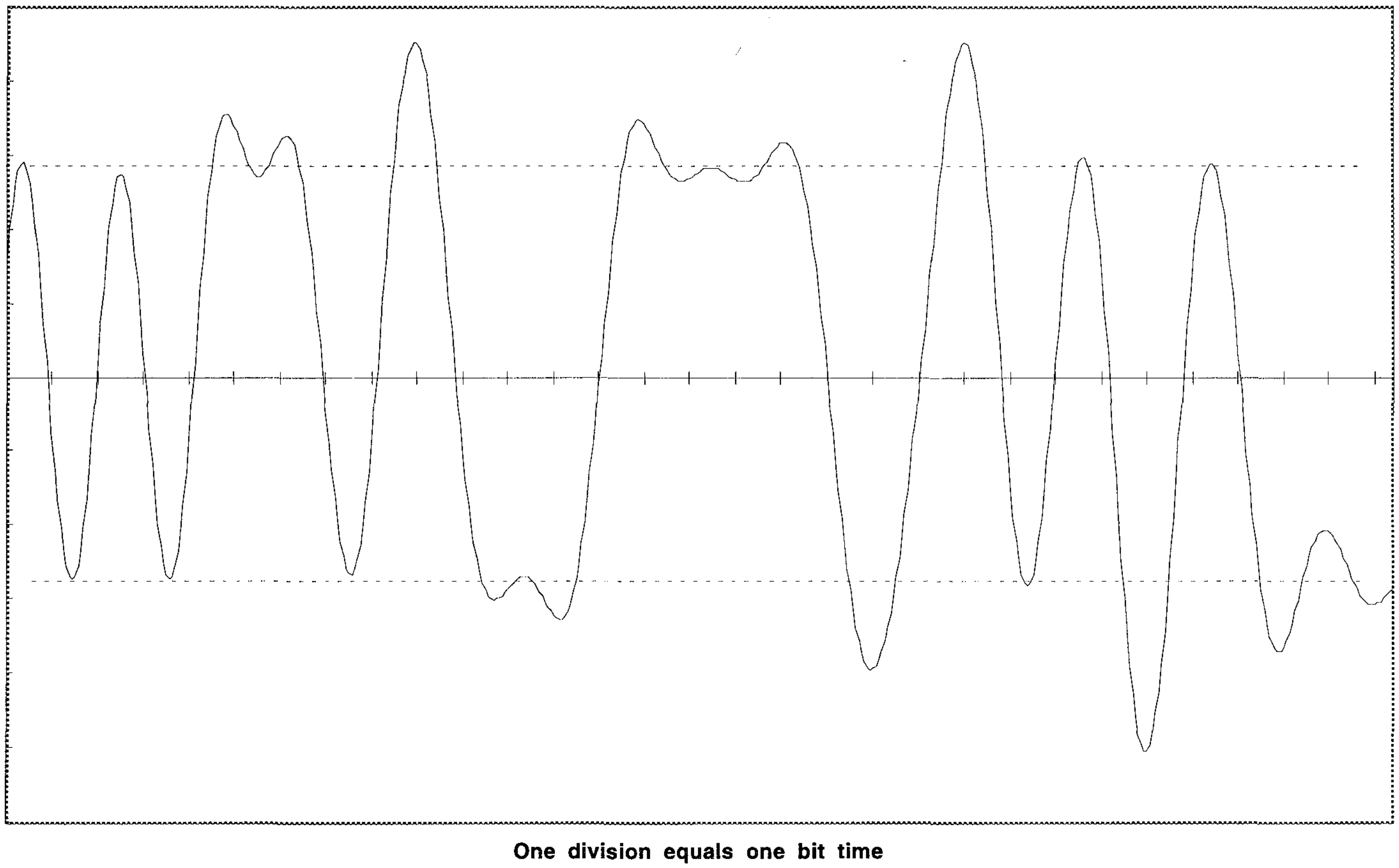


Figure 5-14 - *Time-domain response of x/sin(x) compensated raised-cosine filter alpha = 0.2, sequence = 2^5 - 1 PRBS*

Eye Pattern Generation

Now that the filter PRBS time response to a PRBS pattern has been generated, an eye pattern (or eye diagram) can be generated from this diagram. An eye pattern is just consecutive overlays of each data bit with one another. When using an oscilloscope, an eye pattern is generated by synchronizing the scope to the recovered clock (see section on clock recovery). Since the scope is not synchronized to the data, essentially random data patterns will be displayed. To generate an eye diagram analytically, we take the PRBS response and cut it up into pieces, each one data bit time long. Then all of those traces are overlaid on top of one another and the eye pattern is produced. Actually, it is usually more convenient to plot two or three of these patterns consecutively in time so that there is a visual connection between adjacent bits — it's just a personal preference factor.

Figure 5-15 is the eye pattern of an alpha=0.8 raised cosine filter response to a n=5 PRBS sequence. Notice that the eye is closed somewhat in the center, where we want it to be completely open. This closure represents almost a 6 dB. performance penalty compared to a perfect eye. This is because we have not yet compensated the response for the sin(x)/x roll off. Figure 5-16 is the eye pattern of the compensated filter, and a perfect opening in the middle of the data bit can be seen. We have plotted the response for the alpha=0.8 filter, and that filter does not meet the criteria for minimal zero-crossing jitter. Now we can see that in fact the eye pattern shows some small amount of zero-crossing jitter that is representative of a perfectly adjusted system with an alpha=0.8 filter. Figure 5-17 is the eye pattern of an alpha=1.0 raised cosine filter with compensation for sin(x)/x roll off. Notice the perfectly open eye at the exact center of the data bit, as well as the jitter-free zero crossings. However, the alpha=1.0 filter has wider transmitted spectral width and a wider noise bandwidth. A low-alpha filter has smaller spectral width and smaller noise bandwidth, but needs a better clock recovery circuit.

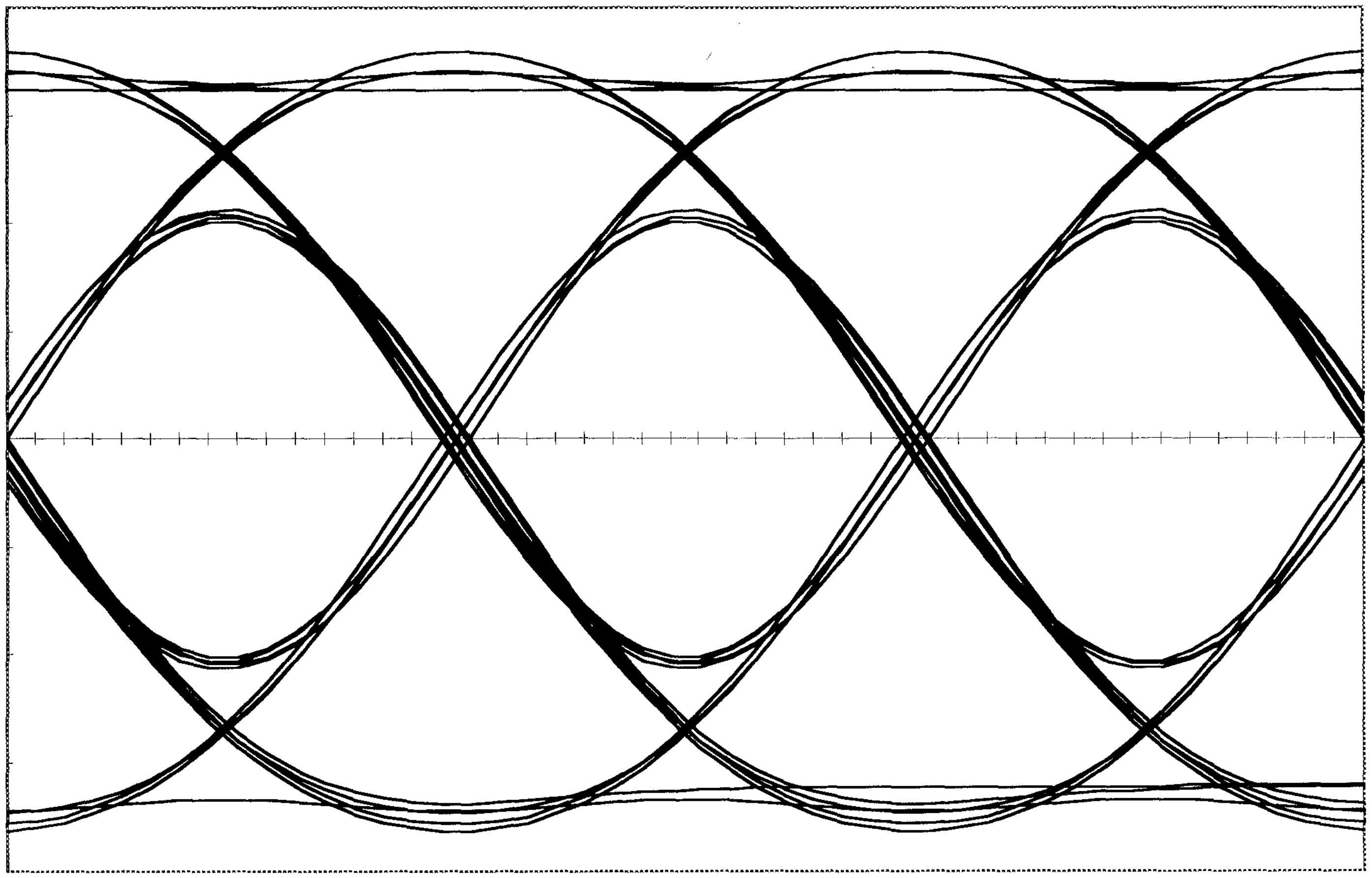


Figure 5-15 *- Eye Pattern - Raised cosine - no compensation alpha = 0.8, sequence = 2^5 - 1 PRBS*

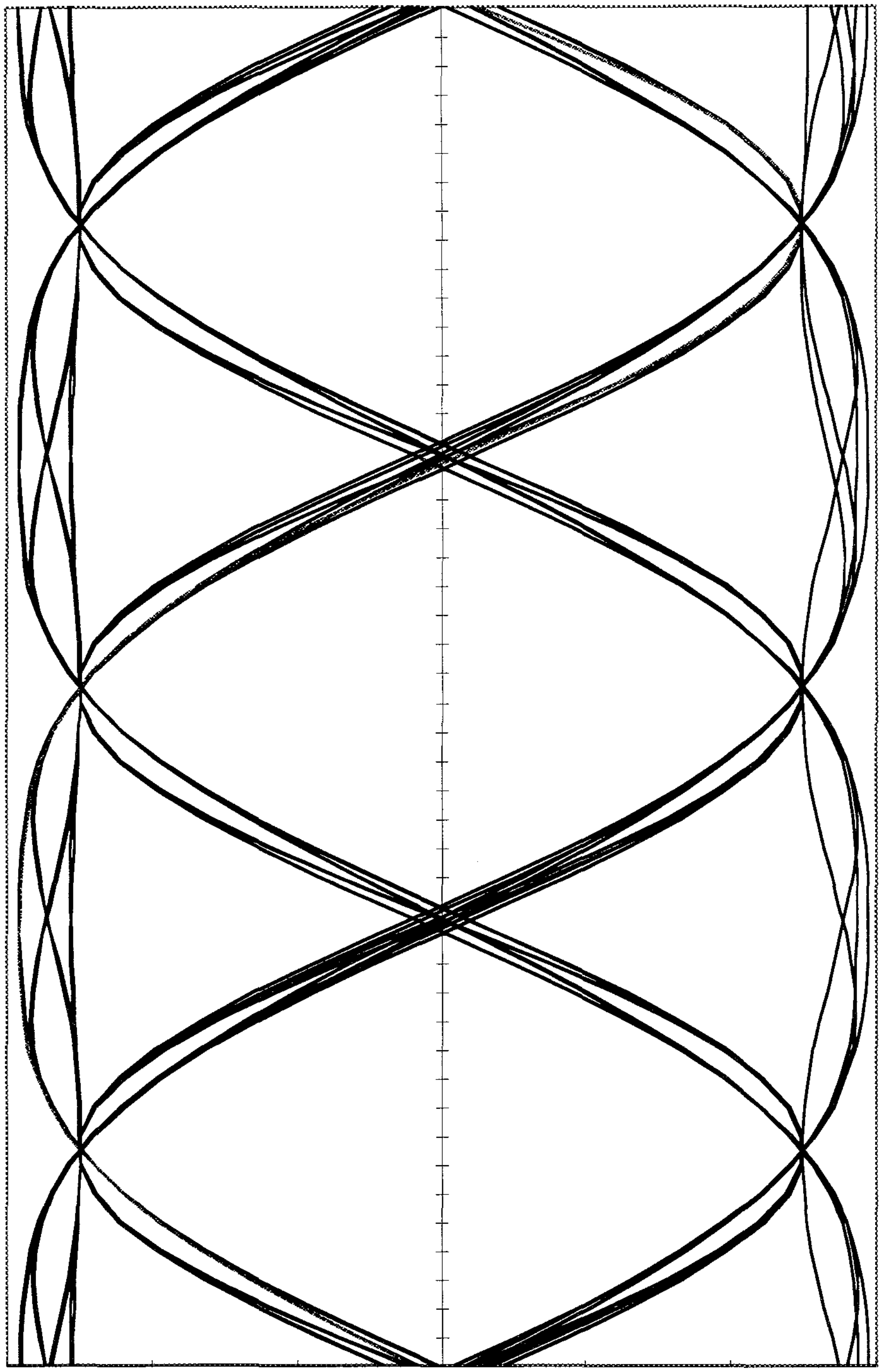


Figure 5-16 - Eye Pattern - Raised cosine + x/sin(x) compensation alpha = 0.8, sequence = 2^5 - 1 PRBS

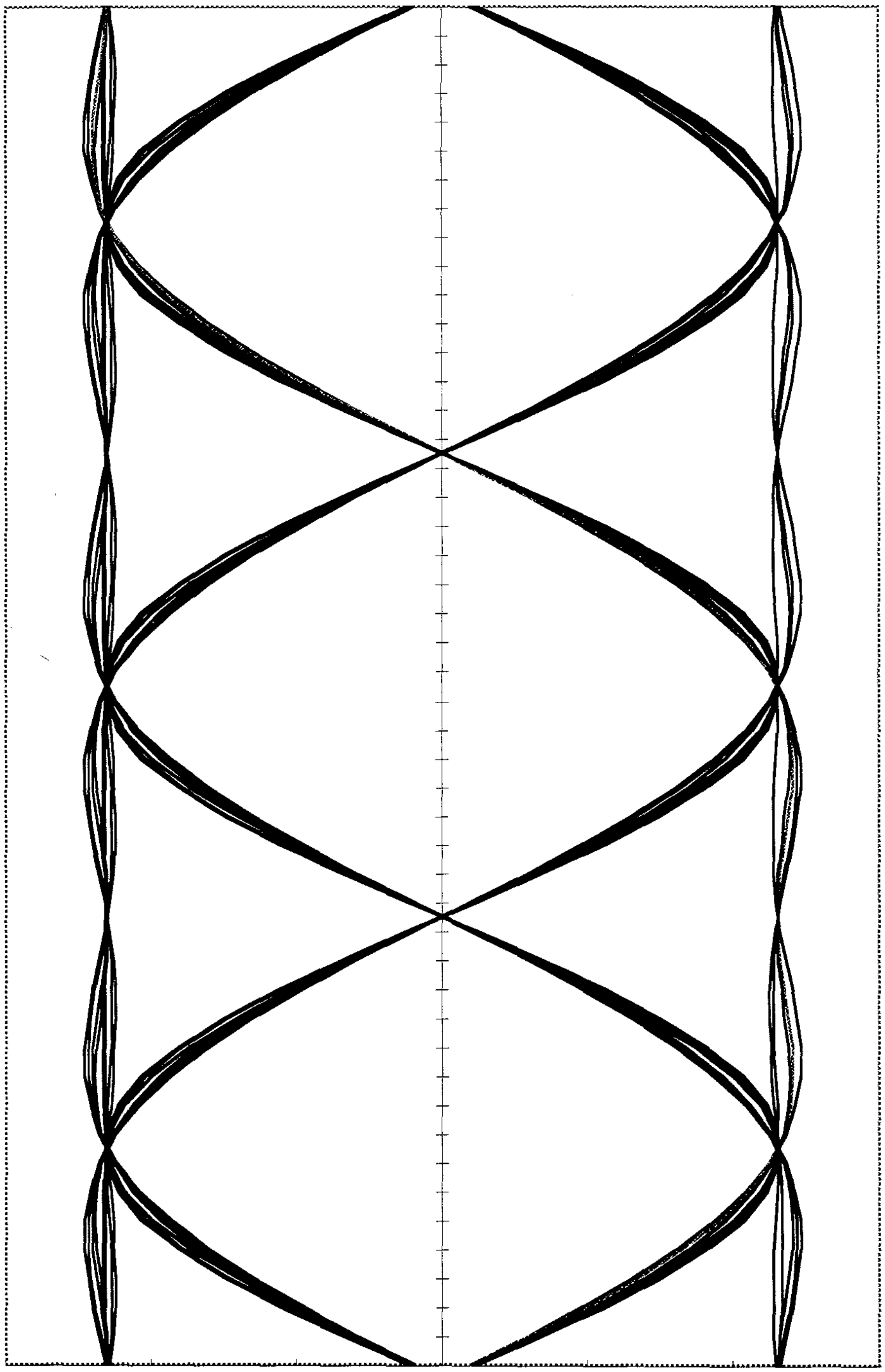


Figure 5-17 - *Eye Pattern - Raised cosine + x/sin(x) compensation alpha = 1.0, sequence = 2^5 - 1 PRBS*

Equivalent Noise Bandwidth

Each receive filter lets through some amount of noise as well as signal. Obviously a wider receive filter allows more noise through than a narrower filter does. In the case of the raised cosine response, the amplitude shape is vestigial — that is, the shape is symmetric around the 1/2 frequency point. The noise, however, is proportional to the square of the response (power is proportional to voltage squared), so the equivalent noise bandwidth of raised-cosine filters changes depending on the value of alpha. A perfect brick-wall filter of 0.5 bandwidth (equivalent to a raised cosine filter with an alpha = 0) has a noise bandwidth of 0.5. See figure 5-18 that shows this effect for a non-brick wall filter. To calculate the equivalent noise bandwidth of other filter shapes, it is necessary to measure the total response compared to a standard brick-wall filter. It may be convenient to normalize the filter width to 1 (rather than 0.5). The noise bandwidth can then be measured by simply measuring the area under the square of the filter amplitude curve from zero frequency up to the point where effectively no energy gets through the filter. If we measure the square-root of raised-cosine filters, which were shown earlier to be optimal for the receiver, then we find that the equivalent noise bandwidth is independent of alpha, and is always 0.5 (or 1, if we normalize to double the bandwidth as with the brick-wall filter). By converting the amount of noise that goes through the equivalent bandwidth to a ratio and expressing it in decibels, it is possible to express the noise penalty of each raised-cosine filter type. Since noise is introduced between the transmitter filter and the receiver filter, only the receiver filter figures into the equivalent noise bandwidth calculation. Table 5-1 shows the equivalent system bandwidth (which is related to the channel efficiency in terms of frequency utilization) for some different raised cosine filters responses.

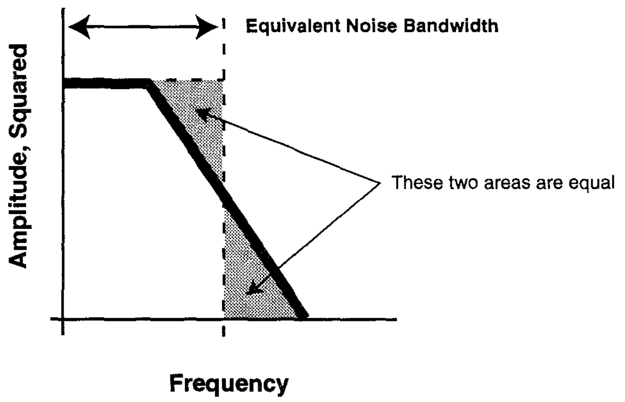


Figure 5-18 - Equivalent noise bandwidth is the point where the areas under the squared amplitude curve are equal.

Alpha	Equivalent Noise Bandwidth
0.0	1.39
0.2	1.28
0.4	1.20
0.6	1.15
0.8	1.12
1.0	1.13

Table 5-1 *- Equivalent Noise bandwidth of Raised-Cosine filter (system response). The receiver noise-bandwidth of a square-root of raised cosine filter is always 1.00, regardless of the alpha factor.*

Some Frequency Response Defects

A few defects are common in trying to generate an accurate raised-cosine frequency response. The most common is that the amplitude response has either a high-pass or a low-pass pole that is undesired. For example, the high-pass pole could arise if the modulation bandwidth of our radio is not sufficiently wide. A low-pass pole could occur if the baseband signal is AC-coupled through a capacitor anywhere in the path. The distortions to the eye pattern from a high-pass pole look similar to the raised-cosine eye patterns that have not been compensated for sin(x)/x roll-off. They both represent a deficient high-frequency response. The distortion from a high-pass pole (AC-coupling capacitor) is very different. Figure 5-19 shows the eye pattern of an alpha=1.0 raised cosine filter including x/sin(x) compensation that has been passed through a coupling capacitor with the -1 dB point at 0.031 baud (i.e.: about 30 Hz for a 9600 baud signal). This is equivalent to a -3 dB point of about 15 Hz. for a 9600 baud data signal. The baseline wander is clearly evident, and the power penalty is surprisingly large, around 2 or 3 dB. Good low-frequency response is quite important in maintaining good performance with scrambled data signals.

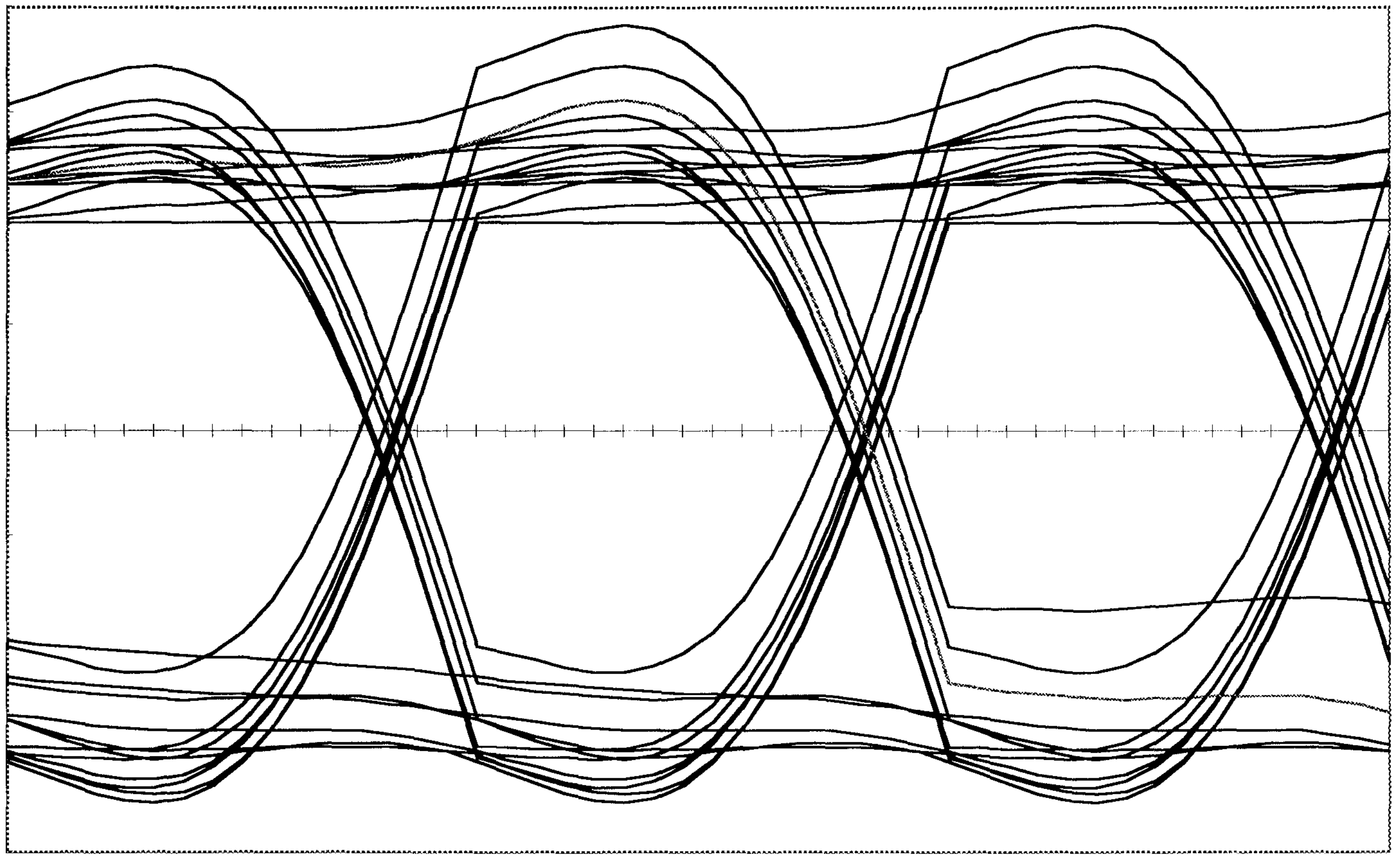


Figure 5-19 *- Eye Pattern - Raised cosine + x/sin(x) compensation alpha = 1.0, sequence =* 2^5 *- 1 PRBS AC-coupled, -1 dB point = 0.031 baud*

Impulse-Response Length Truncation

Could we arbitrarily truncate the impulse response so that there would be no ringing from adjacent bits to cause inter symbol interference in the current data bit? For example, could we multiply, the impulse response with a rectangular window, so the response outside the window time is zero? In the time domain, yes this is possible. However, when this is done, the frequency response (and particularly the out-of-band rejection) becomes severely degraded. There are functions, called windowing functions, that can be applied to the impulse response to smoothly reduce the coefficients to zero at some distance from the central bit time. These various windowing functions have different effects on the frequency response. Most of the time, these responses are used to transform a known filter shape to one with a smaller number of coefficients to compute, but at the cost of the filter transition band (roll-off region) shape changing away from the desired shape.

We can see how critical the transition-band shape is to proper operation of the raised cosine filters. Altering this band can have severe degradation to the eye pattern. If a rectangular windowing function is applied, we simply ignore (or set to zero) the filter coefficients outside some range. This has the effect of limiting the filter length if we compute it as an FIR filter. The eye patterns plotted previously have been computed with 129 filter coefficients, at an oversampling rate of 16. The oversampling rate is this high in order to give nice, smooth eye patterns.

If instead we oversample at 2x (i.e.: 19200 samples/second for a 9600 bit per second system) then it is necessary to compute just 9 filter coefficients to give the same results. It won't look smooth when plotted, but will perform just as well. At an alpha value of 0.2 or less, 9 coefficients are insufficient, as the eye is starting to diverge from a single-valued decision point in the center. At 2x oversampling, 9 coefficients represents 4-1/2 data bits within the filter. What does this truncation to 4-1/2 bits length do to the frequency response? It is possible to compute the shape of the truncated transmit filter, which is x/sin(x) + square-root of raised cosine by doing the following:

1) Derive the desired filter transfer function, then take the inverse Fourier transform (as has been done previously).

2) Truncate the impulse response coefficients by using the rectangular window (set all coefficients outside the range equal to zero).

3) Then Fourier transform the windowed-coefficients back to the frequency domain, and examine the response.

Figures 5-20 and 5-21 are the linear and logarithmic frequency responses of the optimum alpha = 0.4 transmit filter after this truncation process. The error for 5-tap and 9-tap filters is severe, and will yield unacceptable eye patterns. The error for a 17-tap filter is smaller, and yields good eye pattern for some values of alpha, including 0.4. The error for a 33-tap filter is very small, and yields very good eye patterns for most values of alpha. Figure 5-22 is an eye-pattern plot of an alpha=0.2 filter with 33-taps. Compare this to the same alpha=0.2 filter, but with 17-taps instead, shown as figure 5-23. Low alpha-value filters require a larger number of taps than higher alpha filters for equivalent performance. Alternatively, we can look at a log-scale of the filter responses, in order to assess the out-of-band transmit response. Referring back to figure 5-17, even the 17-tap filter provides degraded transmit spectral rejection, however the outstanding 33-tap filter would only be needed for very closely-spaced transmit signals (where a low-alpha filter is required). The first lobe of the 17-tap filter is 28 dB down, although this occurs very close to the passband. At about 25% above the passband, the spectrum is 50 dB down.

Filters with lengths in-between the values chosen above (8+1=9, 16+1=17, 32+1=33 taps) can be designed. The above values were chosen because the impulse responses of these filters is small at the chosen end points. Choosing a filter with a different number of taps would possibly place the truncation of a rectangular window onto the impulse response at a point where the amplitude of the impulse response was significant, leading to some distortion. If a windowing function other than a rectangular function is chosen, the window can be set to roll-off gently (have a small value) at the desired end-points of the impulse response. Several well-known window functions are the Hamming, Hanning, and Blackman windows. However, these windows, while providing improved resistance to peaking, and good out-of-band rejection do so at the expense of changing the shape of the transition region of the filter (the region between pass-band and cut-off). As such, they distort the eye-pattern of the resulting filter. Figure 5-24 shows the eye pattern for an alpha=0.6 filter with Hamming window truncation. The Hanning window provides an almost identical eye pattern. The power penalty with these windows, as compared to the rectangular window is fairly small, roughly 0.5 dB (for a 2-level eye), and would be acceptable for a 2-level eye. Of course the eye closure would be worse for a multi-level eye. A trade-off between the desired alpha, the out-of-band frequency response, the window size, and the window type can yield a smaller number of FIR filter taps for equivalent performance. Some experimentation with these 4 variables can yield an efficient filter, however, this is beyond the scope of this book. The included Excel spreadsheets allows experimentation with the filter, and will generate the eye pattern.

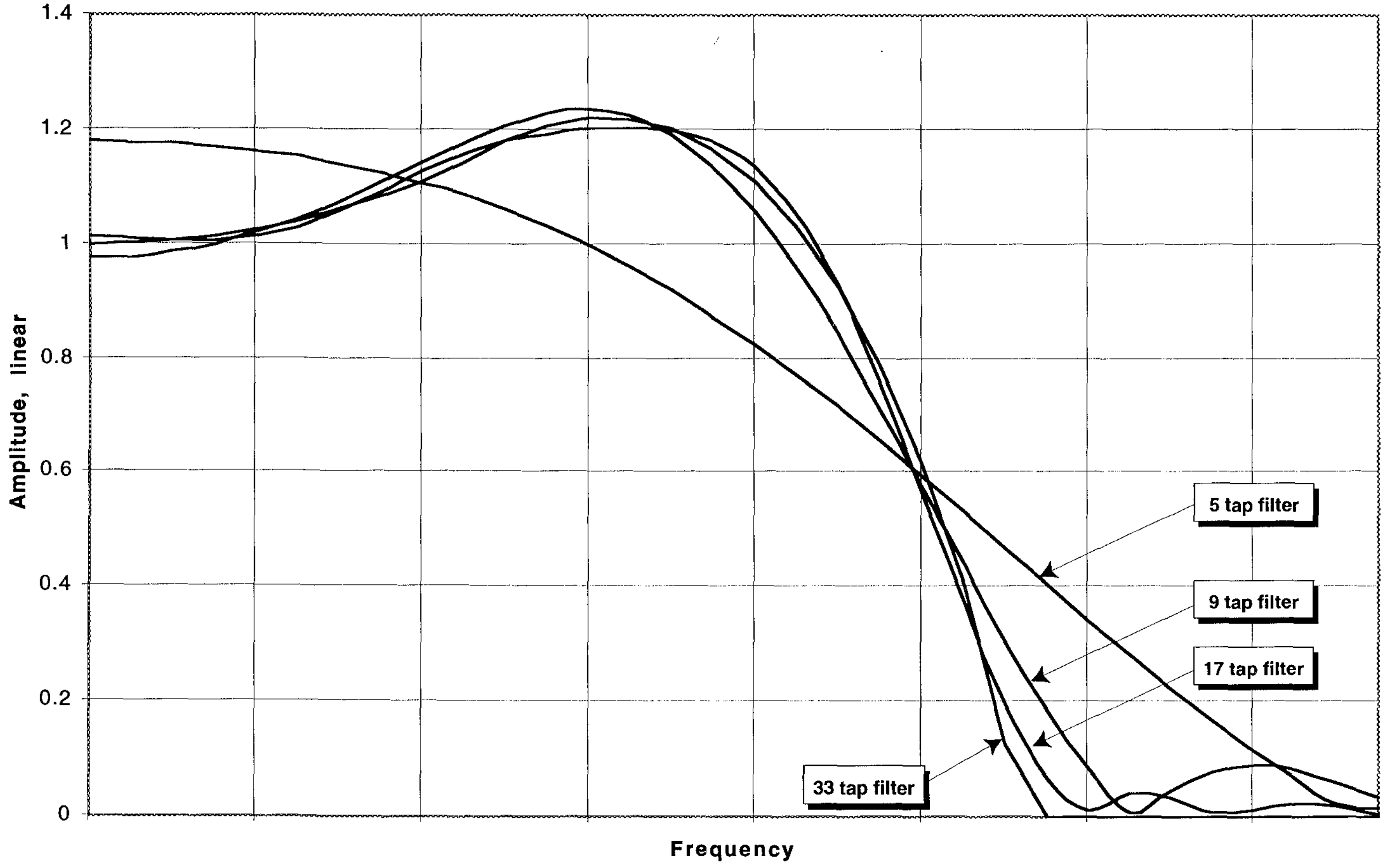


Figure 5-20 - Effect of limiting the number of filter taps for alpha = 0.4 filter (linear scale)

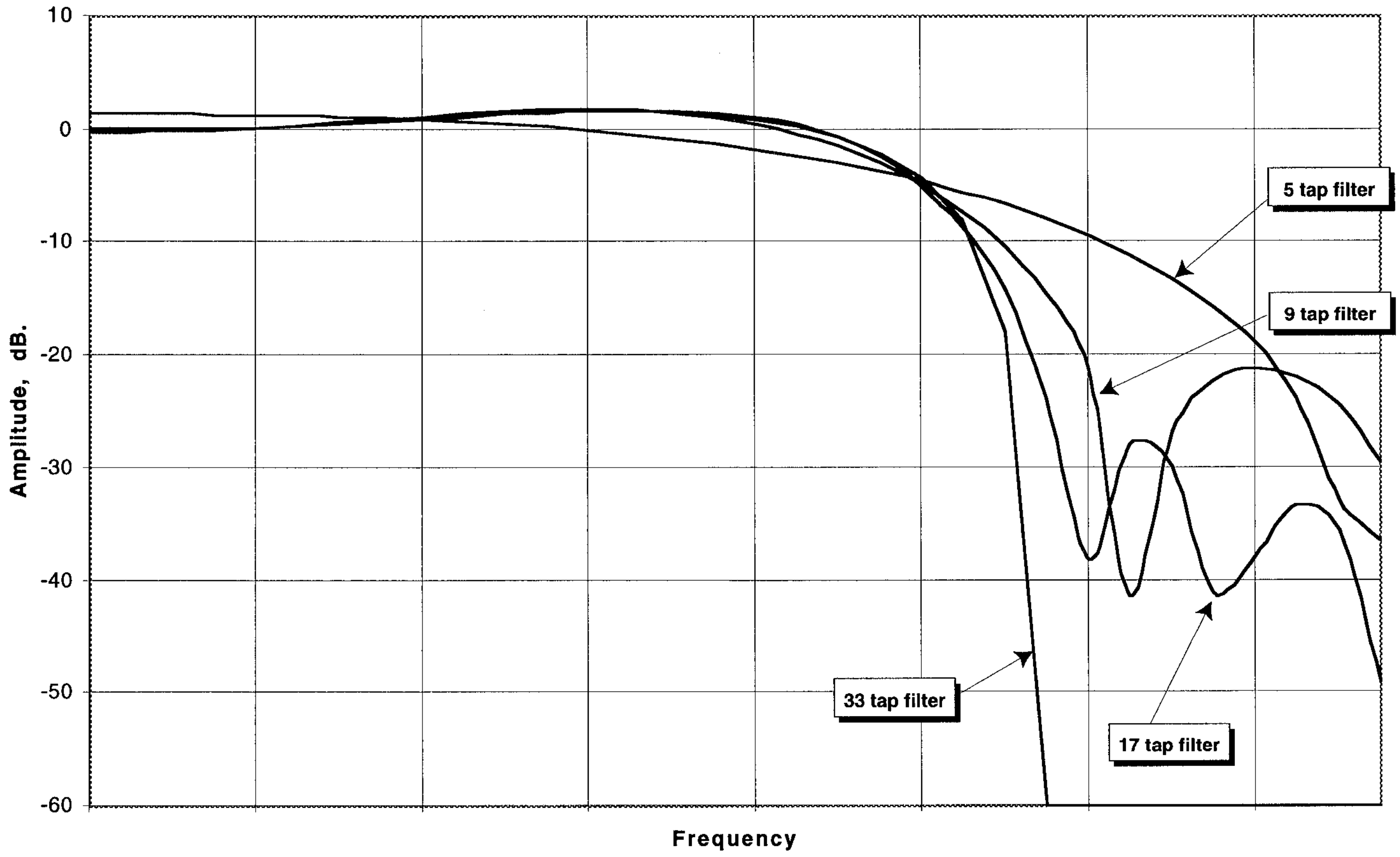


Figure 5-21 - *Effect of limiting the number of filter taps for alpha = 0.4 filter (log scale)*

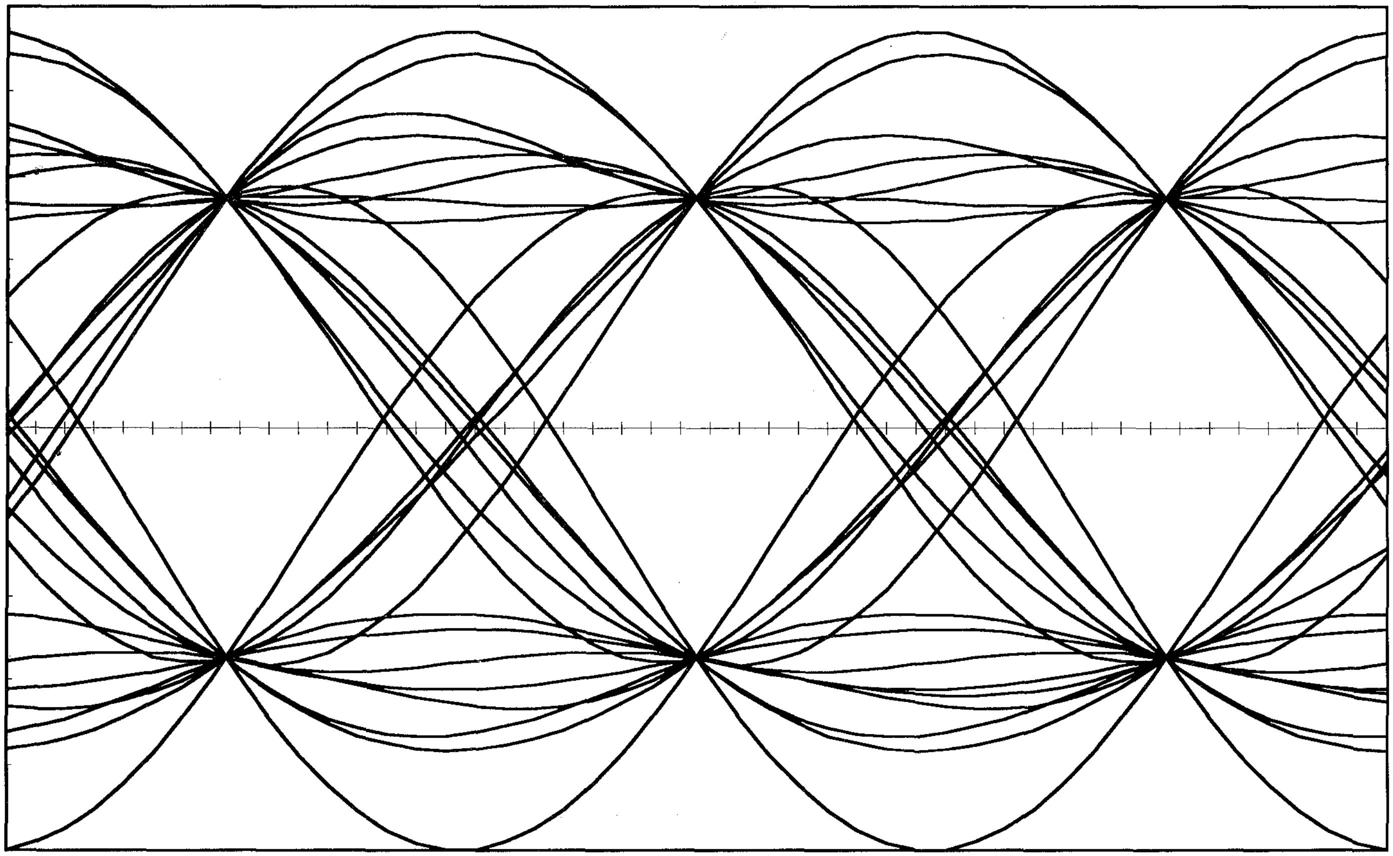


Figure 5-22 *- Eye Pattern - Raised cosine + x/sin(x) compensation alpha = 0.2, sequence = 2^5 - 1 PRBS 33-tap FIR filter equivalent*

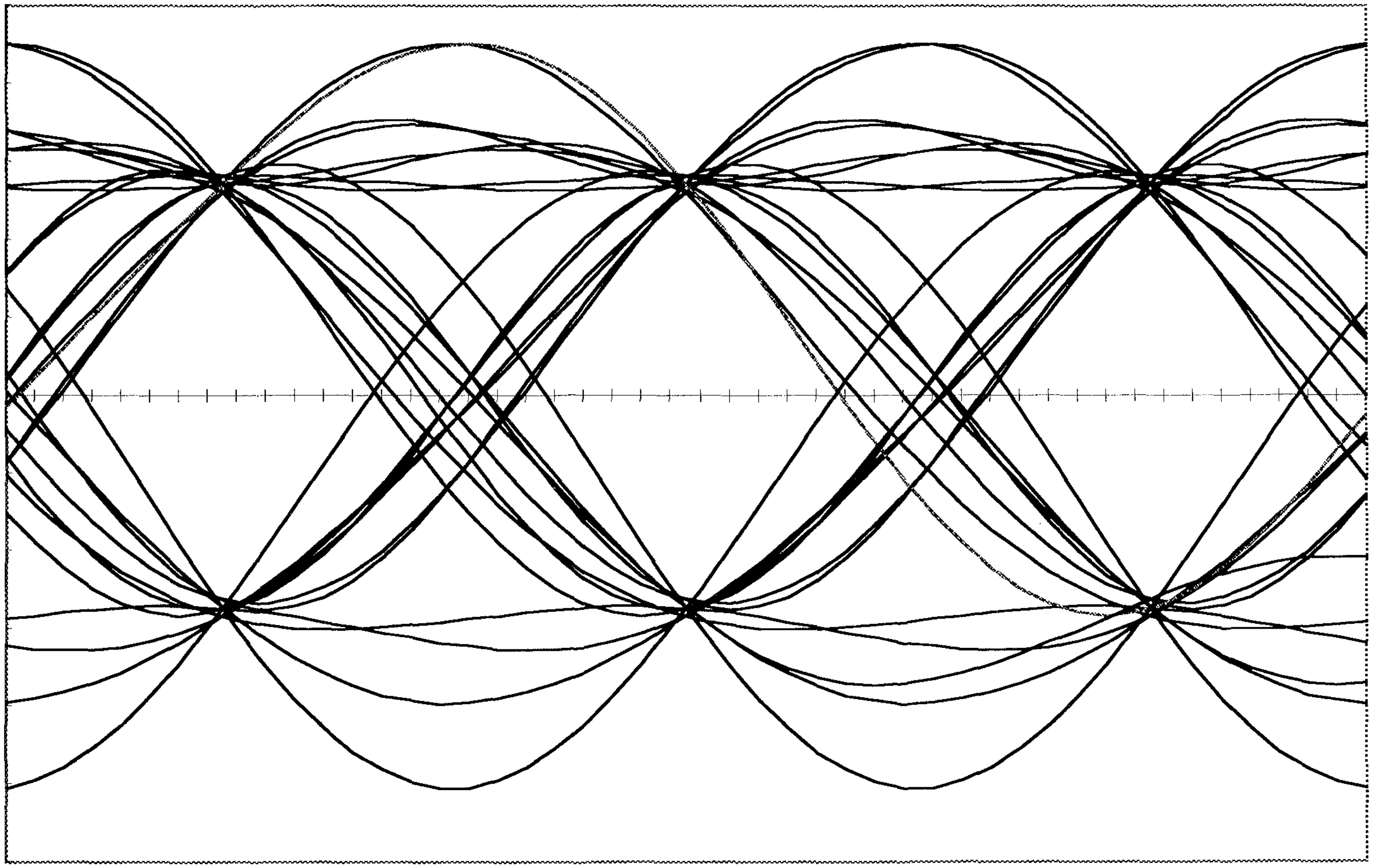


Figure 5-23 *- Eye Pattern - Raised cosine + x/sin(x) compensation alpha = 0.2, sequence =* $2^5 - 1$ *PRBS 17-tap FIR filter equivalent*

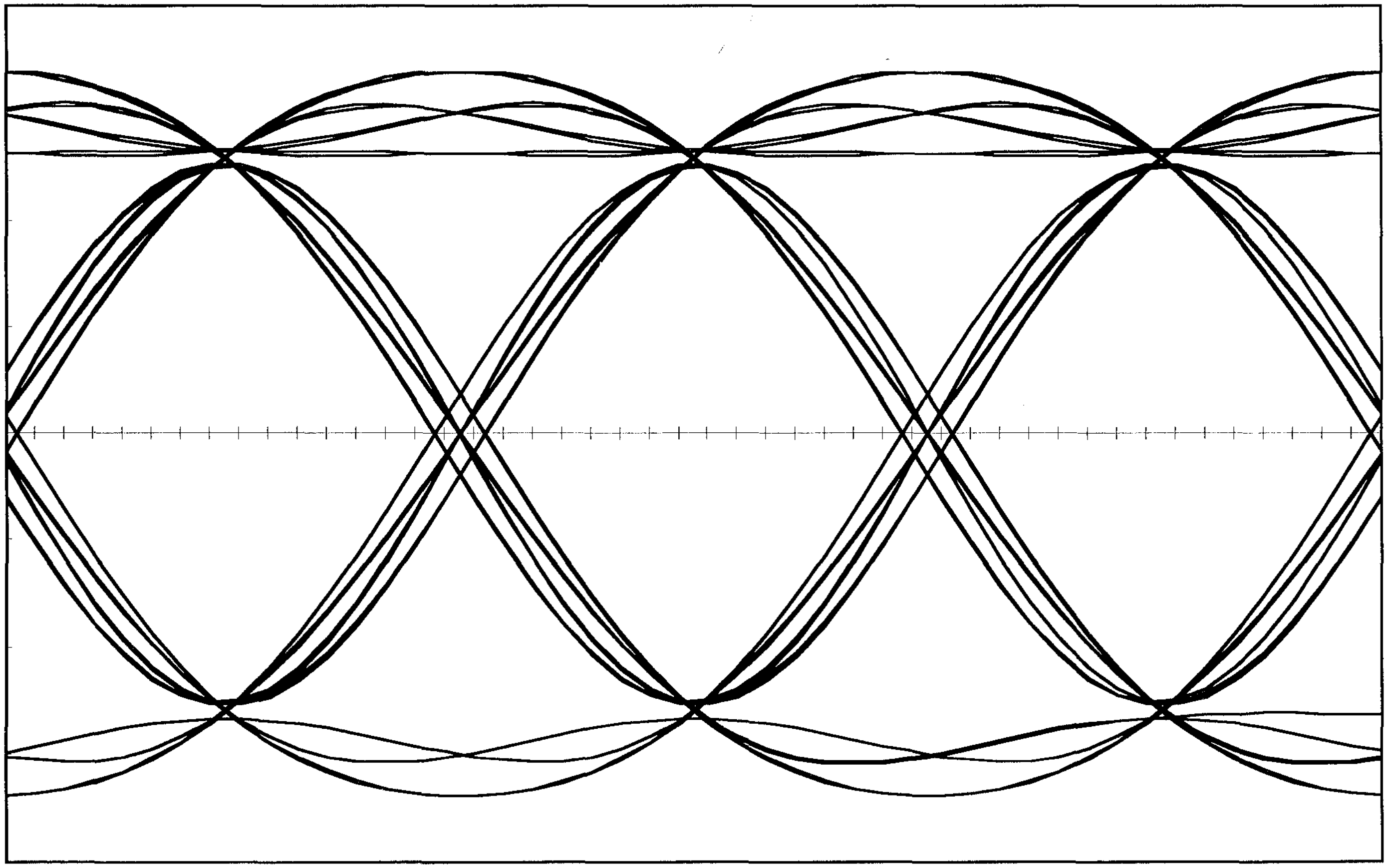


Figure 5-24 *- Eye Pattern - Raised cosine + x/sin(x) compensation alpha - 0.6, sequence = 2^5 - 1 PRBS Hamming-window truncation*

Detailed Step-by-Step Procedure

The following is a detailed procedure for the design of the transmit and receive filters. The included Excel files will help in determining the FIR filter coefficients, and will plot the eye pattern of the resulting filter.

To load the Fourier analysis package, it is necessary to first select Tools -> Add-Ins... and select the Analysis ToolPak option, and click OK. Then the transform is available as Tools -> Data Analysis -> Fourier Analysis. Use of the Fourier Analysis tool requires that a complete installation of Excel 5.0 have been performed.

1. Make a copy of the disks, and put the originals in a safe place. You will over-write the spreadsheets in the following steps. You can make a copy of any file you will alter, and rename it if you prefer, or you may prefer to make a copy and rename it before altering the file.

2. Determine the alpha-value of the desired filter response. If alpha of 0.2, 0.4, 0.6, 0.8, or 1.0 is chosen, then the calculations have already been performed, and are in the included spreadsheets.

3. Derive the x/sin(x)+square-root of raised-cosine transmit filter response. The spreadsheet RAICOSI.XLS contains the formulas. Substitute your value of alpha on this sheet for one of the supplied values, except alpha=0, which has had the formulas hard-coded. The range where you can substitute is C8, D8, E8, F8, G8. When you enter your value of alpha, the new value will replace the designated value in the headers in all locations on the sheet after the calculation. Do not chose an alpha value of 0.0, as some singularities occur which require manually adjusting the frequency response. Hit the F9 key to re-calculate the frequency responses.

4. You must manually execute the Fourier transform. To do this, select the TOOLS menu, and the DATA ANALYSIS... sub menu. This will load the data analysis package. Select the Fourier option. Depending on the column that you decided to put your custom value of alpha into, you must alter the commands given to the Fourier command. If, for example, you chose to enter the new value of alpha in location C8, then you must select C9:C520 as the input range, and I9:I520 as the output range. If you selected D8 as the location for the new value of alpha, then you must select

D9:D520 as the input range, and J9:J520 as the output range. Select the INVERSE transform mode, then click OK. The program will warn that you are over-writing some data, click OK. After the calculation is complete, hit the F9 key again to finish the computations.

5. The top of area of the chart shows the frequency responses for the following 4 filters:

    a) Raised cosine, uncompensated.
    b) Raised cosine + x/sin(x) compensation
    c) Square-root of Raised Cosine
    d) Square-root of Raised Cosine + x/sin(x) compensation

    Filter c) is the receive filter, and filter d) is the transmit filter. Filter b) is the total channel response, receiver plus transmitter, and is used in plotting the eye pattern at the receiver. You may select the various charts, which will show your value of alpha plotted with the others.

6. Decide the number of taps that you want to select. Files for 17-taps and 33-taps are built, and of the form EYEn-17.XLS, and EYEn-33.XLS, where n is the alpha factor previously computed for 0.2, 0.4, 0.6, 0.8, and 1.0, being EYE2-17.XLS, EYE2-33.XLS, etc. Select a file with the same number of taps as you want to copy. After it is open, select OPTIONS, CALCULATE, MANUAL. You must then open the impulse response file RAISCOSI, or whatever you named your customized version of the filter response and impulse response calculation file. For a 17-tap filter, you must select the impulse taps -64 through +64 from the Raised Cosine column with your new value of alpha. For a 33-tap filter, your must select -128 through +128 taps. Then select EDIT COPY, and then select WINDOW, and your EYE file. Select the square directly underneath the RC-Uncomp title (around C10) and then EDIT PASTE SPECIAL... When the PASTE SPECIAL menu pops up, select PASTE AS... VALUE.

7. Repeat the above by selecting the Raised Cosine + compensation column, selecting the appropriate number of taps as above, then changing to the EYE spreadsheet, and doing the PASTE SPECIAL... VALUES into square directly below the RC-Comp label (around B10).

8. Hit the F9 key. The status bar will show you the progress of the calculations. After they have completed, you can select the various charts, which will display the compensated and uncompensated eye patterns, and the time waveform for the compensated filter.

9. The FIR filter coefficients are the impulse response, as listed in step 5 above. In order to derive a filter from the above, you will have to take a subset of the samples. The above spreadsheets are set for 16x oversampling. If you choose 2x oversampling (i.e.: sampling at 19200 Hz. for a 9600 bit-per-second data stream) then you must discard 7 out of 8 values (since 16x is 8 times higher in frequency than 2x). Always keep the center sample at zero. In this example you will keep samples -64, -56, -48, -40, -32, -24, -16, -8, 0, +8, +16, +24, +32, +40, +48, +56, and +64, a total of 17 samples, for the 17-tap filter. For a 9-tap filter at 2x sampling, you would keep samples -32, -24, -16, -8, 0, +8, +16, +24, +32 of the impulse response. You can select other oversampling rates and filter lengths by following these rules.

In general, a low-pass filter for data should be setup with a cut-off of 0.25 of the baud rate or less, due to aliasing of the filter response above a frequency of 0.5. This corresponds to an oversampling rate of 2x with the responses generated by RAISCOSI.XLS. A realizable recontruction low-pass filter cannot remove the alias if it is too close in frequency to the desired passband.

Dolph-Chebychev Transmit Pulses

Certain modems (such as OFDM) transmit multiple carriers simultaneously. In this case, it may be necessary to not only provide good frequency filtering (so that the multiple carriers do not interfere with each other in frequency), but to also truncate the filter response rapidly in the time domain, so that pulses do not interfere with themselves later in time. One useful polynomial for generating the desired frequency and time limited pulses was derived by Dolph in his study on minimizing the sidelobes of an antenna array (Dolph, 1946). It turns out that the same mathematics apply to the problem of determining antenna array currents as to time- and frequency- limited transmit pulse shapes. The frequency response of a Dolph-Chebychev pulse has a principle lobe whose width is as small as possible for a given sidelobe ratio and given number of terms. The frequency response of the pulses, known as Dolph-Chebychev pulses, is described in equation (5-4), from an interesting article on filter design (Helms, 1971).

$$D(x) = \cos\left(P\cos^{-1}\left(\frac{\cos \pi x}{\cos\frac{\pi B}{2}}\right)\right) \tag{5-4}$$

Where x is the normalized frequency (from 0 to 1), B is the width of the main lobe of the frequency response, D is the amplitude, and P is the number of points in the impulse response less 1 (i.e.: if there are 17 points in the impulse response then P=16). The above expression involves computation of the arc-cosine of an argument outside the range of +1 to -1, which requires the use of the hyberbolic cosine function. A simplifying equation to computing the arc-cosine when the argument is outside the range of +1 to -1 is given in Helms (1968), and is described by equation (5-5).

$$\text{if} \quad |x| > 1 \quad \text{then} \quad \cos(P\cos^{-1}x) = \cosh(P\cosh^{-1}x) \tag{5-5}$$

Figure 5-25 plots the frequency response, D(x) obtained from equation (5-4) vs. the normalized frequency, x.

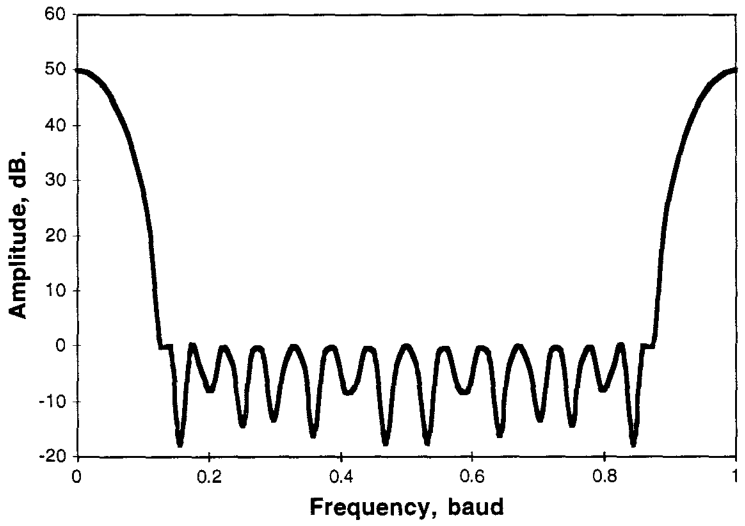


Figure 5-25 *- Frequency Response of Dolph-Chebychev filter with P=16 and B=0.25 baud.*

This filter has 17-points in the impulse response (P=16). The width of the main lobe is 0.25 baud, so the half-width is 0.125 baud. The -6 dB point is at 0.0625 baud. Figure 5-26 is the impulse response of figure 5-25 derived by using the inverse discrete Fourier transform.

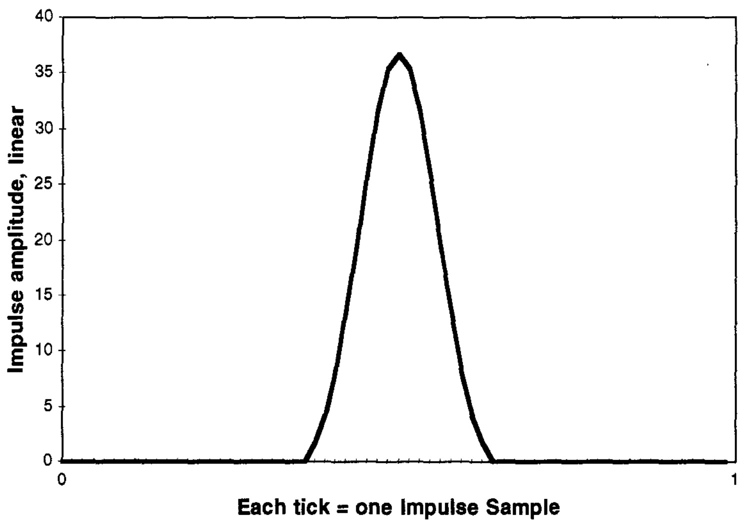


Figure 5-26 *- Impulse Response of the Dolph-Chebychev Pulse Shape of figure 5-25.*

It can be seen that the impulse response is zero everywhere outside the area of interest — thus, limiting the time that the data bit will ring through the filter. The Dolph-Chebychev pulse simultaneously limits the out-of-band frequency and time domain ringing of a pulse rather well. An Excel 5.0 spreadsheet for plotting the frequency and impulse response of Dolph-Chebychev pulses, while allowing adjustment of the number of impulse-response taps and the width of the main lobe is included on the disk — it is the program DOLPH.XLS. As you change the number of taps and the main lobe width, the out-of-band rejection will change — but it always remains flat, as in figure 5-25. The spreadsheet requires pasting the inverse Fourier transform data of locations B13:BM13 to locations B15:BM15 after each manipulation of the parameters (similar to the raised-cosine filters, above).

The receive filters must be carefully matched to the Dolph-Chebychev transmit filter, because none of the important Nyquist characteristics were considered in the design of the transmit filter. Normally, the Dolph-Chebychev filter will be used to filter raw square transmit pulses. Good transmit pulse-to-pulse isolation is achieved in both frequency and time on the radio channel with the Dolph-Chebychev filter. Additionally, the receive filter will have to compensate for both the sin(x)/x roll-off, and will have to establish the proper Nyquist criteria for zero ISI.

Because the D-C impulse response is completely zero outside the main pulse, it is also possible to eliminate ISI by spacing successive data pulses a sufficient time apart from one another. Then each isolated pulse, when convolved with the D-C impulse response, will have zero energy everywhere outside a time interval equal to the sum of the impulse length time plus the pulse duration. Alternatively, it is possible to set the duration of the data bit so that the zero-energy time of a bit arrives starting in the center of the succeeding bit. While these methods do not achieve the densest possible packing of bits in time (compared, especially, to other filter responses), it does result in zero-ISI, and may possibly lead to a simpler implementation of OFDM demodulation.

Chapter 5 - Reference

Bingham, J. "The Theory and Practice of Modem Design." John Wiley and Sons, 1988, Chapter 3.1.

Dolph, C. L. "A Current Distribution for Broadside Arrays which Optimizes the Relationship between Beam Width and Sidelobe Level." Proceedings of the IRE, Vol 35, pp. 335-348, June 1946.

Feher, K. and the Engineers of Hewlett-Packard. "Telecommunications Mesaurements, Analysis, and Instrumentation." Prentice-Hall, Inc., 1987, Chapter 1.2.

Helms, Howard D. "Digital Filters with Equiripple or Minimax Responses." IEEE Transactions on Audio and Electroacoustics, pp 87-93 VOL AU-19, No. 1, March 1971.

Helms, Howard D. "Nonrecursive Digital Filters: Design Methods for Achieving Specifications on Frequency Response." IEEE Transactions on Audio and Electroacoustics, pp. 336-342, VOL AU-16, No. 3, September 1968.

Chapter 6

Matched Filters

In the previous section, raised-cosine filters were discussed extensively, and claimed to be one with optimum channel filter response and good out-of-band transmit spectral suppression. It is also possible to utilize other channel filter responses. This section will discuss how to generate a good received filter response if some other pulse shape is desired (such as transmitting a rectangular pulse).

When a received data bit is filtered (usually at baseband), the lowest error rate is achieved when the received peak pulse signal is maximized compared to the received noise. The proof is a little complicated, but the result can be shown that the optimum receive filter response is:

$$H(\omega) = F^*(\omega)\ e^{-j\omega t} \tag{6-1}$$

Where H(w) is the frequency response of the optimum receive filter, F(w) is the frequency response of the transmitter plus channel, and F*(w) is the complex-conjugate of the frequency response of the transmitter plus channel. See Stremler (1982) or Schwartz (1980) for a derivation of the matched filter formula, equation (6-1). This formula states that the filter's impulse response should be the complex-conjugate of the time-reversed version of the transmit plus channel pulse shape. Although a little complicated sounding at first, it essentially means that the phase components should be made to cancel out (leaving linear phase through the combined filters), and that the magnitude of the impulse response should be a mirror image of the transmit filter's pulse response. The interpretation of this in the time domain is actually very straightforward. Recall, from the section on the Fourier transform, that the convolution of two impulse responses is the same as multiplying their frequency responses. The equation for convolution was given as:

$$y(t) = \int_{-\infty}^{+\infty} h(\tau)\ x(t-\tau)\, d\tau \tag{6-2}$$

To time-reverse the transmit impulse response, change the time relation of the signal x in the above equation. The results in the following equation:

$$y(t) = \int_{-\infty}^{+\infty} h(\tau)\ x(t + \tau)\, d\tau \qquad (6\text{-}3)$$

It can be recognized that this equation is the same as the cross-correlation of the two signals, h, and x. This is the same as saying that the time-reversal of the pulse response is exactly the same as computing the correlation of the received signal with the original transmitted pulse shape. The simplicity of this is that a standard FIR filter is capable of computing this correlation, and thus standard filter implementation techniques are easily used. More intuitively, it states that the best way to optimize the peak pulse received signal is to see how well the current received signal (bit) matches the transmitted pulse shape. Hence, the description of this filter as a *matched filter.*

It would be expected that the optimum received filter response (the matched filter) would then be dependent on the transmitted pulse shape, and that different transmit filters require the use of different receive matched filters. This is exactly the case.

Matched filter for Rectangular Pulses

One very common transmitted waveform is a rectangular pulse. It has been seen that a rectangular pulse generates a sin(x)/x frequency response. This simple transmit filter (really, no filter at all) generates a somewhat broad transmit spectrum. Nevertheless, it is commonly used where simplicity of the transmit filter is important. The transmit pulse shape is easy to describe in this case, and time-reversing the pulse shape is equally simple. Figure 6-1 shows the transmit pulse, the time-reverse received impulse response, and the result of filtering the rectangular transmit pulse with the time-reversed filter.

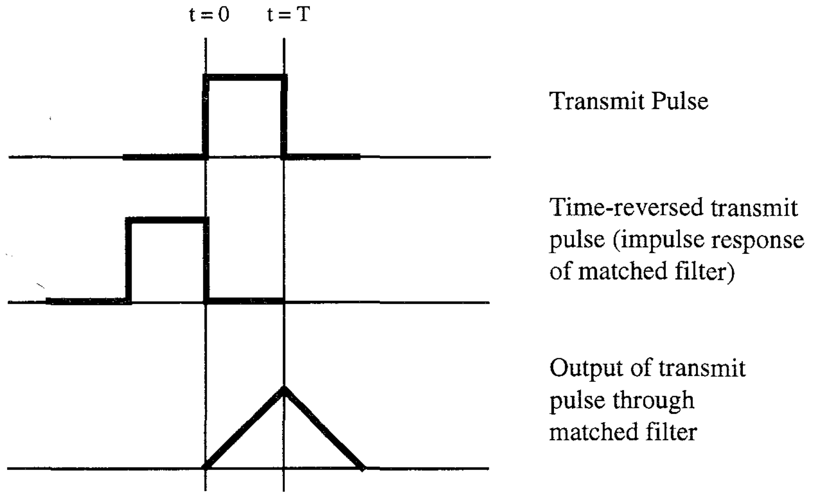

t = 0
t = T
Transmit Pulse
Time-reversed transmit pulse (impulse response of matched filter)
Output of transmit pulse through matched filter

Figure 6-1 *- deriving the matched filter for a rectangular transmit pulse. The output is the same as the cross-correlation of the received signal with the transmit pulse shape.*

Of course, the actual received matched filter cannot exist for *t* less than zero, so a real filter will have a finite delay (it will be causal), but usually this is insignificant, and not any problem. The output of the matched filter should be sampled at time=T, the maximum point of the output. The shape of the above output suggests a very simple implementation of the filter, one that can be done without DSP techniques. It is obvious that the output is a rectangular pulse that has been integrated: the first half charging the integrator, and the last half discharging the integrator. Because second half of the integration is not necessary since the decision is already reached at time=T, the second half can be discarded. Thus, the matched filter for a rectangular pulse is an *integrate and dump* filter.

In HF teletype, and in many HF FSK transmission modes (AMTOR, PACTOR, AX.25), the transmit baseband pulse is rarely filtered, assuming minimal distortion in the transmitter the receiver (primarily the IF filters). Then, an integrate-and-dump circuit is a matched filter, and should be a good choice for receiver baseband filter. One difficulty with the integrate-and-dump filter is that some kind of timing reference has to be extracted from the received signal to identify the time=T point. The time=T is exactly the bit interval, and so the problem is one of recovering symbol-timing, commonly called clock recovery. See the section on clock recovery for suitable techniques. Figure 6-2 shows an illustration of a simple analog integrate-and-dump matched filter. The time constant is not critical so long as it is significantly greater than the period of the data bit, T.

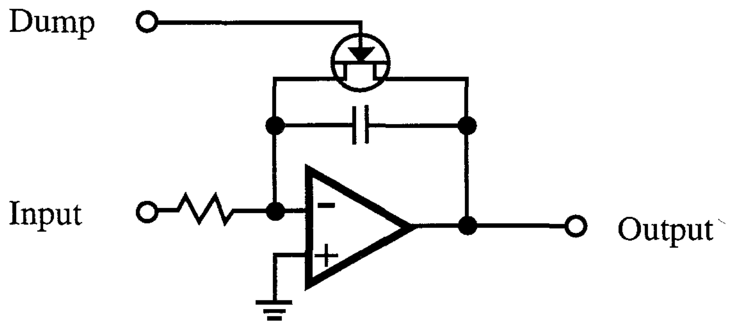


Figure 6-2 *- Simple analog integrate-and-dump circuit. This is a matched filter for rectangular transmitted pulses.*

Matched filter for Square-Root Raised-Cosine Transmit Pulses

In the section on raised cosine filters, the received filter was calculated. In fact, the square-root-of-raised cosine transmit and receive filters correspond to a matched filter pair. Recall that the transmit pulse shape was defined as x/sin(x) compensated rectangular pulses filtered by the square-root-of-raised cosine filter response. The resulting transmit pulse shape was derived by convolution of a rectangular pulse with the two transmit filter functions. But the x/sin(x) compensation of the rectangular pulse merely restored the frequency response of the transmitted pulse back to a flat response. So, the rectangular pulse and the x/sin(x) compensator cancel in the frequency domain. The net result is that the equivalent transmitter response is just the same as transmitting an impulse through the square-root of raised cosine filter. Therefore, the matched receive filter must be a time-reversed impulse response of the transmit square-root of raised cosine filter alone (which in the time domain is the same as the pulse response of the transmitter).

However, the impulse response of the square-root of raised cosine filter is completely symmetric in time. Thus, one cannot distinguish between a time-reversed and a non-time reversed copy of it. The impulse response of the matched receive filter is exactly the same as the impulse response of the transmit filter. The convolution or correlation, (it doesn't matter since the waveforms are time-symmetrical) of the two filters results in an overall raised-cosine filter response. See figure 6-3 for an illustration of these various effects, and the resultant response.

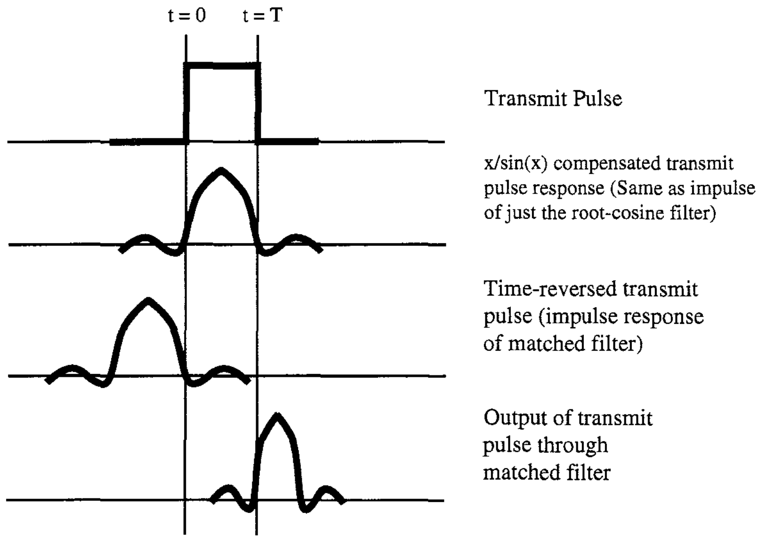

t = 0
t = T
Transmit Pulse
x/sin(x) compensated transmit pulse response (Same as impulse of just the root-cosine filter)
Time-reversed transmit pulse (impulse response of matched filter)
Output of transmit pulse through matched filter

***Figure 6-3** - deriving the matched filter for x/sin(x)-compensated square-root of raised cosine transmit filter.*

Indeed, in the section on raised-cosine filters, eye plots of the raised-cosine filters show the optimum received response (no inter symbol interference). What was not shown previously, but is now known, is that the filter optimizes not only the received signal correlation with the transmit pulse shape, but also the received signal as compared to the received noise. Given that a rectangular pulse plus matched filter has exactly the same properties, either filter response can be used. However, the transmitted spectrum of the square-root raised-cosine filter results in much narrower channel occupancy by the transmitter than the rectangular pulse transmitter would occupy.

Use of the Matched Filter Symmetry to Equalize a Channel

Now a simple rule exists to determine the optimum matched receive filter. Thus, it is possible to construct an arbitrary matched receive filter given the condition that the transmit baseband processing is not completely known by the receiver, or is excessively distorted solely by frequency response (including, possibly phase) errors. If the transmitter can be made to transmit a single isolated data bit (either an isolated one, or an isolated zero), then the transmit pulse response for that isolated data bit will appear at the output of the transmitter. To derive the matched receive filter, the algorithm is as follows:

1. Transmit an isolated data bit.
2. Record the received pulse response for that isolated data bit. This ideally should be a complex-impulse response so that phase errors (and thus group delay) can also be compensated. That is, the I- and Q- components of the received pulse response should ideally be measured. (See Appendix C-3 for I- and Q- notation)
3. Time-reverse then conjugate the received complex pulse response. This becomes the complex impulse response of the receive FIR filter.
4. If the phase of the transmit pulse shows minimum deviation from the linear phase condition, the amplitude of the complex part of the impulse response will be small. In this case, just the amplitude of the received real pulse response needs to be time reversed, and the conjugation operation is null. Otherwise, both the I- and Q- parts of the response must be utilized in the convolution operation. Appendix D discusses complex convolution.

The algorithm compares, in essence, how well a received pulse matches one that was used to train the receiver. When the received pulse exactly matches the pulse used to train the receiver, the correlation of the two is maximum. As a reminder, when implementing such a correlator in a DSP unit, the nominal value of the two data bits, *one* and *zero* are represented as +1 and -1 respectively. So, a received isolated *zero* bit that exactly matches a *one* pulse used to train the receiver has exactly the same correlation maximum as a received *one* pulse, except that the sign is the opposite. Figure 6-4 shows an illustration of the compensation of an arbitrary received isolated pulse.

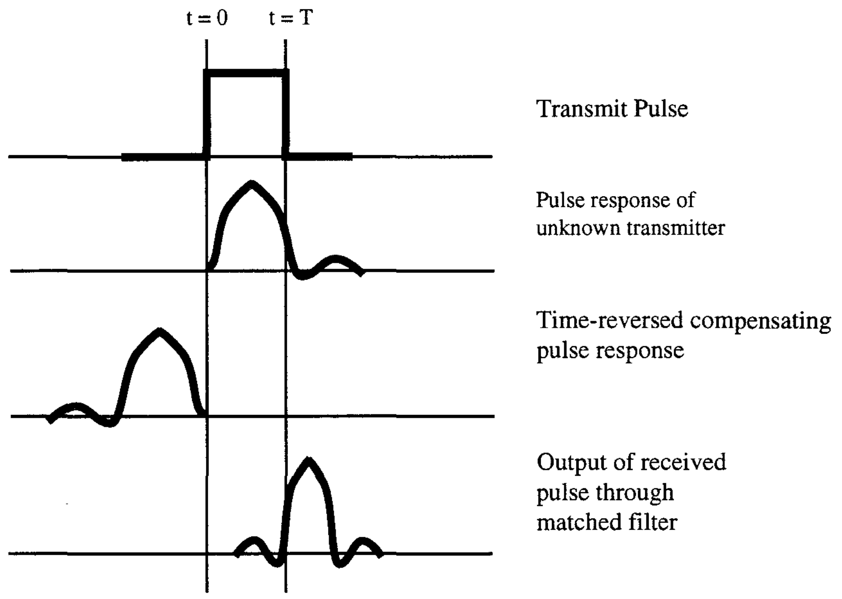

t = 0
t = T
Transmit Pulse
Pulse response of unknown transmitter
Time-reversed compensating pulse response
Output of received pulse through matched filter

***Figure 6-4** - derivation of the compensating matched filter response for a received isolated pulse from a transmitter with an unknown frequency response (a real-valued pulse response is assumed).*

This algorithm will get the receiver 'close' to the desired matched filter. If the received pulse used to train the receiver is noisy, then the fit with the received pulse will be poor. Several isolated pulses should be received and averaged with proper time-alignment of the pulses to assure that the true isolated pulse response emerges from the averaging. This averaged pulse is used as the stored reference pulse in the receiver as the matched filter. Once the receiver matched filter is close, another more optimized adaptive equalization technique can take over the adaptation of the filter response (such as a least-mean-squared steepest-decent gradient equalizer). However, given the wide variety and large deviation of received pulse shapes for current amateur modems, this algorithm may represent a significant improvement over current techniques that is readily attainable, fast to equalize, and simple to implement.

One difficulty that can arise with the use of the above algorithm is that it does not assure minimum inter-symbol-interference, since it does not assure that the frequency response meets the proper criteria. For multi-level signaling, where high accuracy is required, the algorithm is useful mostly to initialize an adaptive filter at a good starting place. Most adaptive filters attempt to minimize the inter-symbol-interference.

Chapter 6 - Reference

Stremler, Ferrel G. "Introduction to Communications Systems," 2nd edition. Addison-Wesley Publishing, 1982, Chapter 7.9.

Schwartz, Mischa . "Information Transmission, Modulation, and Noise," 3rd edition. McGraw-Hill Inc., 1980, Chapter 5-6.

Chapter 7

Data Codes

This section will discuss two basic types of data codes: simple data codes, and forward error correcting codes. We will look at both of these types of codes, and provide their definitions, benefits and drawbacks.

Baseband Data Codes

The first part will look at basic data codes, which usually serve one or more purposes:

1. *Synchronization* - they help delineate the serial bit stream into bytes, or other recognizable chunks, or they identify special control codes.
2. *DC-balance* - they help assure that there are about the same number of ones and zeros in the data stream. In general, baseband codes can help with spectrum control.
3. *Scrambling* - they help assure that transitions between ones and zeros occur relatively frequently, and make sure that there are relatively few long runs of consecutive zero bits, or long runs of consecutive one bits.
4. *Inversion insensitivity* - a transition code can make the decoding of data insensitive to whether or not it has been inverted (logic zero and one states interchanged).

Baseband codes can also have drawbacks, such as error extension, or error multiplication. The first code that is encountered commonly in amateur packet use is a zero-stuffing code. In the High Level Data Link (HDLC) transmission protocol, there is a need to be able to unambiguously distinguish between several special control characters and normal data characters. The zero-stuffing technique solves this problem very well. In HDLC, the FLAG character is the binary word 01111110. How do we distinguish this very important control character (which appears as the packet begin and end characters) from a binary data word of hexadecimal 7E? They both have exactly the same bit representation. In HDLC, the rule is "if a data character is being sent, and 5 consecutive ones occur, then insert an extra zero." Thus the transmitted stream for the 7E character (8 bits) would then become 011111010, a nine-bit sequence. Of course, the receiving station must delete the first zero-bit after five consecutive ones, in order to restore the original data stream. If the receiver should happen to encounter a data stream with 6 consecutive ones, it knows that it could not have been created from any data, but must instead be a control character. Usually, we do not need to be concerned with the zero-insertion and deletion that occurs because it happens inside the HDLC protocol serial chips that are used in packet TNC's. Usually, the modem design does not have to do anything special to interpreting this code.

A transition code can represent a particular data bit value as a change in the output bit stream, rather than the data itself. For example, suppose that every time the input data stream is a zero, then we change the state of the output bit stream. This type of coding is called NRZI - or non-return to zero, inverted. See figure 7-1 for a diagram of the coding process. The terminology for NRZ is not completely agreed upon, sometimes having different meanings in the datacom and telecom worlds. In the context here, we'll refer to the normal telecom definitions:

NRZ — non return to zero. The data bit is either high for the entire bit time (if it is a one), or else it is low for the entire bit time. This is the manner in which uncoded logic signals are generally drawn and described (although in digital logic the states are usually 1 and 0, while as carrier modulation they may be +1 and -1). If the two valid states are +1 and -1, then the signal never assumes the zero state.

NRZI — non return to zero inverted. NRZ data per the previous definition, that has been coded with a one-bit transition code. Essentially, when the NRZ data is a one, the NRZI data does not change state; and when the NRZ data is zero, the NRZI data toggles state.

Generally, we will only be interested in NRZ and NRZI as simple data codes in amateur radio work.

Figure 7-2 is the schematic diagram of an NRZ to NRZI encoder (sometimes called a coder), while figure 7-3 is the schematic diagram of an NRZI to NRZ decoder. The truth table for each is shown in the figure. Note that the output could just as easily have been inverted, depending on the starting state of the flip-flop. The decoder looks for transitions — that is a *one* bit when the previous bit was a *zero*, or vice versa to signal that the coder has encoded a *zero* bit. Notice that since the encoder puts out a transition in place of a *zero*, the decoder correctly recovers the original data stream even if the encoded data stream is accidentally inverted along the way. This type of data inversion could occur if we did not pay careful attention to the polarity of the receive detector or if we accidentally put the local oscillator of the receiver on the other side of the received carrier (for FSK). Using NRZI means that this would not matter. It eliminates the problem of upside-down data.

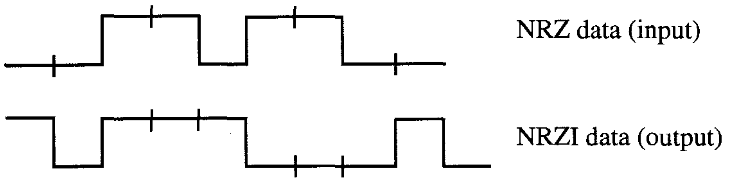


Figure 7-1 *- NRZ to NRZI encoding process.*
A logic zero on the input causes a transition in the output state.

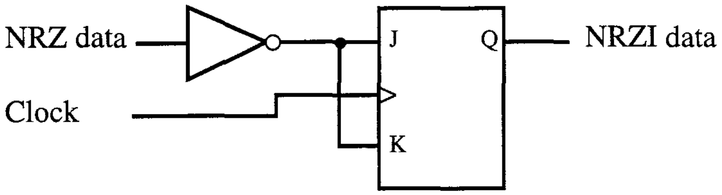


Figure 7-2 *- NRZI encoder.*
A logic zero input toggles the flip-flop.

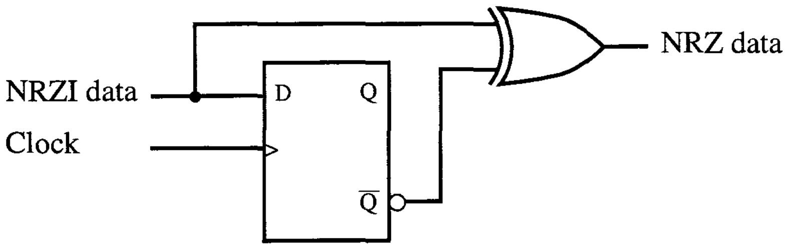


Figure 7-3 *- NRZI decoder.*
A transition in the input results in a logic zero output, no transition results in a logic one output (undoing the action of the encoder).

DC-Balance (One's Density) of the Code

A subtle problem can occur when using HDLC, zero-insertion and/or transition encoding of the data. It is possible to generate a coded data stream that, on the average, does not contain the same number of ones and zeros (deviates from 50% one's density in an NRZ code). Is this important? It can be if any element in the baseband path between the data source and the data sink is AC coupled. This is because as the long-term density (half-one's-bits and half-zero's-bits is defined as 50% one's density) of the one's-bits changes, the average DC value of the signal is changing. If this signal is AC coupled, then this is the same as high-pass filtering the data. This can result in significant performance degradation of modem circuits if care is not taken. Refer to the section on filter response functions to see the large performance loss that is incurred for a high-pass filter -1 dB. cut-off frequency of 0.031 baud (equivalent to a -3 dB cut-off of 0.015 baud). Figure 7-4 illustrates uncoded and NRZI coded HDLC flag characters. Note that both are significantly unbalanced in DC content. One approach to avoiding this problem is to scramble the data. An additional problem can occur if the slicing signal reference detector can be biased by a data stream with an unequal number of 'ones' and 'zeros'. A simple slicing level detector such as a low pass filter will suffer a bias in this condition. See the section on slicing level recovery for more information on this problem.

`01111110011111100111111001111110`	Uncoded HDLC flag characters
`.00000001000000010000000100000001`	NRZI coded flag characters

Figure 7-4 *- Uncoded HDLC, and NRZI coded HDLC flag characters contain an unequal number of one and zero bits - resulting in DC imbalance, and potential problems with AC-coupled circuits.*

Scrambling

A multi-stage scrambler works very much like the single bit transition encoder (NRZI encoder) described above, except that it depends on the value of several past data bits. Also, it keeps a memory about what has been coded previously. The decoder can unambiguously decode the stream by referring to older data bits and using their value in addition to the currently received data bit in order to compute the correct unscrambled data bit value. Appendix B Figures B-4 and B-3 show the schematic of a 5-bit scrambler and descrambler, respectively. There are a number of scrambler and descrambler codes that can be used, depending on some statistical properties that may be needed. It should be noted that the scrambler and descrambler theoretically could be interchanged, and would operate correctly, but this is not done for a very good reason — error circulation. Notice that the encoder (figure B.4) has feedback that causes data bits to recirculate in the shift register. If we were to put the encoder at the receive side (and the decoder at the transmit side), then a single received data bit that was wrong would essentially "get trapped" in the recirculating encoder shift register causing errors to be generated repeatedly. Eventually other wrong data bits would just happen to cancel it out. The decoded error rate would be enormous. So the encoder is placed at the transmitter side, where this problem won't occur. One of the disadvantages of scramblers and descramblers is that they cause error multiplication. This is because we look at several received bits in order to compute the current one. So, if any of those previously stored bits is wrong, then the current bit will also be wrong. For example, if we look at the current data bit and 2 previous bits (see figure B.3), then a single bit error will ultimately impact three data bits at the decoder, resulting in an error multiplication factor of three.

Error multiplication may or may not be a significant problem depending on the strategy that the data link uses to correct errors. If the link has no method of error control or retransmission, nor any method of error detection, then error multiplication is very undesirable. In essence it degrades the performance of the modem substantially. However, if the link uses AX.25 or an ARQ strategy, for example, then the receipt of a single wrong bit means that the entire received packet is defective (will fail the CRC check) and will be thrown out. The fact that either bit or three bits are wrong will not have an adverse detrimental effect on the performance of the modem.

An additional advantage to scrambling the data is that the scrambler shift register, if correctly designed, generates a maximal-length binary pseudo-random sequence, which we are exclusive or-ing (multiplying, or mixing) with the data. This maximal length sequence has some very desirable statistical properties, one of which is that it produces a very flat spectrum. When this is mixed with very non-random data, it tends to generate a spectrum that has fewer isolated spectral peaks and more wideband spectral energy. This can be advantageous in minimizing interference to other receivers listening near to the same frequency as our transmitter. Figure 7-5 shows the spectrum from a pseudo-random sequence. Additionally, scrambling minimizes the probability that most of the encoded bits will be *ones* or *zeros*. Scrambling produces an encoded stream that will average about 50% *ones* density over the long term, and will not tend to deviate too far from 50% density (see section on slicing level recovery).

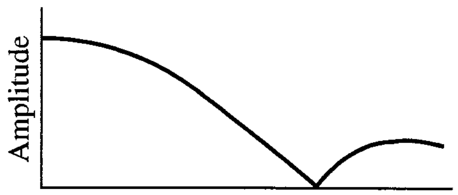


Figure 7-5 *- spectrum produced by a pseudo-random bit sequence.*

Scrambling renders the data unreadable on immediate inspection, but it is not considered a cipher by the FCC. You must, however, disclose the algorithm that you are using if you choose to scramble. Since the receive decoder self-synchronizes in the type of scrambler we have shown in figures B.4 and B.3, anybody can easily decipher the meaning of the data, provided that they have the algorithm (or the schematic). There are no special key or codewords that make the information hidden in any way, so the use of a scrambler does not violate the FCC prohibition on codes and ciphers in amateur telegraphy.

In summary, baseband codes can help us to achieve synchronization, inversion insensitivity, balanced DC content, and spectral flatness.

Forward Error Correcting (FEC) Codes

Another class of codes that we may wish to apply to our data are those that add redundancy to the transmitted data stream. This redundancy can be used, if properly designed, to detect and correct bit errors in the received data stream. Of course, when we add redundancy to data, it is necessary to increase the number of bits that are sent. As a result, the performance of the receive modem will deteriorate due to the increased noise bandwidth of the faster-rate data stream. Interestingly enough, however, we actually gain more in terms of reduced errors due to the detection and correction of bit errors than we lose due to the wider bandwidth. This lucky happenstance is called *coding gain*, and it can vary from about 2 dB for simple block codes up to as much a 9 dB for very complex soft-decision convolutional codes. Additionally, FEC can also dramatically improve the performance of modems in the presence of low-duty cycle high amplitude impulse noise (which is decidedly non-gaussian), such as that produced by some radar systems, both UHF and HF.

We'll first look at some approaches used in error detection that seem obvious, but don't work very well. Then we can derive some simple statistical methods that will let us compute the performance of these codes in terms of their bit error rate performance. Next we'll look at the linear block codes and make some estimates of how well they should work. Later, we'll examine some of the more complex codes.

One of the simplest schemes to correct errors is to send the data twice. Unfortunately, this has only the advantage of detecting that an error occurred. Nothing would tell us which of the two received characters was correct. At the expense of halving the throughput (or requiring us to double the data rate to maintain a similar throughput) we have only provided a method to detect errors. It turns out that much better and more efficient codes are possible. We'll look at 3 of these codes: linear block codes, Reed-Solomon codes (a type of linear block code), and convolutional codes.

Linear Block Codes

A linear block code is one where we take a block of data bits, and several parity check bits, and produce a new block as a result. For example, we can generate a simple code by taking 4 data bits, and 3 parity bits, and together making a 7-bit block. We will designate this as a (7,4) code, meaning that the total block size after coding is 7-bits, and 4 of those bits are our original information bits, obviously (but without stating it) leaving 7-4 = 3 parity bits. A code is linear if and only if the sum of any two codewords is always another codeword. Our code requires that all the blocks be codewords in a set, and the set be linear. For example, the following is a linear code set:

```
00
01
10
11
```

If we add any word in the set, for example 01, to another word in the set, say 11, we get the result 00 which is in the set (our addition wraps-around without a carry). This is also a linear code set:

```
000
010
100
110
```

because the sum of any two codewords is also a codeword. The following is not a linear code set:

```
000
001
011
101
111
```

because if we sum 001 and 011, we get 100, which is not one of the codewords in the set.

How would we produce the 3 parity bits so as to detect any single bit error (regardless of whether the wrong bit was a data bit or a parity bit)? Further more, how would we be able to correct one of the original 4 data bits if it happened to be the bit in error? To do this, we will look at this (7,4) code in some more detail.

In binary logic, the operation we will use is the exclusive-or gate, or XOR. A 2-input XOR gate, with inputs A and B, produces the truth table shown in table 7-1.

Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	0

Table 7-1 *- Exclusive-OR truth table for 2-bits.*

Although the notation is usually different in boolean logic, the plus-sign will be used to denote the exclusive-or operation in this section on coding. Thus, C = A + B will mean that C is the output produced by exclusive-or'ing A and B.

Next, parity will be defined. Even parity is called 0-parity, and odd parity is called 1-parity. That is, the parity of an even number of one-bits is even, and the parity of an odd-number of one bits is odd. For example, if we have a block of bits 1101, then its parity is odd, since there are 3 one-bits in the block. The block 0011 has even parity, since it has 2 one bits. The block 0000 has even parity since it has no one bits, and none is considered even (in this context). It is easy to compute the parity of a block of bits by using exclusive-or gates. For example, assume that the we want to compute the parity of a 2-bit block, then the two bits are input to an exclusive-or gate. As indicated in the truth-table above, the output is a *one* when the parity is odd (a single *one-bit*), and a *zero* when the parity is even (neither or both *one* bits). The concept can be easily extended beyond 2 bits, by cascading two-input exclusive-or gates. Figure 7-6 is a 4-bit parity calculator.

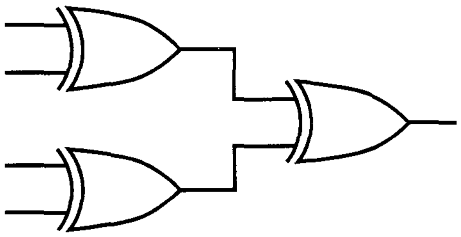


Figure 7-6 *- 4-bit parity calculator circuit*

If there are 0, 2, or 4 *one*-bits on the input, then the output will be *zero*, conversely, if there are 1 or 3 *one*-bits on the input, then the output will be *one*. This circuit could be extended to any number of bits, even or odd, and it would generate the parity of the block. Figure 7-7 is another way to draw a 4-bit wide parity calculator, although it means exactly the same thing as the previous figure. It is possible to generate an even-parity block by adding a single parity bit to another block. For example, if we had the block 100, and calculated the parity (1 in this case) then we could form the 4-bit block dddp, or 1001. This block would have even (or zero) parity, since the new, larger block has 2 *one* bits in it. So, to generate an even-parity block, we add odd-parity to the original block, making it one bit longer. To generate an odd-parity block, we add even-parity to the original block, making it one bit longer.

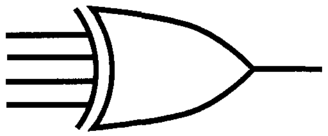


***Figure** 7-7 - alternative representation of 4-bit parity calculator*

Now we can represent the operations that we will perform on our 4-bit information word that we want to add 3 parity bits to. We will call our 4 information bits *data* bits. Assume that the data bits are D0, D1, D2, and D3. We define the parity bits as P0, P1, and P2.

$P0 = D0 + D1 + D3$ (P0 is the exclusive-or of D0,D1, and D3)
$P1 = D0 + D2 + D3$
$P2 = D1 + D2 + D3$

Now, if we add the parity bit to the three data bits, making a larger block so that the resultant larger block has even-parity, then the 3 equations above can be restated as:

$P0 + D0 + D1 + D3 = 0$ (exclusive-or'ing P0, D0, D1, and D3 is always zero)
$P1 + D0 + D2 + D3 = 0$
$P2 + D1 + D2 + D3 = 0$

Figure 7-8 is a circuit diagram of the above 3 equations, which forms the generator for the (7,4) linear Hamming block code.

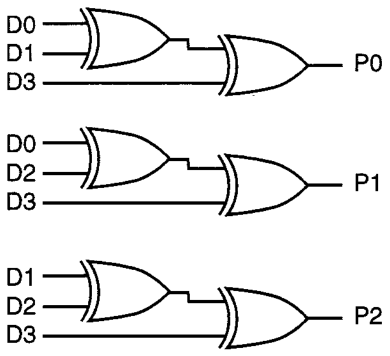


Figure 7-8 *- Parity generator for (7,4) Hamming linear block code.*

We now transmit the 7 bits, which is composed of the 4 data bits and the 3 parity bits in a special order, P0 P1 D0 P2 D1 D2 D3. To calculate for the presence of received errors, we know that we should get an even parity calculation when we calculate the three equations above. Figure 7-9 is the error decoder based on those three equations:

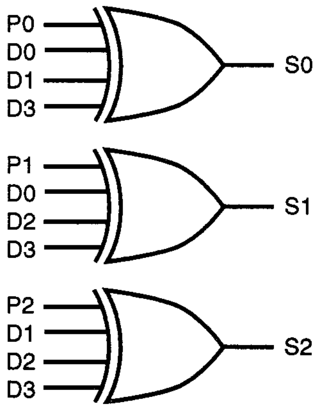


Figure 7-9 *- Error Decoder for (7,4) Hamming code*

Now, since P0, P1, and P2 were generated such that they should yield even parity with their bits, then we expect S0, S1, and S2 to each be zero if there are no errors in the received word. Let's look at the value of S0, S1, and S2 for each of the possible 7 single-bit errors that can occur (any of the 4 data bits, or any of the 3 parity bits could be wrong). W = wrong (received in error), R = right (received without an error). This is shown in table 7-2.

P0	P1	D0	P2	D1	D2	D3	S2	S1	S0
R	R	R	R	R	R	R	0	0	0
W	R	R	R	R	R	R	0	0	1
R	W	R	R	R	R	R	0	1	0
R	R	W	R	R	R	R	0	1	1
R	R	R	W	R	R	R	1	0	0
R	R	R	R	W	R	R	1	0	1
R	R	R	R	R	W	R	1	1	0
R	R	R	R	R	R	W	1	1	1

Table 7-2 - Error Syndrome for (7,4) Hamming Linear Block code.

Notice that the three bits, S2, S1, and S0 have a numerical value that "points" to the location that is wrong, or they have the value 000 if all the bits are right. We call S2,S1,S0 the 'syndrome' of the error, and it is abbreviated simply as S. In the fourth row of the table, D0 is the wrong bit, and S = 011, (=3) so S points to the 3rd bit, which is D0. It is merely necessary to invert the bit that S points towards in order to correct the error. Obviously, it's not necessary to bother with inverting any of the parity bits, since they are thrown away before handing the D0 D1 D2 and D3 data bits to the user. The three parity bits went along on the ride from the transmitter to the receiver, and may have been corrupted by noise, but they have done their job after they help to decode the syndrome word. If we look at all possible codewords that can be generated by valid combinations of the 4 data bits and the 3 parity bits, we notice that they form a linear code set. Any single-bit error causes a codeword that is not in the linear set, so we know it is invalid. That is why this type of code is a linear block code. Figure 7-10 is the schematic of a circuit that corrects the data bits based on the computed syndrome word.

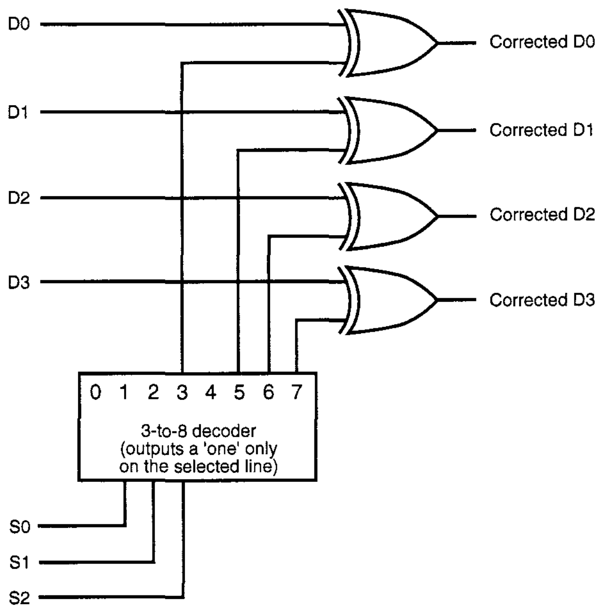


Figure 7-10 *- Error corrector circuit based on the syndrome word, S*

We can form similar codes with more parity and more data bits. The following table lists 4 codes that are in the practical range of consideration. One important item we can calculate is the amount of overhead each code requires. The (7,4) code has an overhead of 3 parity bits for 4 bits of data, or 3/4 = 75%. Four block codes are illustrated in table 7-3.

Code Name	# Data Bits	# Parity Bits	Codeword Size	Overhead
(7,4)	4	3	7	75%
(15,11)	11	4	15	36.4%
(31,26)	26	5	31	19.2%
(63,57)	57	6	63	10.5%

Table 7-3 *- Four linear block codes, and their properties.*

It would seem that the larger codes are good because they are much more efficient, and waste fewer parity bits per data bit. However, we can sense intuitively that a larger codeword has a higher probability of having two bits within it corrupted by transmission (and the above codes cannot correct multiple bit errors, only single bit errors). So, we have a tradeoff in terms of the performance of the code versus it's size. We will now calculate more precisely what this performance tradeoff is.

AWGN Performance of Linear Block Codes

When we calculate the performance of a block code, we will assume that the corrupting source is Additive White Gaussian Noise (see the AWGN section for more details). Recall that AWGN gives an equal probability that any bit is wrong, depending on the signal-to-noise ratio. In order to be fair in our comparison, we need to normalize both the non-FEC and the FEC corrected systems to the same throughput. Since we add redundancy to the FEC-corrected code by adding parity bits that we must transmit, we need to state the throughput of only the data part of the code. We must then calculate the increased bandwidth of the code plus parity bits. In the case of the (7,4) code, if we assume that the uncorrected system operated at 9600 bits per second then the FEC-corrected system would operate at 9600 * 7/4 = 16,800 symbols (bits) per second. All other things being equal, the FEC-corrected system would require a receiver with wider bandwidth in order to decode the 16,800 bps serial stream. Based on AWGN, we can assess the penalty, in dB., of the wider bandwidth as

$$Penalty = 10 \log \left(\frac{B_{FEC}}{B_{non\text{-}FEC}} \right) \qquad (7\text{-}1)$$

In this case, the noise bandwidth penalty for the faster rate is 2.43 dB. Obviously the larger codes have a smaller penalty since their excess bandwidth is smaller than the excess bandwidth of the (7,4) code.

Statistical Performance of Linear Block Codes

In contrast to the loss of performance due to increased noise bandwidth, we gain performance due to the single-bit error correcting capability of the code. Hopefully, this gain more than makes up for the loss due to the noise bandwidth increase, otherwise it's a pretty useless code! Since the probability of any bit being in error is always the same (AWGN assumption), we can calculate the probability of a block of bits being correct for the non-coded and the coded cases. If we assume that k=the number of bits in the block (data+parity) and n=the number of data bits alone, then using the binomial distribution, we can state:

$$P_{block\text{-}correct} = (1 - BER)^{n} \qquad \text{For the non-FEC case} \qquad (7\text{-}2)$$

and

$$P_{bit\text{-}correct} = (1 - BER)^{1} \qquad \text{For a single bit (non FEC)} \qquad (7\text{-}3)$$

Where BER is the bit-error-rate, or the probability that any one bit is wrong, and we calculate the probability that the entire block is correct. Similarly, we can calculate the probability that a code block is correctable (has no errors or one error) as:

$$P_{correctable} = (1 - BER)^{k} + k \bullet BER \bullet (1 - BER)^{k-1} \qquad \text{For the FEC-coded case} \qquad (7\text{-}4)$$

The first term is the probability that all the data+parity bits are correct, and the second term is the probability that one and only one bit (either kind) is wrong. Next, we have to calculate the BER given the signal to noise ratio (discussed in the section on AWGN). Now we can generate a BER vs. SNR plot (as we did in the AWGN section). We have to normalize the block error rate back to the bit error rate, since block errors are not what we are comparing, but rather bit errors are what we are comparing between the two cases. Figure 7-11 is a BER vs. SNR plot of coherent 2PSK, for non-coded case, and for the four codes listed above. We see that each of the codes is actually a net improvement, since the error-correction gain outweighs the noise bandwidth loss. However, we see that the (63, 57) code appears to be the best performer of the four as far as coding gain is concerned. The improvement in the SNR, in dB., at a given BER is called the coding gain of the code. We have to specify at what BER we measure the coding gain, since we get different values of coding gain at different values of BER. The (63, 57) code has about 2 dB. of coding gain at 10^{-6} BER compared to the non-coded data.

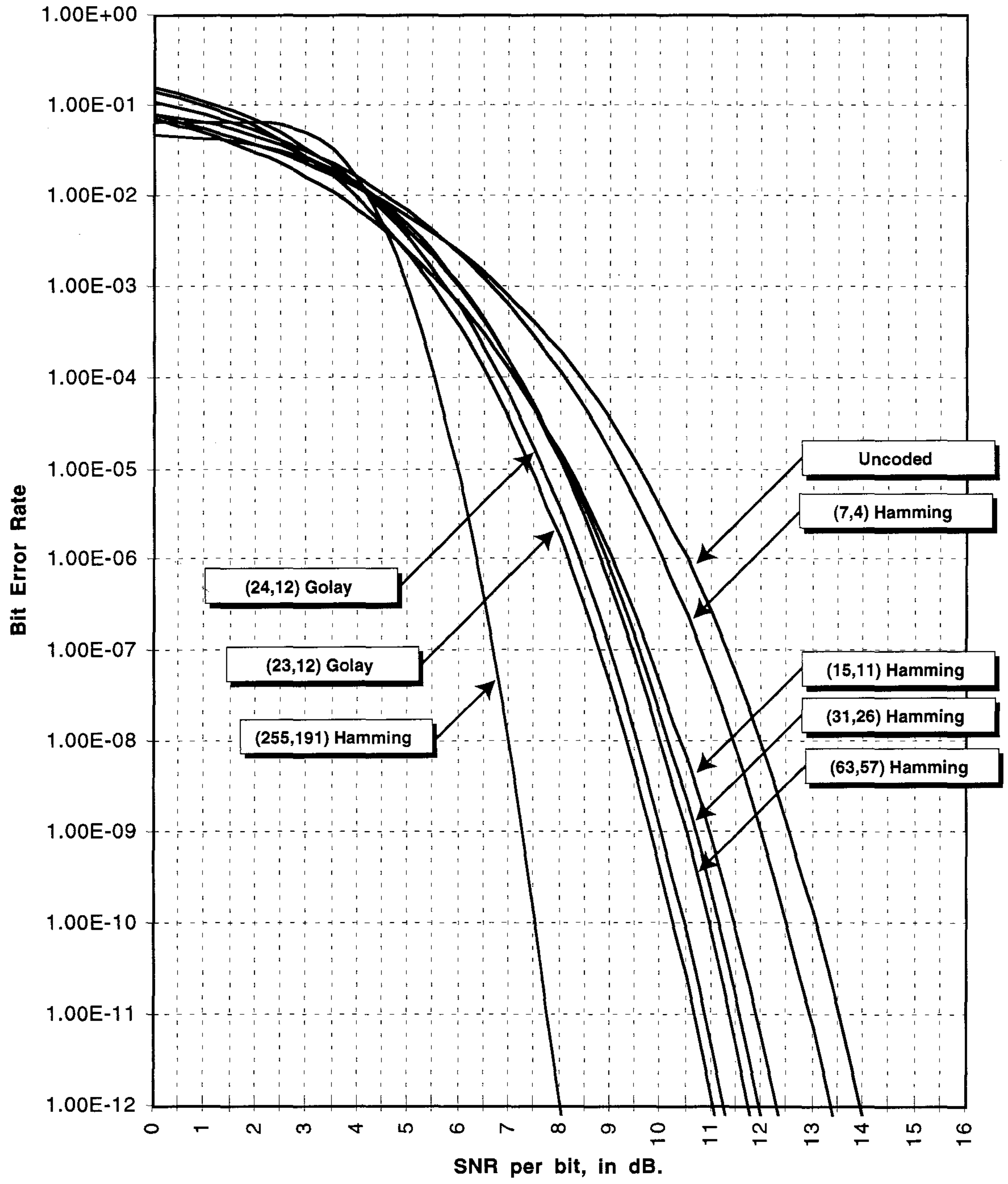


Figure 7-11 - *Bit Error Rate vs. SNR per bit*

Other Properties of FEC Codes

Another desirable property of an FEC code is that it also corrects well for impulse noise. Impulse noise is an occasional noise pulse that is very strong, unlike the AWGN that we studied earlier. If the noise impulse exceeds the data bit amplitude, then we will have a data bit error (half the time). In this case, the impulse noise leads to an error floor. This means that we cannot measure a traditional BER curve, since increasing the signal strength does not decrease the error rate because the impulse is stronger than the data for any reasonable amount of power. However, the FEC-code works very well in this case. Even though we get periodic errors, the FEC code will correct a defective data bit within the block. In this case, FEC turns a non-usable circuit path into a perfectly reliable one! Provided, of course, that the impulse obscures one and only one bit within each block, and that noise does not additionally corrupt another bit within the block that was corrupted by the impulse. Unfortunately, this is not the case, since the noise is non-synchronous with the impulse. Therefore, times will occur when both errors happen in the same block, and we have two errors in one block (which is non correctable). In looking at the charts, the (7,4) and (15,11) codes have the best performance in an impulse-noise environment. This is in contrast to the AWGN performance, where the (63, 57) code has the best performance. From this we can make a general statement: Different codes are better in different ways, there is no one "best" code.

Interleaving the Codewords

We can perform a clever trick, interleaving of the code words, on the blocks that assist us in resisting a cluster or burst of errors. For example, in the case of the (7,4) code, instead of transmitting all 7 bits of the codeword consecutively, we could decide to generate, say, 10 codewords (of 7 bits each) and store them up until we had calculated all 70 bits. Then we could transmit the first bit of codeword number one, then the first bit of codeword number two, etc., up through transmitting the first bit of codeword number 10. Next, we could transmit the 2nd bit of codeword number one, the 2nd bit of codeword number two, etc., up through transmitting the 2nd bit of codeword number 10. Then we transmit the 3rd bit of codeword number one, and we repeat this process until all 70 bits have been sent. We've done nothing except change the order in which we have sent the 70 bits. Each codeword has its 7 bits spread out amongst the other ones. Now let's say that our impulse happens to be two bit-times long. No matter where in time this noise impulse occurs, it does not destroy two bits within the same block. It may destroy two bits, but they will belong to different blocks. In fact, with the 10-way interleaving we have done here, any noise impulse of 10-bit times, or less, will assure that no single codeword contains more than one error due to that impulse. We can then decode the ten different blocks, and each block will be correctable, since each block will contain only one error bit. Then we reassemble the data bits back together in the correct order and deliver them.

In this manner, we have built a burst-error resistant code that resists a burst error-length equal to the interleave factor. We still can have the case where noise corrupts two bits within the same codeword, or noise plus an impulse corrupts two bits within one block. In that case we will still get an uncorrectable error. We can plot the effect of the linear block code in the presence of an impulse error rate. Figure 7-12 shows the BER vs. SNR for an interleaved (15,11) code when the impulse noise causes a 10^{-6} BER even when we have a strong received signal. We can see that the code essentially cuts the saturated error rate from 10^{-6} to 10^{-12}. This is a large improvement, and the performance on an AX.25 link will be essentially perfect at 10^{-12}.

The process of interleaving is essentially making a large block code out of a smaller code. It is always more efficient to use a large block code instead, but this may require an impractical amount of hardware in some cases.

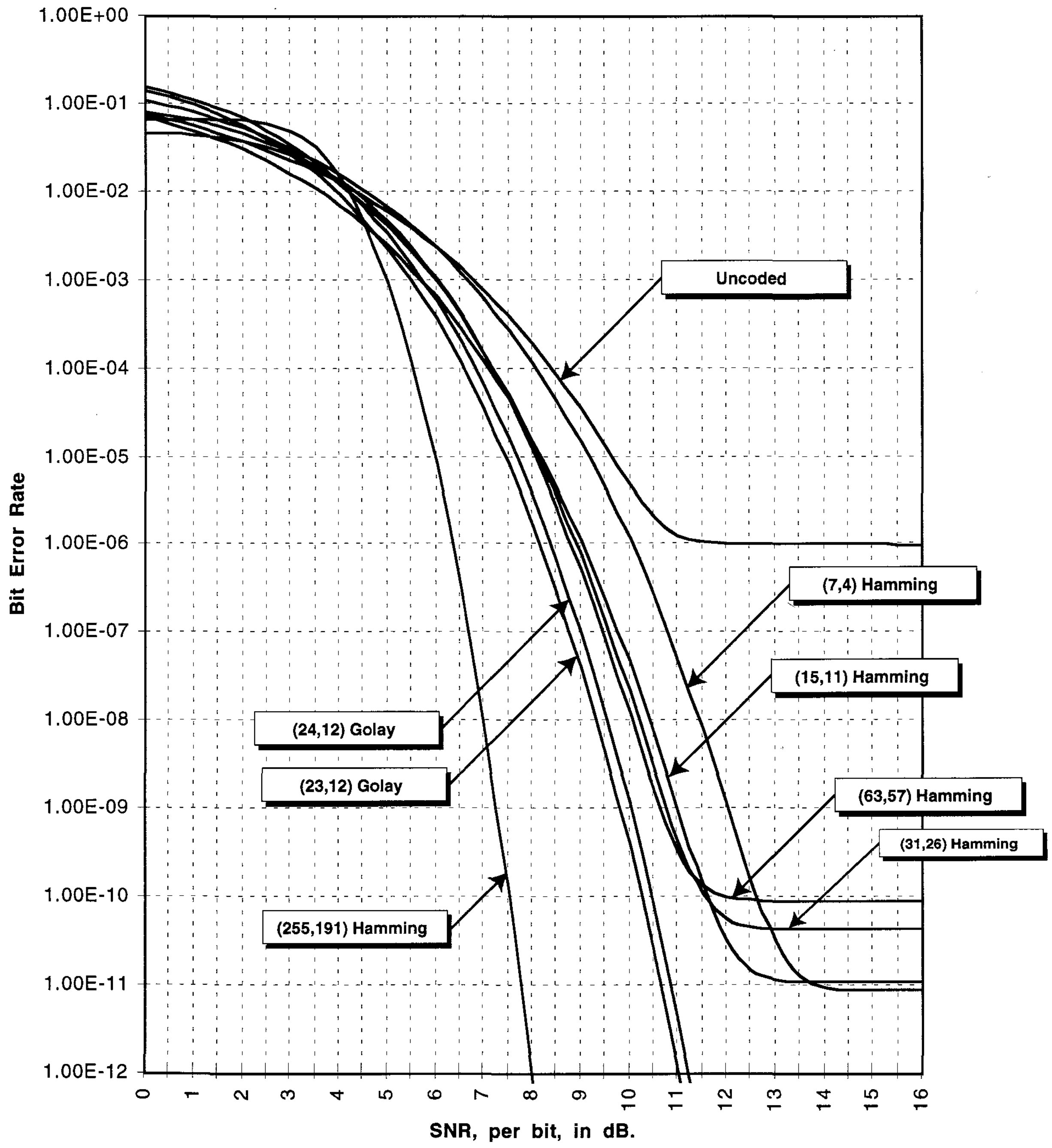


Figure 7-12 - *Bit-Error-Rate vs SNR with Impulse noise = 10^{-6} before decoding*

The characteristic equations for the (15,11) code, and the placement of the parity bits are:

$$P0 + D0 + D1 + D3 + D4 + D6 + D8 + D10 = 0$$
$$P1 + D0 + D2 + D3 + D5 + D6 + D9 + D10 = 0$$
$$P2 + D1 + D2 + D3 + D7 + D8 + D9 + D10 = 0$$
$$P3 + D4 + D5 + D6 + D7 + D8 + D9 + D10 = 0$$

Sequence: P0 P1 D0 P2 D1 D2 D3 P3 D4 D5 D6 D7 D8 D9 D10

The (15,11) code can be readily constructed, and decoded with 8-bit parity gates, a 4-to-16 line decoder, and some 2-input exclusive-or gates. Coding and decoding for the longer hamming codes, such as (31, 26) and (63, 57) involve larger circuits, but the same principles apply.

Golay Codes

Two well-known linear block codes are the Golay (23, 12) code and the (24, 12) extended Golay code (which is formed by the addition of an overall parity bit to the (23, 12) code). The Golay code can correct up to three errors in a block of 23 data+parity bits. The extended code can correct up to 3 bit errors in the block of 24 data+parity bits. We can compute the performance of this code in the AWGN and burst cases in the same way as we computed the performance of the single-bit correcting linear block codes. From the comparison of the Golay code to the smaller codes, we can see that the performance of the Golay codes is better than the smaller Hamming codes for good signal to noise ratios. The (23, 12) code is about 0.75 dB. better than the (63,57) code, while the (24, 12) code is about 0.5 dB. better than the (63,57) code.

Considering that the Golay code can correct up to 3 bit errors in one block, it has an inherent burst-error tolerance built-in. In this regard, it may be easier to implement a non-interleaved Golay code than to implement the interleaved Hamming codes, so long as the burst error-length is 3 or less. However, with the convenience of having the transmission rate being exactly twice the data rate, and be easily able to divide 8-bit bytes into 12-bit words, the (24, 12) code may be very simple to implement.

Longer Codes

A (255, 191) Hamming code that can correct up to 8 bit errors in a block of 255 bits is also shown on the charts. As the codes become longer, it is possible to get more coding gain, but at the expense of more complicated coder and decoder circuitry. The (255, 191) code has 3.5 dB of coding gain at $1.0\mathrm{x}10^{-5}$ BER, and it displays a very vertical fall-off. However, at high bit error rates (around $1.0\mathrm{x}10^{-2}$) most codes are actually worse than uncoded data for high-rate codes. The charts of the BER vs. SNR are not accurate for SNR less than about 2 dB. due to simplifying assumptions made about the error rate bounds. Anyway, this region is usually of little interest.

Reed-Solomon Codes

As the length of a linear block code grows, its efficiency improves. This was shown in the previous section. Another code, known as a Reed-Solomon (RS) error correcting code is also a linear block code with some specific properties. It is probably most well-known due to its use in Compact Disc (CD) audio. The Reed-Solomon (RS) code is constructed not on individual bits, but rather on individual *characters* (words) of a data stream. For example, if a data stream is considered to be composed of 8-bit bytes, then an RS code could be constructed to correct a byte error in the received data stream. In many cases, the choice of the width does not have to bear any relationship to the usage (for example, they do not have to be byte-wide, nor is there any particular reason to favor this width). RS codes perform single character error correction. The RS code can correct up to 8 bits of errors in this example. However, they must all occur within the same character, or else uncorrectable bit errors will occur. Thus, if the data modem is providing in actuality a serial single bit stream, then an RS-8 code will correct a burst error of length 8 — provided that all 8 errors occur within the same word (for a single-character correcting code).

A RS code has the property that there are 2^N-1 characters in a block. Of these, 2^N-3 are data characters, and 2 are error-correcting characters, for a single-character-correcting code. For example, if N=8 (8 bits per character), then the RS block would contain 253 data characters and 2 error — correction characters. If a character were 12 bits (N=12), then the RS block would contain 4093 characters, and 2–error correcting characters. RS codes that correct more than one character error can also be made. It requires two error correction characters for each correctable character. In the previous examples, 2 error correction character means that the code can correct a single character error. To build a ten-character-error correcting code, twenty correction characters are needed. For example, assuming 8-bit characters, then there would be 255 characters in the block, composed of 20 error-correction characters, and 235 data characters which could correct any 10 character errors in the block of 255 characters. To send less than 235 data characters (shorten the block size), the transmitter and receiver can both agree beforehand that some of the data characters are zero. Then the transmitter would not send the zero characters, and the receiver would insert zero characters in place of the unsent transmit characters. RS codes are among the best block codes and are extensively used in many applications.

RS codes can be easily generated with two character-wide shift registers and multiplication circuits (exclusive-or circuits), and can be decoded with a slightly more complicated 3 character-wide shift-register/polynomial circuit. However, one drawback to the RS code is that the entire received block must be saved prior to correcting the error. If the block is long, this means that a significant number of storage elements could be required (for N=12, this would be a 4095-long 12-bit wide shift register in addition to the decoder). Additionally, the absolute delay through such a decoder would be equal to the block size times the character rate. This delay may or may not pose a problem. RS codes are *systematic*, that is the data are not changed or encoded. They are transmitted as clear text; only the error correction characters are specially generated. This property may be desirable in some applications.

RS codes may be used by themselves, or in conjunction with another code, such as a convolutional code to improve the performance. Tandeming such codes is called *concatenation*. Otherwise, RS codes may be interleaved (such as in block coding) except that the characters are interleaved (rather than the bits, as in the linear block codes discussed previously). Thus, adjacent characters (perhaps bytes) would belong to different RS code blocks. The performance of RS codes when interleaved is similar to that of linear block codes when interleaved. The key advantage of the RS code is the ability to construct large blocks in a straightforward manner, resulting in outstanding efficiency with a very small amount of overhead characters (2 per correcting character ability) compared to the number of transmitted data characters.

Convolutional Codes

Among the most powerful error correction codes available, convolutional codes are different in structure from the preceding linear block and Reed-Solomon codes. At lower data rates, such as for HF work, they can be decoded in real-time on a microprocessor. In fact, a dedicated high-performance assembly-language program can decode convolutional codes well into the hundreds of thousands of bits per second, given adequate processor performance. Hardware implementations are available in gate arrays that have been produced by several manufacturers for their proprietary radio products. Convolutional codes derive their name from the fact that the data stream is convolved with code generator polynomials in several binary shift-registers.

There are three important descriptors of a convolutional code. First, the rate of the code. For example a rate 1/2 code means that there is one parity bit for every data bit, or that the data portion of the code is 1/2 of the sum of the data+parity bits. Thus, a rate 1/2 code will occupy double the bandwidth of just the data alone. Usually, the symbol R is designated the code rate. The second parameter is the constraint-length of the code, usually noted with the symbol, *k*. The constraint length is the span between the current data bit and the oldest data bit that enter into the generation of the output. For example, a constraint-length of 3 means that the current plus the two previous bits are used in the generation of the output of the coder. The third parameter is whether the code is hard-decision or soft-decision decoded. This is a new possibility that we did not discuss in our description of block codes, because soft-decision decoding is not very practical with block codes. In hard-decision decoding, the input to the decoder is quantized to a logic one or a logic zero, and the decoder works with these binary values. It is possible, instead, to quantize the received signal more accurately than with one bit, and this is referred to as soft-decision decoding. Intuitively, soft-decision decoding allows the decoder to know how well the received signal was really a *one* or a *zero*. If the received signal is very close to a detected *zero*, then its accurately-quantized value will match closely that of a perfectly received *zero*. If, on the other hand, the received value is nearly midway between a perfectly detected *one* and a perfectly detected *zero*, then the accurately-quantized receive signal will have a lot of error from the perfect one and the perfect *zero* levels (i.e. the confidence is low). With this additional knowledge of the error, the decoder can more accurately decode the convolutional code, and usually about 2 dB. improvement is possible. In practice, 3-bit quantization of the received signal results in most of the 2 dB. improvement being realized on AWGN channels.

Figure 7-13 is a chart showing the decoded Bit Error Rate vs. Signal to Noise ratio for uncoded 2PSK data, and for several convolutional codes. It can be seen that with soft-decision coding, and longer constraint-lengths, approximately 5 dB. of coding gain is possible at $1.0\text{x}10^{-6}$ bit error rate. Figure 7-14 is a chart showing the performance of convolutional codes when corrupted by impulse noise that causes a constant $1.0\text{x}10^{-6}$ background error rate, regardless of the signal strength (such as from a radar system). Obviously, the uncoded signal saturates at an error rate of $1.0\text{x}10^{-6}$ regardless of the received signal strength (within the bound that it is weaker than the radar signal), while the convolutional decoders show good error performance under the same conditions. Both charts indicate the upper-bound error probability, and thus are not so accurate at high error rates (low SNR). Usually this area of the curve is of lesser interest. Both charts have been adjusted to normalize the SNR per bit to the uncoded data rate. In the case of a 1/2 rate code, this equates to a 3 dB. penalty for the convolutional codes. Since the data rate is doubled, the noise at the receive detector is also doubled. Some texts and charts do not normalize the received noise to the code rate, and thus will show performance of a convolutional code to be better than the normalized charts shown here. As the constraint-length increases beyond 7, the performance of the code continues to improve, while the complexity of the decoder grows exponentially (for Viterbi decoders).

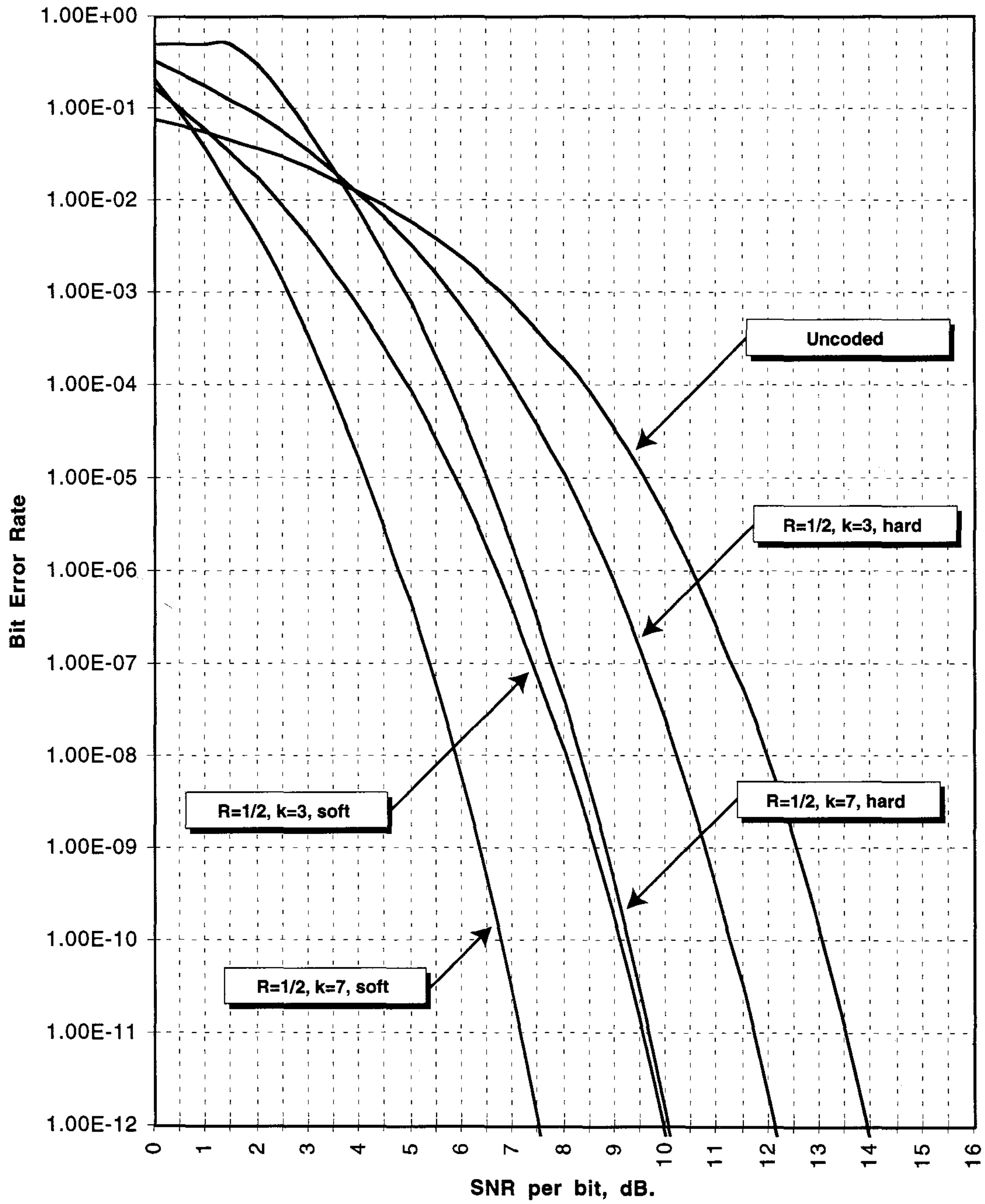


Figure 7-13 - Bit Error Rate vs. SNR for some Convolutional codes

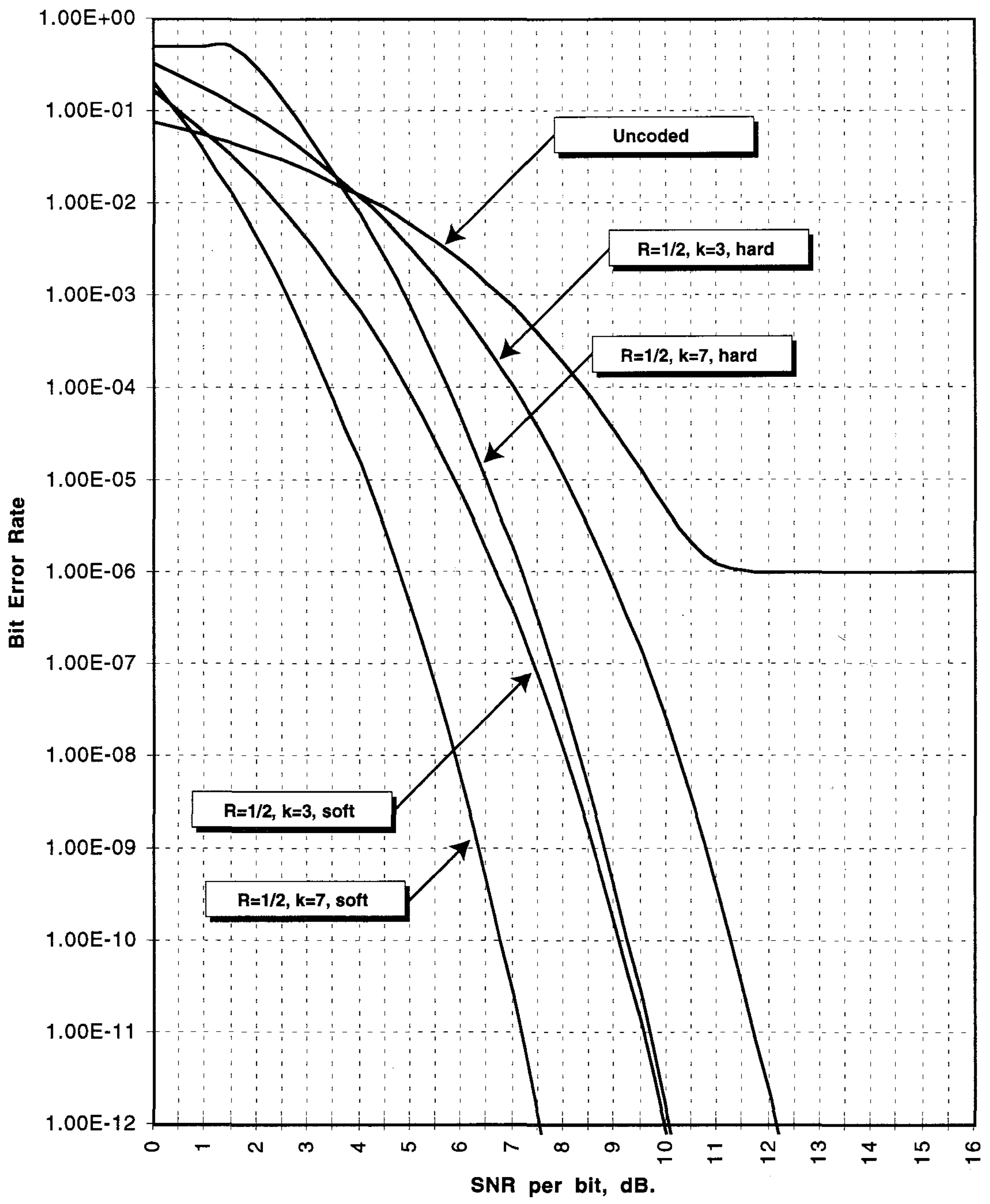


Figure 7-14 *- Bit Error vs. SNR for some Convolutional codes with a constant background error rate of 1.0x10*$^{-6}$

The generation of the convolutional-coded signal is extremely simple, unfortunately the decoding of the signal is not at all straightforward. Figure 7-15 illustrates the circuit for a rate=1/2, k=7 convolutional coder. The clock signals are not shown. The selector switch operates at twice the input data rate, and thus outputs two bits for each bit time, one from the lower coder branch, and one from the upper coder branch. This coder has the generator polynomials:

$$g^{(1)}(X) = X^6 + X^5 + X^4 + X^3 + 1 \quad \text{and}$$
$$g^{(2)}(X) = X^6 + X^4 + X^3 + X + 1$$

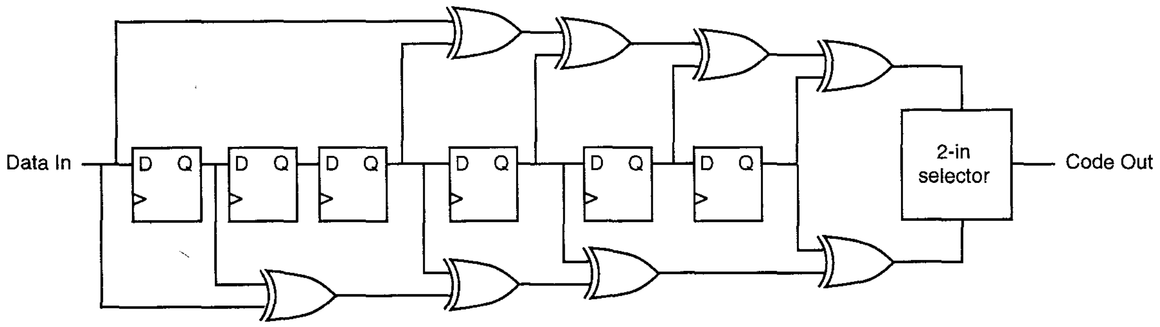


Figure 7-15 *- rate = 1/2, length = 7 convolutional coder.*

Decoding the Convolutional Code

This encoder generates two output-bits per input bit. To decode the convolutional code, and recover the original data stream, it is necessary to examine each two-bit received pair. Since there are 4 possible states of two bits, and only 2 of the states could have been generated (by a logical one or a logical zero input), if an impossible state is received, then a bit error must have occurred. At this point, it is necessary to back-up one pair of bits, and assume that even though the previous pair were correct, perhaps one or both of those bits were wrong. Then, given the 4 possible assumptions about the state of the pervious bit pair, is the current bit pair possible? If so, then the error may have been in the previous bits, not the current bits.

This process of backing up is *recursive*. That is, we can ask if the pair previous to the previous pair is actually correct, and postulate about the 2 valid states of that pair. It can be seen that a tree of possible solutions can be generated — 2 valid branches for the grandparent pair, each of which branches 4 ways for the parent bit pair, each of which branches 2 valid ways for the current bit pair. Thus, 2*2*2 = 8 possible solutions through a tree can be described that yield the last three sets of bit-pairs. The further back into time we go to look at bit pairs, the better the likelihood we will find a path to the current bits and previous bits that minimize the number of assumptions made about which bits are wrong. Starting with the oldest bit-pair, the path through the tree, that requires us to assume the fewest possible bit errors is the correct path. This path tells us the correct decoded original bit stream. This technique is known as the *sequential*, or *Fano*, technique. It is a powerful technique for decoding convolutional codes, but may also be the most intensive, since large trees can result if we back up the tree very many steps.

Another technique for decoding the bit pairs is to realize that the tree structure is very repetitive after a few recursions back into the bits, and that the tree essentially folds back on itself. This reduces the number of computations that may need to be made, and is exploited by the *Viterbi algorithm* for decoding a convolutional code.

Convolutional Decoders

There are several algorithms for decoding the code, and recovering the original data stream. One popular method is called Viterbi's algorithm (VA), and in fact this algorithm has many applications beyond decoding a convolutional code. To decode a convolutional code, the receiver must have detailed knowledge of how a particular encoder works (i.e. it contains a copy of the encoder). The decoder uses its knowledge of the encoder operation, along with the received convolutional code sequence, including received bit errors, and to construct the most likely transmitted data sequence from these two pieces of information.

In describing the decoder, a number of diagrams have become roughly standardized throughout the literature. Understanding these notations is quite important to understanding the algorithm, as the VA optimizes a path through a trellis. It is difficult to understand this optimization without utilizing the trellis diagram. There are four notations used to describe the operation of the convolutional encoder, they are:

1. The bit-sequence diagram,
2. The state diagram,
3. The tree diagram, and
4. The trellis diagram.

The state diagram may be the most familiar of the diagrams. It is used extensively in many digital logic texts, and can be used to represent many different functions. To understand the decoder, we will use a simple encoder and decoder, a rate 1/2, constraint length 3 encoder, and develop the four diagrams for this particular encoder. Worked-out examples for more complex encoders can be found in Proakis (1983).

The encoder for a rate=1/2, constraint length=3 code is shown in figure 7-16.

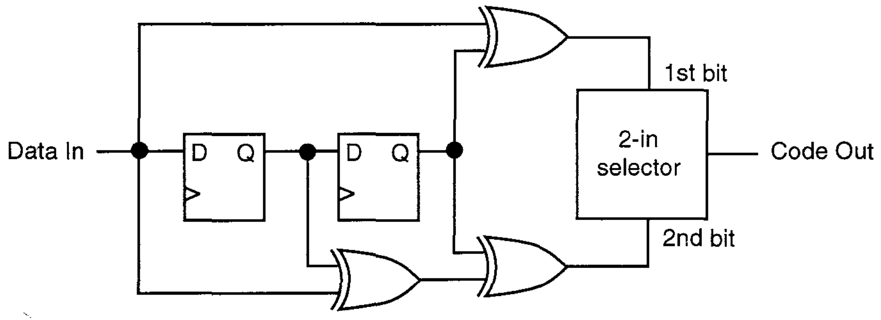


Figure 7-16 *- schematic of rate 1/2, constraint-length 3 convolutional encoder used for the Viterbi decoder example.*

The operation of this encoder can be visualized in a number of ways, and the four methods listed above will be described for this particular circuit. The bit sequence of the encoder describes the bits output by the encoder depending on the bits input to the encoder. Assuming that the two flip-flops are initially in the zero-zero state, the encoder produces the following sequence:

```
Data In   1    1    0    0    1    0    1    0    0    1
Data Out  11   10   10   11   11   01   00   01   11   11
```

This can also be represented as a state diagram, using the traditional state-machine notation, this is illustrated in figure 7-17.

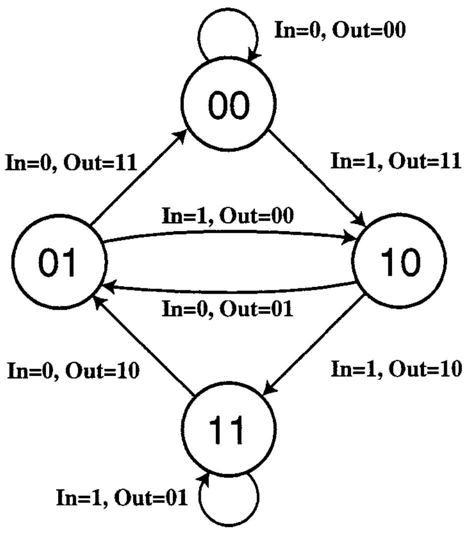


Figure 7-17 - *state diagram of the rate 1/2, constraint-length 3 convolutional encoder. The flip-flop states are inside the bubbles, the left-most digit corresponds to the left-most flip-flop in the previous schematic.*

Note that in the state diagram, the goodness of this code can be observed: for each state, there are two possible inputs (0 and 1), and the two possible output sequences (one for each input possibility) differ in two bit positions - that is the Hamming distance between the output sequences for each possible input is always two.

Another method to describe the operation of the encoder is a tree diagram. In the tree, the outputs of the encoder are shown, along with the input bit, but the states of the encoder are not shown. The tree corresponding to the above example is shown in figure 7-18. In the tree, traversing upwards is caused by a zero data input bit, while traversing downward is caused by a one data bit being input to the encoder.

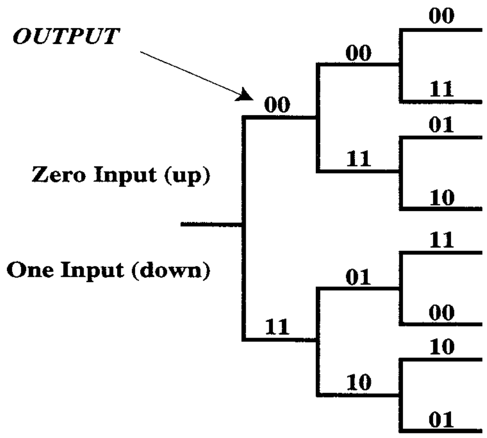


Figure 7-18 - *tree diagram showing the output of the encoder vs. the input (state information is not shown, it has to be inferred)*

Of course, the tree diagram continues on as more data bits are input. The diagram above only shows three bit times. Andrew Viterbi noticed that the tree diagram actually repeats itself after several bit times, and that the tree can be folded back upon itself, in the form of a trellis diagram. Figure 7-19 illustrates the trellis diagram for the above tree. In the trellis diagram, the state of the encoder is also shown (unlike in the tree diagram), but the outputs of the encoder are not shown since the diagram becomes very cluttered. A diagram showing the encoder outputs based on the transitions between states is shown in figure 7-20.

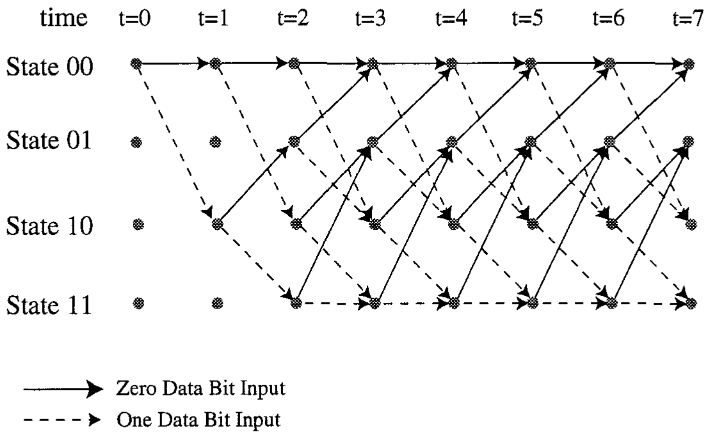


***Figure 7-19** - trellis diagram for the convolutional encoder.*
The states, and the data bits input to the encoder are shown, but the bits output by the encoder have not been shown. At time t=0, it is assumed that the encoder is in the 00 state.

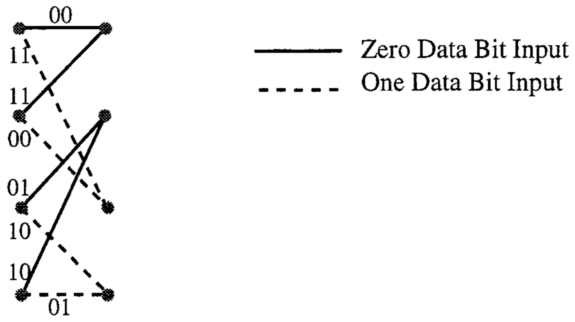


***Figure 7-20** - a portion of a trellis diagram showing the bits output by the encoder based on the current state and the input data bit value.*

In order to decode the code, we look at the received bit sequence, and for each transition ask, "How closely does it match the code that the encoder must have produced for that particular transition?" To quantify the notion of "how close" for a hard-decision decoder (a circuit that outputs only a one or a zero as the received data bit) we use the Hamming distance of the actual received sequence from the sequence that the encoder should have produced. Table 7-4 shows the Hamming distance for several encoder sequences, and some received sequences.

Transmitted Bit Sequence	Received Bit Sequence	Hamming Distance
00	00	0
00	01	1
00	10	1
00	11	2
01	00	1
01	01	0
01	10	2
01	11	1

Table 7-4 - Hamming distance for some transmitted and received sequences.

If each transition vector in the trellis is given a length equal to the Hamming distance of the actual received code from the transmitted sequence, then that distance can be equated to the length of that vector. Assuming that we start at a known state, and that we finish at a known state, the problem then becomes one of finding the shortest distance through the trellis between those two states given the Hamming error distance for each link in the trellis. This technique minimizes the number of receive symbol (bit) errors that we have to assume. To re-phrase: the error-distance for each vector in the trellis is the Hamming distance between the actual received sequence and the sequence that the encoder should have produced (since we know the details of the encoder), based o our best guess as to what the transmitter sent.

At this point, we will now go through an example of the transmission and reception of a bit sequence with errors. We will map this information into a trellis, and then examine the shortest path through the trellis. Figure 7-21 shows the data bit, the transmitted sequence and the receive sequence, and it shows the appropriate error metrics for each vector in the trellis.

				Received symbol (bit) error			
Data bit	0	1	1	1	0	0	0
Transmit Sequence	00	11	10	01	10	11	00
Receive Sequence	00	11	10	11	10	11	00

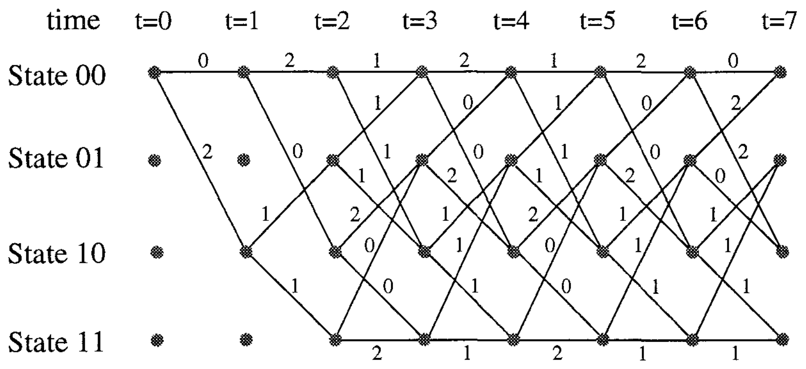


Figure 7-21 *- trellis diagram for a particular received sequence (with one bit error) and a particular transmitted bit sequence, with the Hamming error weights shown for each transition vector.*

Upon examination of the trellis, it can be seen that the shortest path through the trellis has an error length equal to one, and that path is shown in figure 7-22.

				Received symbol (bit) error			
Data bit	0	1	1	1	0	0	0
Transmit Sequence	00	11	10	01	10	11	00
Receive Sequence	00	11	10	11	10	11	00

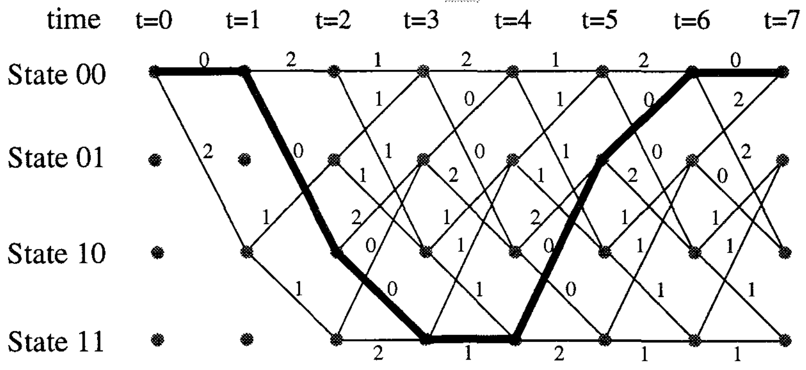


Figure 7-22 *- showing the shortest path through the trellis.*
Each link shows the Hamming distance between what should have been transmitted as a two-bit sequence, and what was actually received as the two-bit sequence. Based on our knowledge of the encoder, we can reconstruct the transmitted data bits based upon this path.

In theory we would need to hold the entire trellis for the complete received message. However, in practice little improvement in the received bit error rate is produced when the number of decoded bits stored is equal to 5 times the constraint length. In this case, we need store only the 15 most recent time tics (30 di-bits), since the constraint length is 3. Additionally, it turns out that the receive process does not really need to know the initial or end states as the decoder will eventually find the correct path. Thus, the code self-aligns the states. We do, however, need to assure that the receiver has correctly synchronized on the di-bit sequences. If the receiver were to misalign by one bit time, nothing would work (i.e.: accidentally taking the last bit of one pair and the first bit of the next pair as a di-bit group). Thus, if the receiver does not know which di-bits belong together it must attempt both synchronization possibilities until it finds that one that produces continually large error paths, while the other produces a consistently small error path. The receiver then chooses the di-bit synchronization state with a consistently small error path through the trellis.

For soft decision decoding, instead of using the Hamming distance between the transmit and receive sequences, the Euclidian distance between the sequences is used as the error metric. As shown earlier, soft-decision decoding can produce up to 2 dB. improvement in BER vs. SNR.

Viterbi's Algorithm

One method for finding the shortest path through the trellis is known as Viterbi's algorithm. In this algorithm, the trellis is examined at each time tic. At each time, the length of the path to each of the possible states is calculated as the error metric, plus the accumulated error metric from the previous state. When two paths converge at a particular state at a given time tic, the one with the worse error metric is discarded, and the one with the better error metric is retained. The retained path is known as a survivor path. There are several paths into each state (two in our example), and several paths out of each state (again, two in our example). At most there can be 4 active paths under consideration in our example. It is not useful to keep non-survivor paths, because they can never be shorter than a survivor path, even when considering later events. We will now examine the operation of the algorithm given the previous example, which consisted of several transmitted bits, and one received bit error.

Starting at time zero, we have assumed that the initial state is zero-zero. The accumulated error metric is set to zero. Then the first possible transitions are calculated based on this state. Figure 7-23 illustrates the computation at time one, based on this initial condition at time zero.

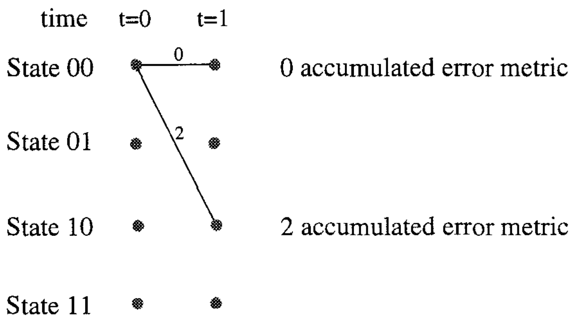


Figure 7-23 *- accumulated error metric at step one of the Viterbi calculation.*

At the next iteration of the calculation, the error metrics between time tics one and two are appended to the calculation. This is shown in figure 7-24.

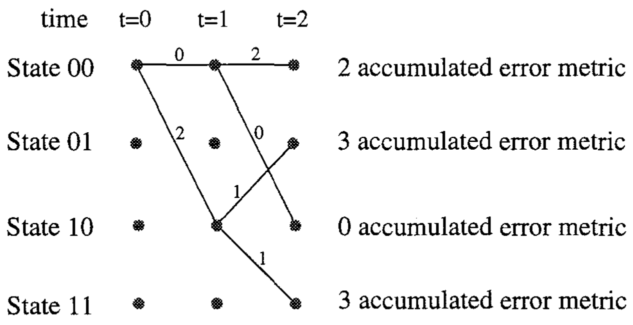

time
t=0
t=1
t=2
State 00
State 01
State 10
State 11
0
2
2
0
1
1
2 accumulated error metric
3 accumulated error metric
0 accumulated error metric
3 accumulated error metric

Figure 7-24 *- accumulated error metric at step two of the Viterbi calculation.*

At this point, no paths have yet been terminated, since there has not been a case where two paths converge at a node (a state at a particular time). However, in the third iteration of the calculation, we can see that some paths have been terminated.

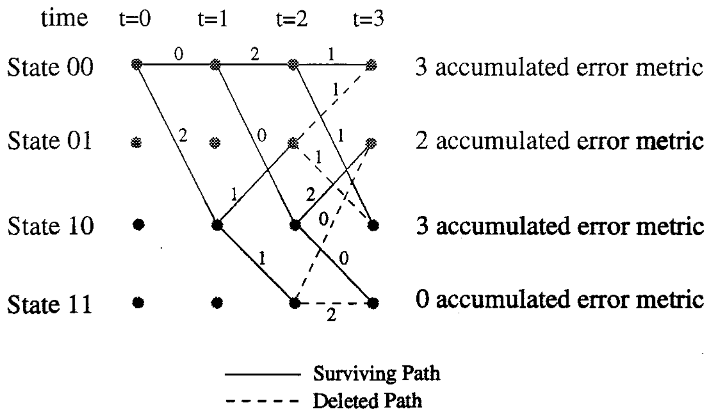

time
t=0
t=1
t=2
t=3
State 00
State 01
State 10
State 11
3 accumulated error metric
2 accumulated error metric
3 accumulated error metric
0 accumulated error metric
Surviving Path
Deleted Path

***Figure 7-25** - third iteration of the Viterbi algorithm.*
Four paths have been deleted since they converge at a node and have a higher error metric than another path arriving at the same node.

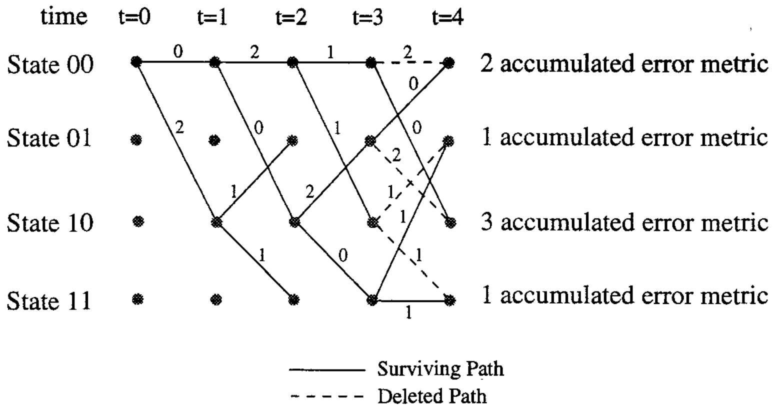

time
t=0
t=1
t=2
t=3
t=4
State 00
State 01
State 10
State 11
2 accumulated error metric
1 accumulated error metric
3 accumulated error metric
1 accumulated error metric
Surviving Path
Deleted Path

***Figure 7-26** - fourth iteration of the Viterbi algorithm.*
Four more paths have been deleted since they converge at a node and have a higher error metric than another path arriving at the same node.

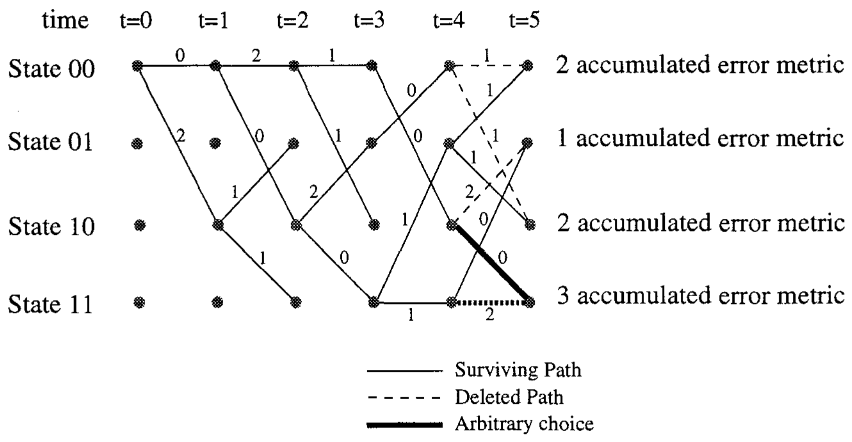

time
t=0
t=1
t=2
t=3
t=4
t=5
State 00
State 01
State 10
State 11
2 accumulated error metric
1 accumulated error metric
2 accumulated error metric
3 accumulated error metric
Surviving Path
Deleted Path
Arbitrary choice

***Figure 7-27** - fifth iteration of the Viterbi algorithm.*
Two paths converging at one node have the same accumulated error metric, so we just make an arbitrary choice (fair coin toss) as to which one to keep.

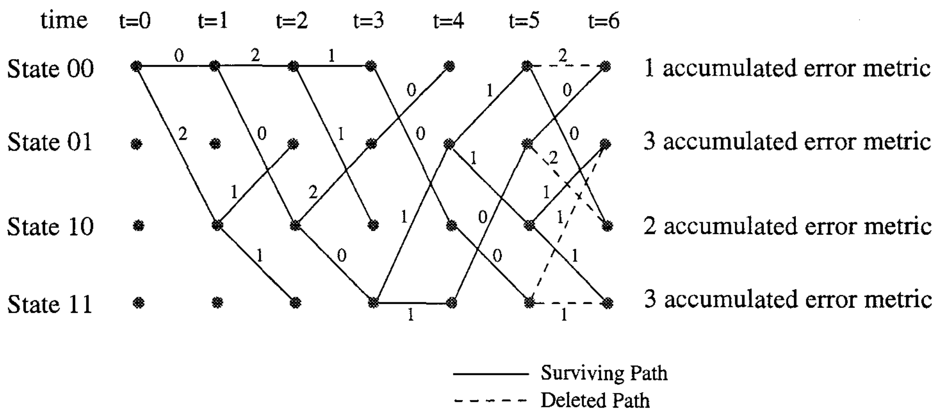

time
t=0
t=1
t=2
t=3
t=4
t=5
t=6
State 00
State 01
State 10
State 11
1 accumulated error metric
3 accumulated error metric
2 accumulated error metric
3 accumulated error metric
Surviving Path
Deleted Path

***Figure 7-28** - sixth iteration of the Viterbi algorithm.*

After 7 iterations the Viterbi algorithm has selected the most likely path through the trellis. In a real application, a bit several constraint lengths old (maybe 15 time tic ago) would be output by the decoder as a valid decision. Thus, the decoder is always processing many bits ahead of what it outputs as its decision. At each time tic, the algorithm has a simple task — to eliminate the highest path metric at each node. In this example, it has to make 8 path comparisons and it has to keep track of the 4 accumulated metrics. Additionally, it has to keep track of 4 paths through the trellis from the point where it output a bit up to the current bit being processed. It may be easier to visualize the path by starting at the upper right in figure 7-29, where the accumulated error metric is 1, and working backwards to the source.

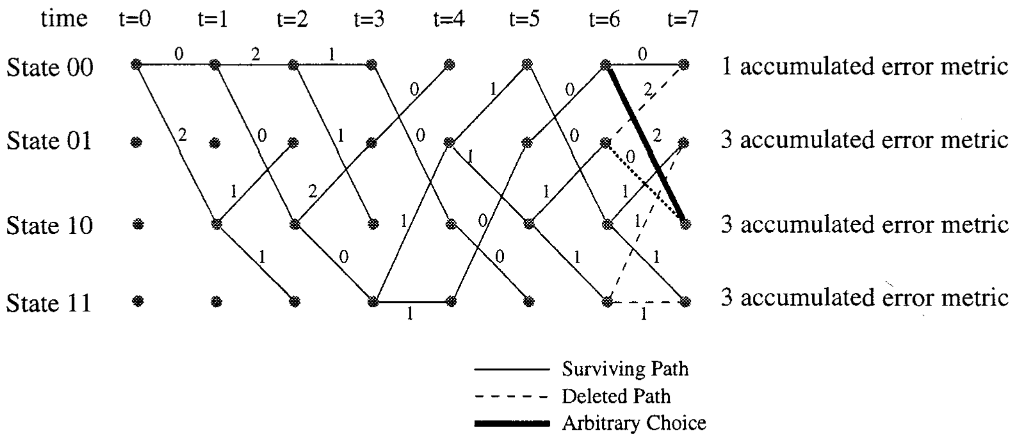

time
t=0
t=1
t=2
t=3
t=4
t=5
t=6
t=7
State 00
State 01
State 10
State 11
1 accumulated error metric
3 accumulated error metric
3 accumulated error metric
3 accumulated error metric
Surviving Path
Deleted Path
Arbitrary Choice

Figure 7-29 - *seventh and final iteration of the Viterbi algorithm. The path with the lowest error metric is selected.*

The decoder keeps a path through the trellis about 15 bit times long. With each additional di-bit pair introduced to the decoder, it produces its decision about the best output bit 15 time tics (30 di-bits, or about 5 constraint lengths) ago in this example. Ideally, the entire trellis would be kept, but this is impractical in a real situation.

Figure 7-30 illustrates the final solution to the path determined by the Viterbi algorithm. It can been seen that this matches the solution described earlier. Once the final path is computed, then the original transmit data sequence can be reconstructed. The algorithm has to save 4 paths through the trellis (in this particular example), since there are always 4 surviving paths at any given time (once two di-bit groups have been processed). The path with the lowest-error count received sequence is assumed to be the transmit di-bit sequence, with the errors corrected. From the corrected transmit di-bit sequence, the original transmit data stream can be reconstructed by the knowledge of how the transmit coder works (figure 7-20).

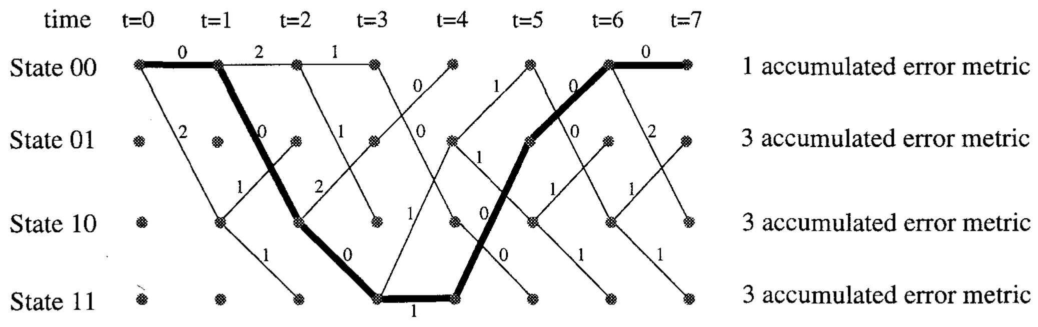

time
t=0
t=1
t=2
t=3
t=4
t=5
t=6
t=7
State 00
State 01
State 10
State 11
1 accumulated error metric
3 accumulated error metric
3 accumulated error metric
3 accumulated error metric

Figure 7-30 - final solution computed by the Viterbi algorithm

In this example, the transmit di-bit sequence associated with the lowest path accumulated error metric is:

```
00 11 10 01 10 11 00
```

and the only possible transmit data bits that could generate that sequence (based upon inspection of figure 7-20, and knowing that 00 is the starting encoder state) are:

```
0 1 1 1 0 0 0
```

Therefore, the original transmit data sequence has been recovered without error even though there was a single bit error in the received di-bit data bit sequence.

Summary of Viterbi Decoder

A summary of the decoder algorithm, assuming that both self-synchronization and the correct ordering of the di-bit groups is achieved is:

1. Assume, initially, that there are four surviving paths through the trellis, and assign them all an accumulated error metric of zero. Setup a memory for each of 4 paths through the trellis, and set them to the all-zeros state.

2. For each received symbol (di-bit) group, perform the Add-Compare-Select calculation, which determines the Hamming weight of the paths from the 4 current encoder states traversing to the next 4 encoder states. Determine the paths with the lowest accumulated error metric. If a tie occurs, use a fair coin toss to select one of them.

3. Destroy the non-survivor paths and their memories. Extend the survivor paths (including both branches for surviving paths that branch out), and their history through time, while also recording the path error metric for the 4 new survivor paths.

4. As a practical matter, paths only need to be recorded for 5 constraint lengths (15 data bits, equal to 15 state times, or 30 di-bits in this example). As each new di-bit group is added, look at the oldest state (15 state times ago) and pick the di-bit pair on the path with the currently lowest accumulated error metric. This is the di-bit pair corresponding to the corrected di-bit sequence 15 state times ago. Derive the transmit data bit that caused the state transition from the 15th-old di-bit to the 14th-old di-bit. This is the corrected receive data bit.

Thus, there is 30 di-bits of delay in this decoder design, since it will output a corrected receive data bit corresponding to some event way back in the path memory of the shortest survivor path.

A number of excellent texts describing convolutional and other codes are available. The references list a few. The first two books Arazi (1988) and Sweeny (1991) are less mathematical than Michelson & Levesque (1985) and Viterbi & Omura (1979).

Chapter 7 - Reference

Arazi, Benjamin . "A Commonsense Approach to the Theory of Error Correcting Codes." The MIT Press, 1988.

Michelson, Arnold and Allen Levesque. "Error-Control Techniques for Digital Communication.", John Wiley & Sons, 1985

Proakis, John G. "Digital Communications." McGraw-Hill Inc., 1983, Chapter 5.3.

Sweeny, Peter. "Error Control Coding." Prentice Hall Inc., 1991.

Viterbi, Andrew and Jim Omura. "Principles of Digital Communication and Coding." McGraw-Hill Inc., 1979.

Chapter

8

Data Slicer and the Slicing Level

When we receive the data signal, it is analog in form. However, the modem has to deliver a digital signal to the device that uses the information. A *slicer* is a circuit that performs the conversion from the analog to the digital domain. For 2-level modulation, it is a one-bit A-to-D converter, for higher levels of modulation, it is a multi-bit A-to-D converter. In slicing, we have to be concerned about two things: 1) that we properly determine the level that separates an analog zero-voltage from an analog-one voltage, and 2) that we accurately compare the received analog voltage against this reference voltage at the right time. If we derive the slicing voltage from the received signal, then step number 2 is usually not much of a problem, since high-accuracy, high-speed comparators, and high-speed flip-flops are readily available, presuming that we have already developed a method of accurate clock recovery. An alternate approach is to use a fixed slicing-level voltage, and to limit or gain-control the received signal so that it is symmetrical around the fixed slicing level. Figure 8-1 shows how the slicing level relates to an eye pattern.

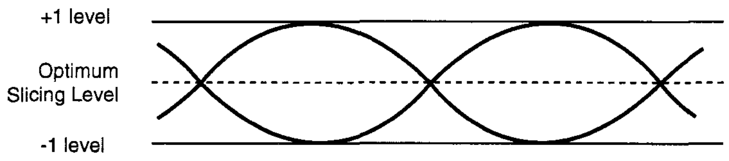


Figure 8-1 *- the optimum slicing level is half-way between the two received signal states.*

For the first type of slicer, where we derive a slicing level, we find that in determining the reference voltage, we must take into account some of the properties of the baseband data signal itself. Also, we may have to take into account some characteristics of the propagation channel and the radio receiver / transmitter pair. In some sophisticated decoders, such as a convolutional decoder with soft-decision decoding, we may in fact want to develop a signal that measures how close to either the ideal one-voltage or to the ideal zero-voltage the received signal is at the moment we sample it. A soft-decision decoder can use the knowledge of how close a corrupted data bit is to the desired value to improve, considerably, its ability to correct errors. See the section on forward error correction for a discussion. In this section we will focus on hard-decision decoding, starting with 2-level modulation.

Slicing Level Determination

If we make the assumption that we have a received signal with, on average, the same number of logical zeros and logical ones being transmitted, then we can easily determine the baseband slicing voltage level — it's merely the average voltage of the input signal. This is easy to develop, simply low-pass filtering the data will provide this voltage. If the received baseband data signal were to drift in voltage for any reason (such as receiver mis-tuning) then the eye pattern would change. In the case of FSK reception, and assuming that we do not exceed the linear range of the discriminator, then the entire received signal would simply have a change in its DC-offset voltage. Our lowpass filter arrangement would properly track this change in average voltage (since the DC-level is the average), and we would be able to keep the slicing level centered midway between the two desired transmission states. See figure 8-2 for a schematic of a simple circuit that derives the slicing level from the average received voltage.

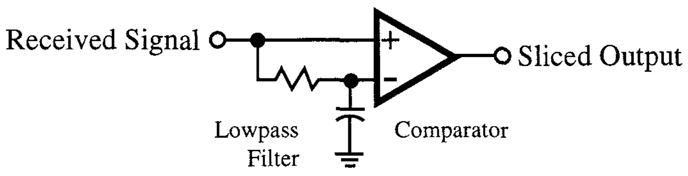


Figure 8-2 *- Simple slicer based on long-term average signal level voltage. The lowpass filter derives the slicing level.*

However, when we state *long-term average* we find that in the short-term, our data will occasionally deviate from the conditions of an equal-number of one's and zero's. If the cutoff frequency of our low-pass filter is too high, then its output voltage will change as the shorter-term fluctuations in ratio of one's to zero's changes. In the worst case, for example, a long string of one's would bias our threshold voltage nearly at the same voltage as a logic one. If this occurred, then just a small amount of noise voltage on a valid logic one signal would cause our comparator to decode a logic zero output. The same effect of course could happen on a long string of logic zero's. In fact, if the ratio of logic one's to logic zero's were, say, 7:1, then our simple lowpass filter arrangement would set the slicing level 7/8's of the way towards the one-voltage from the zero-voltage. Again, this would cause us to be very susceptible to any noise on the logic one analog signal voltage. So, we have some criteria to meet in setting the lowpass filter response: it must be much lower than the duration of abnormal density of ones or an abnormal density of zeros. We could theoretically set a very long time constant to the filter, but then the circuit would not start quickly on the reception of a signal from another transmitter. Thus, we have a trade-off to make in setting the time constant: long for good performance, short so that we can quickly accommodate different transmitters alternately sending to us.

A system using FSK to send unscrambled AX.25 that contains an AC-coupling capacitor will suffer a high error rate due to the effect and due to the large number of FLAG characters.

Source Statistics

In order to determine how to set this lowpass filter, we need to look at the statistics of the data source — that is, its properties with regard to the variation from the ideal 50% one's density case are with time. If our source rarely deviates from 50% ones density, or deviates only for a short period of time, then we can set our lowpass filter very high in frequency. Otherwise it has to be set at a lower frequency. In determining the statistics, we have two general cases to consider: scrambled data, and non-scrambled data.

Non-Scrambled Data

It turns out that most data transmission is very non-random in nature. There is a lot of redundancy in the transmitted information. A long string of space characters happens to be a common thing to send. ASCII characters mostly use the lower half of the 8-bit address space, so the most-significant bit is usually zero. Binary data representing a file might contain long strings of 00h or FFh characters. Additionally, certain layer-2 protocols are also heavily non-random. HDLC is a good example of a non-random, non-balanced protocol. In AX.25 transmission, normally a transmitter will start a packet by sending a long sequence of FLAG characters (7Eh) in order to assure that the transmitter and receiver electronics are fully active and properly receiving the signal, and also that the clock recovery circuits have properly started. Then the actual data are transmitted. This flag character has 6-one bits, and 2-zero bits, and thus has a 6/8 (75%) one's density. Actually, in AX.25, the data is NRZI-coded (see section on coding) and so the transmitted stream during the FLAG time has an even worse one's density of 7/8 (88.5%). So, after the reception of a large number of FLAG characters, the simple lowpass filter slicing circuit will be biased almost at the logic-one voltage (actually, at the 7/8 point). Now when the data characters start arriving, the average density will slip back close to 4/8 (50%) and the slicer will be biased at a very poor point. There will be a large power-penalty in the received data. Even worse, a single data bit error will render the received packet useless. It's clear that lowpass filter slicer is pretty much useless with this kind of data.

An alternative to the lowpass filter is one where the voltage of the logic one and the logic zero are determined independently, and then the slicing level is set half-way in between the two. Figure 8-3 is a schematic of this type of slicing level circuit. The arrangement of the two peak detectors between Vcc and ground prevents a dead-band of two diode drops (1.4 volts) from occurring. Note that the Rdis resistors should be well matched, and the 10-times-Rdis resistors should also be well-matched in terms of resistance accuracy, otherwise there could be a DC-offset error between the desired slicing level and the actual slicing level.

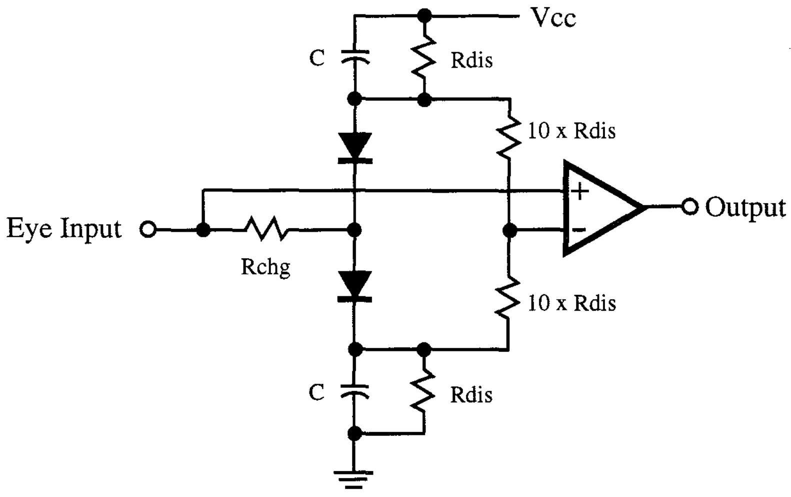


***Figure 8-3** - improved slicing circuit.*
This circuit consists of positive and negative peak detectors.
The output is averaged to find the slicing level, then the signal is sliced against that level.

In looking at the eye patterns of lower-alpha filters, we notice that there is some overshoot to the received signal, depending on the received bit sequence and the exact filter response. However, the overshoot is usually symmetrical if the received data sequence is reasonable and the components and modulation are linear. As the value of alpha is reduced, the overshoot is greater, and the reasonableness of this assumption (such as the long-flags case) gets poorer. For most values of alpha, we can assume good probability and symmetry, and proceed. To determine the logic one voltage, a positive peak detector is utilized. This detector will charge up to the highest voltage, and then hold it. When there are only occasional logic one signals, the detector will still remain charged to the correct level. Similarly, the logic zero voltage of the eye can be determined with a negative peak detector. The slicing level then is taken as the half-way point between the two voltages. The peak detectors must have a time constant, both charging and discharging, to determine how long to hold the desired voltage before it decays. The charging time needs to be sufficiently long, or a single large noise burst would set one of the peak detectors to a very high level from which it would cause successive errors in the received data. Additionally, the discharge time constant needs to be finite if we ever expect the peak detectors to track variations in the incoming signal, or if we expect to receive, quickly, the signal from a new transmitter (perhaps on a slightly different frequency in the case of FSK). How long should the time constants be set? This depends on the source statistics — that is, the pattern and periodicity of ones and zeros in the received data stream.

NRZI-Coded HDLC Calculations

In the case of NRZI-coded HDLC we can be assured that we will get a transition in the data at least every 6 bit times. Therefore, we set the discharging time constant so that our held level is stable for 6 bit periods. For charging, it can be assumed that at least one out of every 7 bits will be our desired bit (for example during the FLAG period) and thus the circuit should adequately peak-charge at this ratio. If the charging time constant is set too short, then the level will easily charge on the desired data bit, but not reject noise.

One problem that is faced with NRZI-coded HDLC is that we should be careful not to AC-couple the received and transmitted baseband signals. If this is done, then the received level is going to vary up and down with the one's density, and there will be distortion of the received eye pattern. This will lead to errors in the ability of the peak detector circuits to track the actual logic one and logic zero error voltages. Alternatively, the AC coupling parameters can be set to a long time, but then the circuit will have a slow response to a new transmitter on a slightly different frequency (long T/R delay). In fact, this problem brings us right back to the situation that was trying to be solved with the lowpass filter slicer.

To determine the time constants, three factors need to be known: 1) how much droop is allowed, 2) over what period of time, and what part of the discharge curve the detector is nominally biased at. The formula that relates the voltage to time for an RC circuit that is being discharged is:

$$V = V_o \left(e^{\frac{-t}{RC}} \right) \tag{8-1}$$

Where Vo is the initial voltage at time t=0. Sometimes the product RC is abbreviated as τ, the time constant of the RC circuit. Figure 8-4 shows V versus time for a time constant of 1, and an initial voltage, Vo, of 1. It can be seen that the one-half voltage point, the slope of the curve, is rather steep. The circuit of figure 8-3 operates in this range of the curve, and in order to provide a small amount of droop over several bit periods, a fairly long constant is needed.

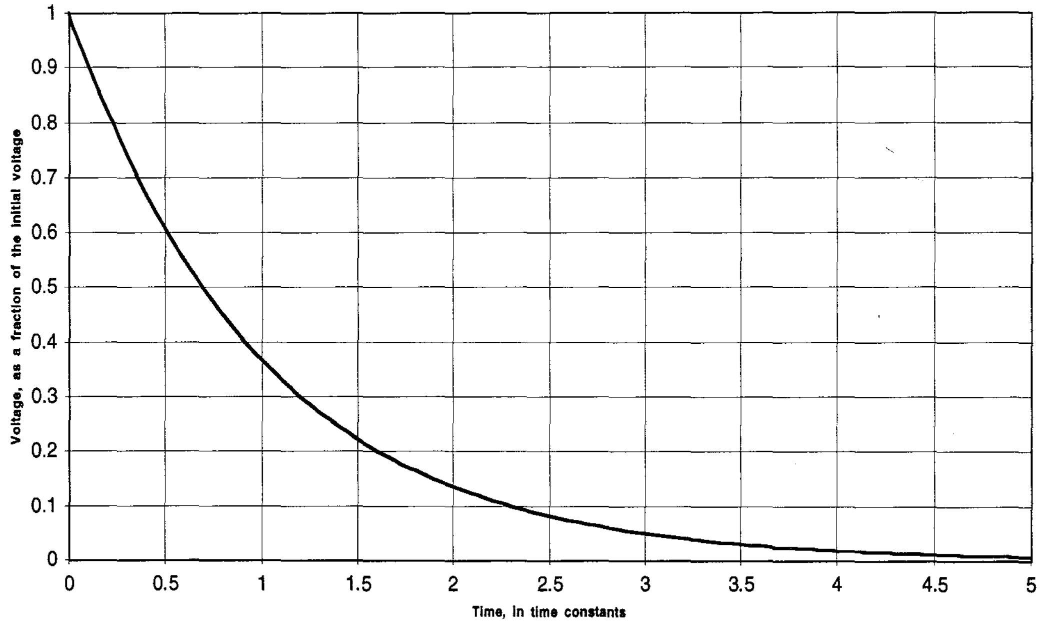


Figure 8-4 *- voltage on an RC circuit versus time, assuming a time constant of one, and an initial charge of one.*

In the case of NRZI-coded HDLC, the design parameters might be set so that the droop is, say 5% over the longest possible run-length of bits, 6 (resulting in the slicing level moving 2.5%). Further, it will be assumed that the RC circuit is one-half charged. Equation 8-1 can then be restated in terms of the time, given that the voltage is known, as

$$t = -\tau \ ln \ V \qquad (8\text{-}2)$$

For $\tau = 1$, the time at which the voltage is 5% lower (V=0.95) is 0.051, or 0.051 time constants. At 9600 baud, then the time constant is:

$$T = \frac{6 * 104\mu \text{ sec}}{0.051} = .012 \text{ sec} \qquad (8\text{-}3)$$

Or, 12 milliseconds. This assumes that the eye voltage is equal to the capacitor charge voltage. For example, if the slicer circuit causes the RC to be no-signal charged to 6 volts, nominally, then 5% droop is with respect to a 6-volt eye pattern (i.e.: a droop of 300 millivolts). If the eye pattern is really only 1 volt, then the 5% droop voltage is 50 millivolts, or .05/6 = 0.833%. Reexamining equation 8-2 with a droop of 0.83% (V = 0.99165) yields 0.00835 time constants. Thus, the time constant for a one volt eye pattern would need to be:

$$T = \frac{6 * 104\mu \text{ sec}}{0.00835} = .074 \text{ sec} \qquad (8\text{-}4)$$

to yield a 5% droop of the one volt eye. As the eye height becomes smaller, the required time constant continues to increase. Thus, the eye height should be on the order of the nominal charge voltage if possible. The charging time constant is shorter and can be approximated by a value on the order of several bit times, or about 500 microseconds in this case.

Scrambled Data Calculations

One solution to the problem of the data varying in one's density is to scramble the data. This is described in detail in the section on coding. Scrambling gives us a much better probability that the transmitter will emit a signal with a 50% one's density, both short term and long term. Systematic non-randomness in the data, such as the long period of FLAG characters or the types and values of the data symbols being sent, is usually smoothed out by the scrambler into an even number of one's and zero's. However, nothing is perfect. There always exists the possibility that a certain non-scrambled signal will occur that exactly causes the scrambled signal to become a long string of one's or zero's. Moreover, this probability is exactly that specified by a binomial distribution, and not the much more frequent probability on non-random events, which occur almost every single packet or data transmission. We can set the frequency of the lowpass filter for our simple-minded slicer based on the random statistics of one's density from the scrambler. To do this, let's examine the properties of a binomial sequence.

Binomial Distribution

A binomial sequence is one where each event has one of two outcomes, i.e.: the outcome of any event is either true or false, one or zero. If there is an equal probability and a random chance of the two outcomes, then we can make some calculations about the probability of a batch or these events (a string of bits) containing a large number of one's and a small number of zero's. One simple way to build this probability is to construct Pascal's triangle. This triangle is a graphical way to determine probability of a binomial distribution. We start with one event, which can have either a one or a zero bit. Since each outcome is equally likely we designate this as 1 1 — meaning one chance of a *zero* outcome, and one chance of a *one* outcome. If we conduct two trials, we have four possibilities: *one one*, *one zero*, *zero one*, and *zero zero*. This actually represents three states as far as one's-density is concerned: one case of both outcomes *one*, two cases of half of each, and one case of both outcomes *zero*. This can be represented as 1 2 1 — a total of four outcomes in two trials, with the two outside 1 characters representing the extreme outcomes. We can continue to construct these trials by a simple algorithm. Each outcome state is the sum of the two outcome states immediately above it to the left and the right.

Events	Outcomes
1	1 1
2	1 2 1
3	1 3 3 1
4	1 4 6 4 1
5	1 5 10 10 5 1
6	1 6 15 20 15 6 1
7	1 7 21 35 35 21 7 1

Table 8-1 *- Pascal's Triangle - the outcomes of a number of binary events*

In table 8-1, t can be seen that for 7 bits, there will be one outcome where all 7 bits are *one*, one outcome where all 7 bits are *zero*, and 35 outcomes where 3 of the 7 bits are *one*. So, the probability of all 7-bits being *one* is 1/128 (128 is the sum of all possible outcomes). We can continue to construct this table above, and as the number of events increases the probability of a long string of ones or zeros becomes pretty small quite quickly. The probability of mostly one's or mostly zero's is higher. Again though, the probability that most of them are well-mixed (the center area of the chart) becomes very large very quickly. In fact, if we continue to increase the number of events (the number of data bits we observe over), it is possible to reach a point where the number of outcomes in the center area of the chart is almost all the events, and the number of outcomes near the edge of the chart are very rare. If the time constant is set long enough, the probability becomes so rare, that we can accept an occasional event of this type biasing our slicing level, because it will contribute only very rarely to a bit error. The above table gets unwieldy for large numbers of events. We can compute the expected outcome more easily with a formula for the number of combinations of binary events. In the formula below, *n!* refers to *n-factorial*, which is the product of *n* and all lesser integers. For example 3!= 3-factorial = 3 * 2 * 1 = 6.

$$P(n) = \frac{N!}{n!(N-n)!}\left(\frac{1}{2}\right)^{N} \qquad (8\text{-}5)$$

where N = the number of events (i.e.: the number of bit times that we observe over), and n = the number of *one* bits present. Then $P(n)$ is the probability of exactly that quantity of *one* bits being present in the collected sample of events. For example, let's assume that we look over 20 bit times, then the probability of no *one* bits is $P(0)$, which is 1/1,048,576. The probability of a single *one* bit (and 19 *zero* bits) is $P(1)$, which is 20/1,048,576. The probability of two *one* bits (and 18 *zero* bits) is $P(2)$ = 380/1,048,576. By symmetry, we can state the same about the number of zero bits, i.e.: the probability of no *zero* bits (20 *one* bits) is *P(20)* = *P(0)* = 1/1,048,576 So, to determine the probability that our lowpass filter will see the extreme of *one's* density or of *zero's* density, we have to add up the probability of the outsides of the curve. How much of the tails of the curve need to be included? If we prevent the slicing level from moving more than half-way between the center and one of the decision levels, then we degrade the noise immunity by 6 dB. If we prevent it from moving by more than 20%, the noise immunity is degraded by about 2 dB. A 20% movement corresponds to the probability of having 20% more *one* than *zero* bits, or vice versa.

Therefore, in the 20-event case, we have to select the undesired events as being = P(0) + P(1) + P(2) + P(3) + P(4) + P(5) + P(6) + P(7) + P(13) + P(14) + P(15) + P(16) + P(17) + P(18) + P(19) + P(20). This probability of occurrence is the same as the probability 1 - (P(8) + P(9) + P(10) + P(11) + P(12)), which evaluates to a numerical value of 0.737. Thus, about one quarter the time the slicing level with a 20-bit time constant will be outside the 20% bounds. If we increase the time constant to 40-bit times, then the probability it will be outside the 20% bounds is 1- (2*P(16)+2*P(17)+2*P(18)+2*P(19)+P(20)). We can do this simplification because P(16) = P(24) so P(16)+P(24) = 2 * P(16). This probability equals 0.846. We are within the 20% range about 85% of the time. Table 8-2 summarizes the probability of being within 20% of the *half-one's* density point versus the number of observation intervals (bits). For very large values of N, the calculation gets rather tedious. Excel has a number of statistical functions that help automate the calculation of this probability.

Number of bits observed	Probability slicing level is within 20% of average
20	0.737
40	0.846
60	0.907
80	0.943
100	0.965
200	0.996
500	0.999994

Table 8-2 - *Using a lowpass filter to derive slicing level for scrambled data. Probability that level is within 20% of mean value (2 dB penalty for two-level modulation).*

Note that around 500 or so bits, the assumption is good, and the power penalty will be 2 dB or less most of the time. At 1000 bits the probability of exceeding the 2 dB threshold is close to zero. Thus, as a good approximation, we should set the lowpass filter time constant to about 1000 data bit times. For higher-level modulation formats, we can see that 20% variation is too much (since we have several eyes), and so this simple scheme is not really practical. Recalculating the above example (assuming a +/- 2.4% variation from the nominal value, and assuming the observation interval is 1000 bits), the probability of the running sum being within the +/-2.4% range is only 0.57. Thus, accurate slicing levels will require a very long time constant, many thousands of bit times long. If the radio link is a fixed, point-to-point link with continuous transmission, then the long time constant may not be a problem. However, with a multi-point system, where the key-up delay is significant, the long time constants are not generally desirable.

The above discussion shows the difficulty in trying to use a low-pass filter for slicing level determination. Furthermore, in the problem with AC-coupling data — the time constants have to be very long to be useful. Thus, the double peak detector circuit may be a better choice than a low pass filter based slicing level circuit, since the time constants may be able to be shorter. In order to determine the discharge time constant for the peak detector for scrambled data, the probability of sequence occurrence must be estimated, since there is no run-length limit as with NRZI-coded HDLC. Assuming that a few bits charge the peak detector sufficiently, then the probability that a long sequence occurs is $1/2^N$. If an arbitrary choice is made that 1 in 10^{-9} times the peak detector will be allowed to droop 5%, then the problem reduces to finding the sequence length. This can be calculated as:

$$N = \frac{\log 10^9}{\log 2} = 29.9 \qquad (8\text{-}6)$$

Or, about 30 bits. Thus, the time constant of the peak detector can be calculated as before (when a 6 bit period was determined):

$$T = \frac{30 * 104\mu\ \mathrm{sec}}{0.051} = .061\ \mathrm{sec} \qquad (8\text{-}7)$$

assuming that the eye height is about the same as the no-signal RC charge voltage. Therefore, the scrambled data peak detector for a 9600 baud signal would have a time constant of about 61 milliseconds for a 6-volt eye and 6-volt RC charge (no-signal). Again, the charge time constant should be on the order of a few bit times.

Fixed Slicing-Level Methods

Another way to perform the data slicing is to set a fixed slicing voltage, and then modify the received signal so that it is symmetrical around that fixed slicing level. One method is to hard-limit the received signal. In essence, the slicing is being performed in the limiting stage. A limiter is composed of a high-gain amplifier and a clipping or limiting circuit. Unfortunately, the problem of setting the DC-bias point of the limiter is, in fact, exactly the same as the problem of deriving the slicing level! Any signal higher than the DC-bias point will become limited at the *one* level, and any signal lower than the DC-bias point will become limited at the *zero* level. Assuming a DC-coupled receive signal with no drift and with a known reference point (perhaps zero volts), then the limiter must be biased at this known point. Alternatively, the limiter could be biased at the average-DC point of the received signal. This is exactly analogous to the case above, where the slicing reference level was derived based on low-pass filtering, and all the restrictions that are faced with long time constants will also concern us with this type of design. A limiter is not usable with multi-level received signals, since all of the different states will be limited into just two values, *zero* and *one*. Nevertheless, it is a feasible circuit design for two-level modulation. See figure 8-5 for a block diagram of a limiter-type slicing circuit.

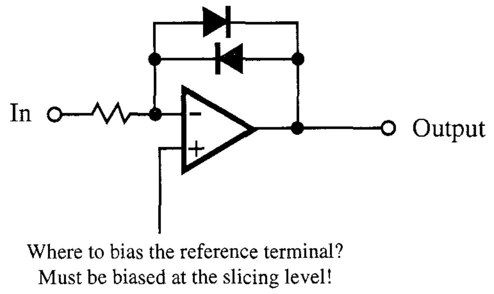

In
Output
-
+
Where to bias the reference terminal?
Must be biased at the slicing level!

Figure 8-5 *- block diagram of a limiter-type slicing circuit.*

A final method that can be used is to process the received signal through an AGC amplifier. Typically, the AGC circuit will measure the amplitude of the output signal, and adjust the internal gain of the amplifier so that the output matches the desired output amplitude. As long as the time-constants of the AGC amplifier are longer than the longest no-transition time of the received signal (analogous to the peak-detector method listed in the variable-slicing level circuit), then the control voltage of the AGC circuitry will maintain a constant gain value. Then the slicing level is set to the point half-way between the + and the - output levels. Since the output levels are constant, the slicing voltage is fixed, and known ahead of time (assuming that the signal does not undergo some type of unusual distortion). Slicing is a simple process when using a comparator with the AGC-controlled signal and the fixed slicing level as inputs. See figure 8-6 for a schematic of a circuit that performs this type of slicing. For multi-level modulation, the AGC type of slicer is generally used because a number of slicing levels must be determined rather accurately.

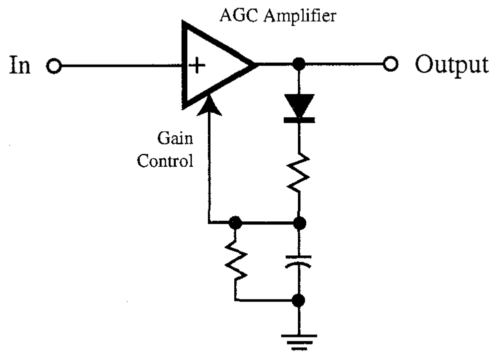


Figure 8-6 *- block diagram of an AGC-type slicing circuit. It is assumed that the data and noise are symmetrical.*

DSP-Based Slicing-Level Determination

With the use of a DSP-based receiver, it is possible to digitally compute the slicing level. If the receiver is properly equalized, then there will be no inter symbol interference (ISI) at the exact point in time that the receive baseband signal should be sampled. Thus, the DSP processor can measure the voltage at that exact instant, and then average a number of these samples (to reject noise). It should also neglect a voltage if it is distant from a good received signal point, indicating a corrupt sample. The averaged voltages then represent the optimum levels of the different symbol values, *zero* and *one* for a two-level eye, or -3, -1, +1, +3 for a four-level eye. The slicing level is then computed based on the distance between the symbol value levels, and the absolute value of the symbol levels (all, of course, at the optimum clock recovery time). This type of slicing level determination is very accurate, and adapts itself to changes in the DC level and in the received eye amplitude, within reason. The DSP software can be set to ignore differences in how many times each different symbol is sent. Of course, the DSP code must be able to acquire the proper levels when the received signal first arrives. So, there should be enough uncompensated accuracy for a stable starting solution. With random multi-level received signals, a DSP should be able to acquire a solution after several symbols of each level are received. See figure 8-7 for a diagram showing the levels and the derivation of the slicing levels.

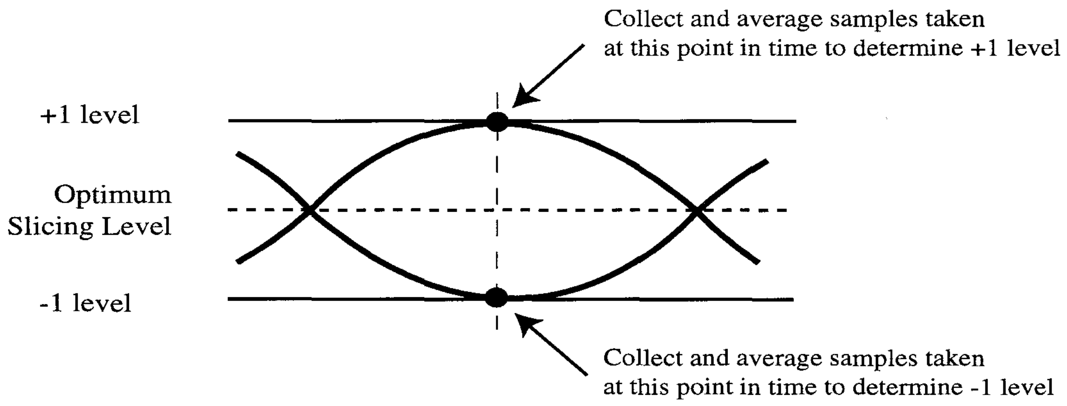

Collect and average samples taken at this point in time to determine +1 level
+1 level
Optimum Slicing Level
-1 level
Collect and average samples taken at this point in time to determine -1 level

Figure 8-7 *- Algorithm for DSP based receiver to acquire slicing level.*

Chapter

9 Clock Recovery

When receiving digital data, it is important to be able to extract the timing information related to the duration and the position of the data bits. This process is known as *clock recovery*, also sometimes known as *symbol timing recovery*. There are 2 reasons that we need to extract the clock of the received signal: 1) to provide a timing signal that allows the ability to optimally decide whether the received data bit is a one or a zero, and 2) to provide a clock signal to downstream devices (such as an HDLC decoder, UART, or decoder circuit) that clocks the bits into that device. We cannot always use our transmitted clock as the received clock (which we do with asynchronous data and a 16x clock) because the transmitter (at the other end) may be at a slightly different frequency than our receiver. Also,the local 16x clock might provide too coarse resolution for high-accuracy decoding of the received signal.

The Importance of Accurate Clock Position

In the chapter on frequency and impulse responses, we saw that careful construction of the channel frequency response would allow us to avoid inter-symbol-interference, but only if we sampled the received data bit at exactly the right time. It is the function of the clock recovery circuit to provide that clock at exactly the right time (that is, with the right phase-relationship to the received data). Figure 9-1 is a received eye pattern of an alpha=0.4 channel. If the data bit value is decided in the center of the bit, then the best possible immunity to noise is achieved. However, if the data bit value were to be decided at some other time, then it is obvious that the eye is not as open at those other times, due to energy from other data bits. Thus, the data bit value would be decided when noise can more easily corrupt the decided value.

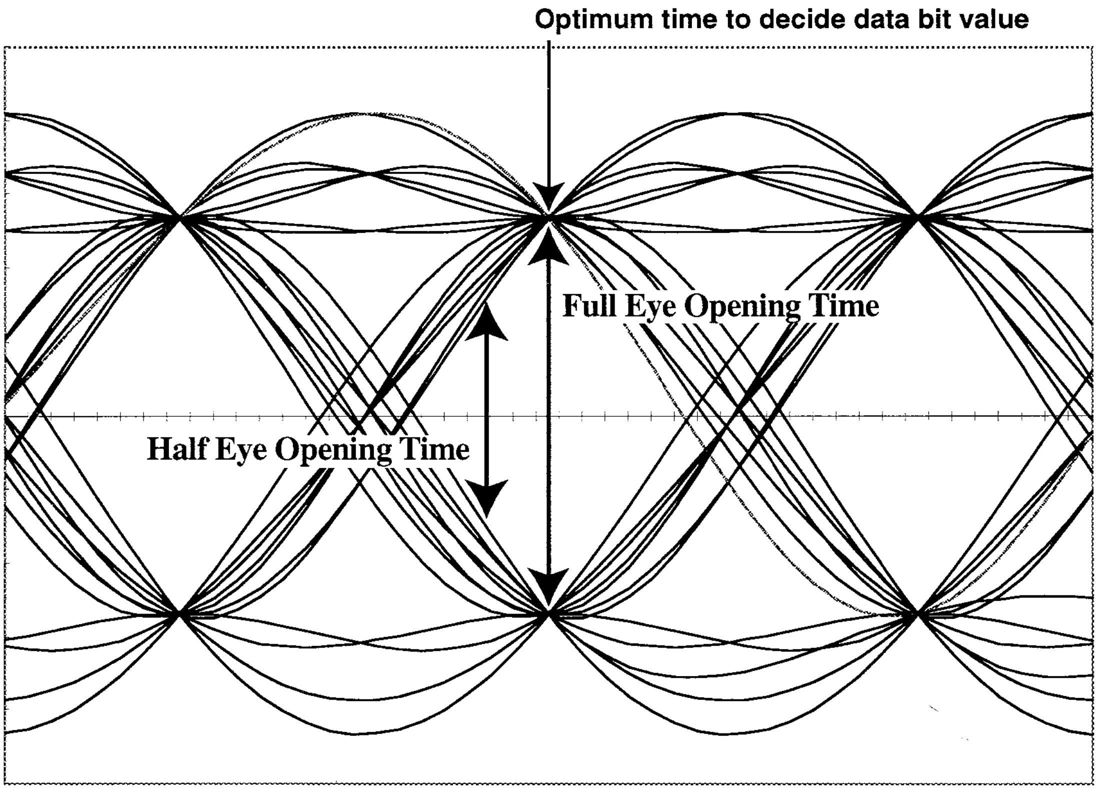


***Figure 9-1** - Received Eye Pattern for alpha = 0.4 filter.*

For the above eye diagram and sequence, it is possible to measure the decrease in eye opening as the decision time is moved away from the ideal sampling point. This allows a determination of how more easily susceptible that signal is to the same noise voltage. For example, if the sampling time is moved away from the center of the data bit by 1/4 of a bit time, the above eye is about one-half closed. This can be approximated as 6 dB. less signal, or a bit-error-rate vs. SNR penalty of about 6 dB. compared to sampling at the proper time. Of course, this only occurs for some data-bit sequences through the filter, but it's an upper-bound approximation that's not too bad. Taking the measured data from the calculation that produced the above graph, and the graph for the alpha=1.0 filter, a rough approximation of power penalty vs. clock position error can be derived (figure 9-2). It should be noted that 11.25 degrees = 1/32 of a data bit period. For low-alpha filters, the penalty for 1/8 of a data bit (45 degrees) can be several dB. For 4-level modulation, the penalty is vastly worse, and the recovered clock must be much more accurately recovered. See figure 9-3, which shows a 4-level eye with a filter alpha of 0.4.

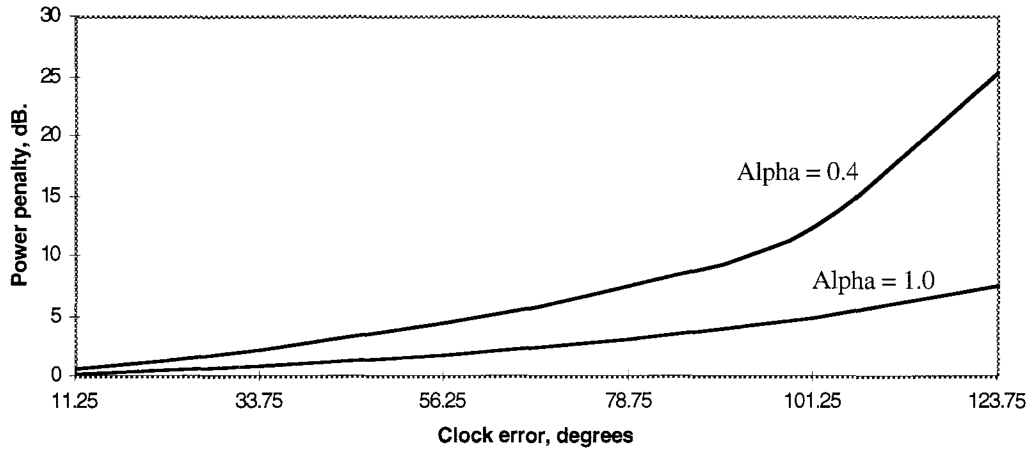


Figure 9-2 *- Approximate Power Penalty vs. Recovered Clock position error for 2-level modulation for some raised-cosine channels.*

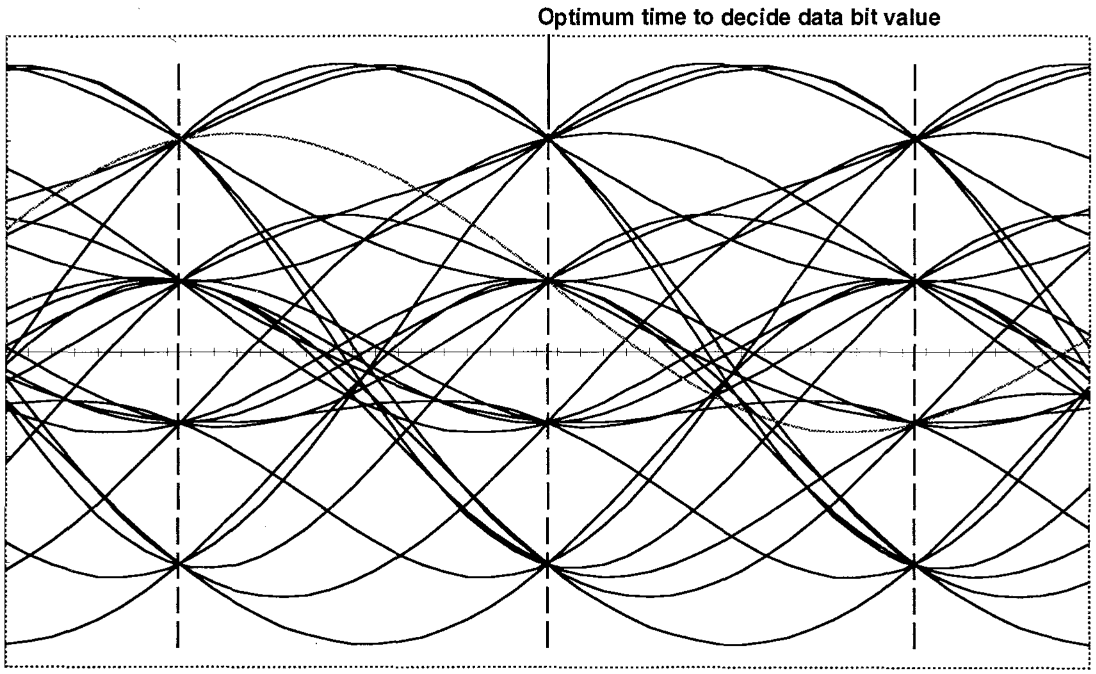


Figure 9-3 *-received 4-level eye pattern for an alpha = 0.4 filter (such as for 4FSK). A high degree of accuracy in the recovered clock position is necessary for a low error rate.*

Methods to Recover the Clock

One of the complicating factors behind recovering a clock that is synchronous with the transmitted baud rate is that the transmitted spectrum of a baseband signal usually contains no energy at the clock frequency! Thus, a simple filter is not possible. We must perform some non-linear operation on the received data in order to generate a new spectrum that does contain a significant amount of energy at the clock frequency. Then, that modified signal can be filtered to recover the clock spectral line. After the clock is recovered the phase of that recovered signal must be carefully adjusted to the correct sampling position in order to allow the detection of the received data bit. Figure 9-4 illustrates the clock spectrum as received, and figure 9-5 is a block diagram of an open-loop clock recovery process.

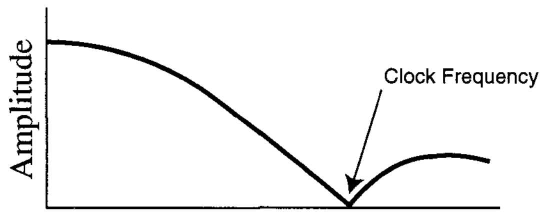


Figure 9-4 *- Spectrum of typical received baseband data signal, showing no energy at the clock frequency.*

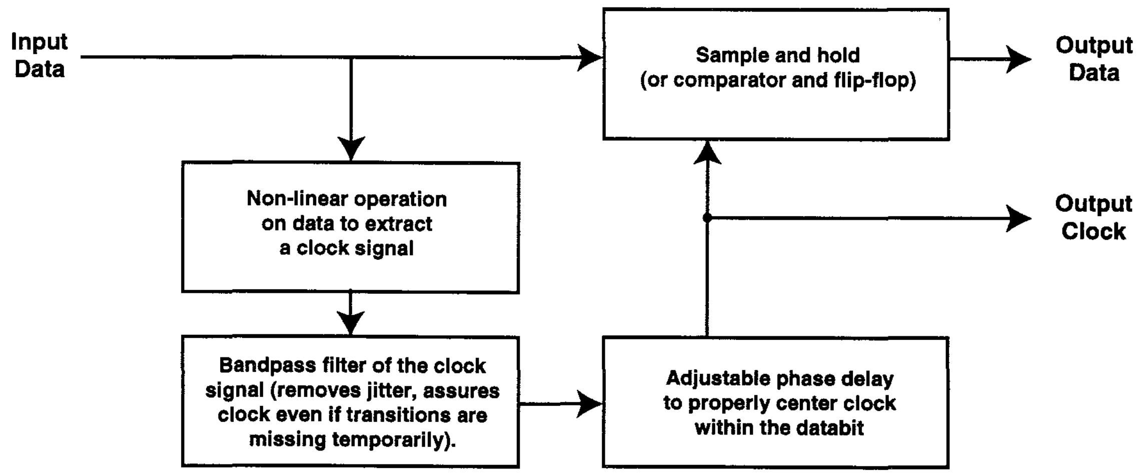

Input Data
Sample and hold (or comparator and flip-flop)
Output Data
Non-linear operation on data to extract a clock signal
Output Clock
Bandpass filter of the clock signal (removes jitter, assures clock even if transitions are missing temporarily).
Adjustable phase delay to properly center clock within the databit

Figure 9-5 *- block diagram of open-loop clock recovery (symbol timing recovery) process.*

After performing the non-linear operation on the data, we need to recall that due to the random nature of the data there will not be periodic transitions in the data. Thus, the non-linear circuit will be missing clock transitions for a few bit times every now and then. So, the output of the non-linear circuit must be filtered in order to supply a clock signal for those bit periods in which no transitions occur. In this case it would be desirable to have the filter-Q be high, so that the bandpass filter would continue to generate clock cycles through the possibility of many missing data bit transitions. Additionally, a high-Q filter will start to drift off of the desired phase position only very slowly during missing clocks. This leads to lower jitter on the received clock and better performance of the data bit detector. On the other hand, it would de desirable for the Q- to be low so that the filter quickly comes onto the right frequency after the start of the reception of a new station. Thus, there are several contradictory requirements for the filter Q-factor.

Several methods are available to perform a non-linear operation on the input baseband data. One method is to full-wave rectify the baseband data signal. Doing so requires that we obtain an accurate half-way amplitude level, which is designated the zero-level to the rectifier circuit. At low speeds, it is possible to use op-amp circuits to do this. First, the circuit has to generate the positive and negative peaks, then generate the half-way voltage as half-way between those two points, then rectify the signal. It is a simple matter to make a precision rectifier at lower frequencies. Figure 9-6 is a circuit for performing this operation. The output of this circuit then needs to be filtered with a bandpass filter. The schematic of a precision full-wave rectifier is shown in the section on carrier recovery. Derivation of the zero-level is the same process as derivation of the slicing level which is discussed in the section on data slicers.

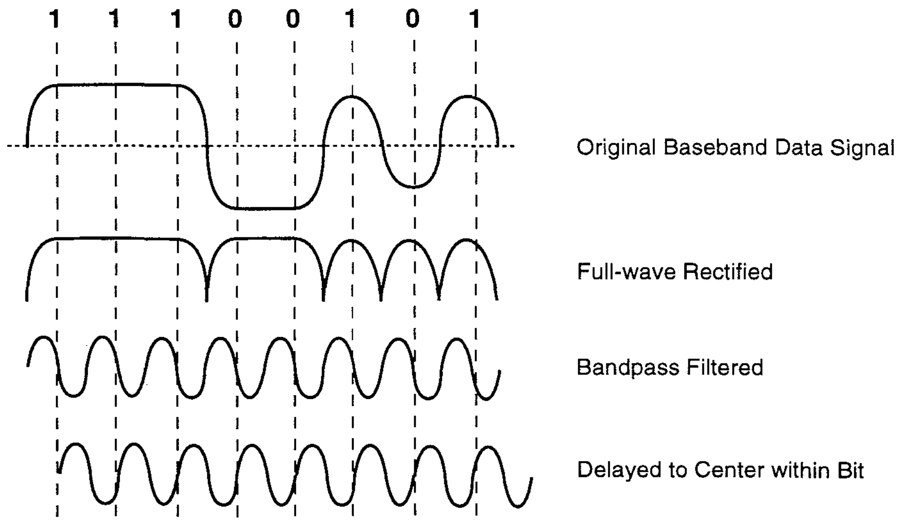


Figure 9-6 *- using full-wave rectification of the baseband data signal to recover clock.*

Another approach that can be taken is to differentiate the baseband data and then stretch the differentiated pulses. One simple approach to do this is with a digital pulse generator circuit. Figure 9-7 is a schematic of such a circuit. When the delay element that stretches the pulses is one-half a bit period long, the circuit will produce the strongest recovered clock. The output of this circuit also needs to be filtered with a bandpass filter, as shown in the figure.

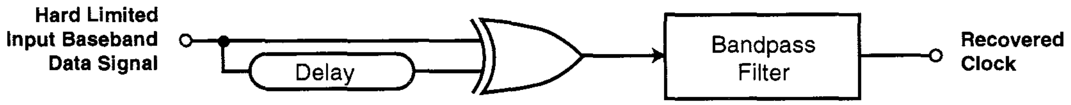


***Figure 9-7** - clock recovery based on transition detector and digital delay generator.*

Alternatively, an all-digital clock recovery circuit can be used. A popular circuit designed by Paul Newland, AD7I, was used on the TAPR TNC-2. In this circuit, a 16x clock runs a digital state-machine. The baseband received data is hard-limited and fed to the state machine. It looks for transitions in the hard-limited data and decides to advance or retard its output clock by one state (1/16 of a data bit time). This depends on whether the generated 1x clock is early or late. This can be easily implemented as an EPROM and an 8-bit register. A later version of this circuit operated at 64x the incoming data bit rate and, thus, had higher accuracy in the recovered clock position. One disadvantage of this general type of clock recovery circuit is that it has no proportionality in the feedback. In other words, the output recovered clock is either early or late, and its position is always corrected either forward or backward 1/16 of a clock cycle when a data transition occurs. This can be equated with a servo-loop known as a *bang-bang* servo. One-half of an adjustment time (1/32 of a data bit in the case of 16x clock) of error is always guaranteed in the absolute best possible case. In practical cases, the clock position error will be more on the order of +1/8 and -1/8 of a bit time (for a 16x clock) due to timing drift and noise. Along with good received signal-to-noise ratios, this error band may be acceptable for 2-level modulation and wide filters, yielding perhaps 2db of power penalty in those cases. Narrower filters, multi-level modulation, or low signal to noise ratios will benefit from better performance than this technique provides.

Closed-Loop Clock Recovery

Another method to recover clock is to utilize a circuit that determines the optimum clock position by deriving an *error voltage* — that is, a voltage proportional to the distance from the optimum clock position. This can then be used to direct a feedback loop to lock the clock to the correct position. An analog implementation of this type of circuit is known as an *early-late* gate synchronizer. It operates by sampling the received eye pattern at two slightly different times. The optimum position is when the two samples are the same amplitude. Figure 9-8 is a block diagram of an early-late gate synchronizer. Figure 9-9 is a diagram showing the early and late voltages as a function of position of the recovered clock based on a matched filter for rectangular transmit pulses. This not only makes the example easier to visualize, but it works for any type of eye pattern. Since it is possible that two consecutive bits could be both *zeros* or both *ones*, it would seem that invalid error information could be obtained. However assuming random data, these cases all eventually cancel out. Only the correct error information remains — the synchronizer only derives accurate clock position error information during transitions. The absolute value circuit assures that regardless of the sign of the bit, the circuit still gives the appropriate error voltage. You may recall that a logic zero is transmitted as a -1 voltage, and a logic one is transmitted as a +1 voltage — zero voltage is an invalid signal level half-way in between a logic zero and a logic one.

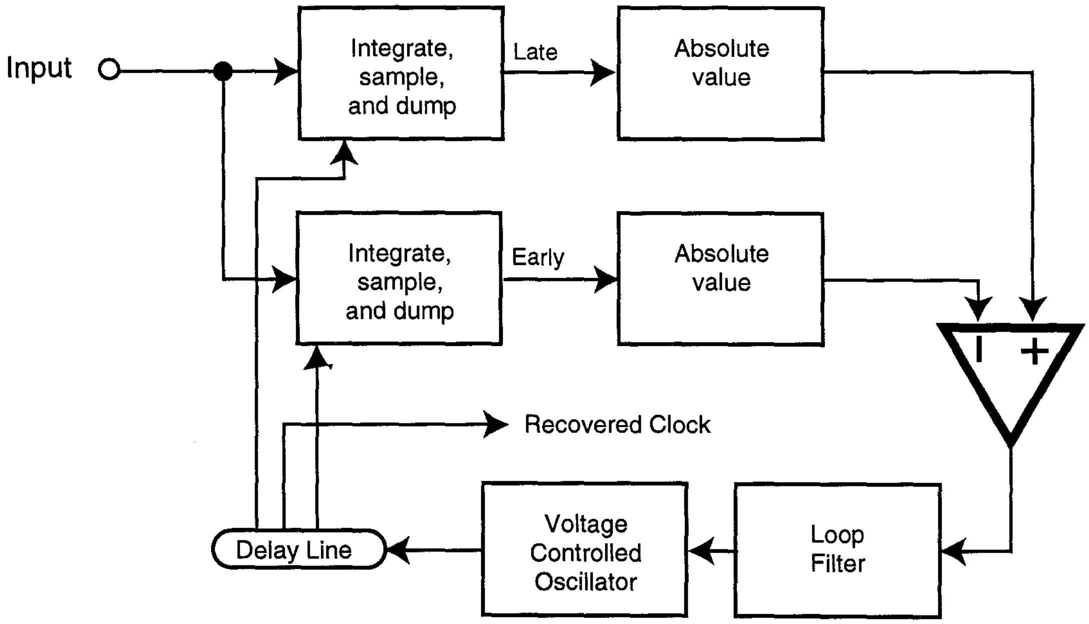
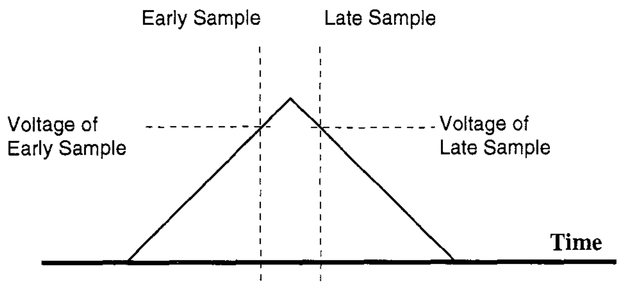
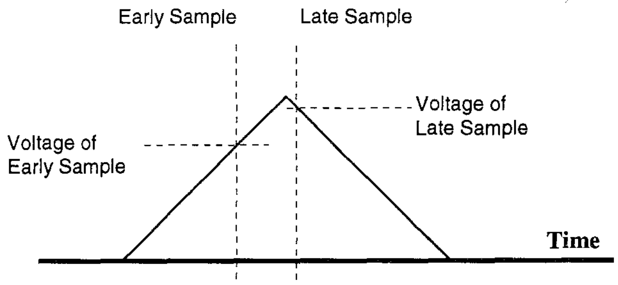

Input
Integrate, sample, and dump
Late
Absolute value
Integrate, sample, and dump
Early
Absolute value
−
+
Recovered Clock
Delay Line
Voltage Controlled Oscillator
Loop Filter

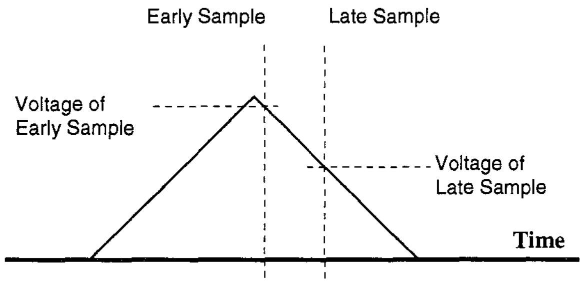
Figure 9-8 *- block diagram of early-late clock recovery circuit. The data must be randomized (scrambled).*



Properly centered clock:
Early and Late sample voltages are the same



Recovered clock leads correct position:
Early voltage is less than late voltage



Recovered clock lags correct position:
Early voltage is greater than late voltage

***Figure 9-9** - Error voltage of an early-late gate clock recovery system. The triangular voltage is the output eye pattern of an integrator circuit (assuming no dump has occurred) for a one-zero bit combination.*

A low-pass filter stores the energy of adjacent similar bits and errors due to noise, resulting in an error voltage that smoothly varies with position of the recovered clock vs. the proper bit time. This circuit behaves like a phase-locked loop (except that the phase detector is a little unusual), and the loop calculations can be performed just like a PLL — see the section on Phase Locked Loops for clock and carrier recovery.

Clock Recovery Filters

The bandpass filter in the clock recovery circuit serves two purposes. First, it assures that when the input data stream is missing transitions (for example, two consecutive one bits will have no transition in between) the output of the filter will ring anyway and produce a clock. Second, the filter removes spectral components near the clock frequency that are caused by data patterns and noise. If this were not done, then the actual position of the recovered clock would drift around and possibly move off of the center position of the received databit, degrading the bit error rate.

The Q-factor of the filter (derived from the use of a ringing tank as a bandpass filter) determines how long the filter will ring in the absence of input transitions. With randomly scrambled data, the probability of a missing transition is a binomial distribution. For example, the probability of 30 consecutive *ones*, or of 30 consecutive *zeros* is:

$$p = \frac{1}{2^{30}} + \frac{1}{2^{30}} = \frac{2}{2^{30}} \qquad (9\text{-}1)$$

which evaluates to about 1.86 x 10^{-9}. Therefore, if the clock position does not drift off from the center position too far in 30 bits, the error rate should be acceptably low. In the case of a filter, the output is on-frequency and in phase-alignment when continuous transitions are received (alternating one-zero pattern for example). When transitions are lost, then there is no input to the filter, the filter output decays down, and the output phase drifts away from the center of the bit position. Thus, as the transition-density of the input changes, the output frequency of the bandpass filter will change, as well as its phase. This causes pattern-dependent *jitter* in the output of the bandpass filter. A rough rule-of-thumb is that the output of a 2nd-order bandpass filter decays to 37% of its output amplitude in Q/PI cycles (Lancaster, 1975). A rough estimate of the required filter Q would be on the order of 100 for use with scrambled data. A much lower filter Q can be used if the received data is guaranteed to have frequent transitions. For example non-scrambled HDLC data will have a transition at least every 6 bit times. When using active filters with high-Q's, care must be taken that the filter is not driven into clipping (since the gain of the filter is likely to be quite high). It is also possible to use a phase locked loop as a filter for clock recovery. Phase locked loop filters are discussed in more detail in the section on Phase Locked Loops for clock and carrier recovery.

Filter Transfer Function

It is important that the filter transfer function be smoothly decreasing from the center of the channel. Otherwise, amplification of the jitter due to data components away from the clock frequency will be enhanced. Figure 9-10 illustrates that the single-sided response of the filter is the same as the baseband response. If the passband response has a dip in the center, then that is equivalent to a baseband response that has more gain away from the center frequency. This causes the noise and data spectrum at the peak frequency to be amplified more than the desired clock frequency which leads to the potential of excess jitter, or movement of the clock away from the desired position in the center of the bit time.

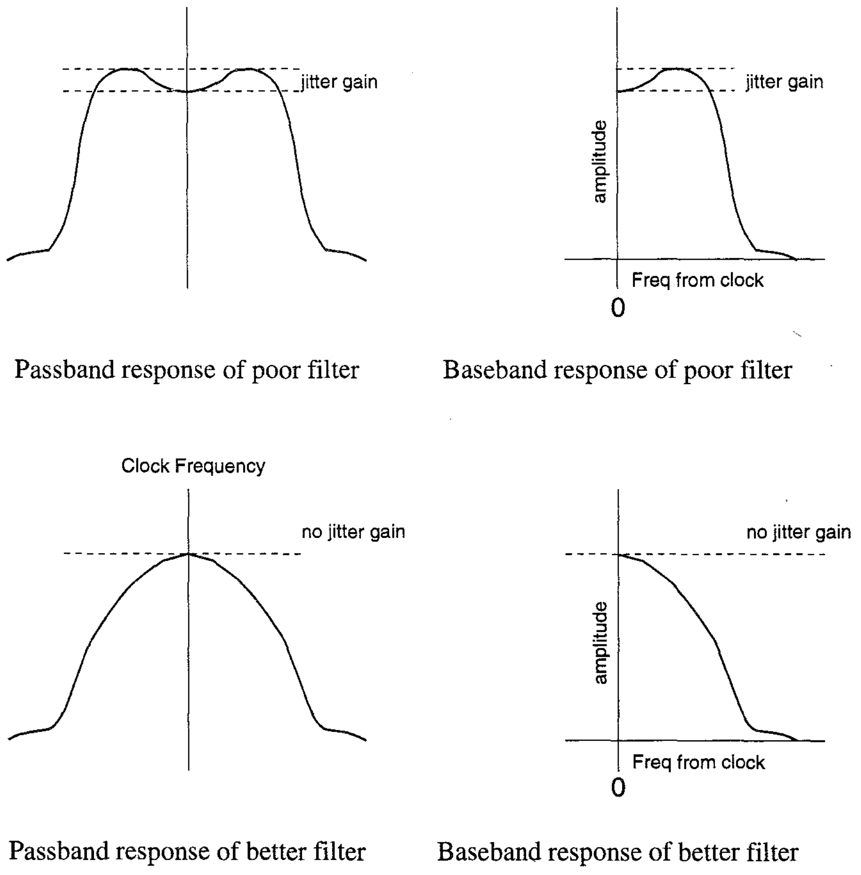


Figure 9-10 *- Passband and baseband equivalent response of clock recovery filter. A smoothly decreasing response has better jitter rejection properties.*

The general subject of data synchronization can cover both clock and carrier recovery, and is addressed tutorially by Gardner (1979).

Modulation-Based Clock Recovery

With the advent of digital signal processors, additional methods to recover clock have been described. One method is called Modulation-Derived Synchronization (MDS) (Grayson and Darnell, 1992). In this method, a single-bit of the received tone is compared against a number of delayed replicas of a single reference frequency. The maximum-likelihood event is the particular time-staggered reference tone that most closely aligns with the received data bit (computed by coherent correlation). It is then assumed that the center of that data bit (or some other defined time relative to that bit) is the proper clock time. In actuality, if several consecutive bits of a particular tone were transmitted, then all staggered-references would become equally-valid, and no clock-position error information would be available for a period of time. The direct extension to this technique is to compare the time-staggered reference tone replicas against a number of bit times, and pick the most likely match based on this longer period of correlation. The referenced article shows that 64 bit-times of correlation gives good recovered clock accuracy, even at low Eb/No ratios. However, this requires a significant amount of DSP computation. In essence, this technique performs a parallel search of the correct clock position.

Another technique for deriving the proper clock position involves sampling the received data bit at a rate much greater than once-per-baud-time. If the data are convolutionally-coded by the transmitter, then the Viterbi error-metric for each of the different sampled phases can be compared. The sampling phase with the lowest Euclidean Viterbi error metric (soft decision) becomes the best estimated time to recover clock (Honary, Zolghadr, Darnell,and Maundrel, 1992). A DSP implementation of this technique can then just use that particular best Viterbi path as the received data stream.

Chapter 9 - Reference

Gardner, Floyd M. "Phaselock Techniques," 2nd ed. John Wiley & Sons, Inc. 1979, Chapter 11.

Grayson, M. and M. Darnell. "Enhanced Symbol Synchronization Technique for MFSK Demodulator". Electronics Letters, Vol. 28, No. 6, March 12, 1992., pp 564-566.

Honary, B., Zolghadr, F., Darnell, M. and M. Maundrell. "Improved HF Modem using Convolutional Coding". Electronics & Communication Engineering Journal, April 1992, pp. 73-82.

Lancaster, Don. "Active-Filter Cookbook." Howard W. Sams & Co., Inc. 1975, pg. 167.

Chapter

10 Carrier Recovery

In all coherent modulation and demodulation operations, it is necessary to recover a synchronous carrier signal with which to demodulate the received signal. This section examines techniques to recover the original carrier. It is assumed that any phase ambiguity in the received carrier will be taken care of elsewhere in the demodulator, probably by differential coding of the data. One of the difficulties in recovering the carrier is that many times no discrete spectral line appears at the carrier frequency. Therefore, some non-linear operation has to occur to regenerate the discrete spectral line before filtering can take place. In the following description, we will assume that a PSK signal is being received. However, the methods can usually be extended to QAM by limiting the carrier at an appropriate point.

Deriving a Carrier Reference

In the simplest case of 2PSK, it is clear that one cannot just filter the frequency around the carrier. This is because the carrier reverses phase periodically, and thus any filter would ring-down before starting back up in the opposite phase. It is necessary to remove the phase changes before the filtering. One way to do this is to double the frequency of the carrier. When the frequency is doubled, the phase states of 0-degrees and 180-degrees also double in phase and become 0-degrees and 360-degrees respectively. Thus, doubling the frequency effectively removes any phase changes. Then it is necessary to filter the double-frequency carrier and then divide by 2 to recover the original frequency. Similarly, for QPSK, it would be necessary to quadruple the received signal so that the 0-, 90-, 180-, and 270- degree phase changes become 0-, 360-, 720-, and 1080- degree changes respectively. This action again removes all phase changes from the incoming carrier. Then the recovered 4x carrier would have to be divided by 4 to return the original carrier frequency. Even though the original carrier frequency is recovered by division, there is an ambiguity in the process — the received carrier could phase align with any of the 4 possible phase states. Thus, the 4PSK data must be coded in some manner (such as differential coding) so that it is unimportant which of the phase states actually is locked to.

Carrier Generation by Rectification

There are a number of techniques to double the carrier frequency (most of which can be applied repeatedly to produce 4x, 8x, etc. carriers). The simplest way is to full-wave rectify the received carrier, then band-pass filter the result. Figure 10-1 is the schematic of a full-wave op-amp type rectifier. Figure 10-2 illustrates the doubling of a 2PSK carrier by this technique. One of the drawbacks of full-wave rectification is that it produces some distortion of the 2x carrier. Accordingly, producing a 4X or 8x carrier requires good band-pass filtering between each rectification step. A carefully adjusted op-amp type precision rectifier (for audio frequency ranges) can give good results.

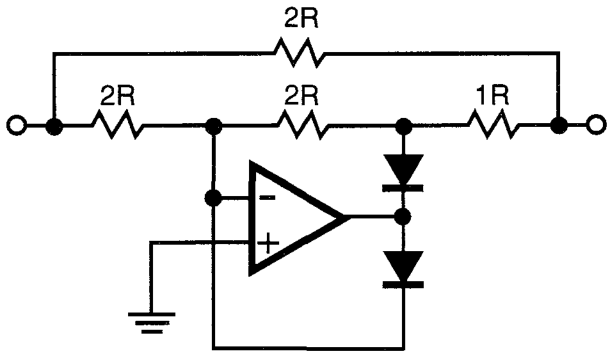


***Figure 10-1** - full wave rectifier*

Note that the exact resistance of the resistors is not so critical, but that the resistor *ratios* of the full-wave rectifier must be accurate to prevent fundamental leakage through the circuit.

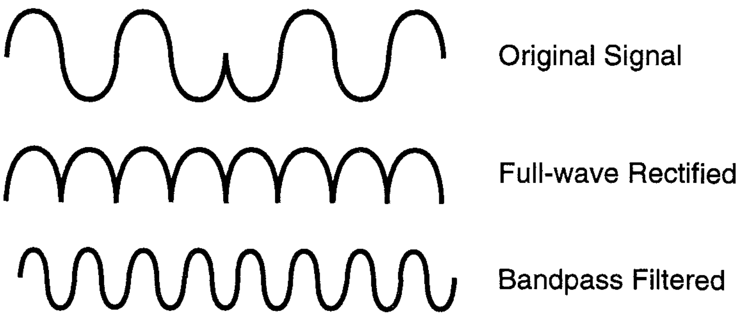


***Figure 10-2** - Doubling the carrier frequency by full-wave rectification*

Carrier Generation by Multiplication

Another method to double the carrier frequency is by multiplication. This can be done either of two ways: 1) squaring the input carrier, or 2) multiplying the input carrier by a quadrature (90-degree delayed) version of the input carrier. To illustrate, assume that the received carrier (without modulation) is equal to:

$$\mathit{Input} = \cos(\omega t) \tag{10-1}$$

Then multiplication of the carrier by itself (such as with a 4-quadrant multiplier) is:

$$\mathit{Output} = \cos^2(\omega t) \tag{10-2}$$

A useful trigonometric identity is:

$$\cos(2x) = 2\cos^2(x) - 1 \tag{10-3}$$

Re-arranging this identity yields:

$$\mathit{Output} = \cos^2(\omega t) = \frac{\cos(2\omega t) + 1}{2} \tag{10-4}$$

Which shows that the output of the multiplication will be a double-frequency carrier of 1/2 the input amplitude, with a DC offset. This output will have to be filtered to remove noise, jitter, amplitude, and missing pulses from the input signal, but not to reconstruct the signal (as with full-wave rectification). Figure 10-3 shows a circuit to produce the double-frequency carrier by squaring.

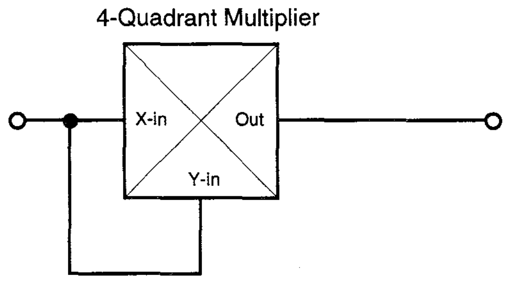


Figure 10-3 *- producing the doubled-frequency carrier by squaring (multiplying the input by itself).*

Alternatively, if the input carrier is delayed by 90 degrees, and then multiplied (such as in a 4-quadrant multiplier) with the input, the resultant signal would be:

$$Output = \cos(\omega t)\ \sin(\omega t) \qquad (10\text{-}5)$$

Again, a useful trigonometric identity is:

$$\sin(2x) = 2\sin(x)\ \cos(x) \qquad (10\text{-}6)$$

Which obviously leaves the output as:

$$Output = \frac{\sin(2\omega t)}{2} \qquad (10\text{-}7)$$

Hence, the output would be a carrier at twice the frequency, 90-degrees removed in phase from the previous carrier, of half amplitude. In practice, delaying the input carrier by 90-degrees is not common, since the same result can be achieved without the delay. Figure 10-4 shows a circuit to produce the double frequency carrier by multiplication.

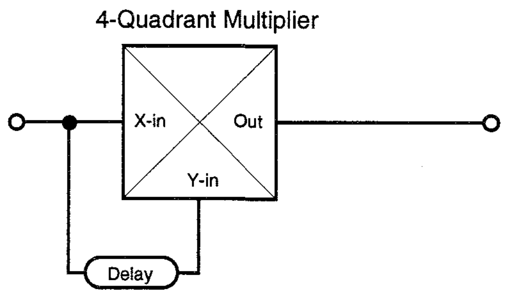


Figure 10-4 *- producing the doubled-frequency carrier by multiplication of the input with a quadrature carrier.*

Other Methods of Carrier Generation

Yet another approach for recovering the carrier is by differentiating the input signal, then delaying it and adding it to the non-delayed version. This is similar to the process used to derive the baud clock. A tight band-pass filter is required since there will be missing transitions during carrier phase changes. Typically a phase-locked loop would be used to reconstruct the carrier. In fact a PLL is commonly used with all the methods shown. The PLL circuit is used to implement a narrow, high-Q bandpass filter with small phase tracking error. One of the disadvantages of the PLL is that its acquisition time may be too long if the loop constants are not matched carefully to the application. When the radio link is full duplex, this is not usually an issue. If the link is half-duplex, then the PLL can affect the T/R turn around time. Normally, the phase detector used should not be a phase-frequency (P-F) detector. The P-F detector will slew the PLL off frequency on a missing input transition, a highly undesirable effect. The use of an exclusive-or type phase detector will degrade the pull-in performance of the loop, but will not adversely slew the output frequency due to a missing pulse.

2PSK Carrier Recovery - Costas Loop

The Costas loop acts to recover the carrier and demodulate a 2PSK signal all in the same circuit. Figure 10-5 illustrates the 2PSK Costas loop demodulator. The two signals that are low-pass filtered are both the data output and the correction to a voltage controlled oscillator (VCO). As the carrier phase from the VCO deviates from the correct point, the product of the two low-pass signals corrects the phase.

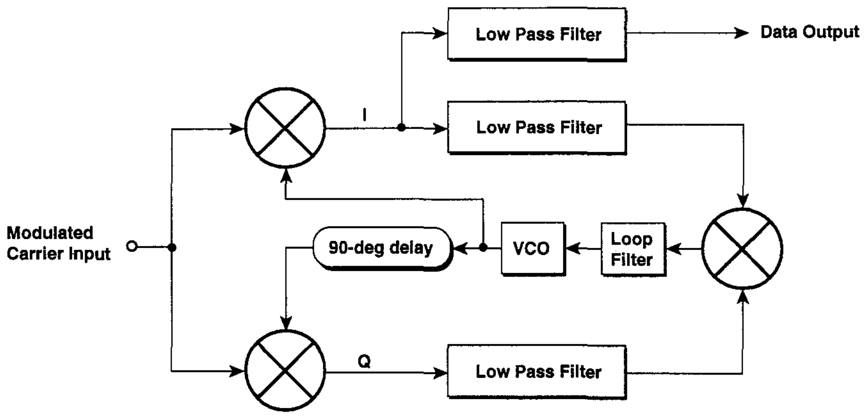


Figure 10-5 *- Costas loop for carrier recovery and demodulation of 2PSK signal*

The operation of the Costas loop can be visualized as follows: when the VCO output is properly phase aligned, then the quadrature channel (the 90-degree delayed version of the VCO) will be orthogonal to the data, and the output of its lowpass filter will be zero (the lowpass filter removes the double-frequency term). This is then multiplied by the upper arm signal, which is either +1 or -1 (depending on the received binary bit value) and the output of the multiplier feeding the VCO is zero (it provides no correction up or down to the VCO frequency and phase). If the VCO slightly leads the desired phase position, then the lower arm will start to develop an output error signal. Because both error signals are of the same sign, the VCO will be corrected the same direction regardless of whether a zero or a one data bit was received. Similarly, if the VCO slightly lags the desired phase position, then the lower arm will develop an error of the opposite polarity, and the VCO will be corrected in the other direction. Thus, error correction signals will always cause the loop to lock. However, if the carrier is significantly off-frequency, then the loop will not lock when the carrier is outside the main lobe of the PSK spectrum, or it can achieve false-lock on one of the side lobes. The VCO has to be close to start with, but once locked it will track carrier frequency variations. At the 45-degree point the error voltage is the greatest, decreasing back to zero at 90-degrees. However, at the 90-degree point the sense of the loop control voltage to the VCO is inverted, so the loop is unstable at this point. Eventually noise or drift will kick the loop out of this point and it will stabilize at either the zero-degree or 180-degree point.

It has been shown that the Costas loop has operational characteristics much like the squaring carrier recovery circuit discussed previously. The advantage of the Costas loops is that all circuitry operates at the carrier frequency (rather than at twice the carrier frequency). Consequently, it may have some practical advantages at very high carrier frequencies. The Costas loop can also be implemented in a DSP-based receiver, and the lower circuit speeds may or may not translate to a DSP implementation advantage.

The design of phase locked loops for carrier and clock recovery is covered in the section on Phase Locked Loops for clock and carrier recovery. Some other methods of carrier recovery are described in (Sari, Karam, & Jeanclaude, 1995)

Chapter 10 - Reference

Sari, Hikmet, Georges Karam, and Isabelle Jeanclaude. "Transmission Techniques for Digitial Terrestrial TV Broadcasting," IEEE Communications Magazine, Feb 1995, pp 100-109.

Chapter

11

Phase Locked Loops for Carrier and Clock Recovery

Phase locked loops (PLL's) are used for carrier and clock recovery in many modem designs. The purpose of the phase locked-loop is usually to act as a narrow bandpass filter. The equivalent bandpass response of a PLL is simply the baseband response translated up to the carrier (or clock) frequency. For example, if the PLL response is -3 dB at 5 Hz. at baseband, then the equivalent bandpass filter will have a width of +/- 5 Hz. at the -3 dB. points. The PLL has the advantage of course, that it can track the exact frequency and phase of the signal it will filter (once it has acquired lock). Proper design of the control loop is important in controlling the PLL loop bandwidth, and in setting the loop dynamic properties (such as settling time, phase margin, and peaking). A block diagram of a PLL is shown in Figure 11-1.

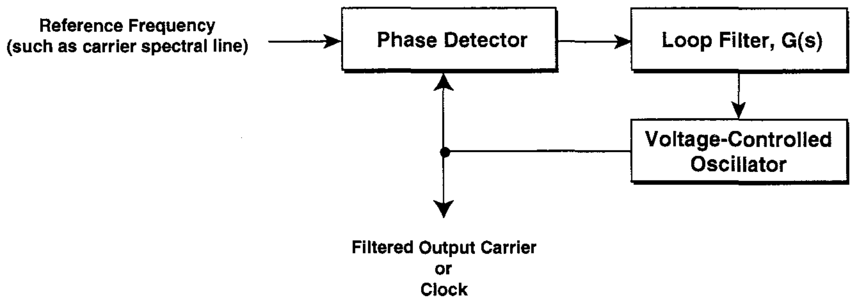

Reference Frequency
(such as carrier spectral line)
Phase Detector
Loop Filter, G(s)
Voltage-Controlled Oscillator
Filtered Output Carrier or Clock

Figure 11-1 *- Block diagram of phase locked loop filter for carrier or clock recovery. The PLL essentially bandpass-filters the carrier or clock frequency.*

The phase detector compares the phase of the two inputs, and then outputs a voltage that is proportional to the phase difference. The loop filter averages the phase detector output voltage, and limits the slew rate and bandwidth of the control voltage to the voltage controlled oscillator (VCO). The voltage-controlled oscillator outputs a sine wave with a frequency proportional to the control voltage. In order to analyze the properties of the loop, a few constants need to be defined:

Kv is the gain constant of the phase detector, in volts per radian of phase difference. For example, one volt output when the reference signal and the VCO signal differ by one radian (57 degrees) in phase.

G(s) is the transfer function of the loop filter, with *s* being the complex frequency variable, equal to *j* times *omega.*

Kf is the gain constant of the voltage controlled oscillator, in radians/second per volt, the change in output frequency for an input control voltage change of one volt. Since the oscillator's frequency and not its phase is controlled by the input voltage, the phase change of the output is *Kf* times 1/*s*.

Closed Loop Response

The closed-loop phase transfer function can then be written in terms of the phase of the input and output signals as:

$$Out = (In - Out)\, K_v G(s) \frac{K_f}{s} \tag{11-1}$$

Solving for the phase transfer function Out/In yields

$$H(s) = \frac{Out}{In} = \frac{G(s)\, K_v\, K_f}{s + G(s)\, K_v\, K_f} \tag{11-2}$$

Which resembles a low-pass filter, which has a transfer function in the form of

$$LowPass = \frac{1}{s+1} \tag{11-3}$$

The closed-loop transfer function *H(s)* describes how the phase of the output relates to the phase of the input reference in terms of the complex transfer function (gain and phase). For example, assume that the loop filter is a low pass filter with a DC gain of 10, and a break frequency of 5 radians per second. This is a simple R-C low pass filter (single pole) with some added gain. The low pass filter transfer function is:

$$G(s) = \frac{50}{s+5} \tag{11-4}$$

Substituting equation (11-4) into equation (11-2) yields the phase locked loop transfer function. This is plotted in Figure 11-2 for a value of Kv = 1, and Kf = 10. This is easily done with Excel 5.0, which has functions for complex numbers and many of the basic complex operations.

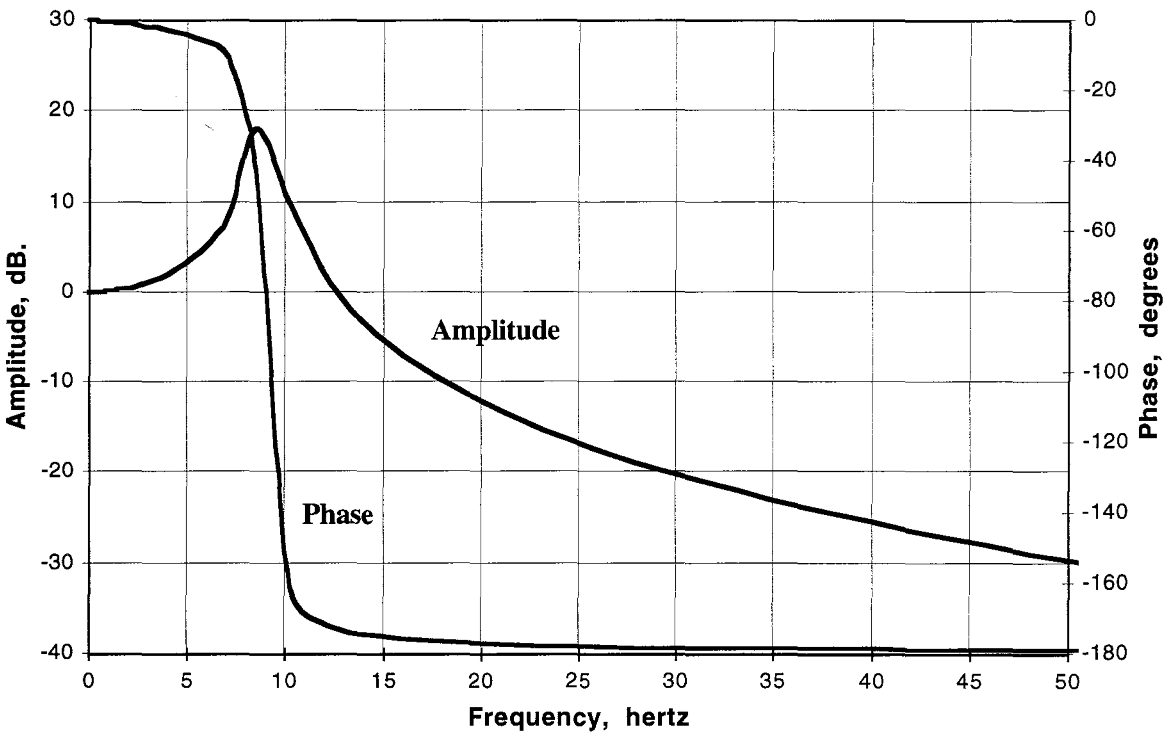


Figure 11-2 *- Loop transfer function H(s) for phase locked loop with a simple low-pass loop filter.*

As can be seen, this loop has very poor properties when used for clock recovery or for carrier recovery. At about 9 hertz, the loop has approximately 20 dB. gain, meaning that noise 9 hertz removed (both plus and minus) from the reference frequency will be amplified by a factor of 10 times, contributing much jitter to the output signal from the VCO. At zero hertz, of course, the output matches the input (no gain) and the phase of the output is the same as the phase of the input (zero degrees). At high frequencies, the jitter noise on the output is -180 degrees compared to the input jitter. This loop is very underdamped.

Open Loop Response

A useful way to view the stability of a loop is to examine the open-loop transfer function. Then stability is achieved if the open-loop gain is less than zero by the time -180 degrees of phase shift is reached. To calculate the open loop gain, equation (11-1) is re-written to remove the feedback of OUT to the phase detector. This can be written as:

$$Out = In\ K_v G(s)\ \frac{K_f}{s} \tag{11-5}$$

The open-loop response is then:

$$\frac{Out}{In} = \frac{K_v\ G(s)\ K_f}{s} \tag{11-6}$$

One of the difficulties in making a PLL stable is seen from examining equation 11-6. The VCO contributes 90-degrees of phase shift due to the 1/*s* integrator pole. So, there is only 90 degrees of total phase to work with in filtering the feedback signal. The feedback is negative, which contributes 180 degrees, while the VCO contributes 90 degrees. When the phase shift is 360 degrees, the loop oscillates (the PLL is unstable) if there is any gain at that feedback frequency. This implies that only a single-pole feedback network can be used (i.e., an RC low-pass filter). The simple example shown in Figure 11-2 illustrates that even with a single-pole feedback network, the loop can be very underdamped (depending on the exact gain and filter parameters). The open-loop response of the PLL in Figure 11-2 is shown in Figure 11-3.

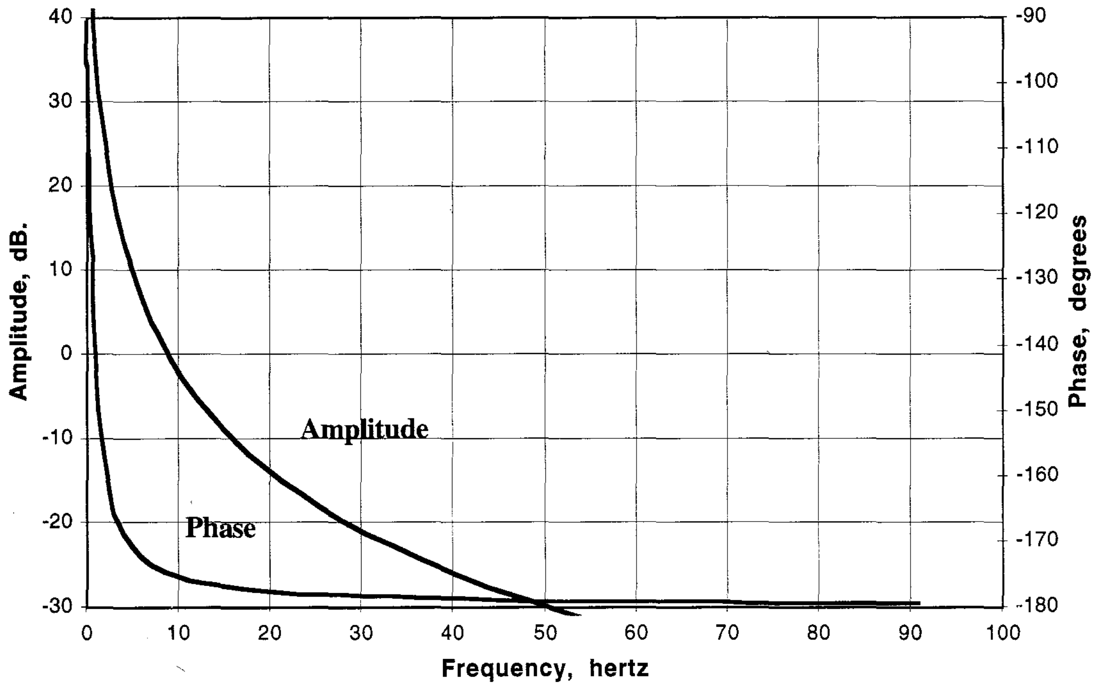


Figure 11-3 *- Open-loop response of PLL shown in Figure 11-2.*

It can be seen in Figure 11-3 that the amplitude crosses 0 dB at about 9 hertz, and the phase is about -174 degrees at that point. This loop has 6 degrees of phase margin, which is very little indeed. Additional poles due to op-amp response etc., may in fact add a few degrees of phase shift at 9 hertz, resulting in loop instability.

Lead Lag Filter

To compensate for the excess phase build-up of the single-pole network, a lead-lag network can be built. This consists of two resistors and a capacitor, as shown in Figure 11-4. The second resistor cancels the phase lag at high frequencies with a lead pole, resulting in a more stable loop response if the two poles are properly placed. The phase and frequency response of the lead-lag filter is shown in Figure 11-5. It can be noted that the phase returns back to zero at high frequencies, helping the PLL loop stability. The loop response of this network is shown in Figure 11-6 for a lead pole of 20 radians/second and a lag pole of 5 radians/second, resulting in a network attenuation of 14 dB at high frequencies (the attenuation can be increased by making the poles farther apart in frequency, but then the maximum phase lag is increased, resulting in less damping). The closed-loop PLL peaking is reduced with the lead-lag network, but the out-of-band response is not as good. In general, improved loop damping which reduces peaking unfortunately degrades out-of-band response roll-off. Lead-lag networks can be used to good advantage if a passive loop filter is needed and the poles and gain are carefully controlled.

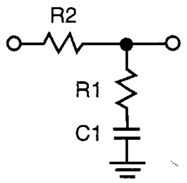


Figure 11-4 - *Lead-Lag filter network*

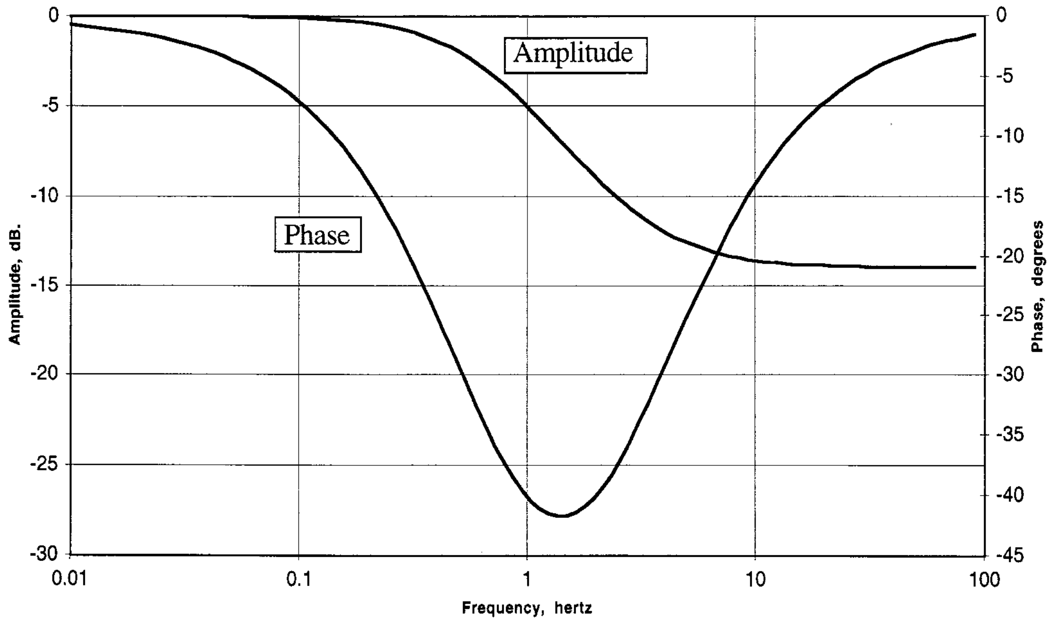


Figure 11-5 - *Amplitude and phase response of lead-lag network alone. Poles of the network are at 5 and 20 radians/second. Note the log frequency scale is in hertz.*

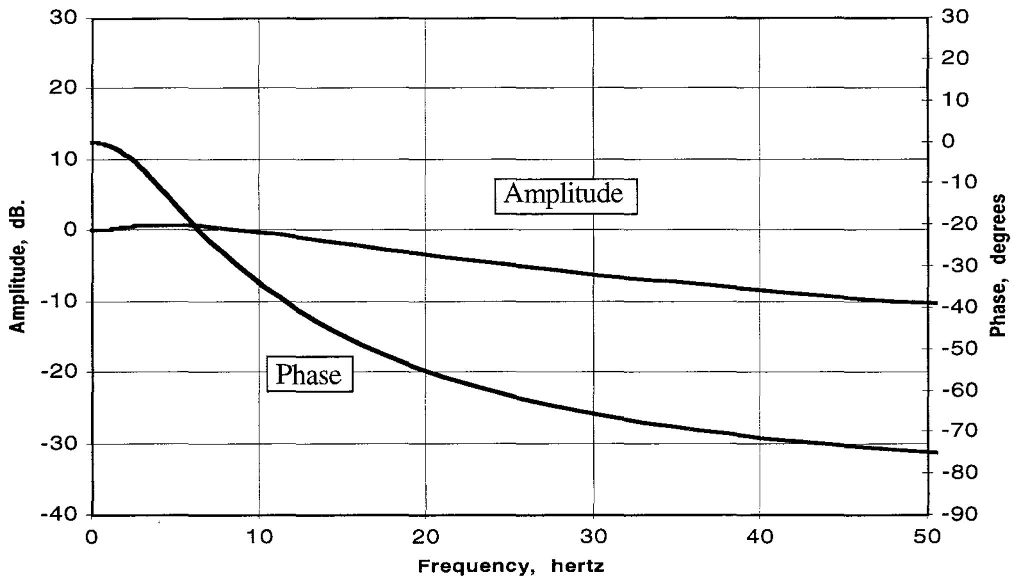


Figure 11-6 - *Closed-loop response of PLL with lead-lag filter.*

Figure 11-7 illustrates the open-loop response of the PLL utilizing the lead-lag filter. Notice that at the point of 0 dB amplitude the phase is -99 degrees, resulting in about 81 degrees phase margin. This loop is very stable and heavily damped. Changing the two pole frequencies in the lead-lag network alters the amplitude of the roll-off and the phase response of the network. The design of the lead-lag network sets the damping of the PLL.

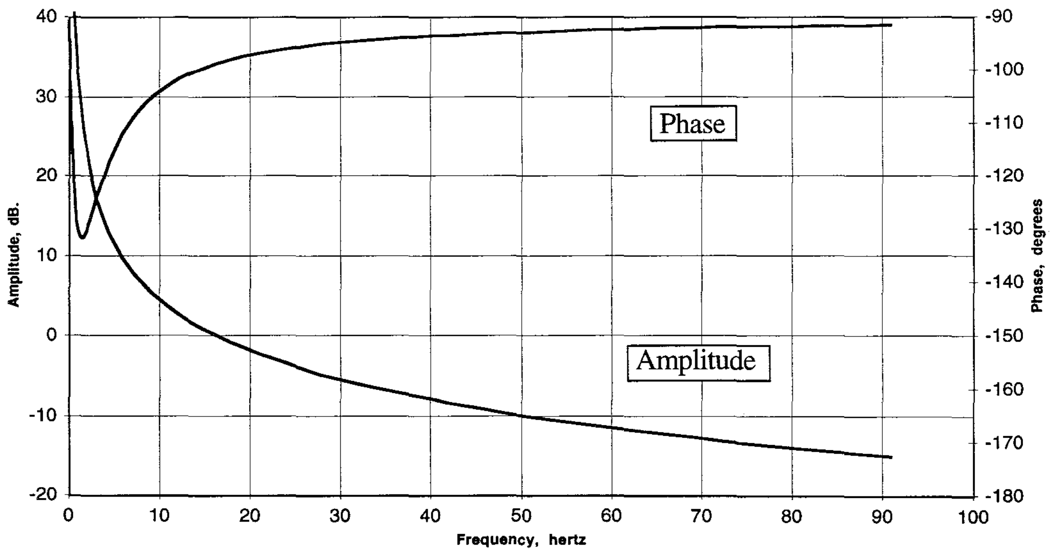


Figure 11-7 - *Open loop of response of PLL with lead-lag filter.*

Higher Order Loop Filter

A slightly better loop transfer function is for the loop to have additional roll-off outside the loop passband. This can be accomplished by altering the lead-lag network to add an additional pole (an additional lag) at a much higher frequency than the lead pole in the lead-lag network. In this way the open-loop attenuation is greater at higher frequencies, but the network phase response goes only to -90 degrees at high frequencies and is less at frequencies where there is loop gain. Figure 11-8 shows the schematic of this type of network. Figure 11-9 shows the closed-loop response, and Figure 11-10 shows the open-loop response.

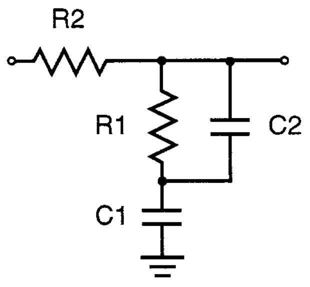


Figure 11-8 - *Lead-lag network with additional pole (lag) at high frequency, due to R2-C2.*

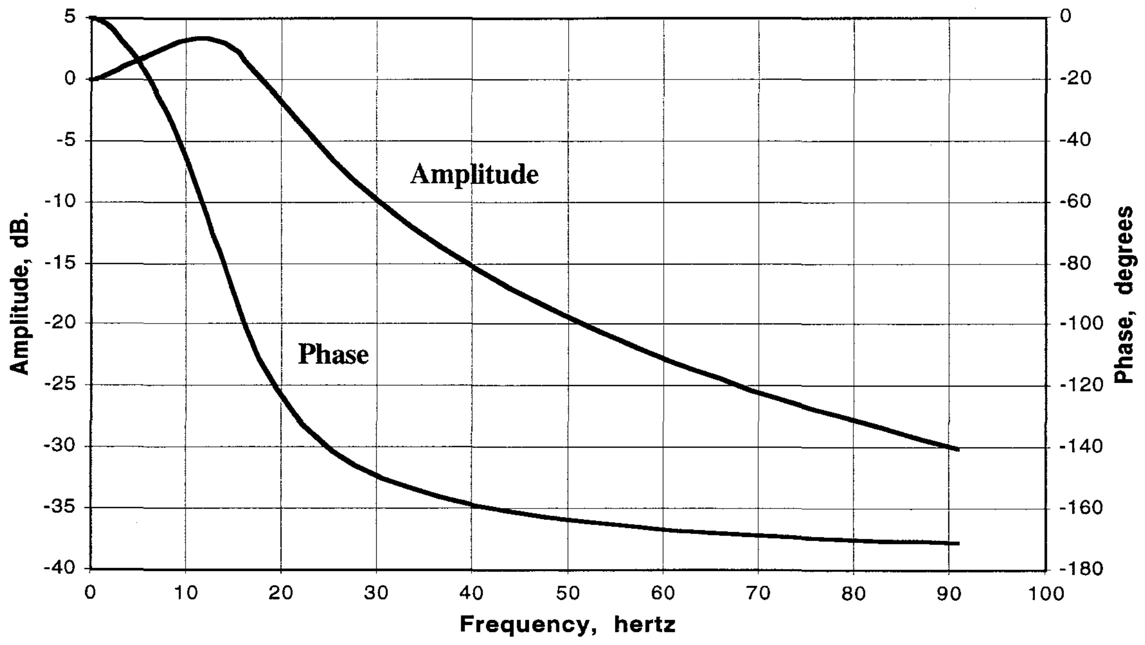


Figure 11-9 - *Closed-loop response of PLL with enhanced lead-lag loop filter.*

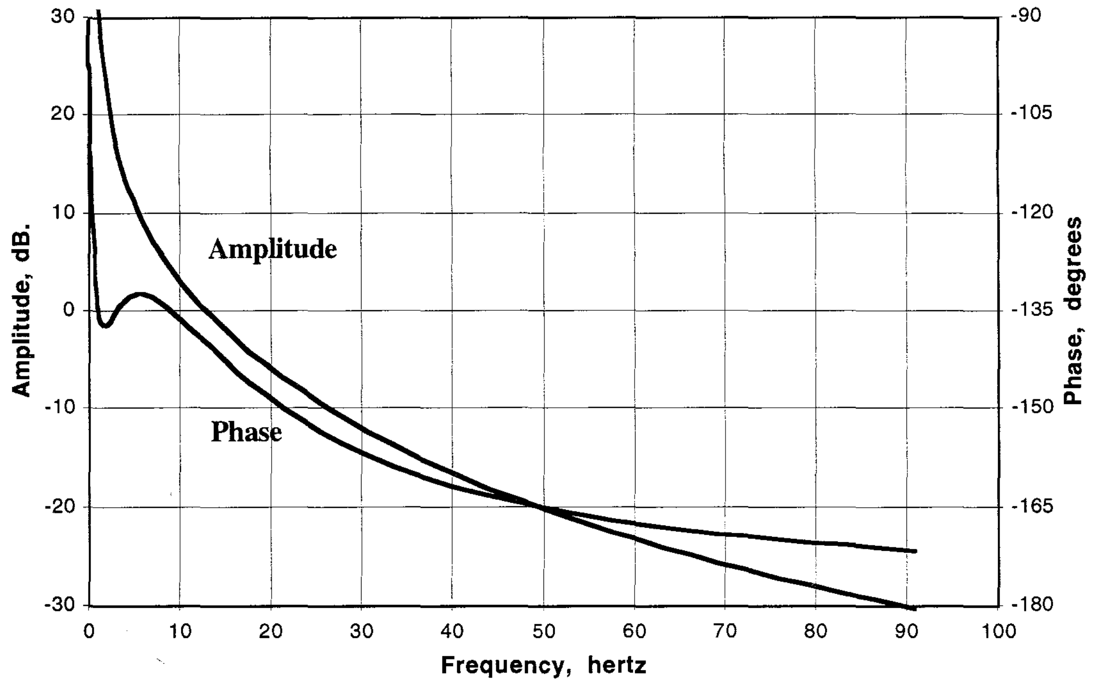


Figure 11-10 *- Open-loop response of PLL with enhanced lead-lag filter.*

The enhanced filter has allowed reducing the out-of band response, but at the expense of a little more peaking in the loop (Figure 11-9). In Figure 11-10 the amplitude response crosses 0 dB. at about 16 hertz, and the phase is -140 degrees, leaving about 40 degrees of phase margin. This is a little marginal in terms of phase margin. It can be improved by moving the R2C2 pole a little higher in frequency. Setting all three pole frequencies in the network allows tailoring the open-loop and closed-loop phase response.

The included Excel 5.0 spreadsheet (PLL.XLS) allows you to adjust the loop parameters and examine the closed- and open- loop PLL responses for all three filter types.

Very good out-of-band rejection to jitter can be accomplished with higher-order filters, provided that they do not contribute phase shift at low frequencies where there is loop gain. This usually places the filter at such a high frequency that it may be dubious as to whether the improvement is worthwhile.

Parasitic Poles

Many of the low pass filter poles that contribute to a little bit of phase shift are not easily found in PLL circuits. The op-amps that make up the loop filter, the modulation response of the VCO, and other undesired and unknown filter responses may creep into the PLL. These are usually known as *parasitic poles*. A second order design with a lead-lag filter has much more tolerance to parasitic poles than does the simple RC filter circuit since the loop phase of the lead-lag filter at high frequencies is 0 degrees, while it is 90 degrees for the simple RC low pass filter. This means that at high frequencies there is 90 degrees of margin with the lead-lag filter, but no margin at all with the simple lag filter. When the PLL is initially designed, the various filter functions should be carefully characterized for excess phase shifts. Even then many times the PLL is unstable. This usually precipitates an exercise known as *the hunt for the missing poles*.

PLL Loop Lock time

In half-duplex transmission systems, a low PLL lock time is important in achieving a fast turn-around link. The loop lock time is determined by the loop filter natural frequency, ω_n. For the passive lead-lag filter, G(0) is the DC gain of the filter (if any), and the natural frequency is given by:

$$\omega_n = \sqrt{\frac{K_v\ K_f\ G(0)}{(R_1 + R_2)\ C}} \tag{11-7}$$

The damping is given by:

$$\zeta = \frac{\omega_n}{2}\left(R_1\ C + \frac{1}{K_v\ K_f\ G(0)}\right) \tag{11-8}$$

and the lock time is approximately equal to:

$$Lock\ time \approx \frac{2\pi}{\omega_n} \tag{11-9}$$

As you be see, the loop locks more quickly when the loop gain at DC is very high (since the natural frequency is increased). However, as seen previously, when the loop gain is too high, the PLL may not be stable or can be seriously underdamped. For the lead-lag PLL in Figure 11-6, the natural frequency is 50 radians/second, the damping is 1.25, and the lock time is approximately 125 milliseconds.

PLL Noise Bandwidth

The noise bandwidth of the phase locked loop is proportional to the damping, ζ. In cases where a jitter free clock must be regenerated over a number of radio hops, large damping helps limit the rate of jitter growth. However, where just the performance of a single modem clock is the attribute to be optimized, then the lowest noise bandwidth may be an optimal solution. Gardner (1979) shows that the noise bandwidth of a second order loop is equal to:

$$B_n = \frac{\left(\zeta + \frac{1}{4\zeta}\right)}{2} \qquad (11\text{-}10)$$

The minimum noise bandwidth of equation (11-10) occurs at a damping of 0.5, rising rapidly with lower damping, and slowly with larger damping. For clock and carrier recovery with no concerns about downstream jitter, setting the damping to 0.5 is usually optimal. If the damping is not easily controlled then it is best to err on the excessive damping side.

Types of Phase Detectors, Lock-in Range

There are two basic types of phase detectors: 1) the *straight phase-detector*, usually implemented by a 4-quadrant multiplier or an exclusive-or gate, and 2) the *sequential phase-frequency detector*, which is strictly digital, and which contains several flip-flops to detect missing edges of one of the input frequencies. The phase-frequency detector is strongly preferred in frequency synthesizer applications because it extends the lock-range of the PLL. However, it should not be used in clock or carrier recovery when there a possibility of occasional missing edges due to noisy input data. A phase-frequency detector would consider this to be a significant off-frequency event and would provide a large correction voltage. The simple phase detector, on the other hand, provides only a small correction voltage under these conditions, which is the desired behavior. Unfortunately, the pull-in and lock range of a PLL with a simple phase detector is poorer than the phase-frequency type detector. In this case, the VCO frequency of the PLL must be located very close to the expected carrier or clock frequency if the loop is to achieve lock. The lock-in range of a second order loop is approximately:

$$\omega_L \approx \pm K_v\, K_f\, G(\infty) \qquad (11\text{-}11)$$

Where G(infinity) is the gain of the filter at a frequency much higher than the lead-pole.

For the PLL in Figure 11-6, the lock-in range is +/- 100 radians/second, or about +/- 16 hertz. It must be assured that the difference between the VCO frequency when the loop is unlocked and the frequency of the received reference signal never differ by more than 16 hertz in this example, including changes due to the variations of temperature, aging of components, etc. Some VCO designs are optimized for very wide pull-range, sometimes exceeding an octave. Although these designs are great for frequency synthesizers, they may be very poor for carrier and clock recovery circuits. A narrowband oscillator may be a better choice in order to assure lock-in and might be mandatory if a phase-only detector is used.

If the circuit preceding the PLL that extracts the carrier or clock reference has sufficient filtering to assure that no edge transition is lost even on fairly noisy input signals, then the phase-frequency detector can be used to great advantage with a much simpler VCO design as a result. Additionally, the dynamic range requirement of the loop is lessened significantly by using a phase-frequency detector. This is usually most easily accomplished with a carrier recovery circuit, since there are usually frequent edges in the recovered carrier reference. However, in the case of clock recovery, there may be no edges (for example a long string of ones or a long string of zeros). In this case, it may not be feasible to pre-filter the clock reference line so that the PLL always sees edges. Thus, in this instance, a phase-only detector may be the only choice, or an entirely different clock recovery scheme may have to be used.

Control Loop Dynamic Range

One difficulty with using a narrowband VCO (implied by the use of a phase-only detector) is that the large DC loop gain needed may cause the VCO control voltage to saturate during acquisition, resulting in poor or non-existent phase locking properties. It is helpful to have a high VCO sensitivity so that the control voltage into the VCO does not exceed the output swing available from the driving op-amp during the acquisition.

Other good material on digital phase-locked loops can be found in Rhode (1983) and Best (1993).

Chapter 11 - Reference

Best, Ronald E. "Phase-Locked Loops Theory, Design, and Applications," 2nd ed. McGraw-Hill Inc., 1993.

Gardner, Floyd M. "Phaselock Techniques." John Wiley & Sons, 1979.

Rhode, Ulrich L. "Digital PLL Frequency Synthesizers Theory and Design." Prentice-Hall Inc., 1983.

Chapter

12

Frame Alignment and Data Carrier Detect

Two subjects that at first seem to have no relation to each other, in fact, are very closely related: 1) the need for frame alignment and, 2) the need for data carrier detect for half-duplex radio channels. *Frame alignment* is the process of finding alignment in an incoming data stream. One example is finding the boundary of forward error correcting (FEC) blocks. Another case is the process for delineating bytes (although sometimes that is done by other devices such as an HDLC chip). It may also be desirable for a modem to find character alignment (such as in AMTOR reception).

Data Carrier Detect (DCD) on the other hand is detecting the presence of a received signal, either for the purposes of preventing transmission on a busy channel or activating some particular received equipment state (for example lighting an LED, or for verifying that a received AX.25 frame should be considered valid). DCD can be detected by several methods, such as the presence of received energy on the receive frequency, or the presence of a signal meeting certain defining criteria, depending on the application. The more general the criteria, the more likely that DCD will respond to any received signal (whether matching the format we desire to receive or not). Also, it will be more likely to *false-declare* (declare carrier when none is present) and to *false-drop* (declare no-carrier when a valid one is actually present). The more restrictive the criteria for DCD, the closer that the received signal must be like, or mimic, the desired receive signal. However, it will more than likely not false-declare nor false-drop.

For some selective methods of acquiring a DCD signal or of finding frame alignment, we may wish to look and see if the actual values of the data bits of the received signal match the criteria. In this case, we would like to reject received data bits that match a different, non-compatible pattern. Here the process for the two purposes is in fact exactly the same: that of finding alignment. Then, if the alignment is good enough statistically, we will declare either valid frame alignment or valid DCD. Thus, we reach the conclusion that the two subjects (frame alignment and DCD) are in fact closely related. First, we will look at some of the simpler techniques for DCD acquisition, and discuss the limitations. Next, we will look into more highly-defined methods of DCD and frame alignment acquisition.

General Data Carrier Detect (DCD) Acquisition

The most general technique to acquiring a DCD signal is, for example, to sample the energy on the received channel by looking for the presence or absence of energy. One method to accomplish this is to AM detect the channel, and filter the received signal. This is an extremely simple method, but suffers from the fact that any energy, including noise, can trigger declaration of DCD. If we set the threshold of detection high enough, we will reduce the probability of false detection. However, at such a high threshold of detection, we will enhance the probability of falsely dropping the detection when a valid receive signal is corrupted temporarily (such as a fade) or we may not even declare valid signal at all on a weak, but usable signal.

To more accurately detect DCD, we must use our knowledge of the characteristics of the received signal to help us develop a more discriminating detector. For example, in the case of 1200-baud AFSK, we know that our receiver should detect 1200-Hz and 2200-Hz tones. Therefore, we could build a circuit that provided two filters: one at 1200-Hz, and one at 2200-Hz. Then, we could compare the amount of energy in the sum of the two filters to the energy present on the channel, but not at 1200- or 2200- Hz., the so-called *out-of-band* energy. Figure 12-1 is a circuit diagram of this type of detector. Note that this circuit is a very simple addition to a traditional AFSK demodulator. It only adds band reject filter(s) to the already needed bandpass filter(s). It also has use for FSK signals detected on a single-sideband receiver, which is an AFSK signal at baseband. A limitation of this type of circuit is that an interfering signal that is out-of-band signal can cause false-drop of the detect signal if the interferer is much stronger than the desired signal. Another limitation of this type of detector is when the energy of the desired signal almost completely occupies the bandwidth of interest. As a result, there may not be much of a non-desired signal to discriminate against.

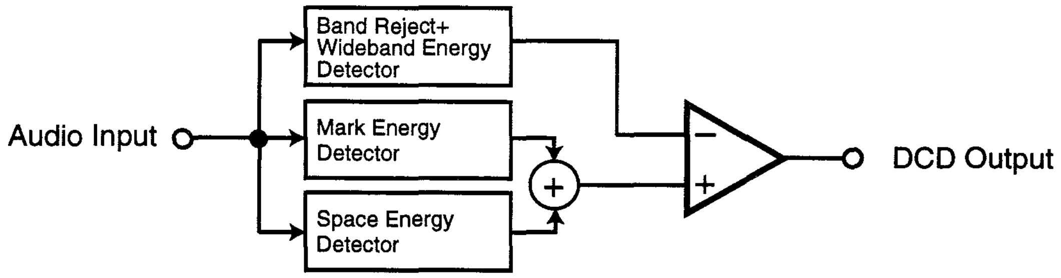


***Figure 12-1** - A simple data-carrier-detect circuit. This type of detector compares in-band to out-of-band energy to determine if a valid carrier is present.*

Another commonly used detection circuit is to declare DCD when a phase-locked loop (PLL) is able to acquire and lock to the received AFSK signal. This circuit can suffer from the false-detect problem, due to random noise occasionally locking the PLL. Notably, a large amount of filtering that matches the desired receive format prior to the PLL dramatically enhances the probability of occasional random lock of the PLL to received noise.

In either of the above cases, it is desirable to add hysteresis and time-constant filtering to the DCD circuit. Hysteresis is useful to prevent *chop* on the DCD circuit from a received signal that fades just slightly above and below the threshold level. Likewise, a time-constant filter reduces the probability of false-declare and false-lock, but at the expense of slowing down the detection time and the loss-of-detection time. Figure 12-2 is an example of hysteresis and time-constant filtering of a raw DCD signal. R1-C1 sets the discharging time, and R2-C1 sets the charging time. The hysteresis is provided by positive feedback to the (+) terminal. Hysteresis prevents *chop* of the output signal when the input would otherwise be near the threshold voltage.

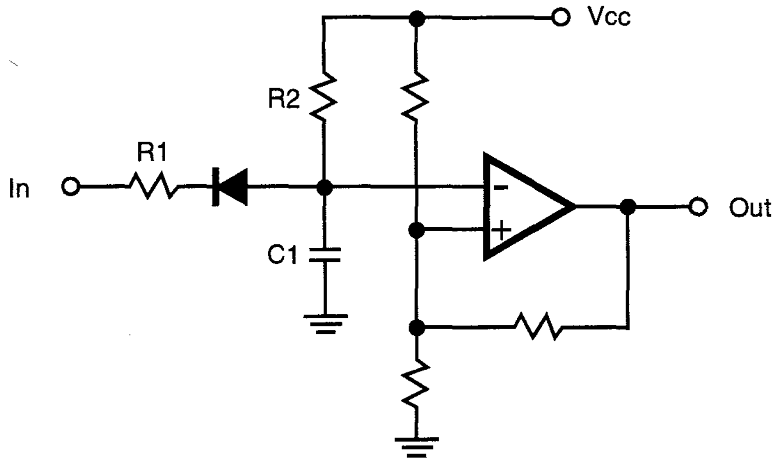


Figure 12-2 *- Circuit to add time constant filtering and hysteresis to raw data carrier detect signal.*

A third technique for DCD detection on a digital signal is to examine the characteristics of the post-demodulation (baseband) received data signal. For example, we could detect the presence of a valid received *eye* pattern. The validating criteria for the eye pattern form the detection algorithms. Two obvious criteria exist: 1) The eye is "open" in the middle. That is, there are few times that the eye pattern assumes an invalid level during the middle of the eye. It is close to a valid signal level. 2) The eye occurs at a frequency that is synchronous with the expected received data rate. Thus, the zero-crossings of the received signal occur at about the right time, on average, when they do occur. A difficulty in determining the right zero-crossing time is that an extended-period of all ones or all zeros in the received signal prevents us from gaining any information about the received signal. So we must use longer time-constants in determining the validity of the timing information.

The technique of determining if the eye is open in the middle of a data bit was demonstrated by Paul Newland, AD7I, in the DCD and clock recovery circuit for the TAPR TNC-2. This is an algorithmic detector based on a state-machine implemented in an EPROM. The clock recovery portion is discussed in the section on clock (symbol-timing) recovery. Assuming that clock has been acquired, then it is a simple matter to look at the received data signal and ask if it is the same value (i.e. one or zero) on both sides of the sampling instant. Of course, noise or interference could possibly corrupt the data bit. Therefore, it is necessary to filter the received indication lest a single noise spike cause false loss of DCD. The clock recovery circuit must track the drifting phase of the received signal so that there is small error in the determination of the center of the data bit. However, this criteria is also required for sampling the received data bit for its actual value, or else the error rate of the modem will be so high that it will be unusable anyway. The circuits discussed previously for hysteresis and time-constant filtering can be used to condition the raw DCD signal prior to its use by other functions.

A more rigorous set of criteria can be developed for DCD detection if a high-accuracy clock recovery function is available. In this technique, we measure the voltage of the received eye pattern at the clock sampling time, and use a window comparator to assess whether the signal is within several dB. of the desired value. In essence, this is an analog implementation of the previous circuit. One additional advantage of this type of circuit is that a pseudo-error alarm indication can be provided. If the data (eye opening) is consistently smaller than the error bounds, the modem may continue to operate without errors, but the margin is small. For example, if the pseudo-alarm thresholds were set half-way between one and the mid-level and the other threshold was set half-way between zero and the mid level, then a consistent violation of the pseudo-thresholds would indicate that less than 6 dB of margin is present. See figure 12-3 for a schematic of this type of detector.

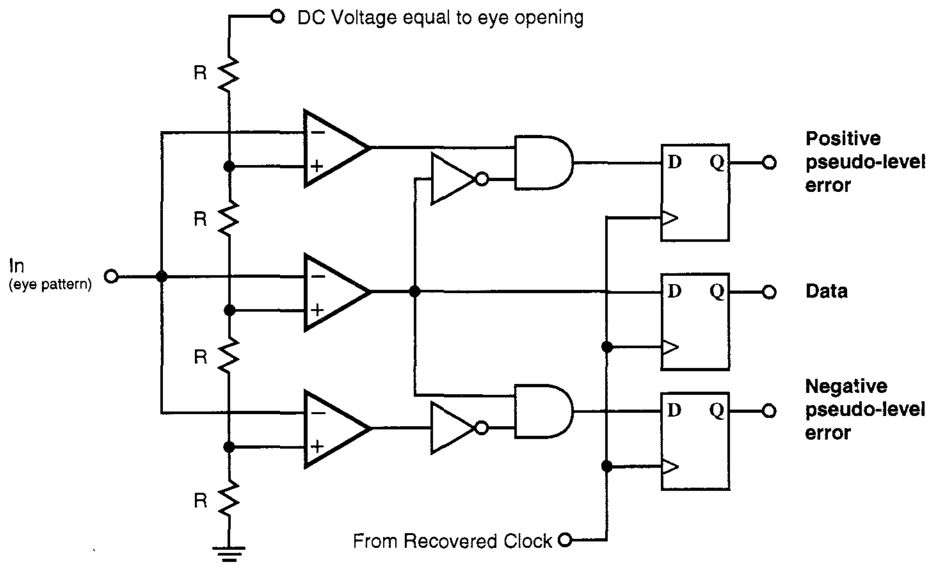

DC Voltage equal to eye opening
R
R
R
R
In
(eye pattern)
D Q
Positive pseudo-level error
D Q
Data
D Q
Negative pseudo-level error
From Recovered Clock

Figure 12-3 - *Pseudo-level error detector. A pseudo-error occurs when the received eye signal is less than half-way open.*

The high-accuracy clock recovery detector circuit in itself provides an indicator of valid DCD. If the symbol-timing detector is based on a PLL, then phase-lock of that detector is a good indication of a valid received signal (or of one matching the data rate, anyway). Thus, with appropriate filtering either a PLL lock-detect signal or monitoring of the PLL error voltage can provide information needed to develop the DCD signal. See figure 12-4 for circuits of PLL lock-detect, and of error-voltage monitoring lock detection. One difficulty with this circuit is that if the voltage controlled oscillator has a control range that drifts significantly with temperature, then the lock detection may not be robust over temperature. Additionally, if the data pattern contains significant missing clock transitions, then there may be some perturbations of the loop control voltage that need to be filtered.

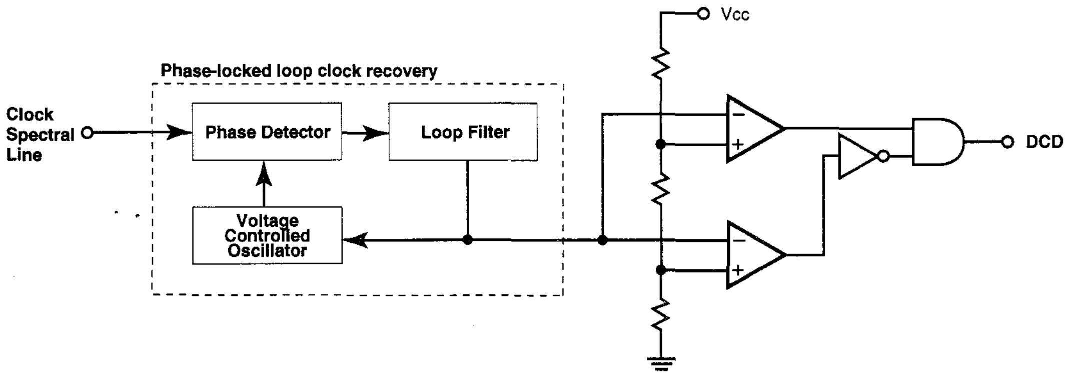

Vcc
Phase-locked loop clock recovery
Clock Spectral Line
Phase Detector
Loop Filter
Voltage Controlled Oscillator
DCD

Figure 12-4 - *DCD detection based on phase-lock of symbol-timing recovery information.*

DCD and Frame Alignment based on Message Content

While simple DCD circuits based on the signal indicators discussed above are common, they are not always particularly robust in the presence of interference or noise. If we have a modem that is decoding the data with a reasonable, but not perfect, bit error rate (BER), and if we know something about the content of the message, then we can develop a yet-more sophisticated circuit or function for either the declaration of DCD or for the determination of frame alignment. From this point on, we will discuss the technique of frame alignment, since DCD derivation is an obvious derivative of this process. A tutorial overview of frame alignment processes can be found in Choi (1990).

In determining frame alignment, there are two general techniques: 1) Include redundant information in the message, such as framing bits, that can be utilized to acquire frame. These bits may be distributed throughout the message, or they may be lumped all together in groups which are then periodically lumped within the message. 2) Utilize some statistical property of the information bits that must be present in order to declare frame (such as an FEC decoder declaring no errors for several consecutive frames).

In either method, the process that needs to occur is for a search method to start examining the bits one at a time to determine a particular pattern. Now, random data can sometimes mimic the framing pattern, or it can mimic the statistical criteria. Therefore, the circuit cannot declare valid frame. We must set a criteria for consecutive-valid-signals that have to be received before the circuit will transition to a state in which frame alignment (or DCD) is declared. Once in-frame, the circuit should not drop alignment if a single bit error, or the error of a few bits occurs in the framing pattern, since the modem is expected to operate properly even at some finite error rate. There must instead be a criteria for what percentage of frame alignment words (or statistical "events") is invalid before the declaration of loss-of-frame.

We will use as an example a simple framing system that consists of additional overhead framing bits added to the data. Let's assume that we add an alternating one-zero pattern to the data, and that each framing bit is added once every 20 bit times. Thus, if the input to our circuit consists of data bits, d, then the following will the output:

```
...dd1dddddddddddddddddddd0dddddddddddddddddddd1ddd....
```

We have 20 data bits, then a framing bit, then twenty more data bits, then the framing bit of the opposite polarity, etc. The receiver must obviously discard the framing bits before sending the data to the consuming device. Our circuit for finding frame operates as follows:

1) select any bit position at random, and hypothesize that it is a framing bit. Store the value of that bit.
2) wait 20 bit times, then test the 21st bit and see if it is the opposite of the bit stored in 1).
3) If the bit is correct then set an output state of *possible frame acquisition.* And search for the next framing bit (go to step 2).
4) If the framing bit being tested fails the criteria, then delay the finder circuit by one data bit time, and go to step 2.
5) If *n* correct consecutive framing tests are successful, then we transition to the state *frame found.*
6) We continue to test each received framing bit to see that it is the opposite of the previous framing bit.
7) Since we are now in the *frame found* state, a single mismatch will not cause us to go out of frame. This is because a single bit error, if it occurred during a frame bit time, would cause loss of frame, and either a reframe, or loss of DCD.
8) If, while in the *frame found* state we start finding a large number of mismatches, then we will transition back to the *out of frame* state, and restart the process by going to step 4.

Median Time to False Frame Declaration

Now we can establish the criteria for declaring in-frame and out of frame. If we require m consecutive successful frame tests then we can compute the probability of falsely declaring frame based on random data as:

$$P\,(\mathit{false\ declare}\,) = p = \frac{1}{2^m} \tag{12-1}$$

For example, if $m = 10$ then the probability of falsely declaring frame acquisition is 1/1024. We require (20+1)*10 = 210 bit times as the absolute minimum to compute a frame acquisition. In any observation interval of 10 framing bits, the probability of the framing condition failing is 1-p, or 1023/1024, which we will call q. If we add another test at the end, or the 11th framing bit, then the probability of bits 1 through 10 failing is q, and the probability of bits 2 through 11 failing is also q. So, the probability of not finding frame in 11 bits = q * q. Ignoring the additional time due to slipping bits when we don't find frame, if we define n as the number of framing bits that we observe, then the probability of finding frame in those bits is:

$$P\,(\mathit{frame}) = 1 - \left(q^{(n-m+1)} \right) \tag{12-2}$$

This resembles a cumulative density function (CDF). We can establish the observation interval n by plotting the CDF, or by making a direct calculation. Assuming we want to find the median time, where P(frame) = 0.5, then we can derive n from equation (12-2) as

$$n = \frac{\log\,(\,1 - 0.5\,)}{\log\,(\,q\,)} + m - 1 \tag{12-3}$$

The number of data bits is 20 * n, since we have 20 data bits between each framing bit, and the total number of bits is 21 * n. For q = 1023/1024, then n = 718. Thus, we expect to see a mean time of about (718 * 21) = 15,078 bits between false detects. Figure 12-5 is a graph of the probability of false-detect vs. the number of frame bits observed for m = 10 and m = 16.

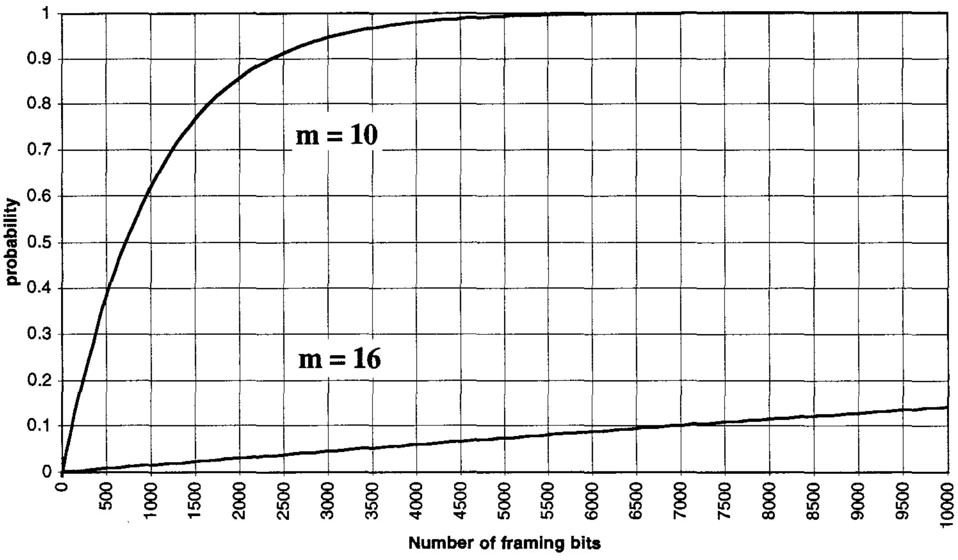
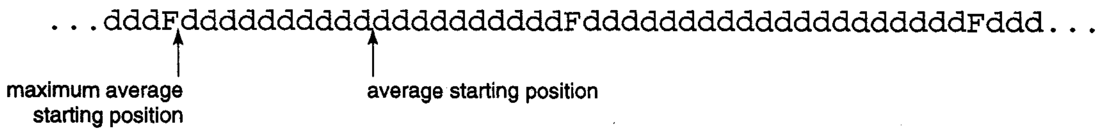


***Figure 12-5** - probability of false frame acquisition vs. number of framing bits observed, assuming random data, for m=10 and m=16*

It can be seen that $m = 16$ consecutive frame bit matches provides much greater mean time to false frame acquisition. For $m = 16$, the equation (12-2) yields about 954,000 bits for the mean time. At 9600 baud, 15,078 bits is 4.73 seconds, while 954,000 bits is 99.4 seconds.

Maximum Average Reframe Time

While increasing m still further results in better prevention of false frame declaration (or of false DCD declaration), it also increases the time that it takes to acquire frame (or to acquire DCD) in the first place. Two items are of concern: 1) The average time to acquire frame (or DCD), and 2) The maximum average time to re-acquire frame if we lose it. The average time assumes that we randomly start half-way through our sequence of 20 bits between data bits, while the maximum average time assumes that we just slipped out of acquisition and are starting one bit later than the correct time. As a result, we have to search through all 20 data bit positions before finally arriving at the framing bit position. F is a framing bit position, and d is a data bit position.



...dddFddddddddddddddddddddFddddddddddddddddddddFddd...
maximum average starting position
average starting position

To calculate the number of required searches, we assume random data bits, *d*. Each time we make another calculation, the probability is 1/2 that the bit we are testing is a match with the framing pattern, since random data may match or not match with equal probability. The average number of comparisons that we make while dwelling at a position is calculated by noting that 1/2 of the time we will have a correct mimic of the frame bit. Then, for those 1/2 that succeed the next test will succeed 1/2 of the time, etc. Summing this series provides the average number of times we dwell before a failure occurs. Fortunately the series converges:

$$c = \sum_{k=0}^{\infty} \frac{1}{2^k} = 2 \qquad (12\text{-}4)$$

Consequently, we spend two test intervals, on average, at each dwell position. Then the average reframe time is, in our example, 10 test positions times 2 test intervals. A test interval is 21 bit times long, so the average frame acquisition time is 10 * 2 * 21 = 420 bits. The maximum average time to reacquire frame (or DCD) is 20 * 2 * 21 = 840 bits. For the case of a 9600 baud modem, this represents 44 msec. for average acquisition, and 88 msec. for maximum average reframe time.

If we increase *m* from 10 to 16 bits required for a correct frame declaration, then the average frame time increases to 16 * 2 * 21 = 672 bits, or 70 msec at 9600 baud. Meanwhile, the maximum average time increases to 32 * 2 * 21 = 1344 bits, or 140 msec at 9600 baud. We can conclude that frame or reframe time is proportional to *m*.

Going Out of Frame

Once in frame (or once DCD has been declared), we transition to a *locked* state wherein we do not slip our framing circuit if an error is found in a framing bit. This is because a single bit error could cause us to go out of frame. Instead, we will require that some number of consecutive frame bit errors, *n*, occur before we declare loss of frame and start the search process over again. Then the question to be asked is how probable is the loss of frame vs. bit error rate for different values of *m* ? Since an error in any bit position is exactly the same as the bit error rate, *b*, then we can state the probability of *m* consecutive errors as:

$$P\,(false\ drop) = b^m \qquad (12\text{-}5)$$

Obviously, $q = 1 - p$ times we will not have m consecutive errors. We can now formulate the problem exactly as we did when asking the question about going into frame. For any group of m bits, the probability of declaring loss of frame is given by (12-4). So, as we shift in each new bit, the most recent m bits have the probability p of meeting the loss of frame criteria. The probability of not losing frame each measurement time is q, and we can directly use equation (12-2) to calculate the probability vs. m, the number of bits required for an error, and n, the number of framing bits we observe. Also, we can use equation (12-3) to calculate the median time. Figure 12-6 is a graph of the probability of false frame loss vs. the bit error rate for different values of m.

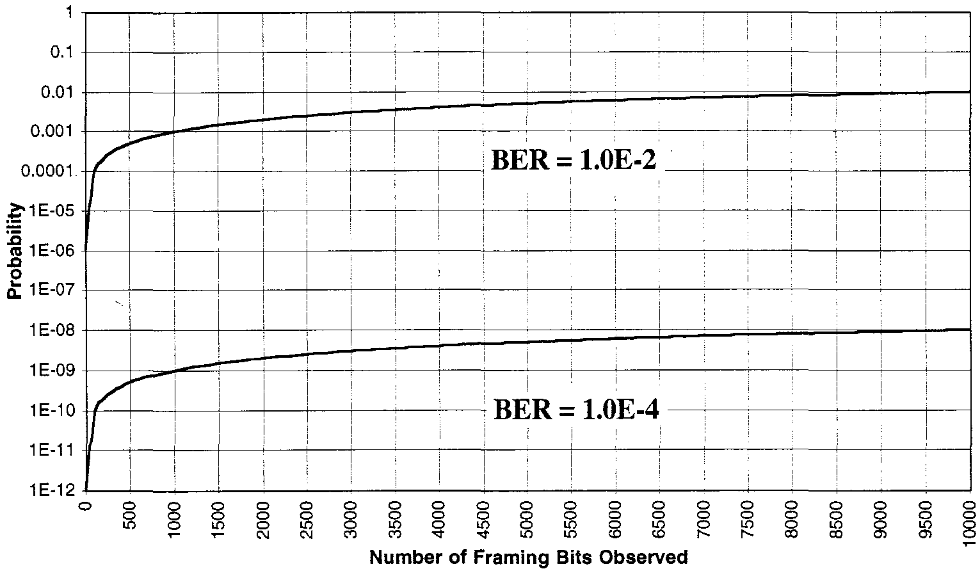


Figure 12-6 *- probability of false loss of frame (or DCD) vs. number of framing bits observed, for m=3 consecutive frame-error bits required to go out of frame, at* 1.0×10^{-2}*, and* 1.0×10^{-4} *bit error rate.*

We can see that for any reasonable bit error rate, such as 1.0×10^{-4}, the probability of losing frame is pretty small. Should we desire more robust false frame loss statistics, we can increase m to 4 or 5, resulting in almost negligible false loss of frame probability.

Other Message Content

Thus, far we have examined the statistics behind framing (or DCD detection) based on the insertion of framing bits into a message. It is also possible to derive suitable criteria without the addition of framing bits. For example, in a valid AMTOR character, there are 4 bits on one polarity and three of the other. When we have validly framed on the correct 7-bit character time, each word will have this ratio (neglecting any bit errors, of course). In the case of linear block coded forward error correction, we will have achieved correct block synchronization when the syndrome bits are all zero. This indicates that no bit errors are present in the received block. Given a measurable criteria, we can implement our previous framing algorithm using the pass/fail criteria as being all-zero syndrome bits. If we are able to calculate the syndrome vector at each and every bit time, then we can find frame based on an update rate of the block size of the FEC word. The algorithm previously described can be modified so that once a valid syndrome vector is found, the time to wait for the next syndrome test is the block size of the code. It is possible to misinterpret 3 bits as a valid syndrome when in fact we are misaligned with the FEC block. However, with random data we will quickly detect a non-zero syndrome at a later test of the framing status.

The combination of framing on FEC code words, and the use of that framing for DCD detect, is a powerful and robust technique that should be readily exploited in the design of modems for half-duplex channels.

Framing on Short Transmissions

So far the discussion on framing has assumed that sufficient number of framing-detectable events have occurred in order to acquire in-frame, which may require the transmission of a significant number of bits. It is possible to speed up the acquisition of frame (or DCD) by a parallel search technique. In this case, rather than testing a single bit and waiting for the next framing occurrence, we start many frame detectors all at the same time. Each of them is one-bit removed in time from the next. In this manner, we search the frame in parallel instead of in series. In the previous example, we should find frame in 21*10, or 210 bit times (for $m = 10$ consecutive valid frame words), which at 9600 baud is 21.8 msec., about twice as fast as the serial search technique, however with a considerable increase in complexity.

Another method to improve the search time is to provide a synchronization block word all at one time in the transmission, rather than distributing the bits throughout the transmission. If ten consecutive bits were transmitted as a synchronization word, then we could build a simple frame finder that looks for 10 consecutive bits that match the Frame Alignment Word (FAW). The word would have to be transmitted several times because the possibility of a bit error might invalidate the reception of the FAW. Additionally, we should assure that the FAW has some desirable properties: 1) it should have negligible auto-correlation, i.e.a time-shifted version of the code word should not match the code word in very many bit positions. 2) the FAW, when concatenated with itself (sent several times in a row) should not have any shifted version spanning two FAW's that matches the FAW in very many bit positions. Both of these criteria minimize the possibility that in the presence of a bit error the frame detector circuit will accidentally align on the word but shifted off of the correct position by several bit times. Patterns known to have the good auto-correlation properties are those generated by the maximal-length pseudo-random bit sequences (PRBS) discussed in the section on coding.

A question naturally arises: Once we have synchronized, how can the loss of synchronization be detected? Four possibilities arise: 1) we can have another FAW pattern at the end of the message that the receiver uses to declare out-of-frame alignment. However, if the signal fades or degrades, we may never see this sequence, so we have to back up the detector with 2) a time-out timer set to slightly longer than the length of a complete transmission. Finally, 3) we can use the FAW to acquire quick frame alignment, and then use distributed alignment to "hold" the alignment. The distributed alignment can be based on FEC code words, or imbedded framing bits. Technique 3 would require the FAW frame to set the distributed framer into the frame state, with all loss-of-frame bit counters set to the state of no-error events detected (all counters set to maximum number of valid frame bits have been received). 4) We can periodically insert FAW words in the data sequence and declare loss of frame if several consecutive FAW words are missed. Because the possibility of a FAW word error is greater than the possibility of a single framing bit error (by approximately the number of bits in the FAW word, at reasonable error rates), we have to be more cautious in declaring out-of-frame condition.

Chapter 12 - Reference

Choi, DooWhan. "Frame Alignment in a Digital Carrier System - A Tutorial," IEEE Communications Magazine, Feb 1990, pp. 47-54.

Chapter

13 Propagation Channel Models

All models so far have assumed that the channel (the radio link) has a simple transfer response — that is, no frequency-dependent attenuation, no multipath reception, nor linear phase change versus frequency. Only the Additive White Gaussian Noise (AWGN) channel has been considered, and all bit error rate performance curves have been based on the AWGN model, due to the ease with which computations are made. In reality, neither HF nor VHF/UHF radio channels exhibit these ideal conditions. The simpler case is VHF and UHF propagation models, where relatively more stationary multipath is the predominant distortion.

VHF and UHF Channel Models

In modeling a VHF or a UHF digital channel, a simplifying assumption will be made that over-the-horizon propagation is not considered. Although there are band openings in the VHF and UHF frequency ranges which are very important, they will not be considered here. Additionally, both VHF and UHF amateur bands are very near to commercial bands. Thus, interference due to overload and 3rd order intermodulation are also significant, but will also not be considered.

The most significant deleterious propagation effect is due to multipath propagation. Multipath, as the shortened form is usually called, is a condition where two (or more) different propagation paths exist between the transmitter and receiver. Usually the first path is consistent, and is the path that is engineered between the transmitter and receiver. A second transitory path sometimes comes into existence when a reflective surface supports an alternate, undesired path between the transmitter and receiver. Either path alone is not a problem, but the combination of the two paths causes frequency-dependent attenuation, frequency-dependent phase non-linearity, and delayed response effects. Whether the frequency-dependence of the distortion significantly affects the channel is dependent on the geometry of the multipath propagation paths.

Figure 13-1 shows a schematic of a multipath propagation condition. The path, *p*, is the primary path, and the path, *s*, is the unintended secondary path. Note that the radio path *s* is physically longer than the path *p*, and thus one individual data bit from the transmitter will arrive at the receiver twice, each one at a different time. Additionally, the phase of the received RF carrier will be different between the two paths. Accordingly, the two carriers will add, either constructively or destructively, and the phase of the received carrier may change as the relationship of the two paths changes. It would be unusual for the secondary path to be tremendously longer than the primary path. If it is, then there is usually a lot of attenuation of the secondary path, and its impact becomes less significant.

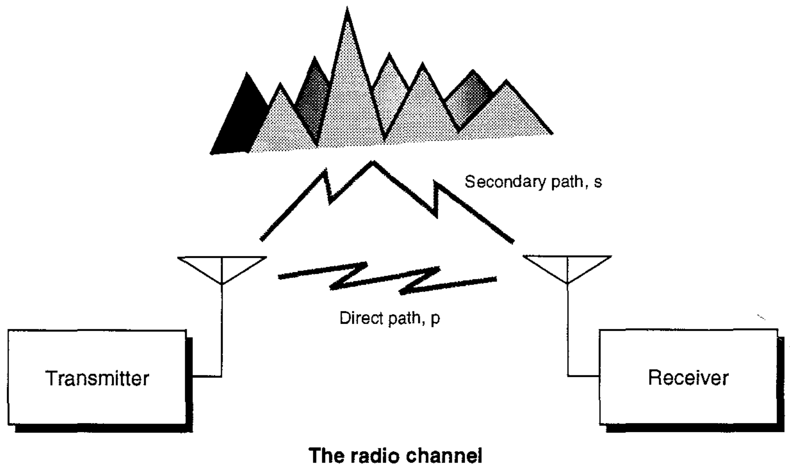


***Figure 13-1** - Multipath is caused by the combination of a primary and one (or more) secondary path(s).*

Comparing the difference in the distance between the two paths allows an estimate of the phase difference between the two paths. To determine frequency-dependent effects the path geometry must be related to the frequency of operation. Let us assume that the path, *s*, is longer than the path, *p*, by 2 kilometers. Also, let us assume that the received signal strength from the two paths, *p* and *s*, are at this moment equal. If the frequency of operation is 445 MHz, then the wavelength of operation is approximately 67 cm. As the frequency changes, due to the difference in frequency between the upper and lower edges of the active channel, the wavelength of the two band edges is also slightly different. At some frequency, the difference in the two propagation paths will be exactly 180-degrees, while at other frequencies it will not be exactly 180-degrees. Thus, it is possible to determine the phase difference vs. frequency over the channel of interest given the two different paths. When the frequency 1 associated with *lambda* (λ) 1 causes destructive interference, then destructive interference will also occur at frequency 2 if it is different in path length by exactly one cycle from the path length at frequency 1. This can be expressed as:

$$\frac{\Delta d}{\lambda_1} = 1 + \frac{\Delta d}{\lambda_2} \qquad (13\text{-}1)$$

Where *delta-d* is the difference in the path length between *p*, and *s*, and where lambda is the wavelength of a particular frequency. When the number of cycles of path length is different by exactly one, then the frequency associated with the wavelength is the frequency spacing of the path nulls. Converting the wavelengths to frequencies f_1, and f_2, the above equation is rewritten as:

$$f_2 = f_1 - \frac{c}{\Delta d} \qquad (13\text{-}2)$$

Where *c* is the speed of light, or about 299,793,000 meters per second. However, the important quantity is the *delta-frequency* between adjacent multipath nulls, which by inspection of equation (13-2) does not actually depend on the absolute frequency. Re-writing equation (13-2) in terms of *delta-f*, then:

$$\Delta f = \frac{c}{\Delta d} \qquad (13\text{-}3)$$

In the above example, where *delta-d* is 2000 meters, the spacing of multipath nulls will be 149.9 kHz. apart. Figure 13-2 shows the frequency spacing of multipath nulls versus the difference in path length, *delta-d*.

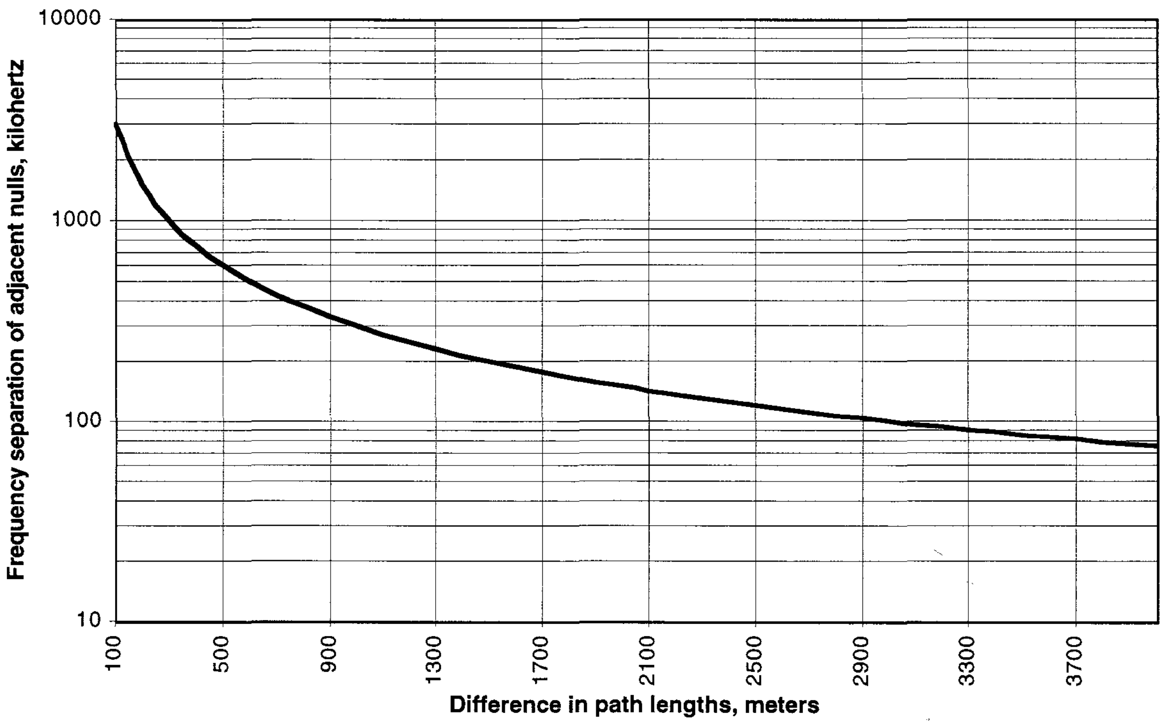


Figure 13-2 *- spacing of nulls in the channel response vs. the difference in the two path lengths.*

As can be seen, a path length difference of 300 meters results in the spectral nulls being about 1000 kilohertz (1 megahertz) apart. Thus, a 30 kilohertz wide channel would have essentially flat fading due to multipath under these conditions. In the case of 2000 meter (2 km) path length difference, the spectral nulls are 149 kHz apart, and there will be some amplitude and phase slope across a 30 kilohertz wide channel.

Minimum-Phase and Non-Minimum-Phase Fades

The two-component multipath amplitude and phase model is computed based on summing the two signals. The main, and the delay propagation components case be written as:

$$Main = e^{-j\omega t} \tag{13-4}$$

$$Delayed = Ae^{-j\omega(t+\tau)} \tag{13-5}$$

Where τ is the time delay of the delayed signal, ω is the frequency in radians per second, and A is the amplitude of the delayed component compared to the main component. It is not particularly important to know the exact time nature of the signal, so $t = 0$ is as convenient as any to calculate. Then the main signal is equal to $1+j0$, and the resultant received signal is the sum of the main and the delayed signals:

$$Received = 1 + Ae^{-j\omega\tau} \tag{13-6}$$

Figure 13-3 shows the received magnitude and phase vs. frequency of a signal and one multipath component with a weaker amplitude, equal to 91% of the main signal magnitude. The multipath signal has been delayed 1 microsecond from the main signal. Notice that the phase changes from about +60 degrees to about -60 degrees around the point of signal cancellation. This means that there is severe group delay distortion near the point of cancellation. There is 6 dB of gain, with zero degrees phase shift when the two signals exactly add (for example, at 445.0 MHz in the diagram below). In microwave radio texts, the case where the multipath component is *weaker* than the main component is called a 'minimum-phase' fade, while the case where the multipath component is *stronger* than the main component inverts the group delay response, and is called a 'non-minimum-phase' fade. Figure 13-4 illustrates a non-minimum-phase fade.

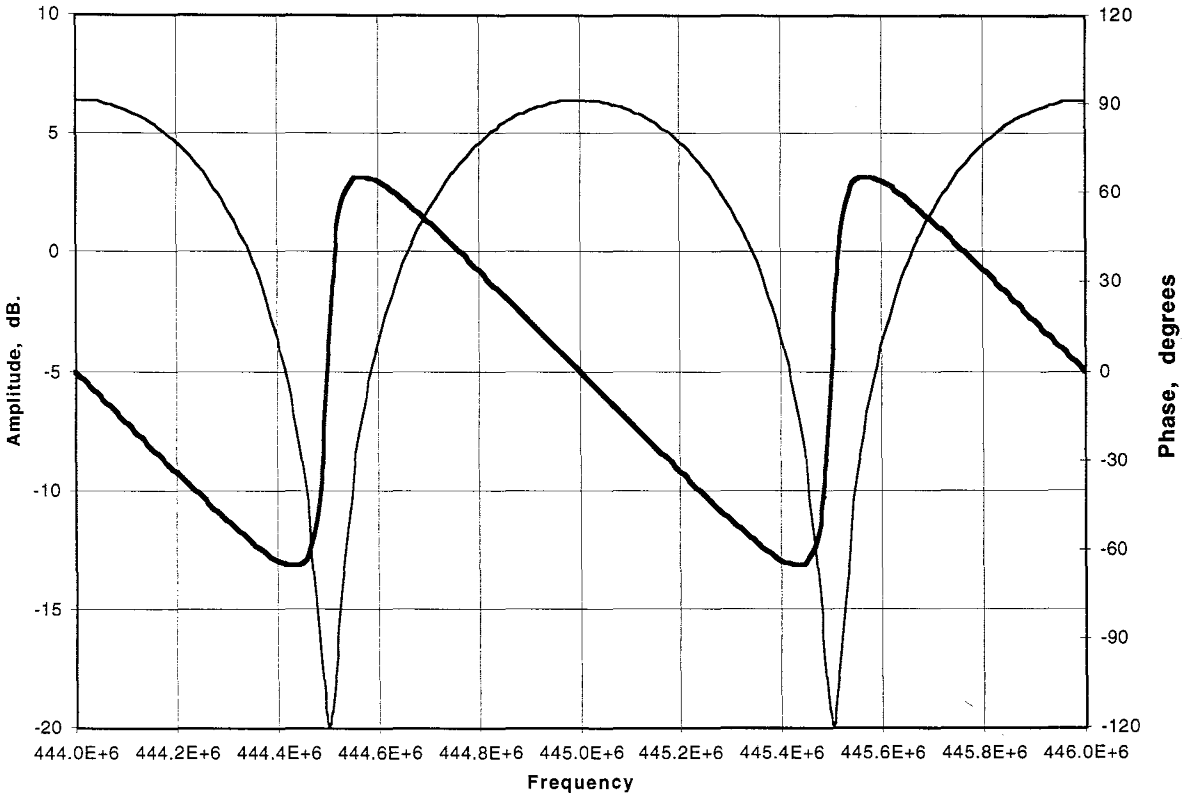


Figure 13-3 - Amplitude and phase of main signal plus one multipath component versus frequency. The multipath component is 91% of the amplitude of the main signal, and is delayed 1 microsecond. This represents a 'minimum-phase' fade.

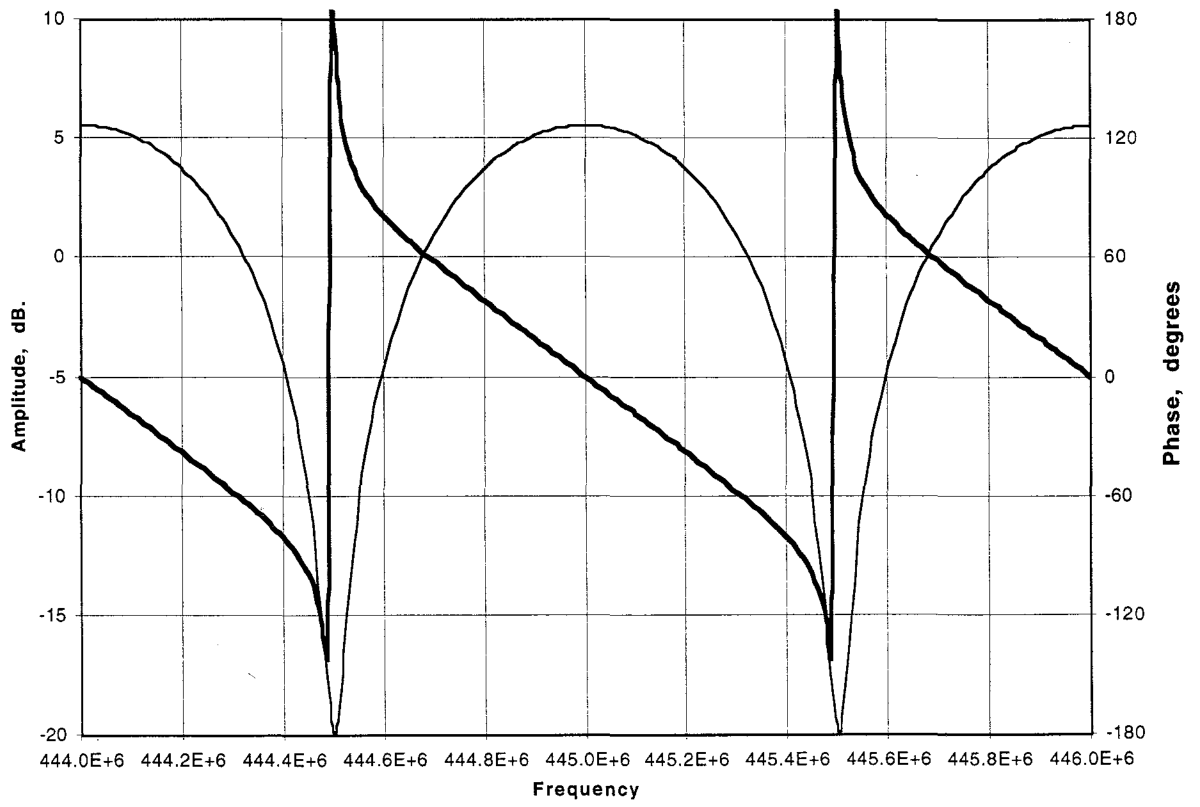


Figure 13-4 *- Amplitude and phase of main signal plus one multipath component versus frequency. The multipath component is 110% of the amplitude of the main signal, and is delayed 1 microsecond. This represents a 'non-minimum-phase' fade.*

Since the phase curves slope in opposite directions near the notch frequency, the group delay will have opposite sign between the minimum-phase fade and the non-minimum-phase fade multipath conditions. Figure 13-5 plots the group delay for these two cases vs. frequency. The analysis increment in figure 13-5 is 100 kHz. A smaller analysis increment would appreciably increase the group delay.

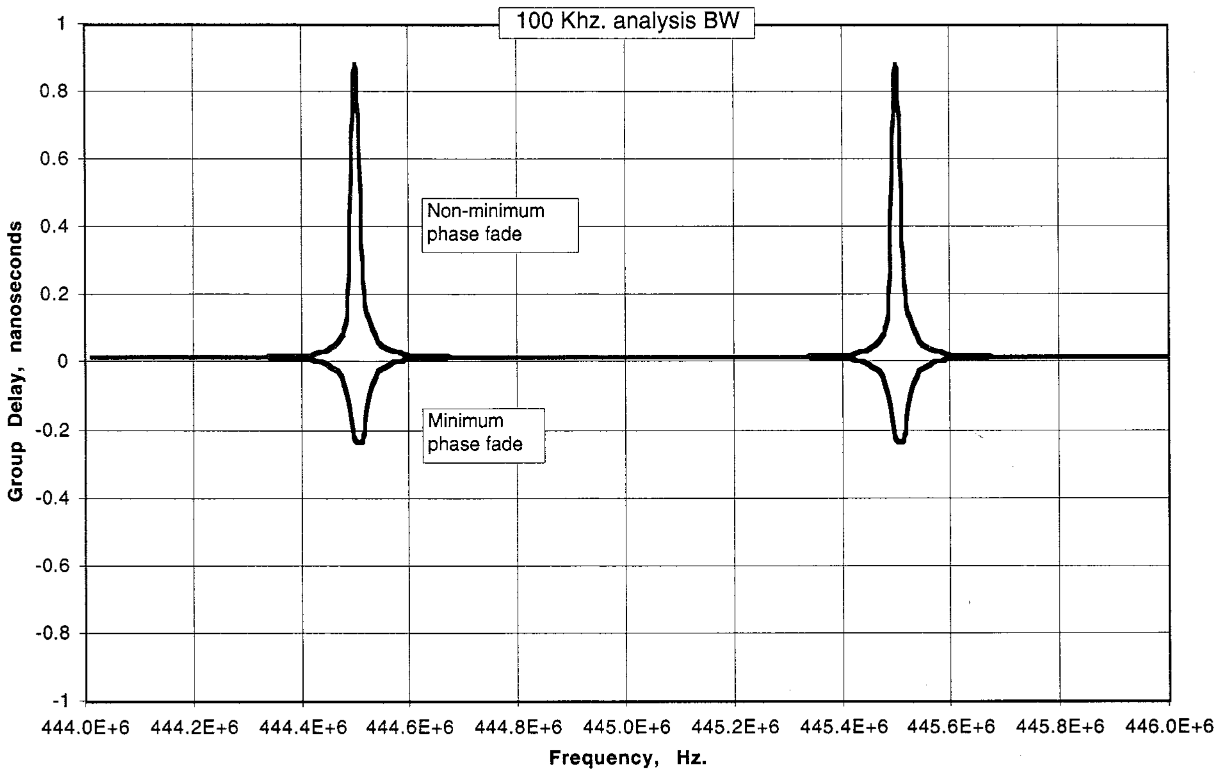


Figure 13-5 - Group delay of minimum-phase and non-minimum-phase two-ray multipath fading vs. frequency.

Rayleigh fading

The above situation with only two paths is not common for complex paths, where there will be several multipath components. When a number of paths exist, the signal amplitude can be represented by a Rayleigh distribution, and the phase by a uniform distribution. The probability of the received signal envelope, R (the amplitude), versus the mean received signal power, S, for a Rayleigh distribution is expressed as:

$$p(R) = \frac{R}{S} \exp\left(-\frac{R^2}{2S}\right) \tag{13-7}$$

Another way to describe the fade is by the cumulative distribution of R, which gives the probability that R is at or below a certain signal level. An approximation, for R smaller than $S/2$, is $p(R) = R/S$. From this a simple, useful approximation for the CDF can be derived as:

$$CDF(R) \approx \frac{R^2}{2S} \tag{13-8}$$

Figure 13-6 shows the PDF, and CDF, of a Rayleigh distribution. Looking at the Rayleigh PDF, we can intuitively understand the meaning of the graph. Most probably the received signal has an amplitude of 1. It can go to zero, but obviously a received signal strength of less than zero is not possible. The probability of exactly zero signal strength is very rare. The received signal can exceed 1 when everything arrives in phase. Only 5% of the time will it exceed 2.5 times the non-multipath signal. The CDF tells us, for example, that the received signal will have a received strength of 1 or greater 60% of the time.

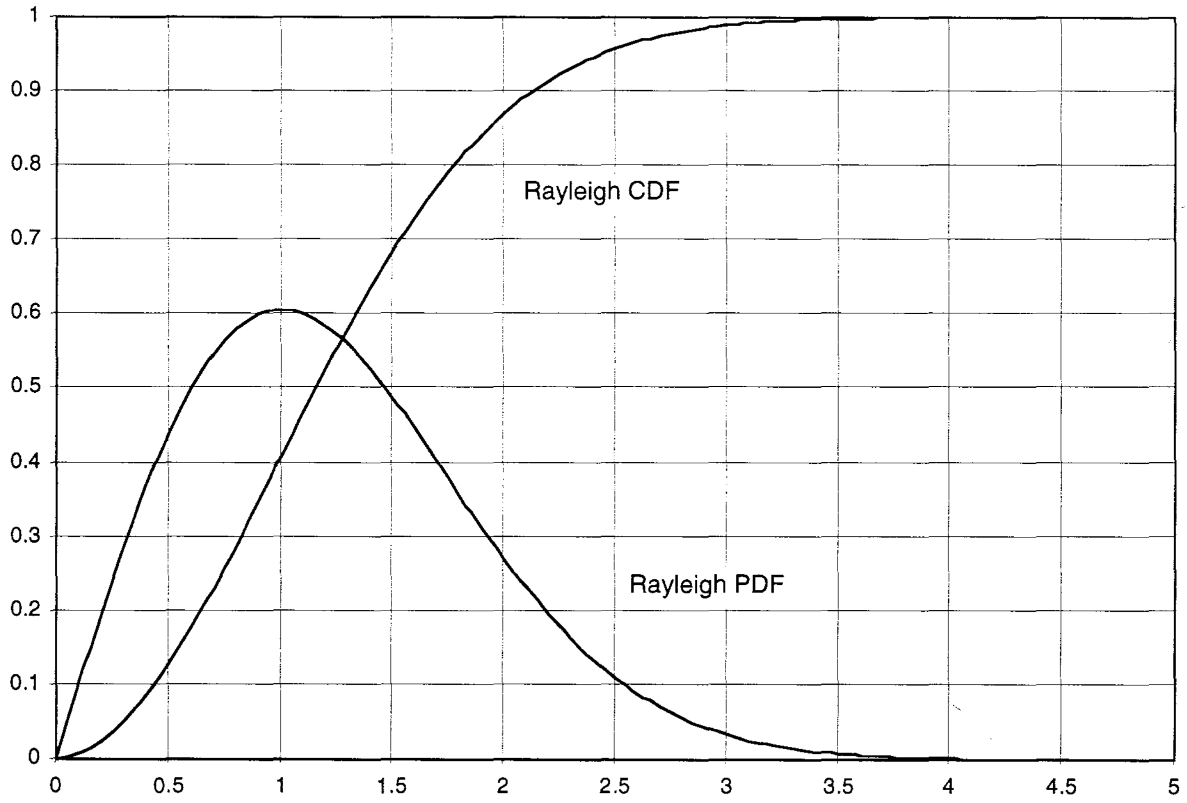


Figure 13-6 *- Rayleigh distribution PDF and CDF, linear plot.*
A Rayleigh distribution models fading in a multipath environment.

It can be seen from the CDF that the signal is less than half-amplitude about 14% of the time. In order to assess the probability of deep fades, the CDF is viewed on a log scale. Figure 13-7 illustrates the CDF versus log amplitude.

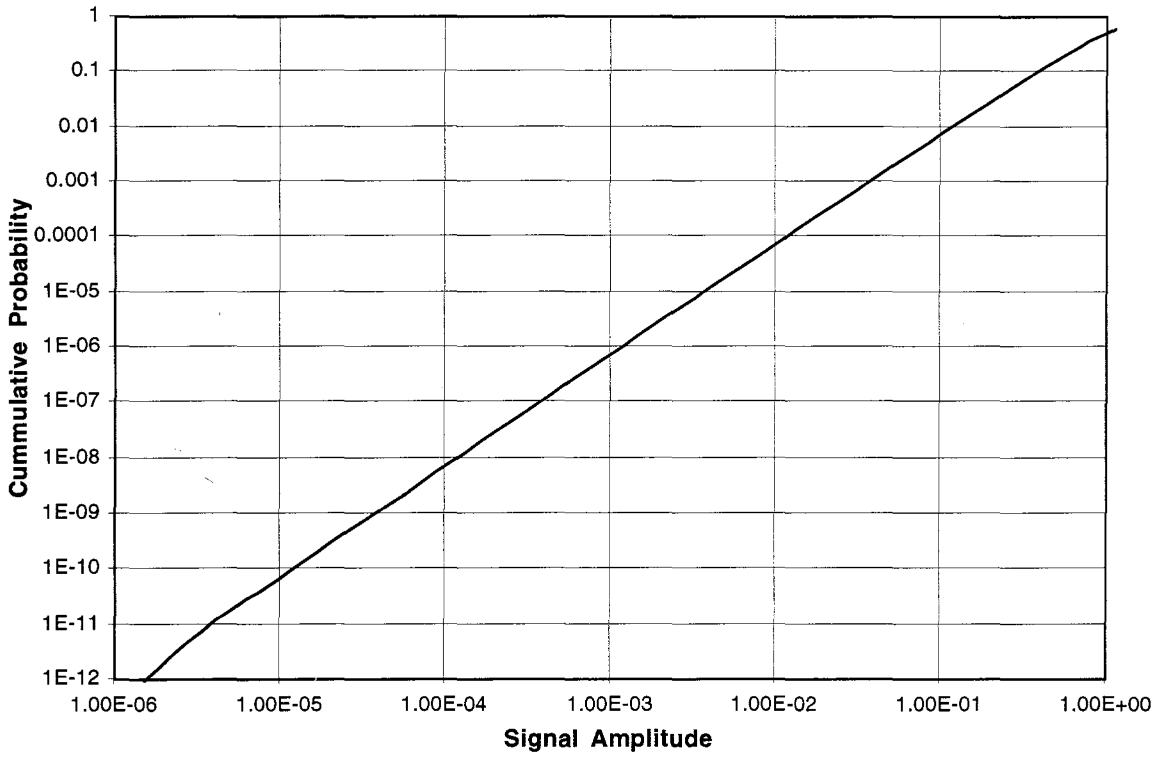


Figure 13-7 *- Rayleigh CDF for small values of R. This plot shows the probability (y-axis) that a faded signal will be less than a given signal amplitude (x-axis). In this case, the signal amplitude will have a fade of .01 (-40 dB) or worse about .0001 of the time (.01% of the time).*

In the Rayleigh faded case, the phase of the received signal has a uniform probability distribution from 0 to 2π, meaning that the received phase can assume any value. Although the phase may or may not change quickly, it will drift so that it has no known constant value.

Generally, the Rayleigh faded path is a pessimistic assumption for VHF and UHF propagation. It models tropospheric and long, poor paths well, but does not model short, direct paths accurately. If one of the propagation components is direct and does not fade, and further if only one or two additional multipaths exist, and they are weaker than the main path, then the propagation is better modeled by a *Rician* distribution. See Schwartz (1980) and Steele (1992) for a discussion of Rayleigh and Rician distributions.

In calculating the fading due to multipath propagation, a sinusoidal carrier model is assumed. However, in a digital data transmission system, a large problem is that the delayed propagation component may be delayed several bit times or more later than the primary propagation path. This causes inter-symbol-interference, because the value of the bit and phase of the long-delayed bit are not related to the current bit. This can cause a significant rise in the bit error rate, and there are no linear operations in the receiver that can compensate for the effect. Of course, error correction coding can correct some of the errors that occur. However, if the multipath causes a severe enough bit error rate, then the FEC may not be able to correct the original data stream.

Circular Polarization

An interesting effect occurs at the reflecting site on the secondary path, the phase of only one polarization component of the reflected wave reverses at the reflection point provided that angle of incidence is greater than Brewster's angle, while the phase of both polarization components reverses at angles of incidence less than Brewster's angle. Stated in other terms: the horizontally-polarized component (with respect to the reflecting surface) always experiences phase reversal at the reflecting surface, *regardless of the angle of incidence*, while the vertically-polarized component experiences phase-reversal at incidence *less* than Brewster's angle, and does not experience phase reversal at incidence *greater* than Brewster's angle (Crawford, 1968). If the reflecting surface causes phase-reversal of only one of the polarizations, then the linearity-sense of a horizontally or vertically-polarized wave is not changed. However, the circularity of a circularly-polarized wave will be reversed upon this type of reflection (large incidence angle). See figure 13-8 which illustrates the circularity of reflected signals. Thus, one method to reduce the impact of this type of multipath reflection is to use circularly polarized antennas at both the transmitter and receiver. The primary path will have the correct circularity, while the secondary path may have the opposite circularity, and it will be strongly rejected by the receive antenna. Tertiary paths that encounter double-reflections will, unfortunately have their circularity reversed twice, resulting in the sense being the same as the originally transmitted polarization, so the receive antenna will not reject the tertiary path circular signal. Normally, however, the amplitude of each multipath component is reduced by each reflection. Consequently, the impact of tertiary paths is much less than the effect of the secondary path.

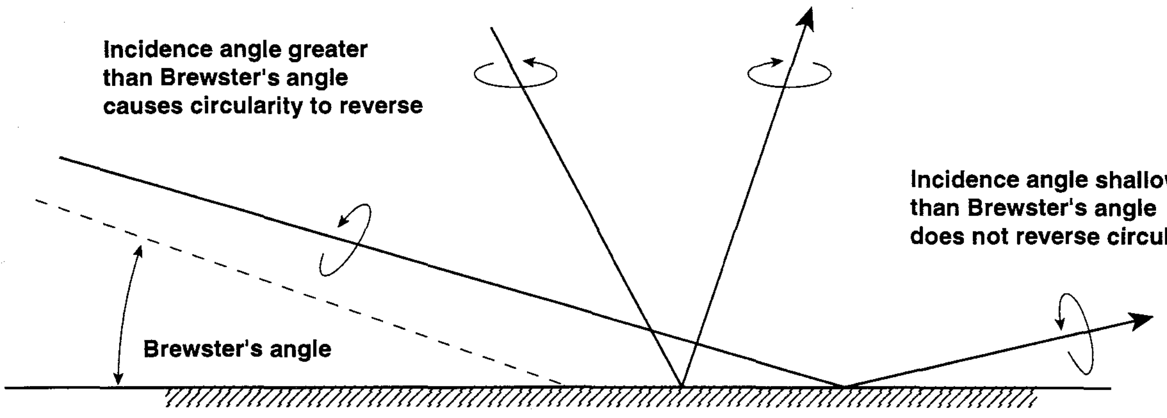

Incidence angle greater than Brewster's angle causes circularity to reverse
Incidence angle shallower than Brewster's angle does not reverse circularity
Brewster's angle

***Figure 13-8** - circularity reversal of polarization depends on the angle of incidence to the reflecting surface.*

Sometimes, multipath is caused not by a reflecting surface, but by changes in the index of refraction of the atmosphere. In these cases an incident wave can be propagated between the transmitter and receiver over two paths that are completely airborne. In this case, the circularity sense of the secondary path will not be reversed, since the second path is due to refraction, not reflection. Thus, the use of circularly-polarized antennas will be ineffective against this multipath distortion. Normally, however, the difference in path lengths is relatively small, and thus the frequency spacing of the nulls is large. In this instance, narrow-band data channels experience flat fading vs. frequency, and a mostly linear change in phase with frequency. In both cases, the channel distortion is mostly attenuation, and not dispersion, so the shape of the received eye pattern does not change significantly.

Wideband Data Channels

The use of relatively high data rates on point-to-point or on multi-point VHF and UHF data links will mean that multipath can cause fading that is very frequency dependent. Additionally, the phase will be non-linear over the channel bandwidth when there is a multipath notch nearby in frequency. Both of these effects will result in significant distortion to the received eye pattern, resulting in a severe degradation in performance. Compensation for frequency-selective effects requires the use of adaptive equalization in the receiver in order to improve the bit error rate during multipath fades. If the multipath component is delayed by several bit times, then the inter symbol interference cannot be easily canceled. In actual practice, the multipath components change with time, and this leads to more complex received signal components.

Rules of Thumb

Several inequalities can be described that bound reasonable criteria for the signaling rate of a channel in multipath propagation conditions. Some conditions and terms are defined:

Tm	is the time spread of the received main and multipath components.
Fs	is the bit signaling rate (Fbaud).
Fd	is the doppler frequency (bandwidth) spread of time-varying multipath components. The channel changes state during a period on the order of 1/Fd. This is the rate at which fading occurs and stops occurring.

To process a received signal coherently, the frequency of the received signal should not vary significantly during the bit interval, so rule number one is:

$$F_s >> F_d \tag{13-9}$$

The second rule of thumb is that there should be no frequency-selective distortion of the received signal, thus:

$$F_s << \frac{1}{T_m} \tag{13-10}$$

Combining the two inequalities bounds the signaling rate that is generally considered useful. This can be expressed as:

$$F_d << F_s << \frac{1}{T_m} \tag{13-11}$$

Of course, it is possible that the channel is so poor that the above rules must be violated. A more detailed treatment of signaling rate bounds for fading channels is given by Rainish and Perl (1989).

HF Channel models

In contrast to the relatively simpler multipath fading models prevalent in the VHF and UHF ranges, the HF ionospheric skywave channel propagation model is more complex. There are 3 general reflection effects that occur on these paths, and that connect the two endpoints of the propagation path:

Density variations versus altitude in the ionization, causing high-path and low-path rays,
Magnetic-ionic effects that cause polarization-dependent paths (the "ordinary" and "extraordinary" rays), and
Non-uniformity of the ionospheric layers.

The three effects each have different delay time spreads. The non-uniformity variations cause differential delays in the 20-40 ms range, while the ordinary/extraordinary and low/high rays each individually result in delay spreads in the 100-200 μs range (Stein, 1987). As the frequency of operation is increased to very near the Maximum Usable Frequency (MUF) for a particular propagation mode, the delay spreads for that mode tend to decrease. For single-hop propagation between 800 km to 2000 km (500 miles to 1300 miles) near the MUF, the propagation tends to be supported from a single region of the ionosphere. Therefore, the doppler spread is small, resulting in very slow fading, on the order of 100 seconds between fades (corresponding to a doppler frequency difference, Fd, of 0.01 Hz. or so).

When the path is short, less than 800 km (500 miles), sometimes multiple-hop and single-hop propagation occurs, with multiple hops occurring at large incidence angles. With long-paths, from 2000 km to 10000 km (1300 miles to 6000 miles) in length, multiple hops are required and even layer-to-layer modes can exist. With these paths, the overall doppler frequency can be up to 2 Hz (fades on the order of 0.5 seconds). On these long paths, delay spreads of 3 milliseconds are possible.

A description of several HF channel model parameters is given in CCIR recommendation 520-1 (CCIR 520-1). The following three categories are defined:

Good conditions
Differential time delay: 0.5 ms.
Doppler Frequency spread (fading rate): 0.1 Hz.

Moderate Conditions
Differential time delay: 1 ms.
Doppler Frequency spread (fading rate): 0.5 Hz.

Poor Conditions
Differential time delay: 2 ms.
Doppler Frequency spread (fading rate): 1 Hz.

Additionally, this recommendation suggests that modems be tested for a two-component multipath signal with equal amplitude, and with a relative frequency offset from each other:

Differential time delay: 0.5 ms
Doppler Frequency spread (fading rate): 0.2 Hz.
Range of Frequency offset: 0 to 10 Hz.

Signaling Rate Bounds

The signaling rates in baud can be estimated from the above parameters and equation (13-11). For Good conditions, the limits are approximately:

$$0.1 \ll F_s \ll 2000$$

For moderate conditions, the rate is on the order:

$$0.5 \ll F_s \ll 1000$$

And for poor conditions, the rate is on the order:

$$1 \ll F_s \ll 500$$

Determining the actual optimal rate is more difficult in practice, since the inequalities "much greater than" and "much less than" are hard to determine, and vary with a particular path. Usually adaptation of the rate is the most successful approach. If one takes a factor 1/10 as meeting the inequalities, then the data rate vs. delay spread would lead to estimates of signaling rates for the good channel as 200 baud, the moderate channel as 100 baud, and the poor channel as 50 baud, which seem reasonable in practice.

HF Channel Simulation

Work on models for, and implementations of HF channel simulation have occurred over the last 30 years. Much of the theory behind modern HF channel models is described well in the work of Waterson, et al. (1970). A good description of a simulator implementation of this model is given by Ehrman, et al (1982), and another description of HF channel simulators is given in CCIR report 549-3 (CCIR 549-3). Several channel simulators have been implemented in the audio frequency range and are generated by digital signal processing (DSP) units. This section describes the channel simulator used by Ehrman. It has been found through experimentation that a 3-path model seems to provide an adequate measure of performance in HF simulation, and thus most implementations have implemented 3 taped delay lines. Figure 13-9 illustrates the 3 tap model. Each tap represents one particular multipath component. The output of the simulator is the sum of the three components (just as the output of the VHF/UHF analysis above involved the sum of two components).

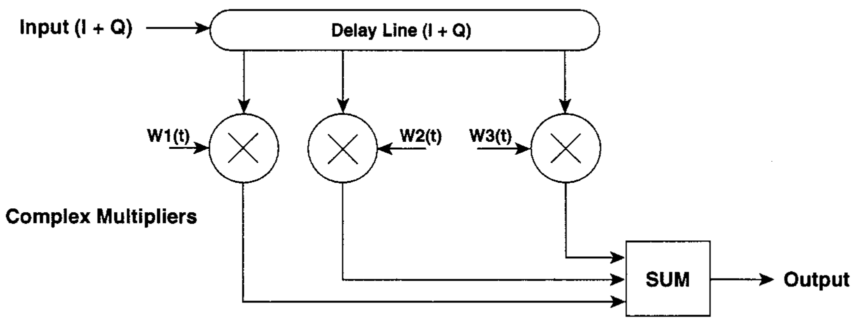


Figure 13-9 *- three tap multipath simulator. W1(t), W2(t), and W3(t) are three complex signals representing the individual path amplitude and phase characteristics.*

In practice, the audio input to the tapped delay line consists of only the in-phase part of the received signal, and so the quadrature input must be generated. This is done through the use of a wideband 90-degree phase shift, which can be accomplished with a Hilbert transform on the input data. Additionally, each of the three paths resolves into the separate ordinary and extraordinary propagation components. Consequently, each path should contain two independent multiplier chains, since the two polarization components are largely uncorrelated and their amplitude and phase vary independently. Finally, additive gaussian noise, and perhaps impulse noise should be added to the resultant output signal to model the channel. Combining these effects results in the simulator shown in figure 13-10.

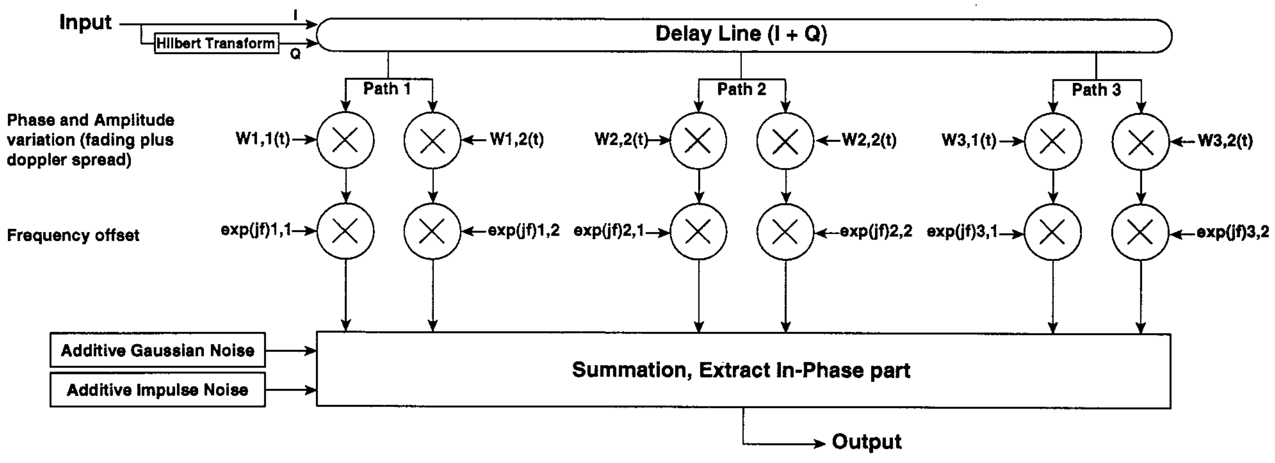


Figure 13-10 *- HF ionospheric narrow-band propagation simulator. All the multipliers are vector multipliers (I + Q).*

In the model, each of the three taps is separated into the two magneto-ionospheric components which are acted upon separately. Although the separation of the two magneto-ionic components more accurately models the ionosphere, it does not necessarily provide much additional insight into the testing of modems, and so can usually be neglected during comparative modem testing (one modem vs. another modem). In Ehrman's model, a frequency offset for each path is introduced through a separate complex multiplier after the amplitude variation and doppler spread is introduced. Normally, each complex multiplier is really four multipliers: two for the in-phase part of the signal, and two for the quadrature-phase part of the signal. However, since only the real part of the output is used, two of the four multipliers (in each path mixing function) can be eliminated.

Theoretically the coefficients for the various paths and magneto-ionic components are slightly correlated with each other. In practice, though, the correlation is small and can usually be ignored. The above simulator is adequate for narrow-band HF operation only, between 2.5 kHz and 12 kHz bandwidth depending upon conditions and the particular path. Extension to a wideband HF model would require modeling dispersion of the HF path (frequency-dependent propagation delay), which is not done in the narrowband models.

The implementation of the simulator portion, excluding the generation of the coefficients for the multipliers, is straightforward and can be readily implemented in a DSP unit. The coefficient generation requires developing gaussian-distributed random numbers with an approximate 5 Hz bandwidth, and the up-sampling and smoothing of those numbers at the sample rate of the delay line. It may be easier to generate the coefficients in a separate computer, and update the DSP unit periodically (Forrer, 1995).

Coefficient Properties

The properties of the coefficients are what is specified in the CCIR recommendation referenced previously. The real and imaginary parts of the coefficients are statistically independent and gaussian-distributed, leading to Rayleigh fading. The coefficient multiplication of each path contributes to the doppler frequency shift. Each tap coefficient *W* can be described by the following equation:

$$W(\nu) = \frac{1}{A\sigma\sqrt{2\pi}} \exp\left(\frac{-(\nu - \nu_0)^2}{2\sigma^2} \right) \qquad (13\text{-}12)$$

Where A is the component attenuation, usually in the range 0 to -30 dB. Two sigma is the RMS doppler frequency spread, usually 0-5 Hz. ν_0 is the frequency offset, usually in the range +100 to - 100 Hz and ν is the frequency variable undergoing random change. Since the real and imaginary attenuation components have equal RMS variation and have zero-mean, they produce Rayleigh fading.

The attenuation components and doppler spread are calculated by first generating, then low pass filtering, two independent gaussian random number generators: one for the real component, and one for the imaginary component, using equation (13-12). The use of thrid order Butterworth filters yields a "good enough" match to a gaussian shape. The bandwidth of the low-pass filters is varied to yield a doppler spread in the range of 0 to 5 Hz. In a separate multiplier, the frequency offset component is generated by rotating the phase of the signal by a constant amount each sample time. The phase angle of the frequency offset is calculated by phase accumulation:

$$\theta(nT) = \theta((n-1)T) + \omega T \tag{13-13}$$

Where omega is the frequency offset of the component and T is the time, n is simply an index variable. The phase, theta, is modulo 2π radians. This phase forms the complex vector to the second of the two multipliers, by Euler's rule:

$$\exp(\theta(nT)) = \cos(\theta(nT)) + j\sin(\theta(nT)) \tag{13-14}$$

Bit Error Rate Performance of Rayleigh Faded Path

Prediction of the bit error rate (BER) of a Rayleigh fading path is very difficult at best. So many assumptions need to be made, especially in the HF environment, that performance verification by simulation is most practical. Still, a rough guide to the asymptotic performance of a system under these conditions can provide some guidance to either the level of results attainable, or the need for counter measures against poor BER. In the Rayleigh-fading path, the bit error rate at any given instant in time is inversely proportional to the signal to noise ratio. So, as a first approximation, the bit error rate overall will be proportional to the amount of time that the path spends at a given poor signal to noise ratio.

Diversity Reception

One common and effective way to improve the received bit error rate on an HF ionospheric path is to employ diversity reception. The diversity can be in the form of different polarizations of two receive antennas, or it can be in the positioning of several different receive antennas several wavelengths apart. The separation of the antennas effectively de-correlates the multi-path nulls in time from each of the antennas. Then, the antenna with the best or the non-faded signal will have the lowest error rate. The error rate of a diversity reception system will be the joint probability of fading occurring simultaneously on all the antennas. This is roughly proportional to SNR to the *M*th power, where *M* is the number of receiving antennas. One rough approximation is described by the following equation (Stein, 1987):

$$P_e \approx \frac{1}{2}\frac{2^M}{\Gamma^M} \qquad (13\text{-}15)$$

Where *M* is the diversity order, and Gamma is the SNR. Figure 13-11 is a plot of the BER defined by equation (13-15) for M=1, 2, and 4 (no diversity, 2-way, and 4-way diversity), and includes a coherent 2 FSK non-faded BER curve for comparison. This equation assumes non-coherently combined FSK signals from the several parallel diversity detectors.

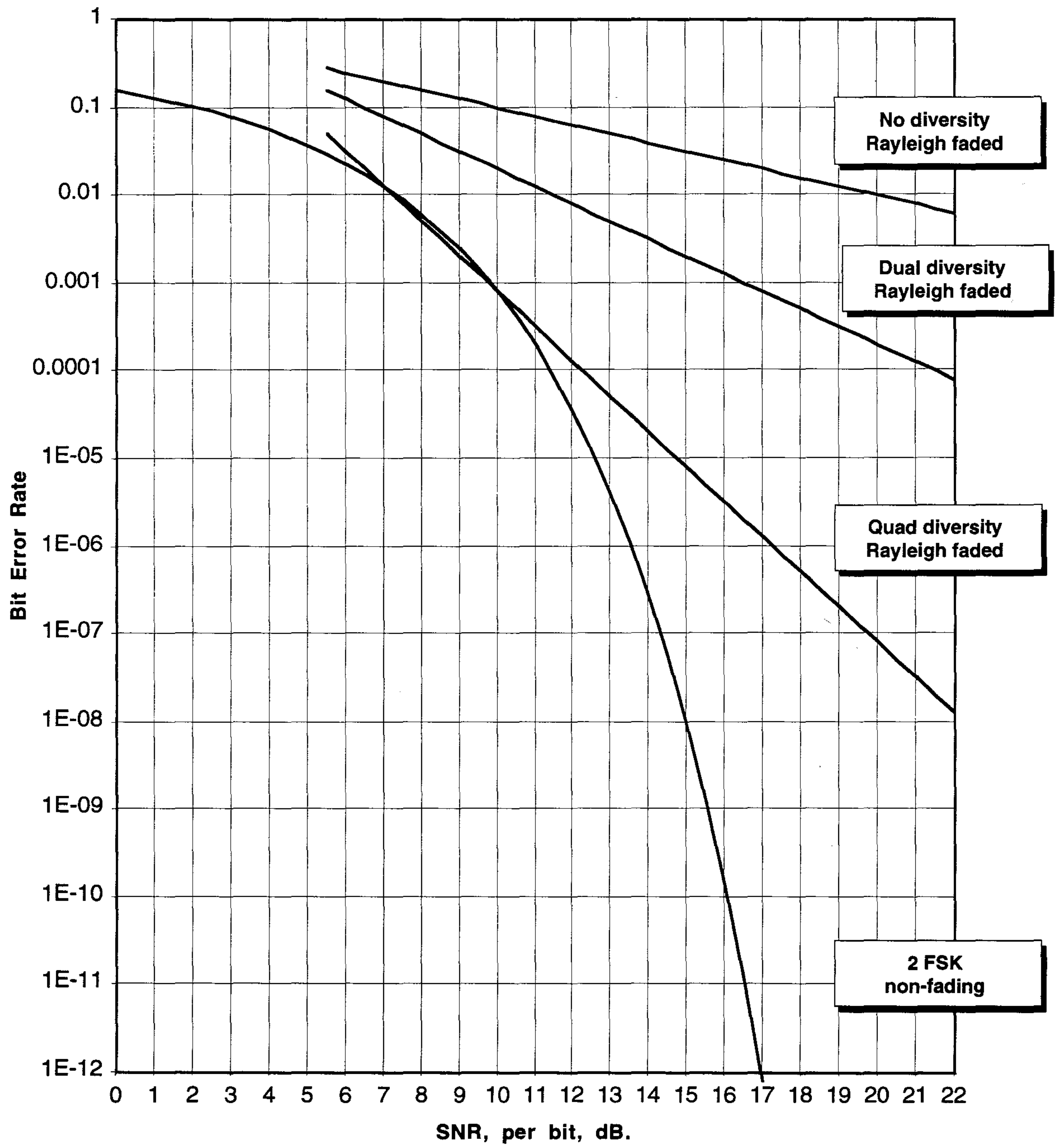


Figure 13-11 - *Approximate asymptotic bit error rate for Rayleigh fading channel*

It is also possible to provide diversity by transmitting on different frequencies, with the correlation of fading events between two different carriers decreasing with increasing separation of the carrier frequencies. Well-decorrelated fading requires that the frequencies be separated on the order of from 10 kHz to a few 10's of kHz in the HF bands. This is generally not practical for amateur usage. Anecdotally, it has been noted that 850 Hz carrier separation offers some decorrelation compared to 170 Hz carrier separation. This suggests that the use of multi-carrier orthogonal FDM, with the carriers carrying redundant (or redundantly-coded) information spaced over a few kHz, might prove effective in providing some diversity.

Another approach to diversity is to transmit information multiple times with the interleave time spaced out, so that it is likely that a single fade does not destroy both copies of the information. An even better method is to use an error-detection-and-retransmission protocol, which attempts retransmission and keeps trying a block of data until it succeeds. This type of diversity is time-diversity. Transmitting twice would be equivalent to dual-diversity, and a correction protocol is equivalent to multi-dimensional diversity.

Approaches to HF Modem Design

Examination of the BER curve illustrates that the HF path must use modems that are very robust to errors. From previous sections, we have seen that the signaling rate is limited due to the time delay spread, Tm. At the high error rates of HF, it is also clear that forward error correction must space out the correction symbols to a time on the order of the doppler fading rate, Fd, to assure that most symbols within the constraint time of an FEC code word effectively miss the fading event. Given these constraints, a number of different systems have been developed with varying degrees of success.

RTTY - Live With the Errors

The earliest system, RTTY, operates at 45.45 baud, effectively a slow enough signaling rate that the HF channel can support the symbols even under relatively poor channel conditions. The modulation is non-coherent FSK, usually CPFSK with 170 hertz shift. Most stations transmit roughly unshaped (or rectangular) data bits, leading to a much wider transmit spectrum than is necessary, such as with baseband shaping (i.e., raised-cosine).

RTTY employs no error correction nor automatic redundancy, so essentially the philosophy is to live with the errors. Error correction is accomplished by the operator manually asking for a retransmission if the received text appears to be incorrect. Certainly not a good technology for sending binary files, where the operator cannot discern good text from bad text!

AMTOR - Simple Error Detection and Re-transmission

Later, a system known as SITOR was developed for maritime use and adapted to amateur use by Martinez (1981). AMTOR, as the amateur variant is called, operates at a signaling speed of 100 baud using non-coherent FSK. It sends each 5-bit baudot character as a 7-bit character. The 7-bit code words are chosen so that 4 bits are of one state, and 3 bits of the other state. Then 3 characters are transmitted in a group. The receiver checks each 21-bit block to assure a correct match to the 4/3 character rule. If any errors are detected, then the receiver requests a retransmission of the block, or else it acknowledges the block. A 4/3 block will detect any single bit error in a character, but will not detect a 2-bit error within a character. Thus, the error detection capability of AMTOR is very limited at high error rates. Modern FEC coding can result in much more efficient detection of errors. The 4/3 scheme was very cost-effective back before the microprocessor was common in modem equipment. The concept of retransmitting a small block at a later time has proven to be effective at combating Rayleigh fading, as at least one of the retransmissions has a good chance of missing a deep fading event. Of course, both the transmission and the acknowledgment must miss deep fading events.

AX.25 Packet Transmission - 300 baud HF

Transmission of AX.25 packet radio, based mostly on the TAPR TNC-2 design or its many derivatives, consists of FSK modulation with 200 Hz shift. AX.25 does not employ forward error correction, and uses a 16-bit CRC to detect errors in a packet. Correction of the errors is by a go-back-n strategy, where all subsequent frames (including the errored frame) are resent, whether or not they have in fact been received correctly anyway. Given the nature of Rayleigh fading, and the high residual error rate, the throughput of AX.25 will be extremely poor on HF. Results to date indicate this. AX.25 with no other coding or FEC at 300 baud is not recommended for HF operation. Alternatively, setting the frame size to one at least helps minimize the number of retransmissions in AX.25.

More Recent Developments

During the 1990's, DSP computing became more cost-effective for amateur radio usage. With it, the advent of more sophisticated and suitable protocols and coding for HF have been developed. Three such systems have come into common usage.

GTOR

G-TOR™, a trademark of Kantronics, Inc. is a rate-adaptive system utilizing FSK keying at 100, 200, or 300 baud, and a shift of around 170 to 200 Hz. The data are forward error correction (FEC) coded using an extended Golay (24, 12) code (see section on coding), which can correct up to 3 errors in a block of 24 bits. The code words are interleaved 48:1 in order to be resistant to a burst error longer than 3 bits. Each frame also contains a 16-bit CRC, independent of the Golay code. G-TOR is a hybrid ARQ scheme, in that the Golay parity bits are not transmitted if the channel supports a low error rate and the 16-bit CRC's indicate no errors. If the transmitter receives a NACK (negative acknowledgment) then it sends the Golay parity bits, possibly allowing the receiver to reconstruct a valid frame. After each frame of several code words are sent, the transmitter must be acknowledged (ack'ed) or negative acknowledged (nack'ed). This type of coding and retransmission, with the resultant delay between retransmissions, is fairly effective against multipath fading. For the case where the baseband signaling is rectangular and the shift is 170 Hz., the bandwidth to the first nulls will be about 2(85+300) = 770 Hz. with significant sidelobes further out depending on the filtering. It is assumed that CPFSK is used, thus reducing the sidelobe amplitude somewhat and resulting in a bandwidth of approximately 1000 Hz. at the -30 dB points. The minimum possible bandwidth occurs with heavily rolled-off baseband signals of (impossible) zero-alpha factor, in which case the bandwidth is 2(85+150) = 470 Hz. The roll-off shaping for G-TOR is not published, and so it must be assumed that it is larger than 500 Hz, and less than 770 Hz between the first nulls when operating at 300 baud. For 300 baud and non-coherent demodulation, the frequency shift should optimally be at least 300 Hz. A lesser shift (such as 170 Hz, or 200 Hz) will result in some power penalty, but this may not be too significant under HF propagation conditions. A description of the protocol can be found in the proceedings of the 13th ARRL DCC (Huslig, 1994).

Pactor II

Pactor II™, a trademark of Special Communications Systems, utilizes a two-tone differential PSK modulation system with raised-cosine roll-off of the baseband signal. This results in good spectral properties and a narrow transmit spectrum. It utilizes two carriers, each DQPSK modulated at 100 baud. Thus, the raw throughput is 400 bits/second. Pactor II utilizes a convolutional code of rate 1/2, and constraint length = 9. Thus, the net throughput is 200 bits/second, assuming the convolutional code can correct the errors. Convolutional codes of rate = 1/2 are fairly robust in terms of error correction capability in AWGN (see section on coding). However, they can suffer from fatal error propagation during an sufficiently long error burst. Generally, they must be interleaved for good performance on HF ionospheric paths, or must be coupled with an effective block-length limit / retransmission scheme in order to provide good performance under deep multi-path fading. The use of QPSK and baseband roll-off results in amplitude variations in each output carrier, and so a linear amplifier is required for each carrier. Since two carriers are used, a linear amplifier is needed anyway. Hence, the flat amplitude properties of offset-QPSK would not really add any benefit. The crest-factor of two carriers requires a 3 dB peak-to-average ratio of the transmitted signal. Additional amplitude variation of each carrier pushes the peak-to-average requirements up a little higher. A description of the Pactor II protocol can be found in the Digital Journal (Rink, 1995).

Clover II

Clover II™, a trademark of Hal Communications Corporation, is a set of modem modulation and protocols generally referred to as Clover. The signal consists of 4 carriers, modulated at 31.25 baud, and spaced 125 Hz (4/T) apart, assuring orthogonality. Ten different modulation formats are used, depending on the received signal quality. The simplest mode is 2 independent 2-FSK transmissions, while the highest data rate is supplied by using 16-phase, 4-amplitude modulation of each of the carriers. Most of the modes are based on various phase-shift modulations, from binary PSK to 16-ary PSK. Each of the four tones is amplitude-shaped, so that the amplitude is zero when the phase transition of the carrier takes place. Clover utilizes Dolph-Chebychev shaped pulses. Additionally, the amplitude shaping results in minimizing the crest-factor of the resultant signal, keeping the peak-to-average ratio around 8 dB. All modulation modes result in the emitted spectral width being less than 500 Hz at the -50 dB points. Data rates vary from 31.25 bits per second for the slowest FSK mode, up to 750 bits per seconds for the 16-PSK+4-ASK (by 4 carriers) modulation mode, although the symbol rate is always 31.25 symbols/second. Clover utilizes Reed-Solomon forward error correction. RS- FEC is an extension of a block code to larger blocks, where the efficiency is better. Clover allows changing the amount of overhead in the RS code from 40% to 0% depending on the need for error correction on the channel. It provides a control block that allows the two ends of the link to measure SNR and phase dispersion on the channel, and to automatically rate-adapt the modems depending on the conditions simultaneous with the transmission of user data. Information on Clover II is derived from HAL (1992).

Future Improvements

At one time the use of class-C amplifiers and mechanical teleprinters mandated rectangular data bits and FSK so that the amplitude of the output remained constant. Today, virtually all amateur stations utilize SSB transceivers with class AB amplifiers and electronic modems which are sufficiently linear to permit good baseband data bit shaping, and the resultant significant reduction in transmitted spectral width. Additionally, mechanical RTTY printers employ a stop bit of 1.41 bit times in length which complicates baseband channel filter design. Many computer generated RTTY signals could utilize 1 or 2 stop bits, restoring synchronicity to the channel. To effect such a change, all would have to choose the same baseband channel response and stop bit length. One good choice would be x/sin(x) compensation plus square-root of raised cosine at the transmitter, and square-root of raised cosine at the receiver. For simplicity of clock recovery, an alpha factor of 1 would be easiest to implement.

Additionally, there are no amateur modems that utilize m-ary FSK or Orthogonal FDM (OFDM). The use of large-m for OFDM means that the integration time constants can be long. However, below a coherence interval of about 5 Hz, the HF channel may distort the received signal significantly on some paths. Multi-carrier OFDM may provide rejection of CW-interference due to redundancy in the carriers, but at the expense of greater bandwidth utilization.

Today's amateur HF practice is focused towards channel bandwidth minimization in order to accommodate as many users as possible. Unfortunately this leads to larger requirements for SNR, and more susceptibility to on-channel interference. Another general approach would be to utilize orthogonal wideband modulation, wherein co-user interference rejection is via code-diversity rather than frequency separation.

Chapter 13 - Reference

CCIR Recommendation 520-1, "Use of High Frequency Ionospheric Channel Simulators", Annex I - Simulator Parameters for Qualitative Testing.

CCIR Report 549-3, "HF Ionospheric Channel Simulators"

Crawford, Frank S., Jr. (1968). "Waves: Berkeley Physics Course, Volume 3." Education Development Center, Inc. Section 8.3.

Ehrman, Leonard, Bates, Leslie B., Eschle, John F., and Kates, James M. (1982). "Real-Time Software Simulation of the HF Radio Channel." IEEE Transactions on Communications, Vol. COM-30, No. 8, August.

Forrer, Johan, KC7WW. (1995). As suggested by private correspondence, TAPR HFSIG newsgroup.

Huslig, M., et al. (1994). "G-TOR: The Protocol." 13th ARRL Digital Communications Conference Proceedings (available from TAPR). pp. 49-79

Martinez, J.P., G3PLX. (1981). "Amtor, an Improved Error-Free RTTY System." QST, June, pp. 25-27

"PCI-4000 CLover-II HF Radio Modem Operator and Reference Manual", Copyright HAL Communications Corp., Urbana Ill., 1992.

Rainish , Doron and Perl, Joseph M. (1989). "Generalized Cutoff Rate of Time- and Frequency-Selective Fading Channels." IEEE Transactions on Communications, Vol. 37, No. 5, May, pp 449-466.

Schwartz, Mischa. (1980). "Information Transmission, Modulation and Noise." 3rd edition, McGraw-Hill, Inc. Chapter 5-9.

Steele, Raymond. (1992). "Mobile Radio Communications." Pentech Press Ltd., London, and IEEE Press, NJ. Chapters 1.2.2 and 1.2.3.

Stein, Seymour. (1987). "Fading Channel Issues in System Engineering." IEEE Journal on Selected Areas in Communications, VOL. SAC-5, No. 2, February, pg. 68-89.

Watterson, Clark C., Juroshek, John R., and Bensema, William D. (1970). "Experimental Confirmation of an HF Channel Model." IEEE Transactions on Communications Technology, Vol COM-18, No. 6, December.

Rink , Tom and Helfert, Hans-Peter. (1995). "Pactor II - Part I." Digital Journal, Vol 43, No 1, January. Parts II and III of the protocol specification are presented in the February and March issues of the Digital Journal.

Chapter

14

Automatic Request for Repeat (ARQ)

In examining methods to correct errors, generally Forward Error Correction (FEC) has been considered. There is another commonly used method for error control, and that is the use of Automatic retransmission (ARQ) of defective transmission blocks. Actually, the two techniques can be intermixed to improve throughput. ARQ has proven useful on severely fading channels, where a complete outage can occur for short periods of time. In fact, ARQ can be considered as another form of transmission diversity. A general classification of ARQ falls into two categories: simple, and hybrid.

Simple ARQ

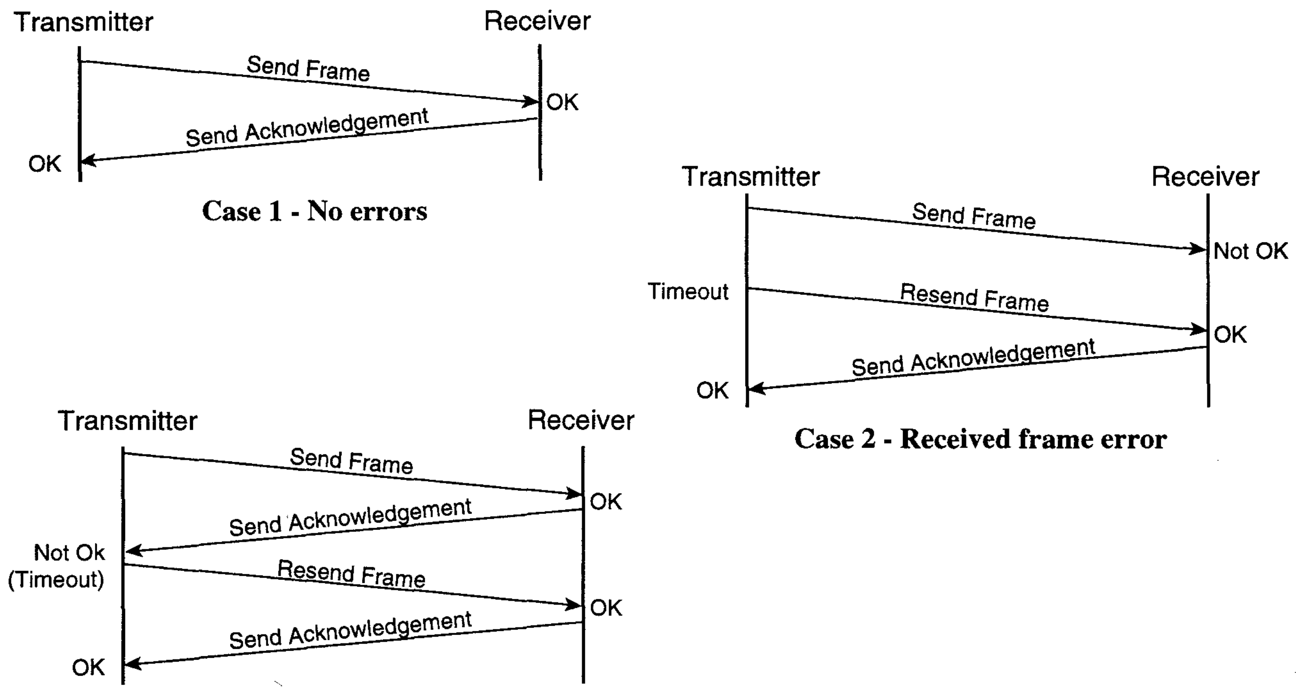
Simple ARQ is a protocol that blocks the transmitted data up into units, called frames or packets. Each frame contains some sequencing information, all or a portion of the data to be transmitted, and a check-sum. The receiver examines each received frame, and determines that it is valid if the received check-sum is valid. Otherwise, it rejects the block. The sequence number in the frame allows the receiver to know whether each valid received frame is the next to be processed, or if some frame(s) prior to the current good frame but subsequent to a previous good frame have been destroyed. In the case of AX.25, the transmit frame sequence counter is a 3-bit counter, that operates modulo-8. So, if the receiver correctly received frames 1,2,4,5, then the receiver would know that frame number 3 was lost. AX.25 utilizes a strategy known as go-back-n, where once an error frame is detected, all subsequent frames are discarded *even if they were received properly*. In other words, the receiver instructs the transmitter to go back and resend all blocks received since the last good one (2 in this example). Another approach is to utilize selective-reject. In this case, the receiver would acknowledge correct receipt of blocks, 1,2,4,5 and the transmitter would only need to resend block number 3. A protocol that allows multiple frames to be sent within each transmission is called a *sliding window protocol.* If the window size is one, then only a single packet can be sent before it must be acknowledged. In this case, the sliding window and the stop-and-wait protocols are exactly the same thing. Many HF protocols are of the stop-and-wait variety, whereas many VHF and UHF protocols are of the sliding-window variety. The stop-and-wait strategy is good when the channel is poor and latency time is determined by channel error rate. A sliding window protocol is good when the latency time is determined by the transmitter and receiver key-up and stabilization time, and the channel error rate is low.

Hybrid ARQ

A hybrid ARQ protocol is one where forward error correction information may be transmitted in order to correct a previously received, but invalid, block of information. One possible characteristic of an FEC code is that it may be *systematic*. A systematic code is one that generates codewords that contain clear-text data plus FEC code symbols. The transmitter generates the codewords, but only transmits the clear-text data and a checksum, while saving the FEC codewords in memory. If the checksum of a received block is good, then the receiver acknowledges the transmitter, and the transmitter can discard the saved FEC code symbols. If, however, the receiver detects an error in the received checksum, then the transmitter can transmit the FEC code words that it has saved. The receiver may be able to correct the received data frame by utilizing these FEC bits. Then the receiver can acknowledge the transmitter which could then discard the FEC symbols for that block. The advantage of the hybrid ARQ scheme is that on a low-error rate channel, the efficiency is very high, since FEC code symbols are sent infrequently. On poorer-error-rate channels, the FEC code symbols get sent more frequently, but succeed in correcting many errors. Finally, the error rate can be so poor that even with the received FEC codewords, occasional transmit blocks must be completely repeated. In this case, the back up simple ARQ stop-and-wait protocol is also used. The combination of all 3 protocols means the throughput of the channel is optimized depending upon channel conditions.

Performance of ARQ

It is possible to roughly model the performance of ARQ transmission in the presence of a channel with errors. The analysis becomes more difficult as the statistics of the channel change. For the assumption of a constant probability of bit error, the rate of retransmission of errored frames can be calculated. The simplest case to analyze is the simple-ARQ channel, with a stop-and-wait protocol. Figure 14-1 indicates the operations that can occur in the stop-and-wait protocol. The first case is when the transmitter sends the first frame. Which is properly received and acknowledged, and the transmitter properly detects the acknowledgment. A second case is where the receiver does not properly receive the frame. The transmitter times-out waiting for the acknowledgment and then resends the frame. Finally, a third case is where the receiver properly receives and acknowledges the first frame, but the transmitter does not receive the acknowledgment due to a channel error. Then, it times-out and resends the first frame, which the receiver has to re-acknowledge.



Case 1 - No errors

Case 2 - Received frame error

Case 3 - Acknowledgement from receiver to transmitter is garbled the first time.

Figure 14-1 *- Stop and Wait Protocol operations*

In the analysis, let us assume that the transmitted frame is *tx* data bits in size, and the acknowledgment is tack data bits in length. Further, the time-out time is T seconds, the probability of a bit error on the channel is *b*, and the data rate is *r*. Additionally, the turn-around time from receive to transmit is *tr* seconds. Then the probability that the transmitted frame is properly received is computed, and the probability that the acknowledgment is properly received is also computed.

$$P_{tx} = (1 - b)^{tx} \tag{14-1}$$

$$P_{tack} = (1 - b)^{tack} \tag{14-2}$$

$$p_s = P_{tx} P_{tack} = (1 - q_s) \tag{14-3}$$

Where *Ptx* is the probability of the transmit frame being received correctly, and *Ptack* is the probability of the acknowledge frame being received correctly. The probability that both events occur is *ps*, and *qs* is the probability that either the transmit frame or the acknowledgment fails. In analyzing the protocol, it is assumed that retransmissions will continue to occur until the frame is eventually successfully received and successfully acknowledged. The number of times that the retransmission has to occur, on average, determines the throughput of the channel. A retransmission occurs for each failure, *qs*, so the number of channel transmissions, *c*, is:

$$c = 1 + q_s + q_s^2 + q_s^3 + \dots \quad = \sum_{n=0}^{\infty} q_s^n \qquad (14\text{-}4)$$

This summation can be solved, and then ps substituted for qs from equation (14-3) as:

$$c = \frac{1}{1 - q_s} = \frac{1}{p_s} \qquad (14\text{-}5)$$

The time of each cycle is composed of turn-around time at the sender, time to transmit the frame, and the time-out time if it fails. If it succeeds, then the time in the backwards direction from the receiver to the sender is the turn-around time at the receiver, and the time to send the acknowledgment. So the total time to send a frame is then:

$$t_{tot} = 2t_r + \frac{tx + tack}{R} + (c - 1)\left(t_r + \frac{tx}{R} + T\right) \qquad (14\text{-}6)$$

Which is composed of one successful transmission, and (c-1) unsuccessful transmissions. The chart in figure 14-2 illustrates this equation with *tx*=1000 bits, *tack*=100 bits, *tr* = 120 milliseconds, *T*=1.2 seconds, for various values of the bit error rate *b*, and for some different data rates, *R*. The channel efficiency, *e*, in percent can be expressed as:

$$e = 100 \frac{tx}{R\, t_{tot}} \qquad (14\text{-}7)$$

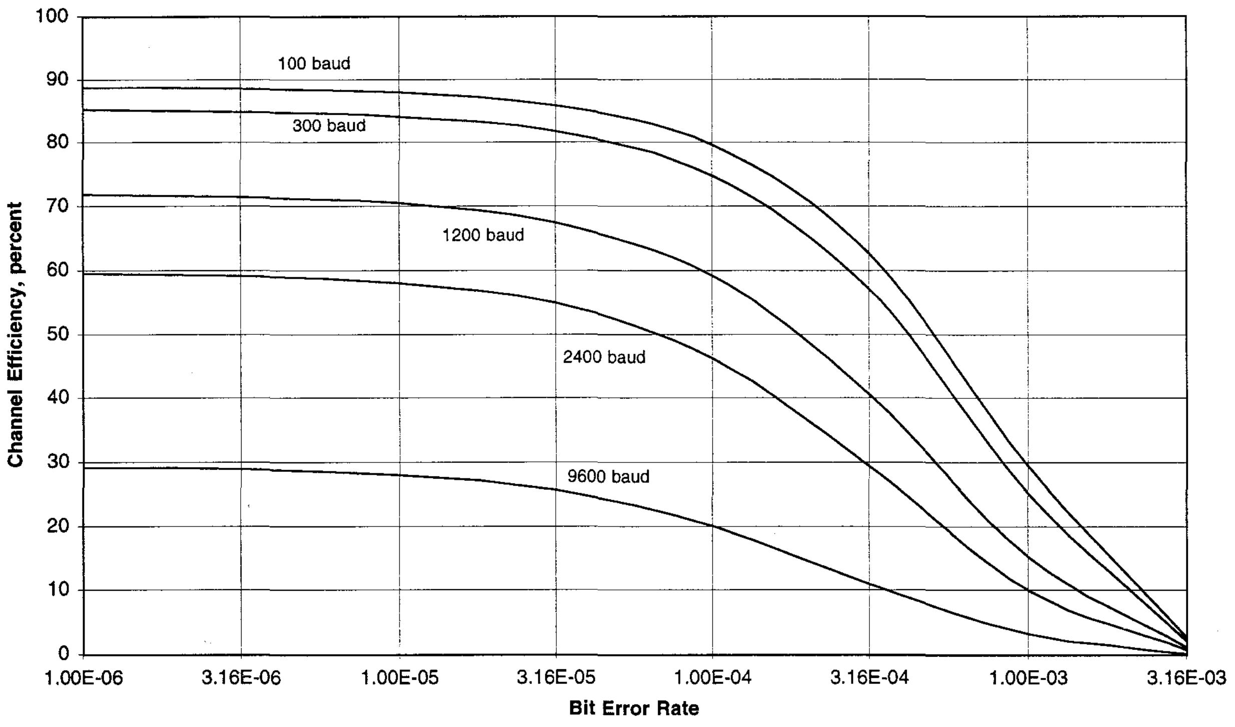


Figure 14-2 *- Channel efficiency versus bit error rate for some different data rates. Stop-and-wait protocol. See text for conditions.*

It can be seen that the channel efficiency starts to fall rapidly when the bit error rate is about one tenth of the transmit frame size. The efficiency decreases with increasing data rate — this is due to the fixed turn around time and the fixed time out time versus data rate assumption. A good rule-of-thumb is that the frame size, in bits, times the bit error rate should not exceed 0.1. For example, if the bit error rate is .0001 (1E-4) then the frame size should be no larger than 1000 bits.

$$\textit{Frame size in bits} \bullet \textit{Bit error rate} \leq 0.1 \qquad (14\text{-}8)$$

Chapter

15

Testing Considerations

Testing a modem for performance has been briefly addressed in the section on propagation, especially the use of the HF channel simulator. However, a number of performance tests need to be conducted to verify proper operation of a modem. Tests generally should measure the bit error rate versus several different factors. As always, modem design is a trade-off between optimum performance under differing assumptions about the channel and radio equipment. Consequently, no one modem is likely to be optimal under all circumstances.

Bit Error Rate vs. Received Signal Level

All other conditions being equal, testing a modem for bit error rate (BER) against the received signal level is probably the single most useful test that can be conducted. However, the other conditions must be well controlled to prevent invalidating the test sequence.

Baseband BER Test

The first test that should be conducted is to verify proper operation of the modem back-to-back, if possible without any radio equipment in place. If the modems are capable of being directly connected, then good control and repeatability of the test is possible. We will assume that the modems are connected at baseband, so that there is no concern about channel frequency response, frequency offset or error, or of propagation anomalies.

The test should inject additive white gaussian noise (AWGN) between the receiver and the transmitter, and the received signal to noise ratio is adjusted. The bit error rate at the receiver is plotted as a function of the received SNR, which should be converted to Eb/No for consistency, and so that comparison against theoretical curves is possible. A pseudo-random data stream should be used as the data source and sink, even if scrambling or other data modifying circuits are incorporated. Modem frequency response errors do not always show up when using patterns with short repetition rates. Typically, 2^{15}-1 patterns are useful to assure that there are no low-frequency content problems. Even better is 2^{23}-1 , but may it be more easily used at higher data rates.

Sometimes, the design of the modulator will dictate that the modulator must supply clock to the pattern generator, rather than the pattern generator supplying clock to the modem. This usually does not present a timing problem except at very high data rates.

One concern with low data rates is that the test time can be very long. In this case, some statistical criteria should be used to terminate the test early. Generally, a single bit error is not a valid criteria for a test interval. At least 10 bit errors should be allowed to accumulate prior to calculating the BER and moving to the next signal level. Figure 15-1 indicates a representative test configuration for BER vs. AWGN testing at baseband.

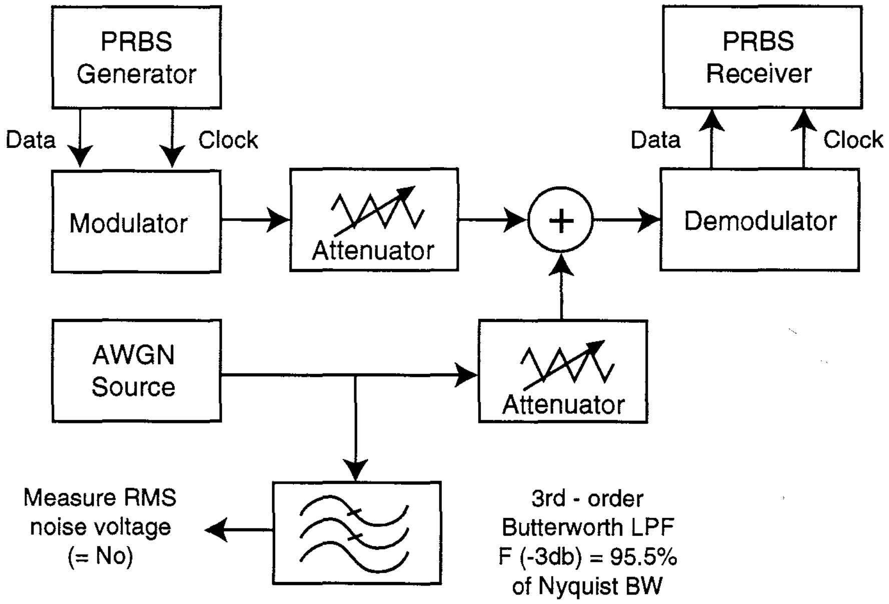


Figure 15-1 *- Test setup for measuring Bit Error Rate (BER) vs. Additive White Gaussian Noise (AWGN) at baseband.*

To conduct the test, the bandwidth of the AWGN source must be known, and the energy content within that bandwidth must be known. An RMS voltmeter that gives accurate readings with high crest factors should be used to determine the noise power level and the modulator signal level. For HF modems one common method dictates measuring the noise power within a 3 kHz bandwidth. It has an advantage in that noise power is easily measured with a 3 kHz low-pass filter. As long as all modems are measured against the same noise spectral density (same noise power and same noise bandwidth), then the test can be comparatively fair. But if 3 kHz bandwidth does not match the noise bandwidth of the demodulator, then the results can be misleading (such as excellent BER with negative SNR ratio). Thus, *tests should always be normalized to Eb/No, not to SNR.* For higher-order filters, the noise bandwidth of a filter is approximately the same as the 3 dB bandwidth. However, for the types of filters used for data transmission (Raised-Cosine, Butterworth, etc.) this is not necessarily true. A third-order Butterworth filter has a noise bandwidth 4.7% greater than its -3 dB bandwidth. A 3rd-order Butterworth low-pass filter whose -3 dB bandwidth is 95.5% of the Nyquist bandwidth has the same noise bandwidth as the Nyquist bandwidth and makes a good calibration filter, since the RMS noise voltage out of it is the same as the RMS noise voltage within the Nyquist bandwidth. This provides a convenient calibration point, for when the RMS signal voltage equals this RMS noise voltage, then Eb/No = 0.0 dB.

In practice, it is easiest to set the noise attenuator to zero, and set the signal attenuator so that the RMS signal into analog summing device is the same as the RMS noise voltage from the Butterworth filter. Then the attenuation value of the noise attenuator will be directly calibrated in Eb/No. This presumes that changing the noise attenuator value does not alter the load impedance of the AWGN source or affect it's output voltage.

Butterworth Calibration Filter

Figure 15-2 is the schematic of an 3rd-order Butterworth low-pass active filter. The 3 dB cutoff frequency is changed by altering R1, R2, and R3. All other component values should remain as shown in the schematic. The resistors are all the same value, and the value can be computed by:

$$R = \frac{1}{2\pi f C} \qquad (15\text{-}1)$$

Where C = 0.01 uF (1.0 x 10^{-8}). Doubling the resistor values lowers the -3 dB frequency by half. To design a filter with a noise bandwidth calibrated to match a 9600-baud data signal, the -3 dB bandwidth should be 4800 * 0.955 = 4584 Hz. In this case the resistors would have the value 3472 ohms (use the nearest 5% standard value). The input terminal must be fed from a low source impedance referenced to ground.

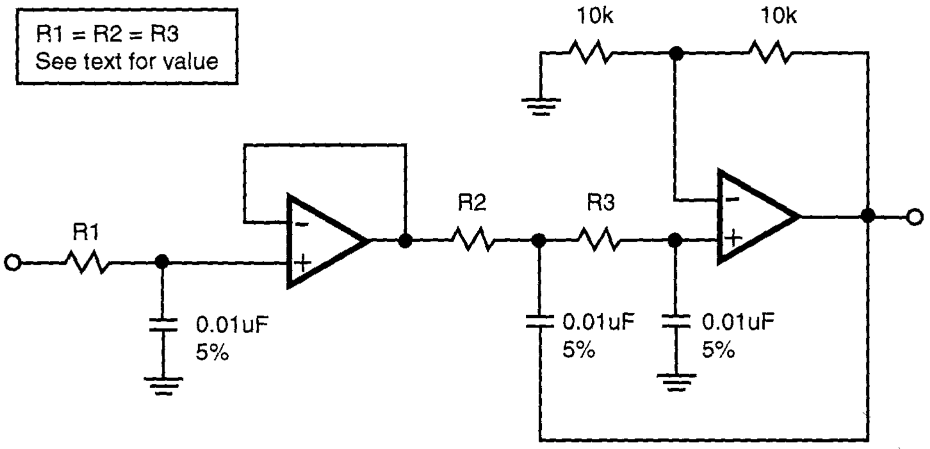


Figure 15-2 - 3rd-order Butterworth filter used to calibrate Nyquist noise bandwidth (No). The value of R1, R2, R3 sets the -3 dB. frequency.

For raised-cosine filters with x/sin(x) compensation, table 15-1 illustrates the excess noise bandwidth versus the Nyquist bandwidth for alpha from 0 to 1 (this is usually the filter for the entire channel, transmitter plus receiver). The excess bandwidth of an uncompensated square-root of raised cosine filter (normally just the receiver portion of the filtering) is 1 for all values of alpha (it is equal to the Nyquist bandwidth regardless of the alpha value).

Noise bandwidth of x/sin(x) compensated raised-cosine filter	
alpha	**Excess Noise Bandwidth, compared to Nyquist BW for channel (RC + x/sin(x)).**
0.0	1.39
0.1	1.33
0.2	1.28
0.3	1.24
0.4	1.20
0.5	1.17
0.6	1.15
0.7	1.13
0.8	1.12
0.9	1.16
1.0	1.13

Table 15-1 *- Excess equivalent noise bandwidth of the complete channel for compensated raised-cosine response.*

To find the equivalent noise at baseband, add the excess noise to the noise of the Nyquist filter (rectangular, with bandwidth of one-half the baud rate, at baseband). For linear modulation (ASK, PSK) at IF, the bandwidth of the Nyquist filter is equal to the baud rate. If the channel response is unknown, the noise power bandwidth of the receiver should be estimated by measuring inside the demodulator at a point where most of the channel filtering has been accomplished. This will set the noise power and the signal power at the same bandwidth. The *Eb/No* can then be derived from the carrier to noise (*C/N*) ratio by:

$$\frac{E_b}{N_0} = \frac{C}{N} - 10 \log \left(\frac{f_b}{B_n} \right) \tag{15-2}$$

Where *fb* is the baud rate, and *Bn* is the equivalent noise bandwidth. Since a square-root of raised cosine filter at the receiver has the same noise bandwidth as the Nyquist bandwidth, no adjustment is necessary if this filter type is used at the receiver and the noise is compared in the channel. Otherwise, appropriate adjustments are needed.

Radio Characterization for Data

After testing of the modem at baseband has established the performance of the modem, then testing of the modem with radio equipment can be conducted. In general, the performance will degrade as compared to the baseband performance. The amount of degradation is directly dependent on the radio implementation, and the modulator and demodulator interfaces to the radio equipment. It is easiest to characterize the radio independently of the modem before starting the combined tests.

Radio-based Test

The characteristics of a radio that need to be measured depend somewhat on the type of modulation being utilized. For example, FSK modulation requires that the frequency modulator be linear, and have adequate bandwidth, but does not impose strong requirements on the amplitude linearity of the final amplifier. In contrast, binary-PSK and ASK both require significant linearity in the transmitter (in binary-PSK, to prevent generation of spurious sidebands, and in ASK to preserve the amplitude characteristics).

FSK - Transmitter Characterization

The three important criteria for an FSK transmitter are 1) frequency linearity, 2) modulation bandwidth, and 3) frequency stability. In the case of utilizing an SSB transmitter to transmit AFSK tones to produce FSK modulation, the SSB transmitter should be characterized as for linear modulation, instead. The transmission of rectangular-data pulses must be accomplished carefully with FSK transmitters due to the potentially excessively wide transmitted bandwidth. We will assume that some filtering of the FSK signal has been accomplished at baseband to control the transmitted spectral width.

The frequency linearity of the FSK modulator is the transfer function of the modulator, expressed as transmitter output frequency vs. input control voltage. If the modulator is DC-coupled, then the test can be conducted very easily. A variable-DC voltage is applied to the modulator, and the frequency of the transmitter is measured over the desired range of the modulator input voltage. The resultant plot should show a linear relationship between frequency and voltage. Figure 15-3 shows a plot of a BB109 varicap controlling the frequency of a crystal channel element for a 445 MHz transmitter, and another plot of a different varicap controlling the same channel element. Notice the difference in linearity between the two. The non-linear curve results in distortion of the received pattern ("squishing" of one of the received data bit levels).

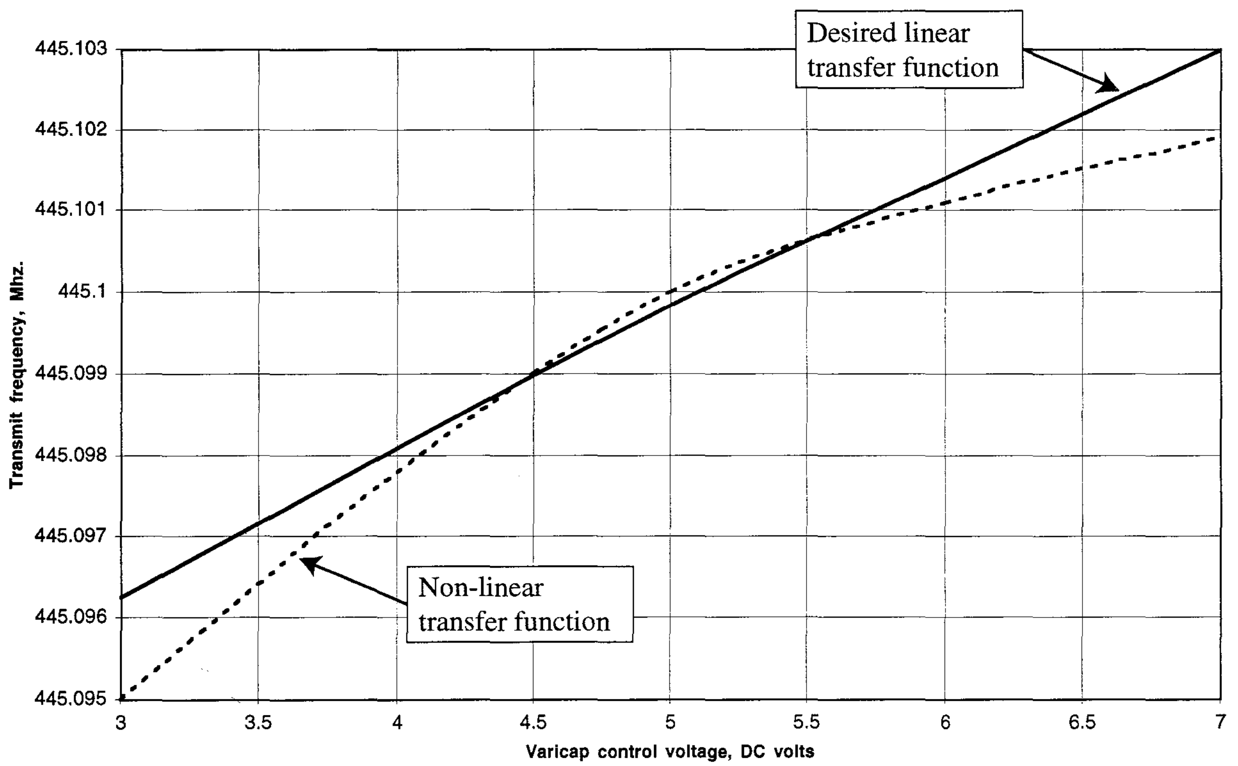


***Figure 15-4** - Transmit frequency vs. varicap voltage on FSK modulator for two different varicap types.*

The modulation bandwidth of the transmitter circuit is important in that the total transmitter frequency response is the product of the baseband filter and the transmitter modulator response. It is easiest to perform all of the response shaping with the baseband filter, in which case the FSK modulator should be wideband, so as not to distort the signal. However, some care to prevent wideband noise from entering the modulator should be taken. The modulation bandwidth can be measured with a variable-frequency audio signal generator. Care should be taken not to exceed the deviation range of the FSK modulator. Figure 15-5 is a test setup for determining modulation bandwidth of the modulator.

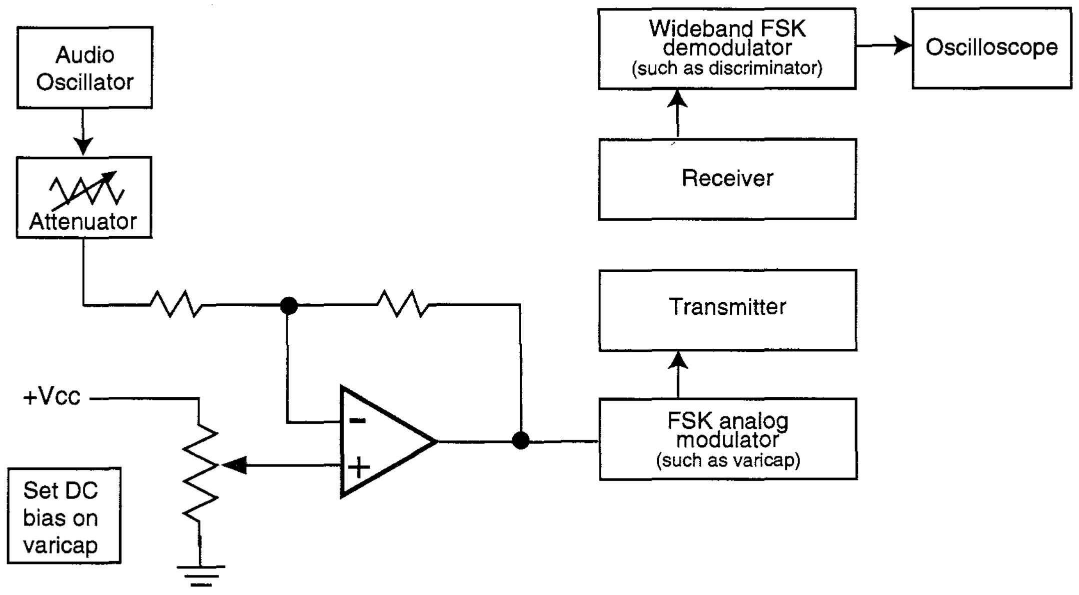

Audio Oscillator
Attenuator
+Vcc
Set DC bias on varicap
Wideband FSK demodulator (such as discriminator)
Oscilloscope
Receiver
Transmitter
FSK analog modulator (such as varicap)

***Figure 15-5** - test setup to measure modulation bandwidth*

The wideband FSK demodulator should be separately verified to have a modulation bandwidth much larger than the Nyquist bandwidth of the data signal of interest. This can be verified with known-accurate test equipment beforehand.

Lastly, the frequency stability of the transmitter should be verified. This should be done with all modulation circuitry in place and with all DC bias voltages properly adjusted. It may be easiest to suppress the modulation and measure the resultant carrier frequency with a frequency counter. The temperature of the transmitter should be varied beyond the expected operational temperature range and the carrier frequency measured at each temperature. If the sum of the transmit deviation plus the frequency error exceeds the receiver's IF bandpass or discriminator characteristics, then the transmit frequency must be stabilized (either via temperature compensation or temperature control). The potential temperature drift of the receiver should be factored into determining what constitutes acceptable transmit frequency stability.

In an FSK modulator, the varicap diode may be temperature dependent, and may cause some drift independent of the crystal oscillator. Different varicap diodes have different temperature coefficients, but sometimes a varicap can be somewhat temperature compensated by the use of a series silicon diode which will contribute a temperature-dependent voltage drop in the DC bias to the varicap.

Other Transmitter Characterization

For other modulation schemes, different parameters of a transmitter may have to be understood. For binary 2PSK, the transmitter should be linear to prevent excessive transmit bandwidth, so the linearity of the unit should be ascertained. Techniques such as third-order intermodulation distortion testing using a two-tone signal and a spectrum analyzer are commonly used for SSB transmitters. Additionally, a transceiver may require that the transmit/receive turn around time (the time between cessation of transmission and the ability to receive properly) be short. This is especially true with AMTOR. The turn-around time is usually specified as the time when a weak signal produces output unperturbed by the prior transmission within some level of recovery.

FSK - Receiver Characterization

The three properties that were characterized on a FSK transmitter need to also be characterized for the FSK receiver. Measuring the receiver discriminator output voltage vs. input frequency is accomplished with a signal generator (plus a frequency counter) and a DC-voltmeter. Measuring the modulation bandwidth requires the use of a transmitter with a known very-wideband modulator. The test setup of figure 15-5 can then be used to characterize the receiver. Frequency stability vs. temperature is characterized in a manner similar to the transmitter, except that the temperature of the receiver is varied. In addition to these three parameters, the response of the receiver IF filter is very important, and can be a significant problem. There are two types of distortion: amplitude vs. frequency, and group delay distortion. For linear modulation (such as PSK and ASK), it is possible to compensate for the amplitude and group delay distortion (as long as the filter bandwidth is sufficiently wide to pass the necessary parts of the signal). But with non-linear modulation (such as FSK), it is not always possible to compensate for filter distortion. This is because with non-linear modulation, there is no direct relationship between IF filter frequency and baseband frequency. For small deviation FSK, however, the signal is reasonably linear, and the filter can usually be compensated "well enough" for practical use.

Assuming a completely linear modulation scheme, figure 15-6 illustrates a typical IF filter amplitude and group delay response, and how the filter response is folded down to baseband by a linear demodulation mechanism (such as mixing). If the carrier is not centered in the IF filter passband, then there are substantial distortions to both the amplitude and delay responses. In general these non-centered distortions are difficult to equalize since the upper and lower sidebands of the received signal are distorted differently. Therefore, it is relatively important to assure the received signal stays centered in the passband if the signal has a bandwidth approaching the bandwidth of the filter. This may require temperature stabilization of the transmitter and receiver.

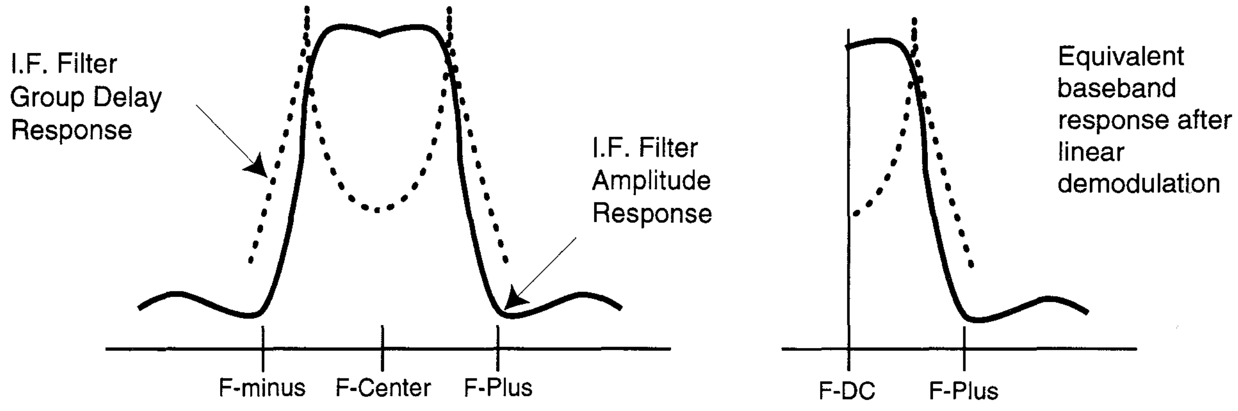


Figure 15-6 - transformation of IF filter response to baseband by linear demodulation

Compensation of the amplitude response at baseband is relatively straight-forward, and was described in the section on raised-cosine filters. The group delay (non-linear phase) can also usually be completely compensated by constructing an inverse-delay function with flat-amplitude response (an all-pass filter). Figure 15-7 is the schematic of an adjustable group-delay compensation all-pass filter, with limited compensation capability.

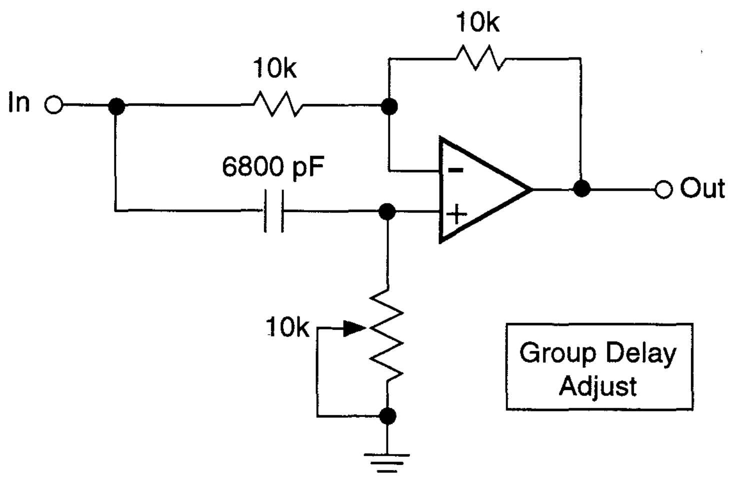


Figure 15-7 - Simple Group Delay compensation all-pass filter.

If the exact filter amplitude and delay characteristics are known, then a digital FIR filter can many times completely compensate the response, although it requires the complex (I, and Q) version of both the input signal and the tap coefficients. The resulting FIR filter is thus significantly slower than one that operates on only the real parts of the signal and tap coefficients.

Measurement of the IF filter characteristics is easiest at the IF frequency of the filter itself. This can be accomplished with a vector network analyzer, but such instruments are not readily available to radio amateurs. Alternatively, a linear modulator (mixer) and demodulator (mixer) can upconvert an analog signal to the IF frequency of the filter and back down to baseband. Then the characterization of the filter (assuming it is symmetrical about the center frequency) can be performed with more readily available equipment. Figure 15-8 is a test setup for measuring the filter amplitude and phase (and hence group delay) response. It relies on a two-channel oscilloscope to perform differential amplitude and phase measurements.

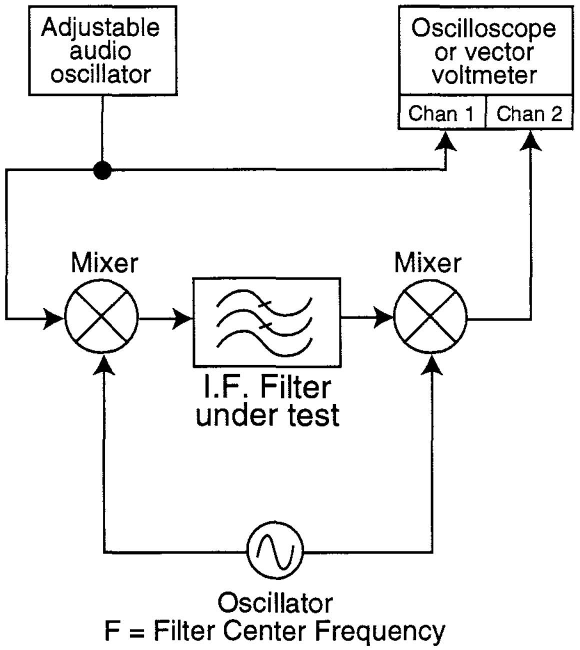

Adjustable audio oscillator
Oscilloscope or vector voltmeter
Chan 1
Chan 2
Mixer
Mixer
I.F. Filter under test
Oscillator
F = Filter Center Frequency

Figure 15-8 *- Test setup to measure IF filter amplitude, phase, and group delay response.*

The amplitude response of the filter is the ratio of the channel 2 amplitude to the channel 1 amplitude versus the frequency of the audio oscillator. The phase response is the relative phase delay of channel 2 with respect to the phase of channel 1 of the oscilloscope versus the frequency of the audio oscillator. A vector voltmeter may prove more accurate than a two-channel oscilloscope. The group delay is the derivative of the phase response versus the test frequency. The mixers should present a fixed, resistive termination to the audio oscillator power divider, or else errors may be introduced into the measurement. Also, the IF filter must be properly terminated on both the input and the output. It is useful to replace the IF filter with a resistive divider, and to measure flat amplitude and phase response versus audio oscillator frequency in order to assure that there are no systematic errors present in the test setup. The local oscillator should be carefully adjusted to the center frequency of the IF filter.

System Characterization

Many measurements utilize a known-flat transmitter or receiver response characteristic. However, if only a non-flat transmitter or receiver is available, then it is difficult to arrive at the characteristic of just the receiver or transmitter. In this case, it may be useful to construct a special wideband flat response transmitter in order to allow calibration of a receiver response. It is generally easier to build and verify the modulation bandwidth of an FSK oscillator, or of a linear modulation source (such as a mixer) than it is to build and verify a known-flat receiver.

Once the three key properties of the transmitter and the 4 key properties of the receiver are known, and the circuits have been designed to provide optimal performance, the characterization of the transmitter and receiver in tandem should be conducted. The key test is bit error rate vs. Eb/No, However, this test is difficult to accomplish easily because the very strong transmit signal makes the generation of a weak receive signal difficult to accomplish. The transmitter must either be placed into an exceptionally well-shielded box, preferably with internal power supply, or else the transmitter must be located remotely. The BER vs. Received Signal Level (RSL) can be measured. Equation (15-3) below will help relate RSL to Eb/No.

Additional practical tests to conduct are bit error rate vs. frequency offset between the transmitter and receiver, and bit error rate vs. carrier-to-interference ratio.

For HF communication systems, it is useful to conduct far more extensive modem measurements using the HF propagation simulator described previously.

A simple, but not always robust, technique of interconnecting a modem and a radio is to connect the radio using the microphone jack and the speaker jack. Great care is required when this is done. Most voice-radio equipment has a frequency response that is tailored to the human voice, and the phase response (and thus group delay) is generally *uncontrolled.* Careful characterization, independently, of the transmitter and receiver should be made. If the modem produces fairly narrowband audio emission centered in the speech audio frequency range, then the technique might prove acceptable. If the modem produces audio emissions that are comparable to the audio speech frequency range, then problems are likely. Additionally, the speaker is a poor place to tap off a received signal, since the resonance of the speaker with the radio cabinet can cause the response to change merely as the receiver is moved within the room or within its own enclosure. Of course, if the modem produces baseband signals (rather than modulated audio signals) then this type of interconnection is completely inappropriate.

Theoretical Eb/No vs. Received Signal Level (RSL)

If the receiver noise figure, received signal level, and baud rate are known, then the theoretical Eb/No can be calculated from:

$$\frac{E_b}{N_0} = R - F - 10\log\left(k\,T\,f_b\right) \qquad (15\text{-}3)$$

Where R = the received signal level in dBW (dB with respect to one watt), F is the noise figure of the receiver, in dB, k is Boltzmann's constant (1.38×10^{-23} Joule/Kelvin), T is the temperature in Kelvin (293K is room temperature), and fb is the baud rate of the data signal. The required receive signal level can be calculated if the needed *Eb/No* is known, by rearranging equation (15-3). Equation (15-3) assumes noise based solely on thermal noise, and is not applicable to low frequencies where band noise exceeds thermal noise.

Pseudo-Level Error Detection Margin

It may be useful to know, at times, the performance of a radio link. However, test equipment may not be accessible, or it may not be desired to take the link out of service to perform measurements. Additionally, usually two sites must be tested simultaneously, with a pattern generator setup at one end and a pattern detector setup at the other end, which could prove inconvenient. Another approximate method for establishing the noise margin non-obtrusively is to design the receiver slicer with several sets of thresholds, one based on the optimum decision point (used for data slicing) and others at sub-optimal slicing points to determine the noise margin. In a perfect "eye" for a two-level receiver, the optimum slicing level is halfway between the one and the zero peaks. If two other levels are placed, one halfway between the one and the optimum slicing level, and the other halfway between the zero and the optimum slicing level, then these two new levels will have 6 dB less noise margin than the optimum level. See Figure 15-9 for a diagram of these three slicing levels. There should be very few times when the decision level occurs in between the optimum level and the sub-optimum levels. Any occurrences in this region indicate less than 6 dB of margin on the link, and a likelihood of poor performance on the link. Of course, the pseudo-levels can be set elsewhere to indicate margin of more or less than 6 dB. Figure 12-3 is a schematic of this type of device.

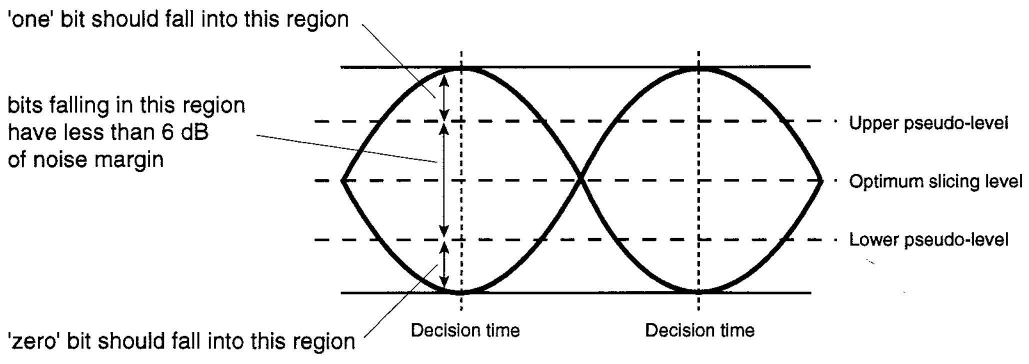

'one' bit should fall into this region
bits falling in this region have less than 6 dB of noise margin
'zero' bit should fall into this region
Upper pseudo-level
Optimum slicing level
Lower pseudo-level
Decision time
Decision time

***Figure 15-9** - eye diagram of pseudo-slicing levels for detection of events with less than 6 dB. of noise margin.*

Adjustment Aid - 'Click' box

A useful aid in setting up a modem is a device to audibly indicate when a bit error occurs. Therefore,adjustments of the modem parameters can be made while listening to this audible indicator. A simple method to accomplish this is to connect a pseudo-random pattern generator and pattern detector to a modem pair, and to route the error pulse from the pattern detector to a simple analog circuit that drives a speaker. Then, a single click will be heard for each bit error (neglecting error-multiplication by the descrambler). When the bit error rate increases, the clicks occur more quickly until at a high error rate they occur so frequently that the speaker sounds like noise. Figure 15-4 is the schematic of a simple circuit to generate clicks from a CMOS-level error pulse (positive-going or negative-going polarity of the pulse doesn't matter).

In testing a 9600 baud modem, for example, at a bit error rate of .001 (1.0 x 10^{-3}), there will be 9.6 error pulses per second, on average. At .0001 error rate (1.0 x 10^{-4}), there will be just under one pulse per second, on average. This circuit will not be very useful on a 300 baud modem because of the slow rate of errors. For example, at .001 (1.0 x 10^{-3}) error rate, at 300 baud, there will be one click about every 3 seconds; at an error rate of .0001 (1.0 x 10^{-4}), there will be one click about every 30 seconds, on average. The clicking detector is more useful on higher-speed data links where a single error bit occurs more frequently at a low error rate as compared to a lower data rate modem.

The click detector can be combined with the pseudo-level detector of the previous example to build an audible detector of poor link margin. In this case the click would be triggered upon the receipt of a bit in the central region of the eye, rather than upon the receipt of an error. This may prove useful when trying to adjust the link or modem for a lower error rate than is feasible when just monitoring the error count. Figure 15-10 illustrates a simple circuit that is handy for this test.

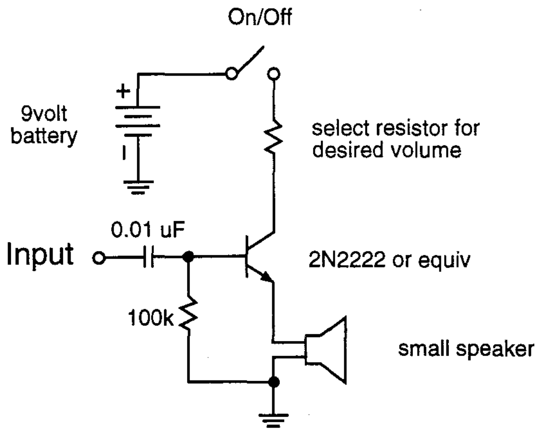


***Figure 15-10** - circuit to generate an audible 'click' in a small speaker upon detection of an event.*

Bit Error Rate Test Statistics

When conducting a bit error rate test, a good question to ask is how many errors are necessary to be recorded before the test is statistically valid. This question has no one right answer, but the question can be answered as: the error rate will be known within some confidence interval. For example, if we conduct a test of the bit error rate of a modem at a particular Eb/No ratio, that produces, say, one bit error every ten minutes, then it is painful to run the test for very long. How confident are we in the validity of the measurement? To answer this question, an examination of the error process is useful. A random sequence of events in time is generally referred to as a Poisson process. The process has a mean rate λ equal to n/t, where n is the number of events (such as the number of bit errors) and t is the time interval of the measurement. Let's assume that we measured 12 bit errors in 2 minutes. The arrival rate is thus $12/2 = 6$ per minute, and if we set up a test interval of one minute, then our estimate of the Poisson mean, μ is $(\lambda \bullet t)$ which is 6 errors. Then the question is simply one of finding the cumulative probability distribution of the Poisson distribution for $\mu = 6$, and determining the range that contains some percentage of the total distribution. For this example, table 15-2 lists the cumulative distribution of Poisson ($\mu = 6$) against the expected number of events, r.

r	Poisson(6,r)
0	0.002478752
1	0.017351265
2	0.061968804
3	0.151203883
4	0.2850565
5	0.445679641
6	0.606302782
7	0.74397976
8	0.847237494
9	0.916075983
10	0.957379076
11	0.979908036
12	0.991172516

Table 15-2 *- Cumulative Poisson distribution for a mean of 6*

Examining the table for r=3 and for r=8 shows that we could expect the number of errors in one minute to be between 3 and 8 for (0.847 - 0.151) = 69.6% of the time. Further, examining the table for r=1 and r=12, shows that we would expect the number of errors in one minute to be between 1 and 12 for (0.991 - 0.017) = 97.4% of the time. Thus, in the experiment with 6 errors in one minute, it can be stated that the bit error rate is between 3 per minute and 8 per minute with 69.6% confidence, and that the bit error rate is between 1 per minute and 12 per minute with 97.4% confidence. It is practical to use a confidence interval of 95% for BER tests. As the number of bit errors per time interval increases, the range of the estimate compared to the mean is reduced. For example, if the test produced 50 bit errors within some interval, then the error rate would be between 34 errors per interval and 67 errors per interval, with 98% confidence. Many texts use an approximation of the normal distribution to the Poisson distribution for values of the mean, μ, of 10 or more. Computer tools make the calculation of the Poisson distribution fairly straightforward for a mean up to about 100. For example, Excel 5.0 contains a built-in statistical function (=POISSON()) to compute the cumulative Poisson error distribution. Above 100, the numeric range of the computation may be exceeded, and the use of the normal distribution as an approximation is easier and quite accurate. The =NORMDIST() function calculates the normal cumulative distribution.

To use the normal distribution, the most accurate results are when:

$$x = r + 0.5, \quad mean = \mu, \quad and \quad std.dev = \sqrt{r} \qquad (15\text{-}4)$$

Table 15-3 lists the confidence intervals near 95% for various values of the mean, μ. To read table 15-3, assume that 6 errors were measured in a given period of time, then there is about 95% confidence that the real error rate is between 1 and 11 errors per unit time, inclusive.

Number of errors measured	Lowest likely number of errors	Largest likely number of errors	Approximate Confidence Interval
3	0	6	90%
4	0	8	95%
5	1	10	95%
6	1	11	95%
7	2	12	95%
8	3	14	95%
9	3	15	95%
10	4	16	95%
20	11	29	95%
50	36	64	95%
100	80	120	95%
1000	938	1062	95%
10000	9804	10196	95%

Table 15-3 *- approximate 90% to 95% confidence intervals for Poisson distribution of various mean values, μ. (Normal distribution approximation for μ = 1000, and 10000).*

To answer the question, suppose that the bit error rate of a modem was measured as 3 errors in 1 minute at a data rate of 9600 baud. The bit error rate is:

$$\frac{3}{1 \bullet 60 \bullet 9600} = 5.2 \bullet 10^{-6} \qquad (15\text{-}5)$$

and the error rate can be estimated to be between zero and $1.04 * 10^{-5}$ with about 90% confidence.

A good text on the derivation and application of Poisson statistics can be found in Cox and Lewis (1966).

Chapter 15 - Reference

Cox, D. and Lewis, John. (1966). The Statistical Analysis of Series of Events. Wiley and Sons, Inc.

Appendix

A

Some Useful Probability Distributions

Throughout the study of modems, we constantly refer to the probability of an event (such as the probability of a bit error). Much of the design concepts rely heavily on a few basic concepts in probability and distributions. This appendix summarizes some useful concepts.

A.1 Probability of Events.

If a single event has a probability, p, of occurring, then the probability of the event not occurring, q, is

$$q = 1 - p \tag{A.1}$$

Given two independent events, each with the same probability of occurrence, p, then the probability of occurrence of both events is

$$P(2) = p^2 \tag{A.2}$$

and the probability that neither event will occur is

$$P(0) = q^2 = (1-p)^2 \tag{A.3}$$

A.2 Histograms, Probability Density, Cumulative Density

For each group of trials of size N, we can calculate the probability of exactly n outcomes for each and every value of n. For example, assuming N=10 (i.e., we flip a coin ten times) then we can calculate the probability $P(0)$, which would be the probability that we got no heads, and 10 tails. We then calculate $P(1)$, the probability of exactly 1 head and 9 tails, on through $P(10)$, the probability that we got exactly no heads, and 10 tails. This graph is known as a histogram, figure A.1.

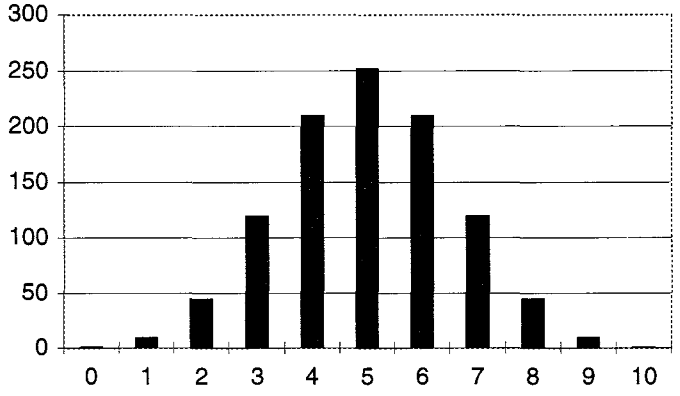


***Figure A.1** - A Histogram -The number of outcomes of exactly 'n' heads in 10 flips of a coin.*

From examining this chart, we see that there is one way to get zero-heads, 10-ways to get exactly one head, 45 ways to get exactly 2-heads, and 252 ways to get exactly 5-heads. If we add up all of the different ways to get outcomes, there are 2^{10}, or 1024 outcomes possible. We can then state that the probability of getting exactly 5 heads is 252 outcomes / 1024 possible outcomes, or 24.6%. If we re-scale the y-axis of the above chart by dividing by 1024 (the number of possible outcomes), then the resultant chart shows the probability of any particular outcome. This type of chart is known as a graph of the Probability Density Function (PDF), figure A.2.

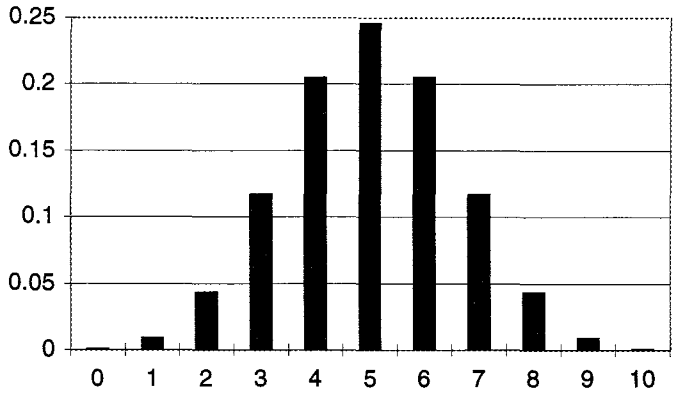


***Figure A.2** - Probability Density Function (PDF) of exactly 'n' heads in 10 flips of a coin.*

If instead of plotting the probability of each individual outcome, we instead plot the probability of *n or less* outcomes for each value of *n* (i.e., the accumulated probability as we move along the x-axis), then we will generate what is known as a graph of the Cumulative Density Function (CDF), figure A.3.

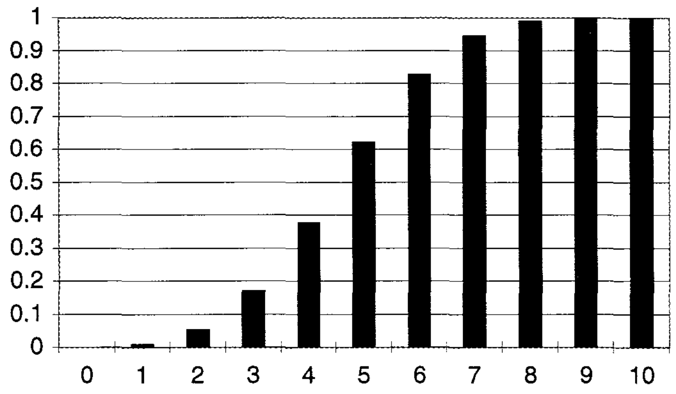


Figure A.3 *- Cumulative Density Function (CDF) of 'n' or less heads in 10 flips of a coin*

For large values of N, we will usually dispense with using the vertical histogram-bars, and instead use a smooth line to plot the function.

A.3 Uniform Distribution

A uniform distribution is one where the PDF is the same for all outcomes. For example, suppose we make ten pieces of paper, write the number 1 on one piece, the number two on another piece, and so on, through writing the number 10 on the last piece. Then, we put all the pieces into a bowl. If we shake up the bowl, and draw one at random, then the probability that we will draw the piece with the number 4 on it is 0.1 (one out of ten chance). The probability of drawing any particular piece of paper in our one trial is exactly the same, namely 0.1. This is known as a uniform distribution. In many programming languages, such as BASIC and C, a random number generator is provided, and the probability of it returning any one number is the same as the probability of it returning any other single number. This generator has a uniform distribution, see figure A.4.

Probability density for continuous real numbers can be converted to the probability of occurrence over a range of variables by multiplying the density by a very small range x, called Δx. Thus, $p = \text{pdf}(x)\,\Delta x$ as Δx becomes small. The probability thus calculated becomes accurate.

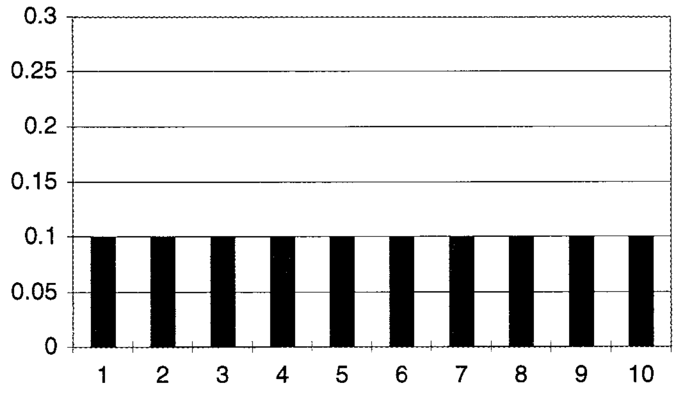


***Figure A.4** - Probability Density Function (PDF) of a Uniform Distribution*

A.4 Binomial Distribution

We will define $n!$ as n-factorial, such that

$$n! = n(n-1)(n-2)(n-3)\ldots(2)(1) \tag{A.4}$$

So, for example, $5! = 5 * 4 * 3 * 2 * 1 = 120$. We define $1! = 1$, and $0! = 1$ also.

Then, given a number, N, of independent random events each with a probability, p, the probability of exactly n outcomes out of the N events is

$$P(n) = \left(\frac{N!}{n!(N-n)!}\right) p^n q^{N-n} \tag{A.5}$$

This is known as a binomial distribution. When $p = q = 0.5$ (such as flipping a coin, or the probability that a completely random bit is a one), then we can simplify the above expression to

$$P(n) = \left(\frac{N!}{n!(N-n)!}\right)\left(\frac{1}{2}\right)^N \tag{A.6}$$

Flipping a coin n times is accurately described by (A.6), and so we can say that tossing a coin several times results in a binomial distribution. We can also say that receiving a completely random sequence of bits also results in a binomial distribution when we count how many "one" bits and how many "zero" bits were received. The histogram, PDF, and CDF charts in the previous section are the graphs of a binomial distribution of 10 independent events, each with a probability of 0.5, such as ten coin flips.

A.5 Gaussian Distribution

The binomial distribution is an exact count of the occurrence of *n* events. However, when *N*, the size of the trial (the number of coin flips) becomes large, then the binomial distribution becomes difficult to calculate, because *N*! becomes an extremely large number. It can be shown that as *N* becomes large, that the above formula can be converted to an approximation that is accurate for the region of the curve near the center of the x-axis, but it is not guaranteed to be accurate near the edges of the curve (i.e., it is not as accurate for 0-heads, or the 10-heads case). This approximation, which is far easier to compute for large *N*, is

$$P(n) = \frac{1}{\sqrt{2\pi Npq}} e^{\frac{-(n-N)^2}{2Npq}} \qquad \text{(A.7)}$$

This distribution is known as a gaussian distribution, and occurs very commonly whenever large numbers of events occur. It is useful to define two new variables, μ or mu, the mean value of a distribution (the average), and σ or sigma, the standard deviation of the distribution. We can express a probability distribution in terms of these two new variables, as

$$P(n) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(n-\mu)^2}{2\sigma^2}} \qquad \text{(A.8)}$$

This is known as the Standard or Normal distribution, and is widely used in statistics of all kinds. Both forms (A.7 or A.8) are alternatively useful in many calculations of modems. The PDF of the Gaussian distribution closely resembles that of the binomial distribution, since it can be a very good approximation of it. A good example is random white noise, which has a gaussian distribution, and usually we will choose a mean value of zero (no DC offset) and a standard deviation of one (which is the RMS, or equivalent power) of the noise signal.

A.6 Poisson Distribution

The gaussian approximation is very good except at the "tails" of the curve. In the case where we want to compute some infrequently occurring events, we need a different approximation to the binomial distribution. One distribution that is accurate at the tails (i.e., n much smaller than N), provided that the probability of an event p is small, is the Poisson distribution, where

$$P(n) = \frac{\lambda^n}{n!} e^{-\lambda} \tag{A.9}$$

$$\text{given} \quad \lambda \equiv Np \tag{A.10}$$

Thus, λ or lambda is defined as mean of the distribution (mean of n). Say, for example, the probability of a bit error is 0.006 (six in one thousand). We would like to estimate how many errors we are going to get in 1000 bits. Obviously, the mean number is .006 * 1000 or 6 errors. Sometimes, we will get 5 errors, or 7 errors, or 10 errors, or even zero errors. The Poisson distribution is easier to calculate approximation for the binomial distribution that would be accurate in this case (since n (6) is much less than N (1000), and the probability of occurrence is small (0.006). The probability density function for the number of bit errors we expect to see in this block of 1000 bits is depicted in the figure below, the probability density function of a Poisson distribution of mean=6. We expect to see 6 bit errors 16% of the time, and no bit errors 0.2% of the time.

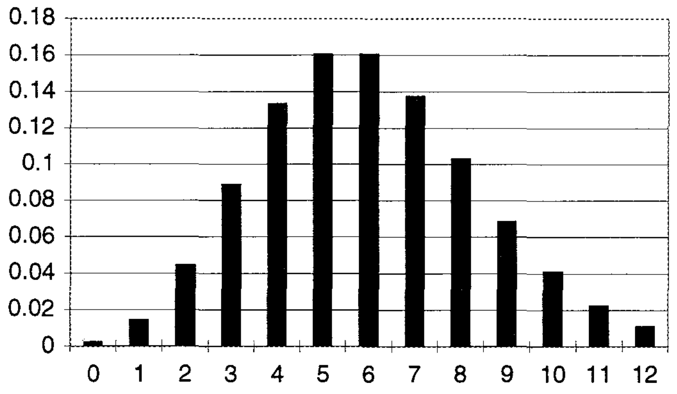


***Figure A.5** - Probability Density Function of a Poisson Distribution with mean = 6*

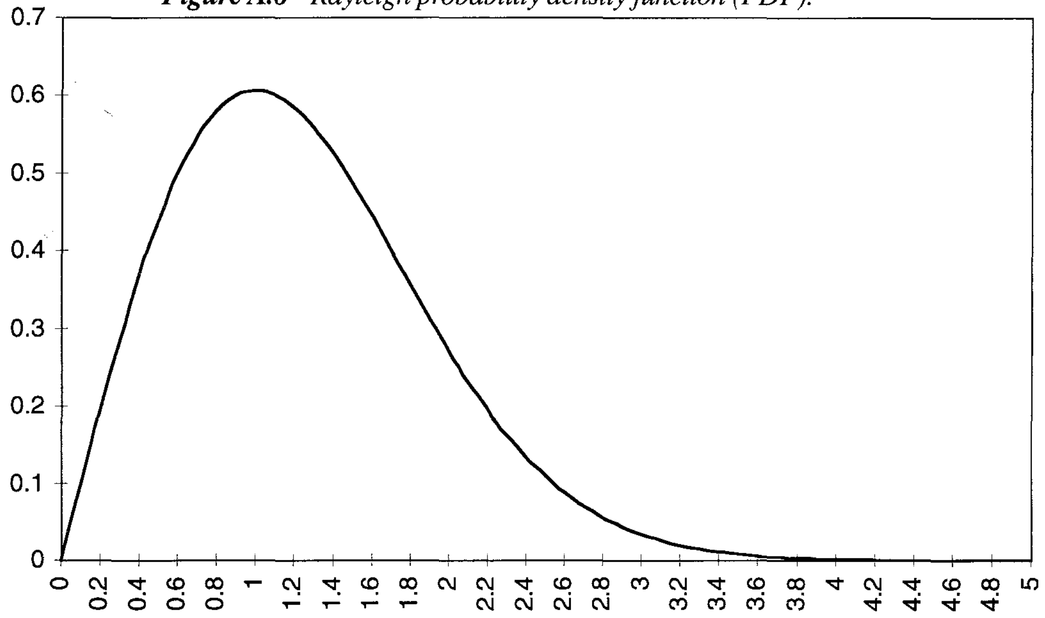
A.7 Rayleigh Distribution

One distribution that is very common in the study of fading signals (multipath) is the Rayleigh distribution. It is generated when the magnitude of a distribution of complex independent gaussian numbers is examined. Since the magnitude can never be less than zero, the distribution is non-symmetrical and becomes small at zero. The Rayleigh distribution is described by the following equation:

$$P(R) = \frac{R}{S} e^{-\left(\frac{R^2}{2S}\right)} \qquad \text{(A.11)}$$

Where S is the expected value, and R is the actual value. This distribution is shown in figure A.6.

***Figure A.6** - Rayleigh probability density function (PDF).*



Appendix

B

Pseudo-Random Binary Sequence (PRBS) Generators

Pseudo-Random Binary Sequences (PRBS), and PRBS generators are very useful in the design and test of data communications equipment. This is because they have many desirable statistical properties.

B.1 Properties of a PRBS

Some of the properties of a PRBS sequence are:

Approximate DC-balance: the sequence has a single more one than zero, which for even small values of n results in small or negligible DC offset.

Maximal-length: the sequence is as long as possible without repeating. This generates significant low-frequency content. The length of the sequence is $2^k - 1$ where k is the number of bits in the shift register.

Almost ideal auto-correlation properties: the sequence does not match itself well when shifted in time. The auto-correlation sequence closely matches that of white noise, and so the PRBS generator may be used as a white noise generator with good results.

Self-synchronizing: a simple circuit can quickly and easily synchronize to the sequence.

B.2 Construction of the PRBS Generator.

A PRBS generator is formed by using a shift register and some feedback taps. The output from the tap(s) is exclusive-or'd and fed back to the input of the shift register. Figure B.1 is a schematic of a 5-stage (k=5) PRBS generator. We can see that taps 2 and 5 have been fed back to the input of the shift register. The output can be taken from any of the flip flops. It must be assured that the all-zeros sequence (all 5 flip flops are zero) never occurs, since the generator will never escape this sequence.

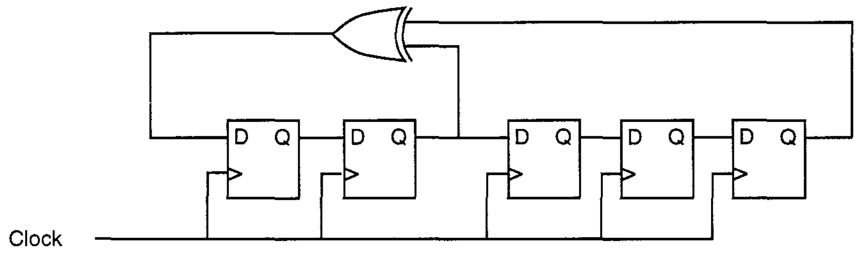


Figure B.1 *- 5-stage PRBS generator with feedback taps 2,5.*
This generator produces a 31-bit sequence, consisting of 16-ones and 15-zeros.

Table B.1 lists PRBS feedback taps for shift registers from 2 to 34 bits in length. It is derived from Peterson and Weldon (1972). Although, there are actually many possible feedback tap positions for each generator, only one possible arrangement is shown.

Pseudo-Random Sequence Generators		
Number of stages	**Sequence Length**	**Feed-back taps**
2	3	1,2
3	7	1,3
4	15	1,4
5	31	2,5
6	63	1,6
7	127	3,7
8	255	2,3,4,8
9	511	5,9
10	1,023	3,10
11	2,047	2,11
12	4,095	1,4,6,12
13	8,191	1,3,4,13
14	16,383	1,6,10,14
15	32,767	1,15
16	65,535	1,3,12,16
17	131,071	3,17
18	262,143	7,18
19	524,287	1,2,5,19
20	1,048,575	3,20
21	2,097,151	2,21
22	4,194,303	1,22
23	8,388,607	5,23
24	16,777,215	1,2,7,24
25	33,554,431	3,25
26	67,108,863	1,2,6,26
27	134,217,727	1,2,5,27
28	268,435,455	3,28
29	536,870,911	3,29
30	1,073,741,823	1,2,23,30
31	2,147,483,647	3,31
32	4,294,967,295	1,2,22,32
33	8,589,934,591	13,33
34	17,179,869,183	1,2,27,34

Table B.1 *- Pseudo-Random Sequence Generators*

B.3 Autocorrelation Properties

The periodic autocorrelation value, ϕ, as a function of the shift, s, of any PRBS sequence is:

$\phi(s) = n$ for $s = 0$ (i.e., when the sequence is perfectly aligned with itself)

$\phi(s) = -1$ for s = any other value (when the sequence is offset from itself)

A calculation of the autocorrelation function for n=3 (a 7 bit sequence) illustrates the above property.

0011101	The original 7-bit PRBS sequence
0111010	Shifted forward by one bit time (circular shift)
ADAADDD	3 agreements, 4 disagreements = -1 correlation
0011101	The original 7-bit PRBS sequence
1110100	Shifted forward by two bit times (circular shift)
DDADAAD	3 agreements, 4 disagreements = -1 correlation
0011101	The original 7-bit PRBS sequence
1101001	Shifted forward by three bit times (circular shift)
DDDADAA	3 agreements, 4 disagreements = -1 correlation
0011101	The original 7-bit PRBS sequence
1010011	Shifted forward by four bit times (circular shift)
DAADDDA	3 agreements, 4 disagreements = -1 correlation
0011101	The original 7-bit PRBS sequence
0100111	Shifted forward by five bit times (circular shift)
ADDDADA	3 agreements, 4 disagreements = -1 correlation
0011101	The original 7-bit PRBS sequence
1001110	Shifted forward by six bit times (circular shift)
DADAADD	3 agreements, 4 disagreements = -1 correlation
0011101	The original 7-bit PRBS sequence
0011101	Shifted forward by seven bit times (circular shift)
AAAAAAA	7 agreements, 0 disagreements = +7 correlation

The autocorrelation of the 7-bit sequence is diagrammed in figure B.2. Note how the periodicity of the sequence is easily measured by the autocorrelation.

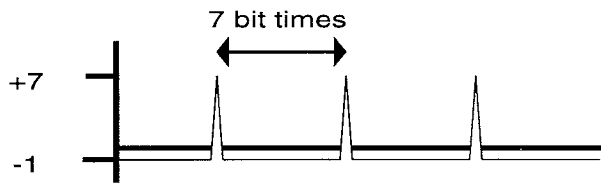


Figure B.2 *- Autocorrelation of a 7-bit PRBS sequence, the peaks occur every 7 bit times.*

B.4 Self Synchronization

It is simple to transmit and receive the PRBS sequence and to verify the accuracy of the received copy. The received detector will achieve synchronization in *n* bit-times. The use of a PRBS sequence for testing equipment is very common. It is simple to construct the generator, as per figure B.1. The circuit in figure B.3 will synchronize to the sequence. The output of the exclusive-or gate will pulse when the input sequence does not match the PRBS, thus it will count bit errors. Note that since a wrong bit participates in the error detect output several times, the output error rate will be an integer multiple of the true bit error rate. For the *n*=5 case, with two feedback taps, the error multiplication will be 3 times.

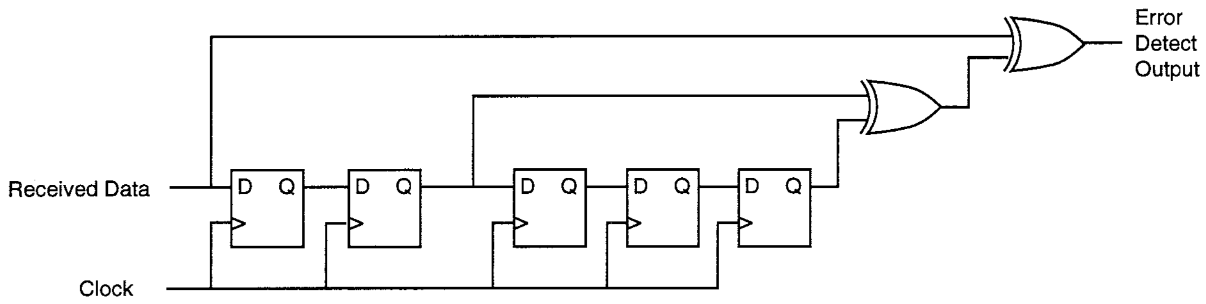


Figure B.3 *- PRBS receiver and error detector circuit*

Essentially what happens is that instead of feeding back the output to the shift register input, we compare it against the received data bit instead. If what we would have fed back agrees with the received bit, then no error has occurred, and the receive shift register gets loaded with the new correct value of the received sequence. If a bit error occurred, and the received bit is wrong, then it will show up in the error detect output three times: first upon entering the shift register, then at tap 2 and again at tap 5.

B.5 Self Synchronous Scrambler and Descrambler

The above PRBS generator and receiver can instead be used as a data scrambler and self-synchronizing descrambler. This is accomplished by injecting the data into the scrambler and altering the feedback. The error detector becomes a comparison of what should have been produced by the sequence and what actually arrived. This is the originally transmitted sequence. Figure B.4 is the schematic of the scrambler, and figure B.3 is a schematic of the descrambler.

Another way to visualize the operation of the scrambler and descrambler is to look at the mathematical operation performed on the data. The circuit in figure B.4 takes the input signal *Tx(t)* and generates the output

$$Rx(t) = \frac{Tx(t)}{1 + x^2 + x^5}$$

While the circuit of figure B.3 takes the input signal *Rx(t)* and produces

$$Descrambled\ signal = (1 + x^2 + x^5)\,Rx(t) = Tx(t)$$

It can be seen that the output of the scrambler plus descrambler pair produce the original input signal (but with an error multiplication of 3 times).

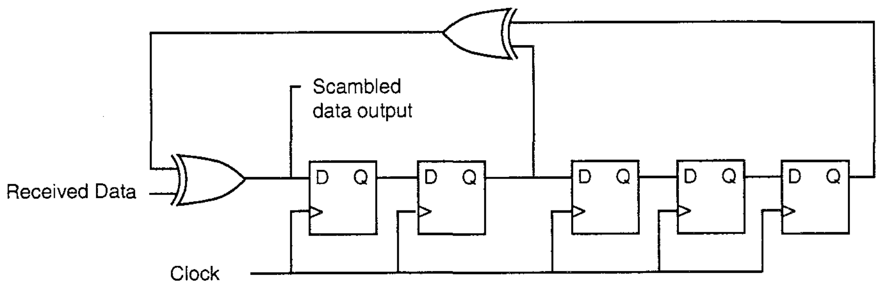


Figure B.4 *- PRBS scrambler for n=5. If the data input is tied to zero, then this becomes a PRBS generator.*

B.6 Binary Polynomial Arithmetic

The arithmetic operators on binary polynomials are defined as follows:

Addition: equivalent to the "exclusive-or" operation for like terms.
Multiplication: equivalent to the "and" operation for all cross terms.

In this context, "like terms" means the corresponding coefficients of the same power of x. The coefficient can be either zero or one, and if zero that particular term is not included in the equation. Thus, for example, the polynomial represented by:

$$1 \bullet x^0 + 1 \bullet x^1 + 0 \bullet x^2 + 1 \bullet x^3$$

would be written as:

$$1 + x^1 + x^3$$

since x^0 is one. The addition of two polynomials would be computed as follows:

$$(1 + x^2 + x^3) + (1 + x + x^2 + x^4) = x + x^3 + x^4$$

While the multiplication of those same two polynomials would be computed as follows:

$$(1 + x^2 + x^3)\,(1 + x + x^2 + x^4) =$$

$$(1 + x + x^2 + x^4) + (x^2 + x^3 + x^4 + x^6) + (x^3 + x^4 + x^5 + x^7) =$$

$$1 + x + x^4 + x^5 + x^6 + x^7$$

Appendix B - Reference

Peterson and Weldon. (1972). Error-Correcting Codes. MIT Press

Appendix

C

Discrete Fourier and Inverse Discrete Fourier Transforms & Complex Numbers

The Fourier and Inverse Fourier transforms are fundamental to the operation of digital filters and in the design and operation of DSP modems. This appendix reviews briefly the operation of the transforms, and some properties. The relationship between the frequency response of a filter, and its time-domain properties are described by the transform.

C.1 The Fourier Transform and the Impulse Response

In a nutshell, if the complex frequency response of a filter is known, then one can completely determine the impulse response, *and vice versa.* This duality is provided by the Fourier transform. That is, if we take the Fourier transform of the complex impulse response of a filter, we will get the complex frequency response of that filter. Conversely, if we take the inverse Fourier transform of the complex frequency response of a filter, we will get the complex impulse response of that filter. This can be stated in the following way:

Impulse Response $\Rightarrow$ Fourier Transform $\Rightarrow$ Frequency Response
Frequency Response $\Rightarrow$ Inverse Fourier Transform $\Rightarrow$ Impulse Response

Where the frequency response is complex, and the impulse response is also complex (A complex signal is one whose real and imaginary components are known. One way to acquire these components is to sample both the in-phase (I) and quadrature (Q) components).

In standard textbooks on this subject, little is said to give an intuitive understanding of how the Fourier transform accomplishes this magic. How this occurs will be done by dissecting the formula for the forward transform in sampled time space. This will show that the procedure that is taking place is actually very straightforward.

Prior to this, understanding of some simple properties of complex numbers must be explained, so that we can graphically construct what the Fourier transform is doing.

C.2 Complex Number Representation

In the use of complex numbers, electrical engineering texts usually refer to the imaginary operator as "j", while most other texts refer to it as "i". In either case, it is equal to the square root of -1. A common way to view a complex number is as a combination of real and imaginary parts. This is quite obvious in rectangular format. For example: 5+j6 represents a complex number whose real part is +5 and whose imaginary part is +6. To add two complex numbers, the real and imaginary parts are added separately:

$$(4 + j5) + (2 + j3) = (6 + j8)$$

To multiply two complex signals, the four possible combinations of the parts are multiplied:

$$(4 + j5)\,(2 + j3) = 4*2 + j5*2 + 4*j3 + j5*j3 = 8 + j10 + j12 + j*j15$$

Since j = sqrt(-1), then jj = -1, so the above simplifies to:

$$8 + j10 + j12 - 15 = (-7 + j22)$$

Another way to represent this number is in polar form, where the magnitude of the number is the length of the vector, and the angle is the angle of the vector with respect to the real positive axis. This yields the familiar equations for converting from rectangular to polar coordinates:

$$\textit{Magnitude} = \sqrt{\mathrm{Re}^2 + \mathrm{Im}^2} \tag{C.1}$$

$$\textit{Phase}\ \theta = \textit{Arctan}\left(\frac{\mathrm{Im}}{\mathrm{Re}}\right) \tag{C.2}$$

Where Re and Im are the real and imaginary parts of the number, respectively. In the computation of the Fourier transform, we will use another property of the complex numbers, Euler's rule:

$$e^{j\theta} = cos\theta + j\,sin\theta \tag{C.3}$$

Using this rule, it can be seen that if one takes an ever increasing complex angle and exponentiates it, then it describes a vector that travels around the unit complex circle — that is, a vector of length one that draws a circle in the complex plane. This circle represents simultaneously a cosine wave in the real part, and a sine wave in the imaginary part. This is important to understanding how the Fourier transform works.

Another view is that the projection of the rotating vector onto the real axis describes a cosine wave, while the projection of the rotating vector onto the imaginary axis describes a sine wave (shown in figure C.1).

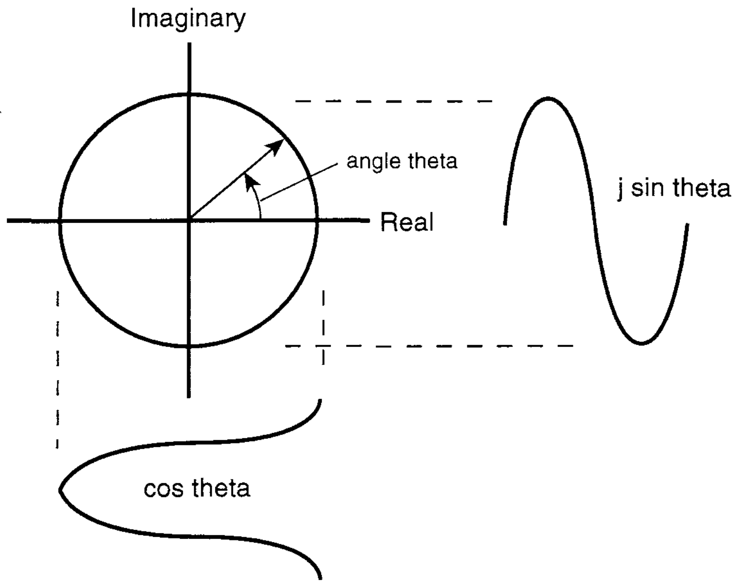


Figure C.1 *- Plot of exp(j*theta) as theta increases from zero*

C.3 Description of a Signal using Complex Notation

A single frequency (CW) signal, without any modulation, is described as:

$$a(t) = Ae^{j(\omega t + \varphi)} \tag{C.4}$$

where omega = 2 * π * frequency. *A* is the amplitude of the signal, omega (ω) is the frequency, in radians per second, and phi (φ) is a phase offset. The real part of this signal is the cosine part, yielding the more familiar real-valued description of a CW signal as:

$$a_{re}(t) = A\cos(\omega t + \varphi) \tag{C.5}$$

The value of the exponential notation is that the product of two signals, or the summation of two signals with unknown phase relationships, can be viewed either in-phase or in quadrature-phase with a certain economy of notation.

Assume that we wish to calculate the magnitude and phase of a signal *U* known only to be of the same frequency as *A*. We can generate two local signals, *Acos(wt)* and *Asin(wt)*. Let the amplitude of the local signal, *A*, equal one. Then if we independently multiply both local signals with the incoming unknown signal (and low-pass filter the results), we produce two outputs:

$$I = UA\cos(\omega t) \tag{C.6}$$

$$Q = UA\sin(\omega t) \tag{C.7}$$

Then *I* and *Q* are known as the *in-phase* and the *quadrature-phase* amplitude components of the unknown signal *U*, respectively. This is because the value *I* is maximized when *U* is phase-aligned with *A*, and the value *Q* is maximized when *U* is 90-degrees out of phase with *A*. These are sometimes also known as the *real* and *imaginary* amplitude components. The magnitude of the signal *U* with respect to *A* is:

$$Magnitude = \sqrt{I^2 + Q^2} \tag{C.8}$$

and the phase of the signal U with respect to the reference carrier A is:

$$Phase = Arctan\left(\frac{Q}{I}\right) \tag{C.9}$$

The arctangent calculation must be a 4-quadrant calculation (so the sign of I and Q are important). Figure 2 illustrates the development of the I- and Q- components of U.

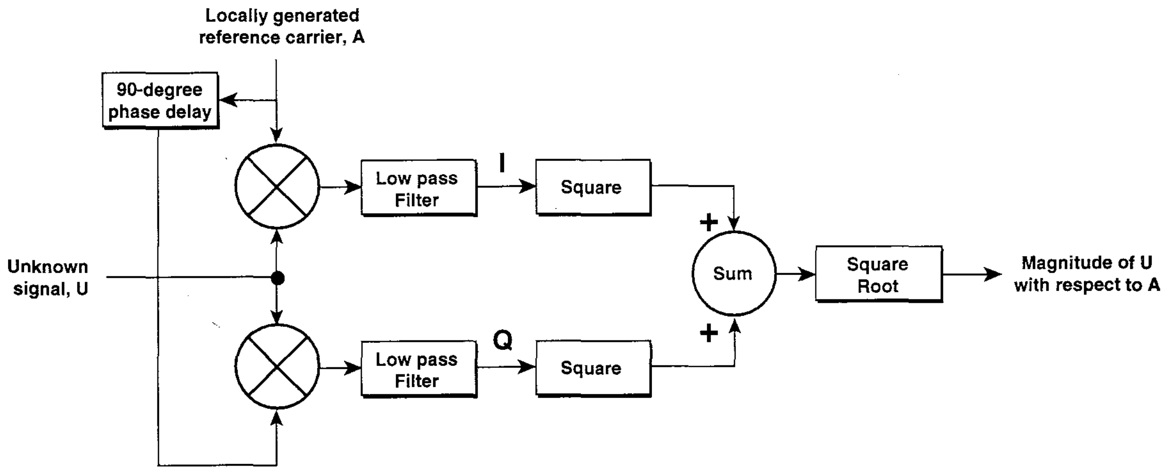


Figure C.2 *- generating the I and Q components of the signal U with respect to A. The magnitude of U can be computed from these two components.*

C.4 Complex Conjugate

One important notation of a complex number is the complex conjugate. This is usually written as an asterisk, for example F* would mean the complex conjugate of F. The complex conjugate of a number is equal to the real part plus the negative of the imaginary part. For example, the complex conjugate of 5+j6 would be 5-j6. The complex conjugate of 3-j4 would be 3+j4. The complex conjugate is important in the study of Fourier transforms because it naturally describes the power of a signal. The magnitude squared of a voltage is proportional to its power. The product of a number and its conjugate yields the magnitude squared. Assume that F = 3+j4. Then F* = 3-j4, and the product FF* is 25, which is the magnitude squared (real squared plus imaginary squared).

C.5 Discrete Fourier Transform

We will use the discrete Fourier transform since it may be simpler to visualize discrete values. We must remember that the Fourier transform is only defined for periodic sequences (those that repeat) with the period equal to *N*. The equations for the Discrete Fourier Transform (DFT) is:

$$F(k) = \sum_{n=0}^{N-1} x(n)\, e^{+j2\pi kn/N} \qquad \text{(C.10)}$$

Where *F(k)* is the *k*th Fourier coefficient, *x(n)* is the time sample at time n, and N is the number of coefficients we wish to have. Similarly, the Inverse Discrete Fourier Transform (IDFT) is:

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} F(k)\, e^{-j2\pi kn/N} \qquad \text{(C.11)}$$

Looking at the Forward transform, which converts a time-series to its frequency components, the expression $e^{+j2\pi kn/N}$ describes a vector that travels around the unit complex circle *k* times per computation. Take the case where *k*=0. This is the 0th Fourier coefficient, or the DC component. In this case, the exponential always evaluates to 1+j0, and so we are merely adding up all of the samples in the time series (waveform). The positive and negatives values of the waveform cancel some of each other, and the result is that the net DC component remains after the summation. When *k* = 1, the vector travels once around the unit circle as we sum all of the time components. We can break the travel around the circle into two parts, the real part and the imaginary part, and then sum them individually.

The real part is a cosine wave that completes one cycle of a cosine wave in one pass through the time samples. Thus, if the time signal of interest has any spectral components that match the frequency of one cosine wave, and those spectral components are in phase with the cosine wave, then those parts will always add together in phase, since we are multiplying the cosine wave with the time samples. So, any portion of the time samples that are at that frequency and that phase will synchronously multiply with the cosine wave, and they will add. Other waves that are multiples of the cosine wave, such as twice the frequency, or spectral energy of the same frequency but 90 degrees out of phase, will cancel out since half the time they will agree with the phase of the cosine wave. Half the time they will disagree, canceling. We can see that for k=2, the exponential will generate a cosine wave that completes two cycles, and thus will synchronously detect all energy in the time waveform at that frequency with the proper phase.

The imaginary part of the exponential is a sine wave that completes one cycle of a sine wave in one pass through the time samples when k=1. However, it is 90 degrees out of phase with the cosine wave above, so it will emphasize spectral energy at the frequency that happens to be correctly phase aligned with the sine wave. The same applies for higher frequency components of the waveform as it did with the previous paragraph.

Now, for each pass through the transform (for each value of k) we will generate one spectral component. It will have a real part (demodulated by the cosine wave) and an imaginary part (demodulated by the sine wave). We then add the two parts together, forming a complex number that represents the real and imaginary parts of the spectral component. What do the two parts of the spectral component tell us? The real part tells us the amplitude of the spectral component that is in-phase with our reference, and the imaginary part tells us the amplitude of the spectral component that is 90 degrees out of phase with the reference. We can, of course, convert this to magnitude and phase with the rectangular to polar equations.

To sum up the example, if we look at the 1st Fourier coefficient, the transform generates a cosine and a sine wave that complete one cycle (since it's the 1st coefficient). We multiply the point on the cosine wave by its corresponding point in the data for all points, and we sum the results. We do the same for the sine wave and the data. If the data has the same period and phase as the generated cosine wave, then they will both always be positive and negative at the same time. Thus, summation of all the components will yield a large positive result. A data signal at twice the frequency would agree in sign half of the time with the cosine wave, and disagree half the time. When we sum up the result they would cancel leaving zero output. If the data were exactly 180 degrees out of phase with the cosine wave, then the result of the multiplication would always be negative, and the resultant sum would be a large negative result. A large negative result means "lots of energy at that frequency, but 180-degrees out of phase". A large result in summing the sine wave component means "lots of energy at the frequency, but 90-degrees out of phase". A large sum in both the sine and the cosine cases means "a lot of energy at that frequency, but 45-degrees out of phase". The signs of the components and their amplitudes let us determine exactly the magnitude and phase of the component.

So, for each coefficient calculation (determining the next frequency component) we generate a cosine wave and a sine wave of exactly that same frequency as that component. Next, we multiply those waves with the time-sequence data for each sample, and then we sum all the results. This has the effect of integrating the time stream at exactly that frequency, and producing maximum output when the time data has components that match that particular frequency. Essentially we have a bank of filters, where once at each place we calculate a coefficient. Notice that we do not have to calculate all the coefficients if we chose not to, so we can implement some simple tone filters using this technique.

It should be noted that the DFT / IDFT are not the most efficient computation method to derive the Fourier transform. As can be seen by how we generate a complex exponential that travels around the unit circle repeatedly, we are generating a number of duplicate calculations. Another algorithm, called the Fast Fourier Transform (FFT), takes advantage of the duplications, and uses a clever approach to minimizing them, resulting in much faster computations. It achieves the same results, however. When using the FFT, we must compute all of the coefficients, and we must have a binary-number of coefficients (2,4,8,16,32,64 etc.) The DFT has neither of these restrictions, which can be useful occasionally. Both transforms yield the same final results (when one calculates the same number of coefficients).

C.6 Power Spectral Density (PSD)

In the analysis of random numbers the Fourier transform cannot tell us the spectrum, since a random number is by definition not periodic. However, the power vs. frequency can be computed from a random signal by taking the Inverse Fourier transform of its autocorrelation integral (see appendix D). The autocorrelation integral is roughly the correlation of a signal with its complex conjugate, yielding the magnitude squared as a function of time difference. Phase information is lost, so the resultant Fourier transform output is only real-valued, giving the power vs. frequency of the random signal.

Fourier transforms have many other interesting properties that have been well studied (Bracewell, 1978).

C.7 Symmetry Requirement for Frequency Response

When only real-valued samples are available, the Fourier and Inverse Fourier transforms do not have sufficient information to yield a spectrum throughout the range 0 to 1. In fact, the response from 0.5 to 1.0 is the mirror-image of the lower half frequency (0.0 to 0.5). If complex-valued samples are available, then the transform does not form a mirror-image. In transforming back and forth between time and frequency it is easy to forget the symmetry properties of the transform. In general, with real-valued input data the amplitude must be even-symmetric, and the phase must be odd-symmetric about 0 (or about 1). This means that if, for example, the sampled frequency response from 0 to 0.5 is available, then to generate the response from 0.5 to 1, the lower-half response is mirror-imaged onto the upper half response, except the *complex-conjugate of the response must be used.* That is — the amplitude is a mirror image, but the upper half phase is a mirror image of the negative of the lower half phase.

If the signal is viewed from -0.5 to +0.5, rather than from 0 to 1, then the symmetry conditions become visually clear, since the phase is smooth across the zero frequency point, and discontinuous across the 0.5 and -0.5 frequency points. This is illustrated in figure C.3.

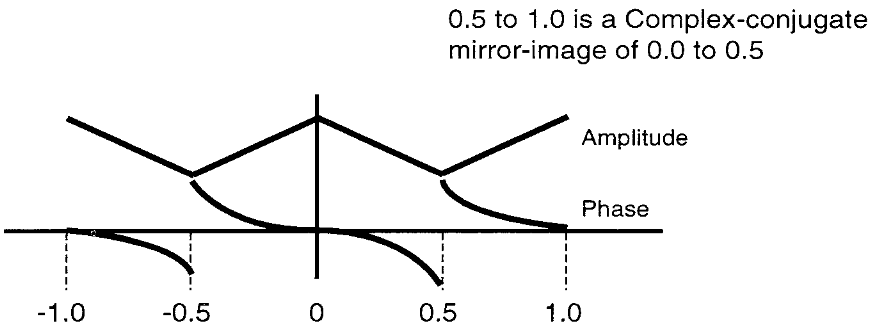


Figure C.3 *- Symmetry conditions for the Fourier transform in the sampled-frequency domain.*

Appendix C - Reference

Bracewell, R.N. (1978). The Fourier Transform and its Applications. 2nd Edition. McGraw-Hill.

Appendix

D Correlation, Convolution, and Laplace Notation for Filters

Correlation and Convolution are two time-domain operations that are extremely useful in the design and analysis of modems. Convolution is related to filtering in the frequency domain, and autocorrelation tells much about the spectral properties of a signal.

D.1 Convolution and Multiplication

One very useful duality illustrated by the Fourier transform is the duality between multiplication and convolution, which is extremely useful in the implementation of digital filters for modems. Simply stated: the effect of multiplication in the frequency domain is the same as the effect of convolution in the time domain. What does this mean? If we take two filters and cascade them, then we will achieve a filtration effect that is the product of the two filters in the frequency domain. That is, if one of the filters is a low pass filter and the other is a high-pass filter, then the product of the two would be a bandpass filter. We can compute the response of the two filters by multiplying the complex amplitude response of the first filter with the complex amplitude response of the second filter at each and every frequency of interest.

Conversely, we can achieve the same effect in the time domain. If we take the inverse Fourier transform of the first filter's complex response, which yields the impulse response of that filter, and if we then convolve that response with the impulse response of the second filter (which we computed by taking the inverse Fourier transform of the second filter complex response), then the resulting impulse response will be that of the two cascaded filters. We can then take the Fourier transform of the cascaded impulse response, and it will yield the frequency response of the combined filters.

The convolution operation has another name in the sampled-time world of digital filters and that is the Finite Impulse Response filter, or FIR filter for short. We can directly implement the impulse response as an FIR filter. Combining the frequency and time domains with convolution and multiplication yields a very powerful set of tools for solving filter design and implementation problems, and we will rely heavily upon them for both modeling and actual filter designs.

D.2 Convolution

The convolution of two signals in time can be calculated as

$$y(t) = \int_{-\infty}^{+\infty} h(\tau)\; x(t-\tau)\, d\tau \tag{D.1}$$

Where *x(t)* is the input to the filter at time *t*, *h(t)* is the impulse response of the filter at time *t*, and *y(t)* is the output of the filter at time *t*, while τ is the variable of integration. It turns out, of course, that we can transform this equation to sampled-time for use in digital filters. In that case, the limits of integration are finite, and extend to cover the significant portions of the impulse response. If too small a part of the impulse response is utilized, then the frequency response of the equivalent filter will be different, and errors will be introduced. Converting to sampled time, the above equation is written as:

$$y(t) = \sum_{k=-n}^{k=+n} h(k)\, x(t-k) \tag{D.2}$$

The graphical interpretation of this equation is easier to do in sampled time. In the above equation, *k* is the time index. Consider the impulse response *h(k)* = ...,0,0,0,0.5,1,1,0.5,0,0,0,... which is a low-pass filter with fairly poor roll-off. Also consider the input signal *x(t)* as ...,0,0,+1,0,-1,0,0,0,... which is a positive pulse followed by zero, then a negative pulse. Figures D.1, D.2, D.3, and D.4 illustrate the convolution of those two signals, and the resultant output *y(t)*.

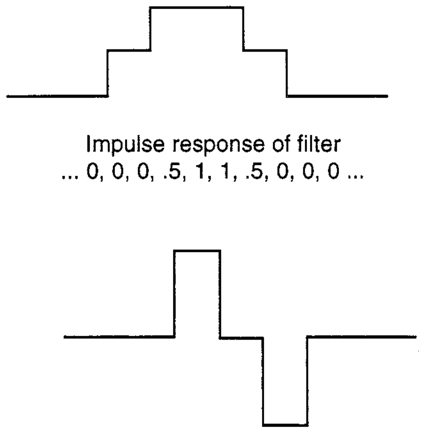
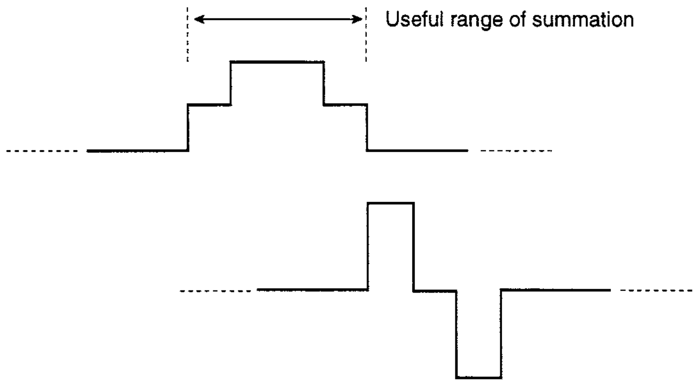
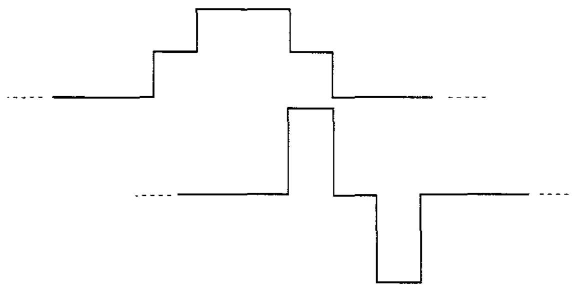


Figure D.1 *- Two signals to be convolved (filtered)*

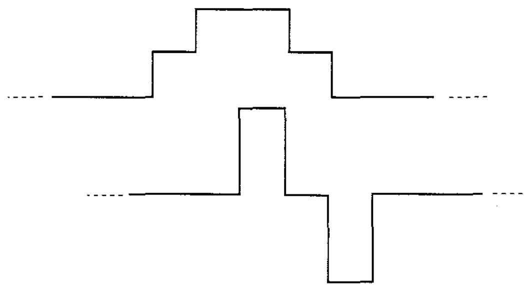


At time t=0, the summation is 0+0+0+0 = 0

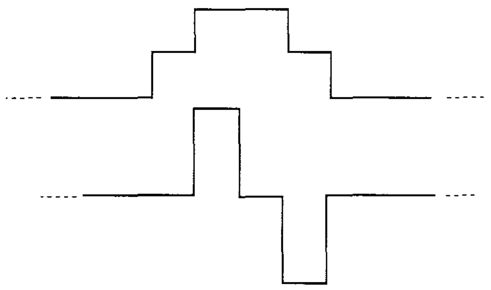
Figure D.2 *- at time t=0, there is no overlap of the two signals, so the output is zero.*



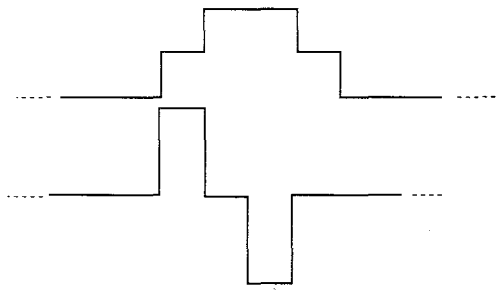
At time t=1, the summation is 0 + 0 + 0 + 0.5 = 0.5



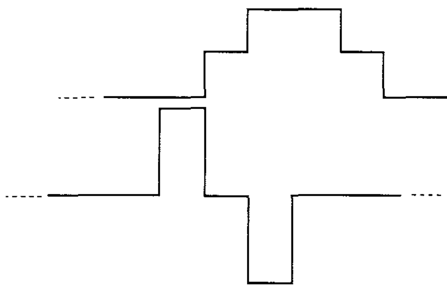
At time t=2, the summation is 0 + 0 + 1 + 0 = 1



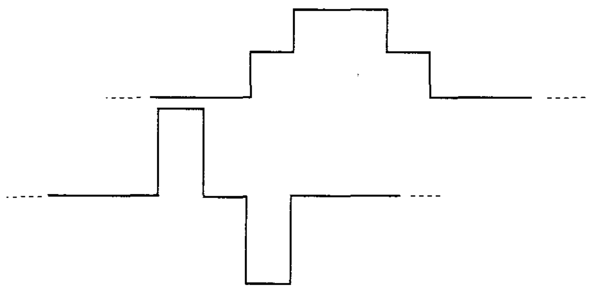
At time t=3, the summation is 0 + 1 + 0 - 0.5 = 0.5



At time t=4, the summation is 0.5 + 0 - 1 + 0 = -0.5



At time t=5, the summation is 0 - 1 + 0 + 0 = -1



At time t=6, the summation is -0.5 + 0 + 0 + 0 = -0.5

Figure D.3 *- calculating the filter response at times t=1,2,3,4,5, and 6. The summation theoretically occurs over all time, but since the two signals have only a finite time when both are non-zero, the summation interval can be limited to that range with no error.*

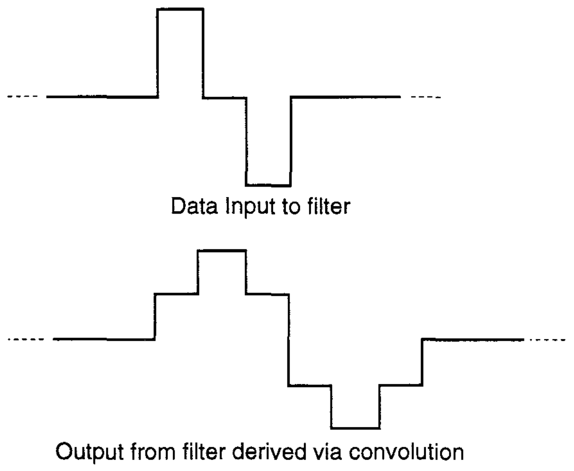


Figure D.4 *- Resultant output of the filter compared to the input*

The low-pass filtering is obvious as the width of the pulses has been increased, the slope is not as fast, and the output has been delayed from the input.

An FIR digital filter is based on the direct implementation of equation (D.2). Figure D.5 is a block diagram of a digital FIR filter algorithm.

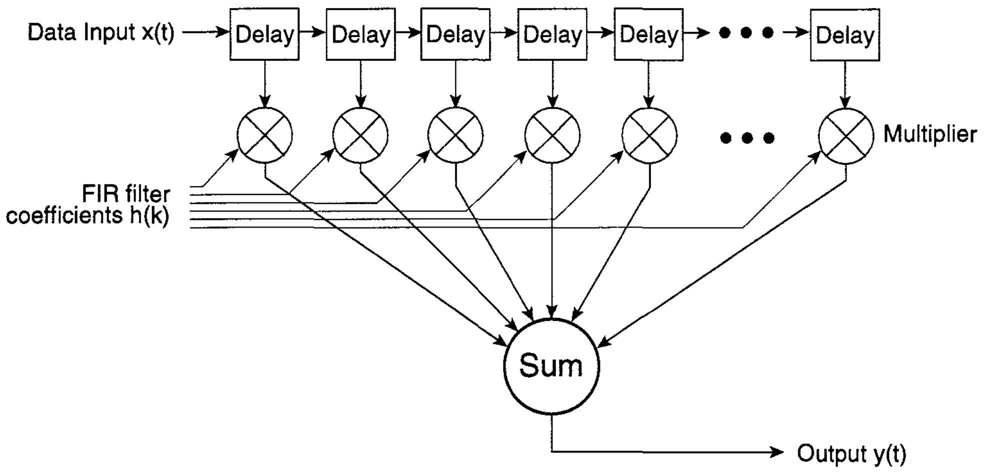


Figure D.5 *- Algorithm for implementation of digital FIR filter*

D.3 Frequency and Impulse Response

Armed with the knowledge of the Fourier transform and of the properties of convolution, we can solve the modem related problem: namely what is the impulse response of a filter, and how does the ringing of that filter affect the analog data waveform? We will first use our brick-wall filter with a cutoff of half of the bit rate, postulated as being useful. *The inverse Fourier transform of the frequency response of a filter yields the impulse response of that filter.* Remember that the data filter is periodic. This is not easily seen until we remember that we are dealing with a sampled system, and it is sampled at the symbol rate in baud, or some multiple thereof. Let's assume we are dealing with a 9600 bps binary (two-valued) eye pattern, and we use a 19200 Hz sampling frequency, and the filter cutoff is 4800 Hz. The filter response plus its alias (the portion from 0.5 to 1.0 of the baud rate) looks like that shown in figure D.6 below.

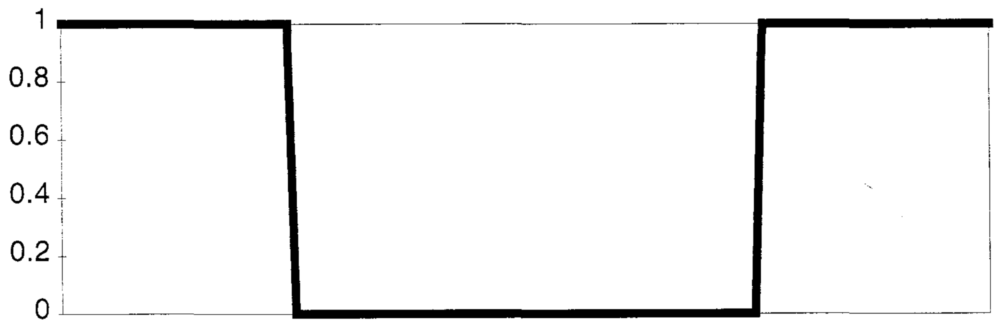


Figure D.6 *- Frequency response of a hypothetical filter*

This figure is the frequency response. It has a value of 1 from DC to 1/2 the baud rate, zero from 1/2 to 1-1/2 times the baud rate, and 1 from 1-1/2 to twice the baud rate. Alternatively, we can view this response as zero from minus baud rate to minus one-half baud rate, one from minus one-half the baud rate to plus one-half the baud rate, and zero from plus one half the baud rate to plus baud rate. Both descriptions are common and illustrate two different views of the same response.

Figure D.7 is the inverse Fourier transform of the preceding frequency response (Figure D.6). In the upper chart are the coefficients directly from the transform. In the lower figure the samples have been delayed and shifted by half of the total to put zero in the center of the chart — it's easier to interpret this way. This time response contains ringing that lasts very long, since this figure shows 64 bit periods in time. For complex signals (those with non-zero phase part, such as a Hilbert transform filter), when going from the impulse response to the frequency response (forward Fourier transform) the samples should be rotated back to the form shown in the top part of Figure D.7 before transforming in order to compute a real transform.

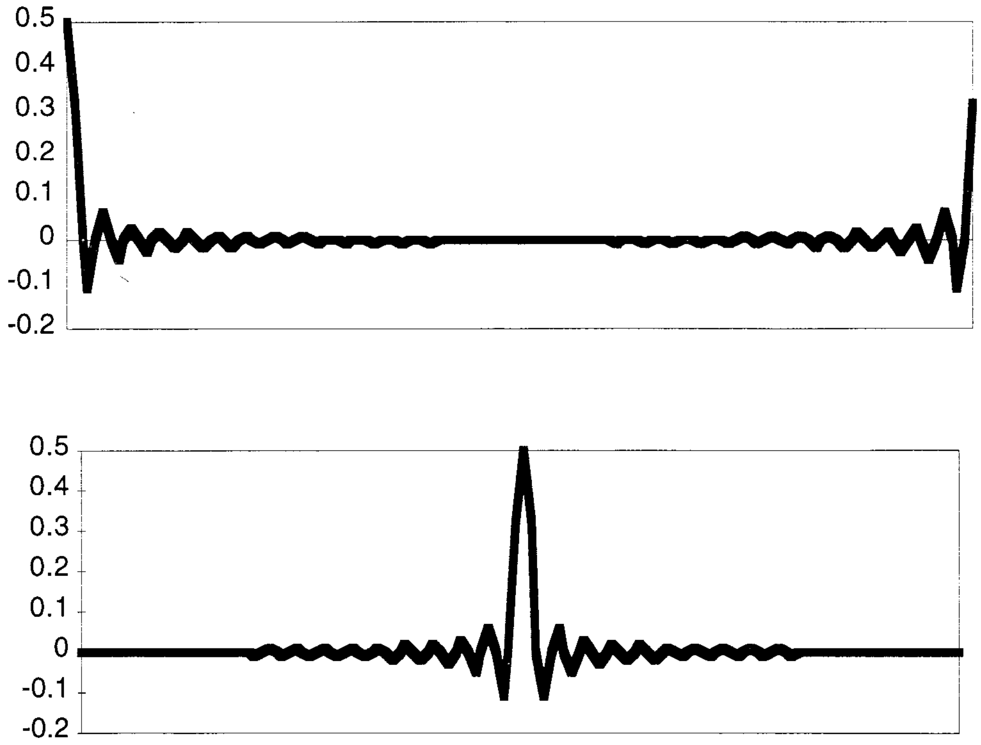


***Figure D.7** - Impulse response of filter in Figure D.6.*
Before and after rotation of samples to center the peak in the middle of the chart.

Figure D.8 is just the central portion of the previous figure magnified so that we can see the impulse response in the critical area, and can see the individual time ticks, which occur twice per data bit time (since we sampled at 19200 Hz.). This chart shows 10 bit periods of time (20 tick intervals).

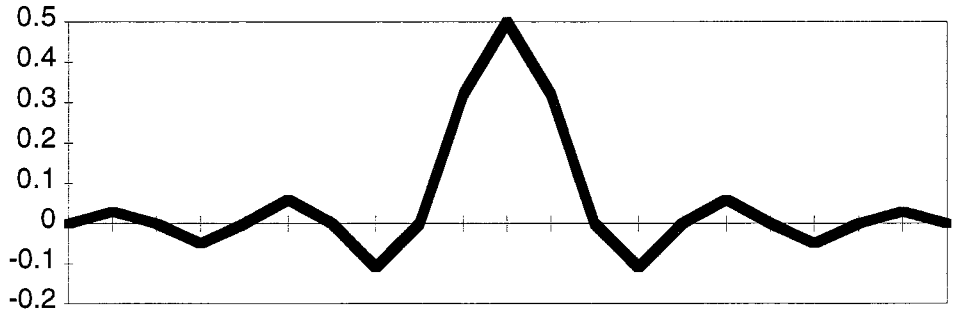


Figure D.8 - Magnification of central portion of figure D.6.

How do we read these plots? If we look at the peak of the impulse response, we will define that as the time of the desired data bit maximum. This is the center of the data bit where we desire to sample the bit (the instant in time when we clock the slicer flip-flop) since this represents the point of maximum eye opening. Two ticks later, we see that the curve crosses through zero. This corresponds to the time of the next data bit maximum. Two more ticks later again, the curve crosses back up through zero, representing the time of the center of the second data bit. In each case, the curve crosses through zero at exactly the time of the center of the next data bit. This is highly desirable, since it means that we can sample the desired data bit. Furthermore, if we sample it precisely at the tick-mark time, then all preceding data bits will have no net energy added to the current data bit. This is, in fact, the criteria for zero Inter symbol Interference. How did we achieve this lucky circumstance? It's due to the fact that we set the filter cutoff to exactly one half of the data bit rate, and the filter is exactly symmetric about this 1/2 frequency point. Any filter in which we do this will theoretically yield zero ISI.

Unfortunately, we notice that the filter attains some significant response during the odd-tick marks, that represent the data bits one half time period away. So, during the part of the eye pattern where we are one-half of the time away from the maximum eye opening, there is significant corruption of the waveform due to energy from many preceding data bits. These corruptions in fact significantly distort the eye pattern, but they do leave us with a perfect eye opening in the center of the bit at the best possible sampling time. Due to other distortions in the overall system response, we will find, however, that we are very sensitive to slight errors in timing and synchronization with the above response. What we would like is a filter with an impulse response where the oscillations of the response away from our desired data bit are fairly small, and also that the curve crosses through zero exactly at multiples of the data bit period corresponding to the best time to sample our desired bit (in the center of the eye).

D.4 Correlation

Correlation is a term that relates how closely two signals match each other. It is similar mathematically to convolution, the only difference is the sign of the time index in the expression. Thus, by changing how the variable *h()* is defined, from the impulse response of a filter to a reference waveform, the computation of correlation is very similar to that of convolution, where the impulse response is stored within the filter coefficients. In fact, the FIR filter structure of a DSP computation can perform the correlation function merely by reversing the direction of the time indexing. The continuous-time definition of correlation is:

$$c(t) = \int_{-\infty}^{+\infty} r(\tau)\ x(t+\tau)\, d\tau \qquad \text{(D.3)}$$

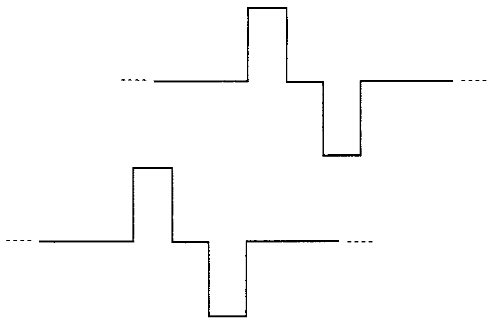
Where *c(t)* is the correlation function in time, and *r(τ)* is the reference waveform against which *x(t)* is being compared. This can be expressed in discrete-time form in a manner similar to convolution, with the expression:

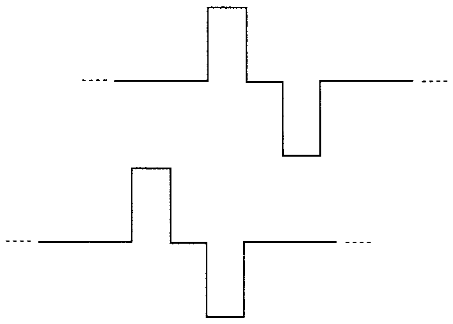
$$c(t) = \sum_{k=-n}^{k=+n} r(k)\, x(t+k) \qquad \text{(D.4)}$$

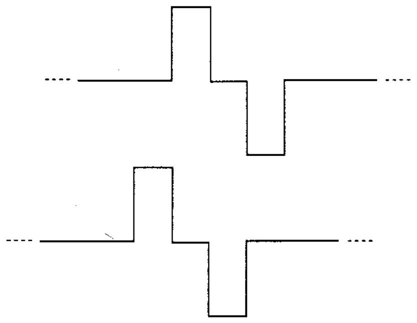
And, excepting the change of sign, and redefinition of variables it appears to be like the expression for convolution. For $t = 0$, then if $r(k) = x(k)$ for all k, then the value of $c(\ t = 0\ )$ is obviously maximized. This says that two signals match each other most closely when they are the same signal, and they are not shifted relatively to one another, an obvious statement. Additionally, if $r(k) = -x(k)$ for all k, then the value of $c(\ t = 0\ )$ has the most negative possible value, and the signals are strongly correlated, but of opposite sign. Correlation is a good detection method in modem design, with +correlation related to one data bit value, and -correlation related to the opposite data bit value in a binary transmission system. In matched filter design, correlation is the technique used to optimize the received signal.

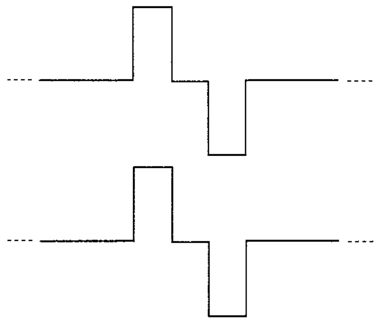
D.5 Cross Correlation

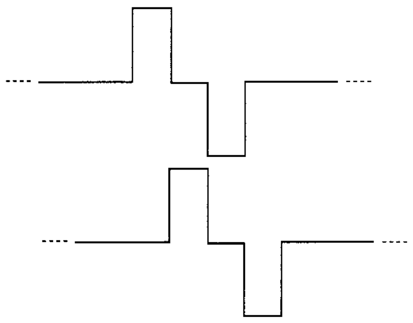
Cross correlation is the computation of the correlation function between two different signals. In this case, r() and x() are two different signals being compared. In the above examples for convolution, if the impulse response is replaced with the stored reference waveform, and if the direction of movement of the input data signal occurs from left to right, then the same graphical technique performs the cross correlation function. Figure D.9 illustrates the cross-correlation of two signals that strongly match each other.

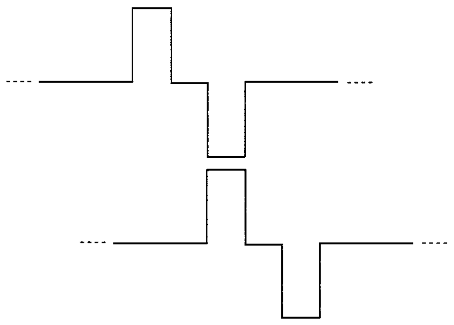

At time t=0, the cross-correlation is 0 + 0 + 0 = 0


At time t=1, the cross-correlation is -1 + 0 + 0 = -1


At time t=2, the cross-correlation is 0 + 0 + 0 = 0


At time t=3, the cross-correlation is + 1 + 0 + 1 = 2


At time t=4, the cross-correlation is 0 + 0 + 0 = 0


At time t=5, the cross-correlation is 0 + 0 - 1 = -1

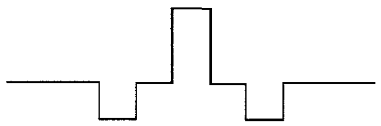

Resultant cross-correlation of the two signals

Figure D.9 *- graphical cross correlation of two signals*

D.6 Autocorrelation

Autocorrelation is essentially the correlation of a signal with itself. When complex signals are used, autocorrelation is really the cross correlation of a signal with its complex-conjugate. For a real-valued signal this is the same as the signal itself. Recall that multiplication of a signal with its complex conjugate provides the magnitude squared of the signal, which is proportional to the power of the signal at that instant in time. In practice, a signal has to be recorded so that the autocorrelation can be computed after the fact. Autocorrelation has many uses and describes several useful properties of a signal. For example, a good maximal-length pseudo random signal or a good frame-alignment word would be characterized by autocorrelation integrals similar to figure D.10. Two properties that autocorrelation provides are 1) an estimate of the periodicity of a signal, and 2) an estimate of how random a signal is. The expression for autocorrelation is:

$$\phi < t > = \int_{-\infty}^{+\infty} r(t)\ r(t+\tau)\, d\tau \tag{D.5}$$

Where phi is the autocorrelation versus time, and r is the signal. This form of autocorrelation measures the total energy, and is most useful with transient (as opposed to continuous) signals. White noise is a good example of a signal with no periodicity and complete randomness. If one performs the autocorrelation of a random noise signal, then something like Figure D-10 results. This is a complete match of a signal with itself only at $t = 0$, that is when the signal is exactly aligned with itself. At all other times, there is no consistent match between the signals.

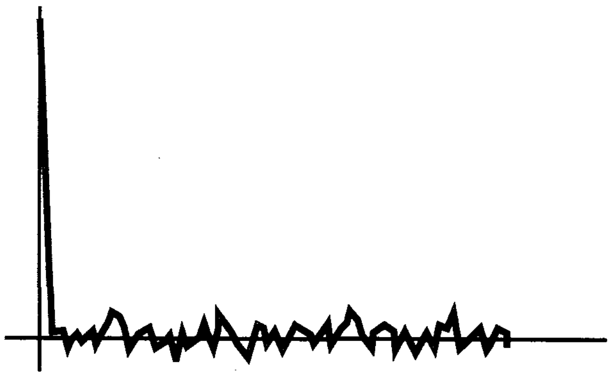


Figure D.10 - Autocorrelation of random noise

If there is a periodicity within the signal, the autocorrelation will show a peak at a time separation corresponding to the frequency of the periodicity. Figure D.11 shows the autocorrelation of a noisy signal with some weak periodicity.

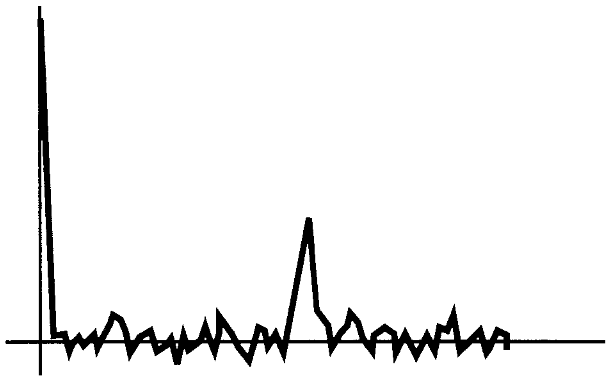


Figure D.11 *- autocorrelation of a noisy signal with a weak periodic spectral component*

One last example of autocorrelation shows when a signal has been low-pass filtered. In this case, adjacent signal samples are related to each other since the low-pass filtering removes high-frequency noise, and thus adjacent samples cannot be too unlike one another. In this case, the signal is like itself for small values of t (small offset between the signal and itself). Figure D.12 is an autocorrelation of a low-pass filtered noise signal.

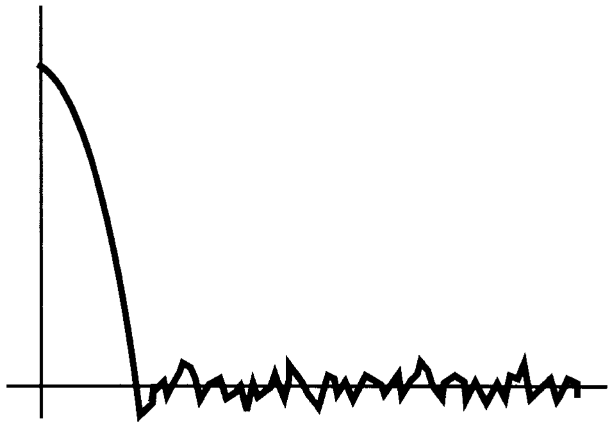


Figure D.12 *- autocorrelation of white noise that has been low-pass filtered*

It is clear that observation of the autocorrelation integral tells much about the signal. Taking the inverse Fourier transform of the autocorrelation integral in fact gives the power spectral density of the signal itself. Autocorrelation of carefully designed sequences (such as the PRBS) can have a single peak at time t=0. These sequences are valuable for synchronization with noisy signals, since there is great immunity to accidentally synchronizing at the wrong time.

D.7 Complex Signals

While the mathematical operations for performing convolution and correlation should now be clear, it may not be so obvious how to perform these operations on complex signals. In many DSP calculations, it may be easiest to separate the calculation of the real and the imaginary portions into separate operations. Then the real and imaginary impulse response components and signal components can be used. The product of the impulse response and the input signal is then the multiplication and addition (accumulation) of complex signals. The real input signal can be obtained by quadrature sampling of the input, or it can be derived from the input sequence via a Hilbert transform of that sequence (which produces the wideband 90-degrees phase shift, hence quadrature sequence). The design of Hilbert transforms has been covered in books on digital filter design (Parks and Burrus, 1987).

The equations for deriving the convolution of a complex signal with a complex impulse response are:

$$y_i(t) = h_i(t) \otimes x_i(t) - h_q(t) \otimes x_q(t) \tag{D.6}$$

$$y_q(t) = h_i(t) \otimes x_q(t) + h_q(t) \otimes x_i(t) \tag{D.7}$$

$$y(t) = y_i(t) + jy_q(t) \tag{D.8}$$

Where ⊗ represents convolution, and (i,q) are the in-phase (I) and quadrature phase (Q) components, respectively. Tne input is x(t), and y(t) is the output of the filter with an impulse response h(t). This algorithm can be depicted as a diagram, shown in Figure D.13. Further discussion of complex convolution can be found in Steele (1992).

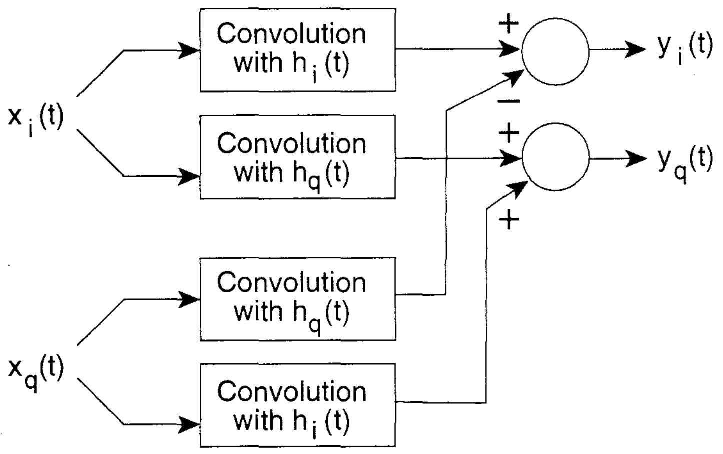

$x_i(t)$
$x_q(t)$
Convolution with $h_i(t)$
Convolution with $h_q(t)$
Convolution with $h_q(t)$
Convolution with $h_i(t)$
+
−
+
+
$y_i(t)$
$y_q(t)$

Figure D.13 *- Complex convolution algorithm*

D.8 Laplace Notation for Complex numbers

One common notation for complex signals, especially when describing filters, is the use of the Laplace notation. We won't go into the derivation of the notation here except to summarize the notation. In this notation, the variable s represents a complex number:

$$s = \sigma + j\,\omega \tag{D.9}$$

We can describe the reactance of a capacitor or an inductor with this short hand. To perform a sinusoidal single-frequency analysis, we let the real part (sigma) be zero. Thus, the impedance of a capacitor versus frequency is:

$$Z = \frac{1}{j2\pi fC} = \frac{1}{sC} \tag{D.10}$$

Similarly, the impedance of an inductor is:

$$Z = j2\pi fL = sL \tag{D.11}$$

We can then describe a simple low-pass filter using this convenient notation. The impedance of the resistor is R, and the impedance of the capacitor is 1/sC. See Figure D.14.

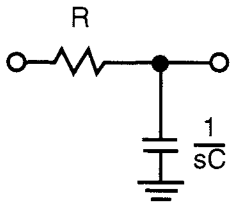


Figure D.14 *- RC low-pass filter, with impedances of the two elements expressed in Laplace notation.*

We can then derive the transfer function of the low pass filter very easily. Some simple algebraic manipulation of the equation puts the filter into a very useful form.

$$H(s) = \frac{\frac{1}{sC}}{\frac{1}{sC} + R} \tag{D.9}$$

This is the familiar voltage-divider equation. Multiplying both numerator and denominator by *sC* yields:

$$H(s) = \frac{1}{(sCR + 1)} \tag{D.12}$$

In this equation, RC is the time constant of the filter. The -3 dB cutoff frequency of the filter expressed in radians/second is 1/RC. Thus, replacing RC by 1/ω, and then multiplying both numerator and denominator by ω yields:

$$H(s) = \frac{\omega}{s + \omega} \tag{D.13}$$

If we normalize the above equation to a frequency of one radian per second, we replace ω with 1:

$$H(s) = \frac{1}{s + 1} \tag{D.14}$$

This expression characterizes both the frequency and the phase response of the RC filter in a single compact equation. To analyze the response when the input is a simple single-frequency sinusoid, *s* is replaced with *j*. At DC, *s* is equal to zero, and so *H(s)* is 1 (no attenuation at DC for this filter). At a frequency of unity (the -3 dB frequency), $s = j1$, and the above equation evaluates to:

$$H(s = j1\ rad/sec) = \frac{1}{j1 + 1} \tag{D.15}$$

The amplitude of this value is 0.707 (-3 dB), and the phase is -45 degrees. Thus, a filter with a capacitor of one farad and a resistance of one ohm has a -3 dB cutoff frequency of one radian/second, and has 45 degrees of phase lag at that frequency. Figure D.15 shows the amplitude (in dB.) and the phase (in degrees) versus frequency (omega, in radians/second) for equation (D.14). The amplitude rolls-off at -20 dB/decade (-6 dB/octave) — commonly referred to as a single-pole roll-off.

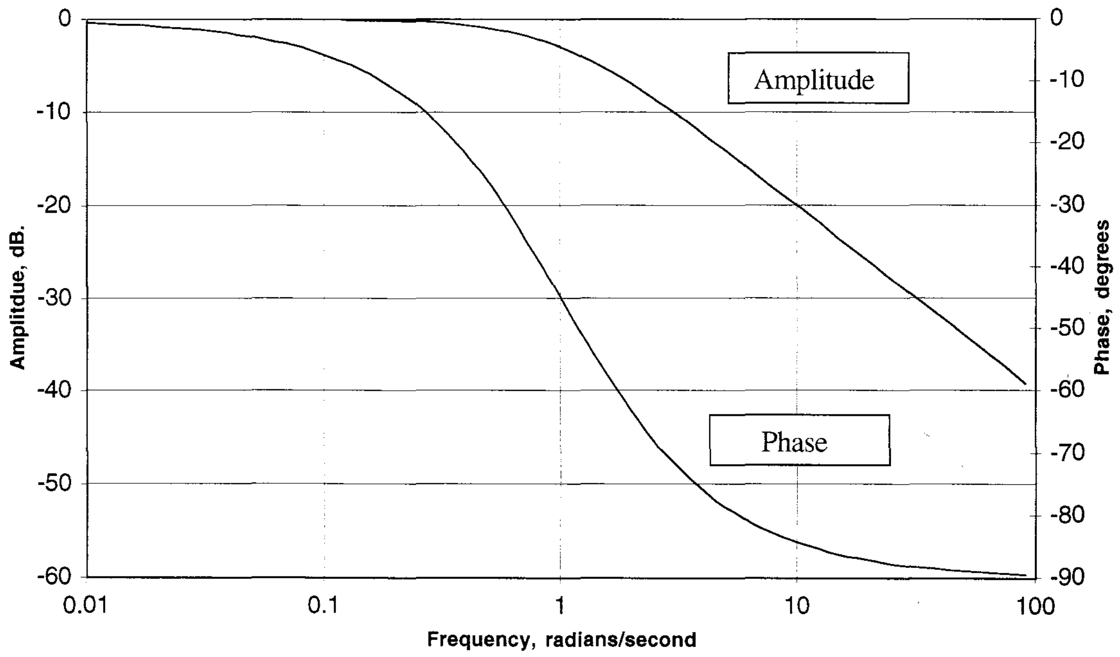


Figure D.15 - RC lowpass filter described by equation (D.11)

A high pass filter with a series C and a shunt R has a transfer function of:

$$H(s) = \frac{S}{S+1} \tag{D.16}$$

When normalized to one radian per second. Figure D.16 shows the amplitude (dB) and the phase (degrees) versus frequency (omega) for the highpass filter of equation (D.16). Notice that the phase is positive (leading) for the highpass filter.

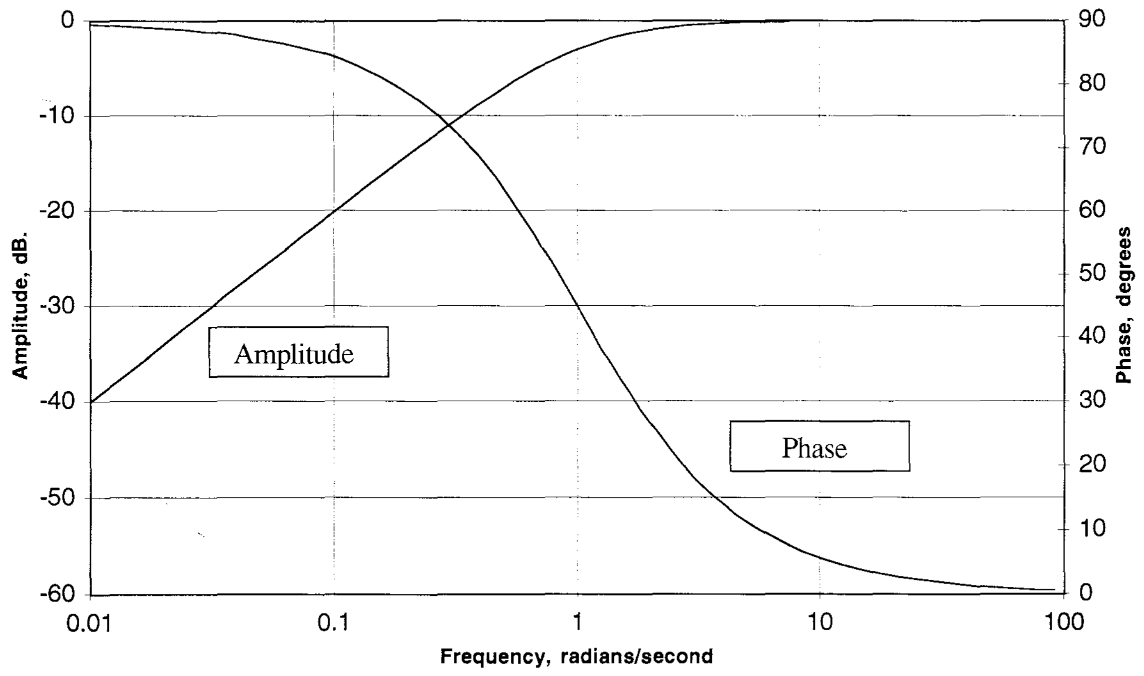


Figure D.16 *- RC highpass filter described by equation (D.16)*

The numerator of the function is called a *zero* of the transfer function, since when it is zero, there is no response from the network [$H(s) = 0$]. In the high-pass filter, the numerator is zero at DC, so the high-pass network blocks DC (which is obvious). The denominator of the function is called a *pole*, since when it evaluates to zero, the transfer function is infinite (actually, undefined). In the above equation, the denominator evaluates to zero when s = minus 1 (a real, not imaginary number). Poles with negative, real values produce exponentially-damped responses. Poles with purely imaginary values (sigma = 0) produce undamped sinusoidal responses. Complex poles (negative real + some imaginary) produce responses with both exponential decay and sinusoidal components. Poles with positive real values produce oscillators!

Other more complicated networks can be analyzed in the same manner as those above, and can be numerically evaluated with a spread sheet such as Excel which can handle complex numbers. The file APPEN-D.XLS, included on the disk, calculates both of the above formulas versus frequency, and the charts produced from them.

A good derivation of the Laplace transform is in Best (1993).

Appendix D - Reference

Parks, T.W., and Burrus, C.S. (1987). Digital Filter Design. John Wiley and Sons, Inc.

Steele, R. (1992). Mobile Radio Communications. Pentech Press Ltd, London and IEEE Press, NJ. Chapters 1.2.4 and 2.1.3.

Best, R.E. (1993). Phase-Locked Loops: Theory, Design, and Applications. 2nd edition, McGraw-Hill, Inc. Appendix B.

TAPR wants you!

Zero in on the international organization that started the packet revolution in the early '80s and continues today to work on new technology, standards, and publications. Combine this work with the Packet Status Register, an authoritative source since 1982 for up-to-date user and technical information about digital communications, and you can't go wrong when you join TAPR!

Don't be left out! TAPR is the place to get connected on an international, national and regional basis. Find out what others are doing and what ***you*** might do next in digital communications!

Goals:

- Support R&D in amateur radio digital communications
- Disseminate information on digital communications
- Provide affordable and useful kits
- Pursue and help advance the amateur radio art of communications through publications, meetings, and standards

Membership:

Membership in TAPR is $20 per year (US, Canada, and Mexico) and outside North America, membership is $25 per year.

History:

If you have used a packet radio TNC, then you are already a part of TAPR history.

TAPR was founded in 1982 as an international organization with interests in the areas of packet and wireless digital communications. Today, TAPR continues as a membership supported non-profit amateur radio research and development organization. TAPR currently has more than 2500 members, worldwide. TAPR continues to develop kits for the amateur radio community and is working actively on publications and communications standards

Special Interest Groups:

Spread Spectrum
HF Digital Communications
DSP
Networking
. . . ***and more!***

Long Term Technical Projects:

- Spread Spectrum Communications research and kit development.
- Improved HF digital communication methods.
- Develop faster local user access methods.
- More technical and educational materials to supplement kitting and design projects.

Join the wireless digital communications fun !

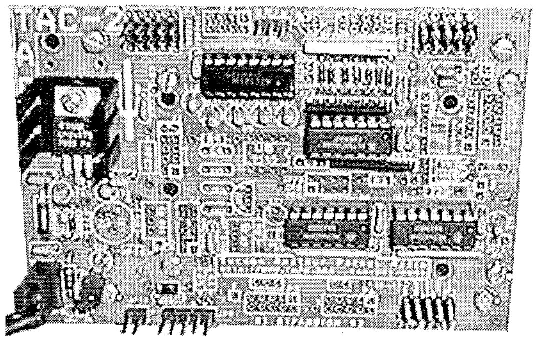
http://www.tapr.org

Product Overview

March 1998

TAPR has been active in the development of amateur radio kits since its origination. What follows is a limited listing of TAPR kits. A current price list and full product description can be attained by contacting the TAPR office or via the Internet (tapr@tapr.org • http://www.tapr.org)

Totally Accurate Clock



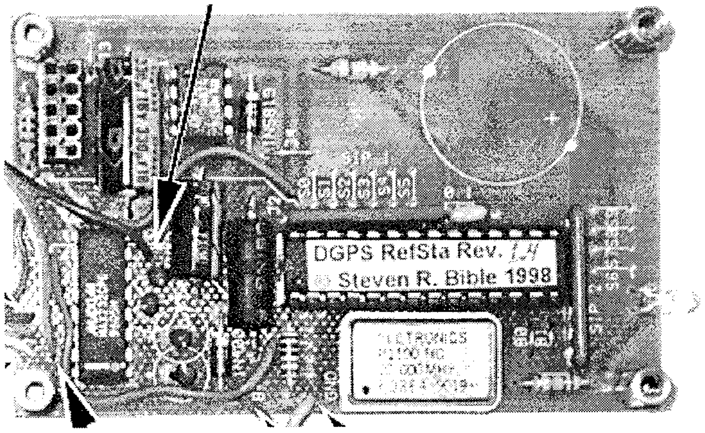
If you need an accurate clock, then the TAC-2 (Totally Accurate Clock), designed by Tom Clark, W3IWI, is for you. Depending on the selection of a GPS, RMS timing precision of 50nsec or less can be obtained. This is 3-4 orders of magnitude better than any of the WWVB or DCF77 systems -- like the "Most Accurate Clock" units provided by Heathkit in years past.

The TAC-2 kit can interface to several popular GPS units including the Motorola ONCORE, Garmin GPS-25 and Trimble SK-8 — which can all be mounted on the TAC-2 board. The TAC-2 adds a number of desirable features in this kit

The TAC-2 kit is intended to serve several purposes: It provides a "universal" electrical and mechanical interface for a number of common OEM board-level GPS receivers including specifically: Garmin GPS-25, Motorola Oncore, and Trimble SK8. It provides interfaces for the 1 pulse-per-second (1PPS) signal generated by the GPS. It supports Low-impedance, fast rise-time 1PPS signals for "laboratory" applications. Has RS232 level 1PPS signals for computer applications. Supports specialized 1PPS interfaces for an add-on PCB that will stabilize a low-cost crystal oscillator to an accuracy ~ 1 part-per-billion. It can be configured with several different power supply options to make use of your GPS receiver easier. The kit has provision for an Uninterruptible Power Supply (UPS) to buffer the GPS receiver through brief power outages. The TAC-2 can provide Battery Backup so the GPS receiver can wake up "smart."

The TAC-32 program (provided) displays the UTC time, date, day-of-week, day-of-year, local, Greenwich Mean Sidereal times, JD and MJD, and even the current GPS week. You can enable audible "WWV-like" time ticks to assist you in setting the formatter (or your wrist-watch). You can have the software automatically reset the PC's internal clock with about 25 msec accuracy. All the time display updates and audible ticks happen synchronously with the GPS 1 PPS signal because the PC reads the tick on its DCD line. TAC-32 allows you to enter timing offsets, thus allowing easy corrections for time delays in cables and the instrument and it tells you (with 1 nsec resolution) the actual epoch of the 1PPS tick and it gives you an estimate of the accuracy of the tick. SHOWTIME gives you a nice display of which satellites you are using and which satellites are above the horizon and includes a bar-graph "S- meter" for each of the GPS satellites currently in lock. Full details on the software is available on the web page. (**http://www.tapr.org/tapr/html/tac32.html**)

DGPS Ref Station



The Differential GPS (DGPS) Reference Station Interface Board connects to a Motorola Oncore VP OEM GPS receiver to create a low cost 8-channel DGPS Reference Station. The Reference Station provides pseudo-range differential GPS corrections that conform to the RTCM SC-104 Type 1 Version 2.1 message format. These can be transmitted via data link to remote users. Possible data links are radio modems, terminal node controllers (TNC), or telephone modems. Remote users receive the corrections and apply them to their DGPS ready receiver to calculate a DGPS solution. (http://www.tapr.org/gps)

Possible uses include:

- Search and Rescue
- Fire Fighting/Flood Area Boundaries
- Automatic Vehicle Tracking / APRS Trackers
- Parades and Marathons
- Balloon Tracking
- Research and Development
- Anything that requires increased GPS accuracy

Amateur Radio

Spread Spectrum

Visit the TAPR Spread Spectrum Web Page

http://www.tapr.org/ss

TAPR CD-ROM

Over 650 Megs of Data in ISO 9660 format. TAPR Software Library: 30 megs of software on BBSs, Satellites, Switches, TNCs, Terminals, TCP/IP, and more! 150Megs of APRS Software and Maps. RealAudio Files. Quicktime Movies. Mail Archives from TAPR's SIGs, and much, much more!

AN-93 HF Modem: RTTY, AMTOR, PCTOR

The AN-93 kit provides any PC user with the capability for operating RTTY, AMTOR, and PCTOR with this simple modem-only design. AN-93 is the equivalent of a BayCom, BayPac, or PMP setup, but for HF digital operations. This very simple kit should be ideal for many who have wanted to play on HF, but didn't want to pay the money for an expensive multi-mode controller. With the AN-93, only three components are required for HF digital communications: a PC-compatible computer, the AN-93 modem, and software that performs the encoding and decoding. The AN-93 comes with a tuning indicator to allow visual tuning and the unit also provides audio output for oscilloscope display. The AN-93 requires 9volts @ 150mA. Power Supply is not included in the kit. Interfacing to the radio is through a DB-9. TAPR provides a simple method tuning the modem. The kit will be shipped with the A/D convertor providing full-memory ARQ capability for PCTOR. The kit provides both FSK and AFSK outputs for use with more common amateur HF radios.

TAPR 9600bps Modem

NOTE: This kit requires some prior hands-on experience with radio modifications.

The TAPR 9600bps modem is a full-duplex baseband modem compatible with K9NG and G3RUH standards. The TAPR 9600bps modem, like other 9600bps modems, requires an interface to your radio other than via the microphone and speaker jacks. The modem fits internally into the case of many popular TNCs and supports a 20-pin modem disconnect header which allows for a second external modem (such as the TAPR PSK modem) to optionally be connected.

The transmit path includes an independent watchdog timer. The modem selection logic correctly routes the PTT command from the TNC to prevent, for example, keying your 1200 baud radio when running 9600bps. The receive filter includes a compensation adjustment to tailor the modem to the radio receiver and AC-coupling to accommodate frequency drift on the channel. An improved state machine is used for clock recovery. The DCD circuit works on both signal quality and the presence of a signal

Index

Supplied Software

Given the proper tools, it can be very intuitive to see the relationship between modem filter and channel responses and the resulting eye pattern, and thus modem performance. The supplied software will allow you to experiment with different modem filters and visually see the effect of the changes, plotted directly as eye patterns on screen.

The software supplied on the enclosed disk was used to generate the channel response graphs, and the eye patterns in section 5 of the text. With this software, you can model many different channels and modem filter designs, and make different assumptions about the precision of the filters, the lengths of the filters, and you can model many different responses.

For example, if you have a radio with known response defects, it may be helpful to model the frequency response and the resultant eye pattern from that radio. Given the measured amplitude and phase characteristics, you can decide what kind of performance degradation will occur with the use of that radio by simulating the eye pattern it will generate, and then measuring the eye closure due to intersymbol interference directly on the plotted graph. Or you can measure the jitter of the clock recovery system in your modem, and using a plotted eye pattern for your radio you can estimate the performance penalty due to the non-optimum positioning of the clock. You may wish to build a filter to compensate for the defects present in a current modem or radio design, and then model the resultant eye pattern by combing all of the channel responses.

Finally, you may wish to design your own modem filters, and the supplied software will allow you to calculate good impulse response coefficients for a digital filter implementation, and then calculate the eye pattern that should result from that filter. Or, you may wish to design other filters, and the supplied software will allow you to specify some interesting filter shapes, and will determine the impulse response coefficients.

The eye pattern calculation requires two steps, first inputting the frequency (and possibly phase response) of the filters and then determining the equivalent impulse response. This is done using the RAISCOS.XLS spreadsheet. Secondly, the impulse response is convolved with a pseudo-random data stream, (a pseudo-random data stream is generated, and then ‘filtered’ by your filter), and the resultant output is plotted as an eye pattern, this involves the use of one of the ‘EYExxxx.XLS’ spreadsheets, depending on the filter length you want to use. The procedure is described near the end of section 5, and involves some cutting and pasting between the two spreadsheets. You can easily modify the resolution and oversampling rates by changing the spreadsheets.

The response of simple low-pass and high-pass filters in covered in the spreadsheet APPEN-D.XLS, which models the filters discussed in Appendix D. This spreadsheet allows modification of the pole frequencies, and observation of the amplitude and phase responses. More complex filters can be simulated directly from the transfer function of the filter. The enclosed spreadsheet shows how the transfer functions of the basic filters are mapped into the spreadsheet, and you can easily expand the spreadsheet to include more complex filters. The simulation of the filters gives a good 'intuitive' understanding of how the transfer function relates to the filter response.

Phase locked-loops are sometimes mysterious because the properties of the feedback loop have to be understood in terms of both the amplitude and the phase of the feedback. The spreadsheet PLL.XLS provides both the open-loop and the closed-loop response of several common PLL filter functions, and allows looking at the amplitude and phase response of a phase locked loop. the critical parameters are easily varied, and the resultant response is plotted in familiar terms. You can measure you loop parameters, and then determine if the loop will be stable, or will have desirable properties. If not, its easy to change some parameters, and simulate the resultant response.

Lastly, Dolph-Chebychev filters are fascinating, and not only provide unusual frequency response, but also finite impulse response duration. The enclosed spreadsheet allows modeling the frequency response and determination of the impulse response coefficients for different lengths of DC filters. Additionally, they describe the antenna currents desirable for a phased-array antenna since the mathematics behind array pattern formation is the same Fourier transform as the frequency-response / impulse response relation.

Installing the Software

The software has been compressed into a PC self-extracting archive file (.EXE). The files occupy a little over 3 megabytes of disk space when expanded. All of the EXCEL spreadsheets require Microsoft Excel 5.0 or later to run. The Excel files themselves can be run on either Macintosh or Windows versions of Excel. The program REMEZ.EXE must be run either in DOS, or from a DOS window from within Windows, it will not run on a Macintosh.

Unarchiving instructions for files using a DOS based machine

First, copy the file DISTRIB.EXE into the root directory of the desired destination hard drive. For example, to install the distribution on your "C" hard drive, insert the diskette into your floppy drive, and type:

```
c:                  where "c" is the destination drive!
cd \                to start from root directory
copy a:distrib.exe  where "a" is the disk drive containing the distribution diskette
```

Type 'distrib' to extract all the necessary files from the DISTRIB.EXE file.

Unarchiving instructions for files using a Macintosh

If you are unable to read DOS formatted disks, the DISTRIB.EXE can be found on http://www.tapr.org in the publications area under products for downloading to whatever format you require.

Copy the DISTRIB.EXE file on the diskette to a folder on your hard disk. Expand the DISTRIB.EXE file using StuffitExpander (freeware) and DropStuff (shareware), which are recommended. See below for a source for these programs if you do not have them already.

If you are using StuffitExpander, simply drag the DISTRIB.EXE file onto the StuffitExpander icon. The archive will be expanded and placed into a folder.

StuffitExpander and DropStuff may be obtained via anonymous ftp from
mac.archive.umich.edu   /mac/util/compression/

You'll need both: stuffitexpander3.52.sea.hqx and dropstuff3.52.sea.hqx

Supplied Software Information

RAISCOSI.XLS

Excel 5.0 spreadsheet containing raised-cosine filter frequency and impulse responses. Thes can be modified for custom responses, and the impulse response can be pasted into the ey pattern sheets to plot the resultant eye pattern.

EYE4-9.XLS

Excel 5.0 spreadsheet containing an alpha=0.4, 9-tap eye pattern plotter for 2-level eyes. Yo can paste a 9-tap impulse response into this sheet.

EYE2-17.XLS

Excel 5.0 spreadsheet containing an alpha=0.2, 17-tap eye pattern plotter for 2-level eyes. Yo can paste a 17-tap impulse response into this sheet.

EYE2-33.XLS

Excel 5.0 spreadsheet containing an alpha=0.2, 33-tap eye pattern plotter for 2-level eye: You can paste a 33-tap impulse response into this sheet.

PLL.XLS

Excel 5.0 spreadsheet with open- and closed- loop plots for 3 common PLL loop filters. Yo can vary the gain and breakpoint frequencies, and the plots will update automatically.

APPEN-D.XLS

Excel 5.0 spreadsheet with simple low-pass and high-pass filter calculations using Laplac notation. You can modify the filter equations, and the plots will update automatically.

DOLPH.XLS

Excel 5.0 spreadsheet containing Dolph-Chebychev pulse description, including frequenc and impulse response.

REMEZ.EXE

DOS executable program. Calculates impulse response for general low-pass, high-pas: bandpass, stopband, raised-cosine, square-root raised cosine, sinc-compensated square-roo raised cosine filter. Also calculates a Hilbert-transform filter and a wideband differentiato This program uses the Remez-exchange technique and is derived from the Parks-McClella algorithm. The program writes its output to the standard output, so you may want to redire it to a file. For example: REMEZ > MYFILE.DAT would place the listing into the file name MYFILE.DAT.