AN INVESTIGATION INTO A DSP IMPLEMENTATION OF PARTIAL RESPONSE SIGNALING FOR 4800 BITS PER SECOND FULL-DUPLEX DATA COMMUNICATIONS OVER M.1020 TELEPHONE LINES

Prepared by: Russel Horwitz

Prepared for: Dr. R. M. Braun
Department of Electrical and
Electronic Engineering
University of Cape Town

A thesis submitted to the Faculty of Engineering, University of Cape Town, in partial fulfillment of the requirements for the degree of Master of Science in Engineering

Cape Town 1990

DECLARATION

I declare that this thesis is my own, unaided work. It is being submitted for the Degree of Master of Science in Engineering at the University of Cape Town. It has not been submitted before for any degree or examination in any other University.

Signature removed

_____

(Signature of candidate)

30th day of September 1990

To my folks, Norman and Wendy, who thought it would be a good idea to send me to University.

ACKNOWLEDGEMENTS

I would like to express my appreciation to:

Dr. Robin Braun, my supervisor for his invaluable advice and support during my thesis.

Les Devonish, of STC, for making funds available for the project.

Steve Schrire for his excellent technical assistance.

Tim Courtenay for his freindship, support and encouragement during the last two years.

TERMS OF REFERENCE

This thesis investigation into a DSP-based partial response signaling system was requested by Dr. Robin Braun, Senior Lecturer in the Department of Electrical and Electronic Engineering, University of Cape Town and Mr. Les Devonish of STC in Boksburg. The investigation was requested in January 1989.

The specific instructions were:

1. To investigate the use of PRS as a means of upgrading the CCITT V.22bis modulation scheme to a 4800 bps full-duplex scheme.

2. Digital signal processing techniques are to be used wherever feasible to improve performance and reliability.

3. To limit the scope of the project it can be assumed that the amplitude and phase response of the channel will be equalized over the bandwidth of M.1020 leased lines. The modulator and demodulator sections only are to be investigated.

4. The objective of the project is to investigate the feasibility of a DSP implementation of PRS for this particular application, rather than to deliver a complete working modem.

5. The thesis must be submitted by 29 September 1990

SYNOPSIS

This thesis investigates high-speed digital transmission over a conditioned, voice-grade telephone circuit (M.1020), using a technique known as partial response signaling, or PRS. In particular, the case where 4800 bps, full-duplex transmission is required in a CCITT V.22 type format is investigated. The main V.22 criterion to be adhered to, is that frequency-division multiplexing (FDM) is to be used as the means of separating the transmit and receive channels. The carrier frequencies should be 1200 Hz and 2400 Hz respectively. The investigation concerns the modulation and demodulation sections only.

Conventional, memoryless data transmission systems can usually achieve a spectral efficiency of 1 - 1.5 bps/Hz in the case of binary transmission. This is due to the impracticalities associated with designing an ideal lowpass filter that will give zero intersymbol interference (ISI). The highest speed V.22 type modem standard for operation on M.1020 lines is the V.22bis standard, which provides 2400 bps full-duplex transmission (excluding any data compression). It uses a multilevel, memoryless modulation scheme, known as 16 QAM.

PRS is a technique which uses a transversal digital filter to correlate bits is such a way that it becomes possible to transmit the signal in the minimum bandwidth stated by Nyquist's theorem. The result is that the spectral efficiency is increased to 2 bps/Hz for the binary case.

In this project, a multilevel class-1 (duobinary) PRS scheme, called 49 QPRS, is investigated as a means of achieving 4800 bps transmission in the V.22 type format. The two data channels will use the same carrier frequencies as the V.22 schemes, namely 1200 Hz for the low band and 2400 Hz for the high band. Pilot tones are used for receiver synchronization. One pilot tone is transmitted with each band and is arranged in such a way that the receiver

sampling clock, carrier frequency and symbol timing signal can all be unambiguously recovered from a single pilot tone.

Digital signal processing is a widely used technology whereby complex functions are implemented digitally, which offers considerable advantages over analog techniques. The modem design in this project is based on a the Motorola DSP56001 digital signal processor. The DSP chip is mounted on a plug-in PC card and is programmed via the host PC. The processor implements all the complex modem functions, including encoding, Nyquist type lowpass filtering and modulation functions.

A raised-cosine lowpass filter is used for Nyquist filtering and is evenly divided between transmitter and receiver in order to conform to matched filter criteria. This filter is implemented with a linear phase, finite impulse response (FIR) design.

In order to fulfill the requirements of a half-thesis, it was not necessary that the entire modem be built. The work carried out in this project included the following:

a) A 49 QPRS transmitter prototype was constructed, which is able to generate both low and high band signals, complete with pilot tones. The design included DSP56001 assembler software, a Turbo Pascal host program to initialize the DSP56001 and a hardware design. The hardware included external memory, D/A converter, output smoothing filter, pseudorandom sequence generator and a clock generator.

b) The receiver software was written and successfully tested to demonstrate the implementation of the receiver on the DSP56001. This included a DSP56001 assembler program and a Turbo Pascal host program. The hardware was not built.

c) A computer simulation was carried out to investigate crosstalk between the two channels. This is caused by a out-of-band frequency components of the transmitted signal echo overlapping with the received signal.

The transmitted signals were observed to be extremely clean, and contained no visible intersymbol interference at the symbol timing instants. The frequency spectra of the signals were equally well defined. Very sharp bandlimiting was achieved with good suppression of sidelobes (up to 60 dB).

The most important result of the crosstalk simulation was that transmitter echoes up to 10 dB above the received signal will have negligible effect on the performance of the system, even if no bandpass filtering of the received signal is performed. In ideal conditions, echoes up to 24 dB above the received signal could be tolerated.

It was also shown that, if performance is to be optimized, the FIR filter lookup table should be as long as possible, a windowing function should be employed, and a moderate amount of excess bandwidth (in the order of 5 - 10 percent) should be used.

The results obtained were very promising. It is recommended that future work should aim to complete the design, which would include a real-time implementation of the receiver, the clock recovery circuits and an adaptive equalizer.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

AM	amplitude modulation
A/D	analog to digital converter
DSP	digital signal processing
D/A	digital to analog converter
FDM	frequency division multiplexing
ISI	intersymbol interference
LSB	least significant bit
PAM	pulse amplitude modulation
P(e)	probability of symbol error
PLL	phase locked loop
PRBS	pseudorandom bit sequence
PRS	partial response signaling
QAM	quadrature amplitude modulation
QPRS	quadrature partial response signaling
SNR	signal to noise ratio
VCO	voltage controlled oscillator
α	filter roll-off parameter

1. INTRODUCTION

The tremendous advantages of digital over analog telecommunications systems have resulted in an ever increasing number of these systems utilizing digital techniques. In order to maximize these advantages, it is necessary to transmit digital signals at the highest possible rate. This thesis concerns the full-duplex transmission of 4800 bits per second data over specially conditioned telephone lines. A paper on this thesis was published in the COMSIG 90 proceedings and can be found in Appendix A.

The modulation scheme is intended to upgrade the scheme used in V.22bis modems. The required method of separating the two data directions is to transmit the signals in non-overlapping frequency bands, a technique known as frequency division multiplexing, or FDM. The maximum speed at which transmission can take place on a telephone channel is limited mainly by the following factors:

a) The bandwidth of the transmission channel
b) Channel group delay and magnitude response
c) Noise introduced by the channel
d) Noise introduced by the hardware
e) Hardware complexity and cost

Conventional binary and multi-level signalling schemes, also called memoryless schemes, are based on the requirement that each pulse must be confined to its own time slot as far as possible [1.1]. In systems such as these, also called zero-memory systems, the pulse tails of the bandlimited pulses are a major source of intersymbol interference and must be eliminated or, at least, minimized. In practice, binary systems using the above criterion cannot achieve the Nyquist rate of 2 bits/second per Hertz of available bandwidth, as this would place unrealistic demands on the bandlimiting filter. Consequently, a maximum efficiency of 1 bps/Hz is usually achieved.

A technique proposed by Dr. Adam Lender in 1962 yields a doubling of the transmission rate, without compromising bandwidth. Using the technique, a bandwidth efficiency of 2 bps/Hz can be achieved for the binary case, which is the theoretical maximum. Controlled amounts of intersymbol interference (usually 100%) are introduced and this alters the time domain and spectral properties of the signal. Because the pulses are combined in a known way, the original data can be obtained at the receiver by applying a simple decoding rule. This is known as correlative level coding, or partial response signalling (PRS). A special case of PRS, known as 49 QPRS, is investigated in this project.

The implementation of PRS schemes is similar to that of conventional pulse-amplitude modulated schemes. A disadvantage of PRS is that correlative encoding introduces more signalling levels. There will is corresponding increase in probability of error, due to the decision distance being decreased. Nevertheless, the high bandwidth efficiency of PRS is such that it can achieve a higher data throughput than conventional techniques in a specified bandwidth, given the same noise density.

Digital signal processing (DSP) technology is a concept which is becoming more popular in telecommunications as faster processors become more available. The crux of DSP is that signals are digitally processed and are interfaced to the analog world via A/D's and D/A's. The processing can either be carried out using digital hardware or, in more complex applications, using software that is run on a DSP chip. As a result of DSP techniques, highly tuned analog circuitry becomes unnecessary and complex modulation schemes can be implemented with relative ease.

In this project, Motorola DSP56001 digital signal processors mounted on PC plug-in cards were used. Two cards were purchased from PERALEX, a Cape Town based company. The following programs were supplied with the cards:

a) Cross-assembler for assembling programs written in DSP56001 assembler.

b) Program which allows the assembled code to be downloaded to the DSP56001.

c) A simulator program which allows all the DSP56001 functions to be simulated.

d) Various Turbo Pascal utility programs which were used mainly for interfacing between the PC and the DSP56001 via the host interface of the DSP56001.

The practical research carried out included the following:

The transmitter

A transmitter prototype was implemented, which is capable of generating 49 QPRS signals at one of the two carrier frequencies (for frequency division multiplexing). This included both software and hardware design. Various hardware, such as memory, latches and interfacing logic was wire-wrapped onto the DSP56001 card, while other hardware, which is described later, was built on separate printed-circuit boards.

The receiver

In order to limit the scope of the project to the requirements of a half-thesis, the receiver hardware was not built. Instead, the operation of the receiver was investigated by writing a DSP56001 assembler program that samples data from a file of transmitted data stored on disk. The file of transmitted data was generated by altering the real-time transmitter program in such a way that it sampled binary data from a file on disk and stored the transmitted

data on disk in another file. The receiver program then recovers the bitstream which is also stored on disk. It can then be compared to the original data in order to verify the correct operation of the data. A real time version would require an A/D and minor adjustment to the program, but would otherwise operate in the same way as the program which was written.

Clock recovery

A method of clock recovery was proposed for the modem. The scheme uses pilot tones, which offer reliable performance with a simple implementation.

Interchannel crosstalk

Interchannel crosstalk between the two data directions was investigated by computer simulation. It should be pointed out that due to the digital nature of the design, the computer simulation could be written in such a way that it can be assumed to be fairly accurate.

A flowchart of the thesis investigation is given in Figure 1.1 to better illustrate the scope of the research that was carried out

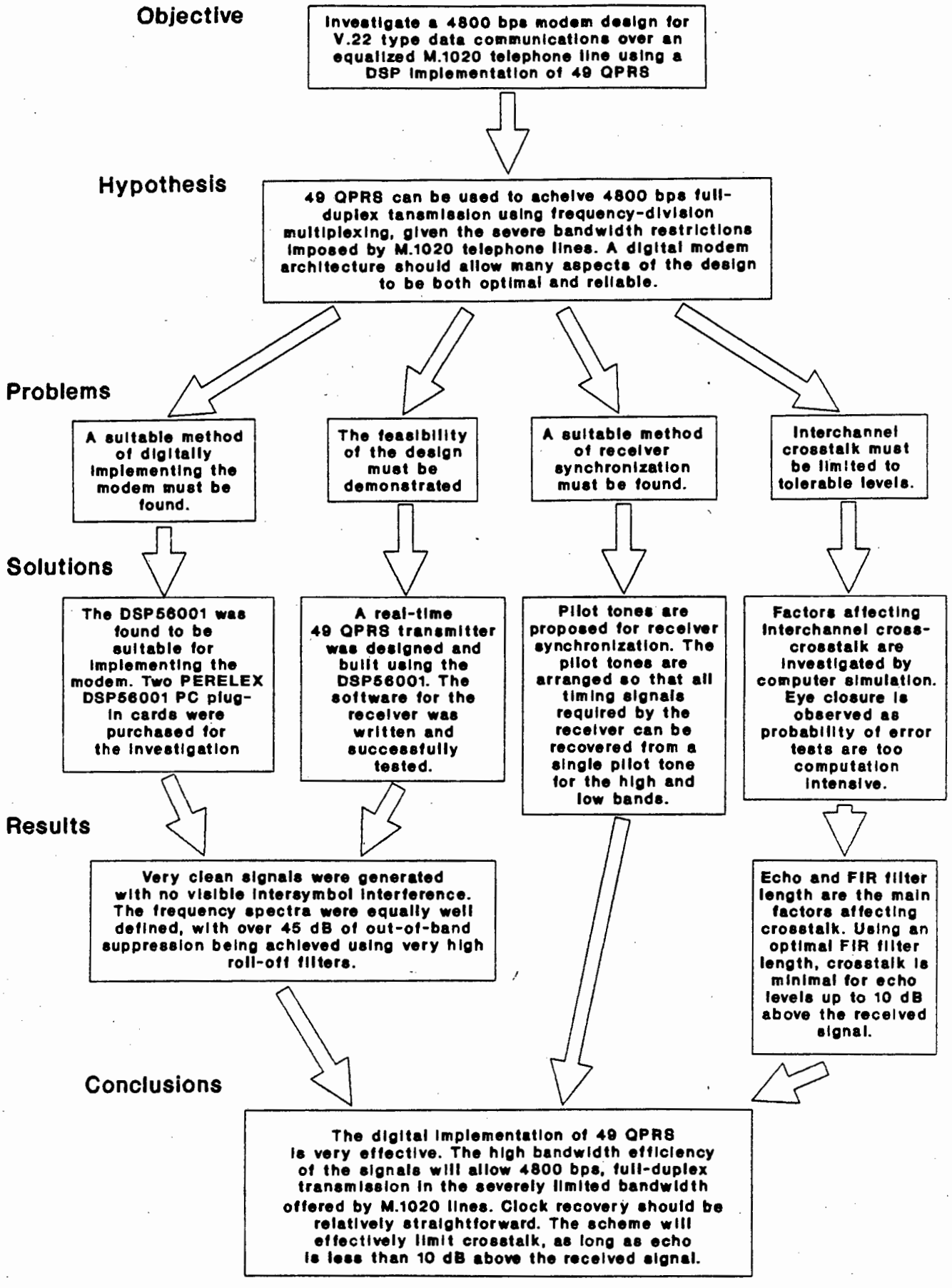


Figure 1.1 Flowchart of the thesis investigation

The objectives of this thesis report are :

a) To give a description of fundamental communication theory relevant to the project.

b) To provide an overview of PRS, leading up to the 49 QPRS scheme which is investigated, and to compare it to conventional schemes.

c) To describe the proposed method of implementing 49 QPRS as a means of achieving 4800 bps full-duplex transmission on an M.1020 telephone line using digital signal processing technology.

d) To describe the practical investigation, which includes hardware design, Turbo Pascal software and DSP56001 assembler software.

e) To investigate the possible drawback of crosstalk in the proposed scheme.

f) To draw conclusions from the research and to make recommendations for future work.

The report begins with an introduction of conventional, non-correlative techniques, leading up to 16 QAM, which is the scheme used for 2400 bps full-duplex transmission (CCITT recommendation V.22bis). An overview of partial response signalling, including the 49 QPRS scheme, is introduced and compared to QAM type schemes. The type of telephone channel that will be encountered is discussed, along with a brief description of the V.22bis implementation. This is followed by a detailed explanation of the proposed modem scheme and how it is implemented using DSP techniques. An investigation into crosstalk, caused by out-of-band frequency components overlapping into the adjacent channel, is then presented. Finally, conclusions are drawn and recommendations are made for future work.

REFERENCES

[1.1] Feher, K., **Digital Communications: Microwave Applications**, Prentice-Hall Inc., 1981, Chapter 7.

2. MEMORYLESS SYSTEMS

2.1 NYQUIST'S THEOREM AND PULSE SHAPING

Memoryless systems are digital communication systems in which pulses are confined as much as possible to their own time slot. Intersymbol interference (ISI) is thus kept to a minimum. Most conventional modems used today are based on this principle. The technique uses Nyquist's first criterion which can briefly be explained as follows [2.1]:

> Nyquist's first theorem states that the ISI at the sampling points between successive digits can be eliminated. Digits are independent and uncorrelated, and each digit can be recovered without resorting to the past history of the waveform. For maximum spectral efficiency, the transmission filter is made rectangular, with a cut-off frequency at $\frac{1}{2}T$ Hz, where T is the bit period. This results in a $\mathrm{sinc}(\pi t/T)$ impulse response. The response to two successive impulses is shown in Figure 2.1. Notice that, at time $t = T$, $g(t-T)$ is at a maximum, while $g(t)$ is zero. It follows that any number of pulses can be passed through the filter with zero ISI, as long as they are exactly T seconds apart. In theory, a binary system using the above criteria will have a spectral efficiency of 2 bps/Hz.

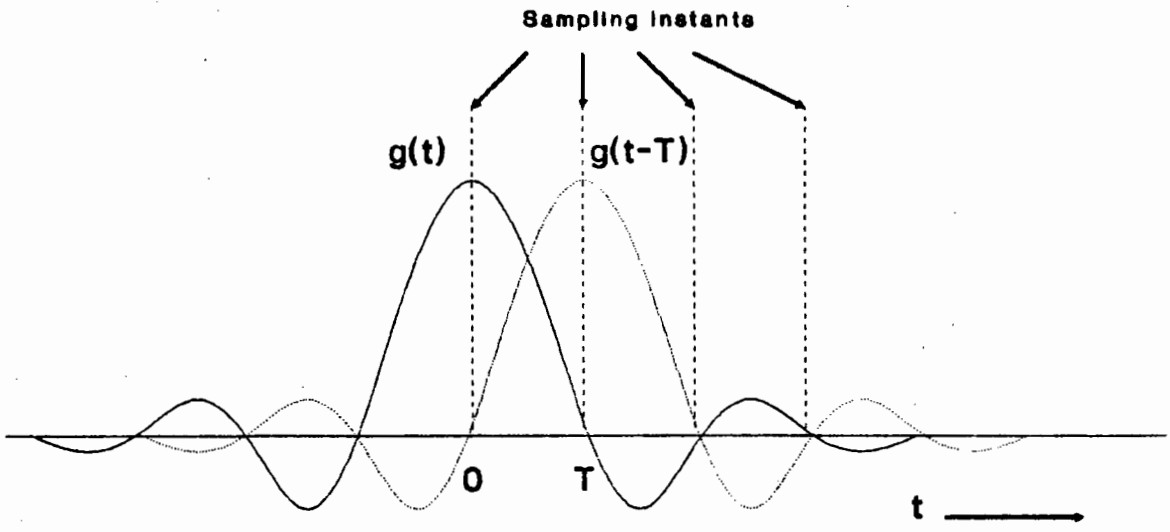


Figure 2.1 Response of a Nyquist filter to two successive impulses.

Unfortunately, such a filter is physically unrealizable. Even if it could be approximated, for example using digital filters, it would still not be practical because of the excessive ISI at the transition points, caused by large overshoots of pulse tails which decay as 1/t. As a result, even the slightest deviation from the bit rate of 1/T bits/sec at the sampling instants would render the system unusable.

The practical solution to this problem is to use a raised-cosine filter [2.2]. A raised cosine filter characteristic consists of a flat portion at low frequencies followed by a roll-off portion that is symmetrical about the cutoff frequency. The impulse response of this filter has nulls at the same instants as the rectangular filter but the pulse tails decay much faster, depending on the particular raised cosine characteristic that is chosen. The larger the roll-off portion of the frequency response (i.e. the more excess bandwidth used), the faster the pulses will decay. This is the fundamental trade-off when using these filters as it may

be desirable to confine the signal to the smallest possible bandwidth. The characteristic can be expressed in the following form:

$$X(w) = \begin{cases} T & 0 \le |w| \le (1-\alpha)W \\ T/2 * \{1 - \sin[\frac{\pi}{2\alpha w}(|w| - W)] & (1-\alpha)W \le |w| \le (1+\alpha)W \\ 0 & |w| > (1+\alpha)W \end{cases} \qquad (2.1)$$

where:

$W = \pi/T$

α is the roll-off factor and is defined as the excess bandwidth divided by the minimum Nyquist bandwidth (i.e. $\alpha = 0$ is equivalent to a rectangular filter)

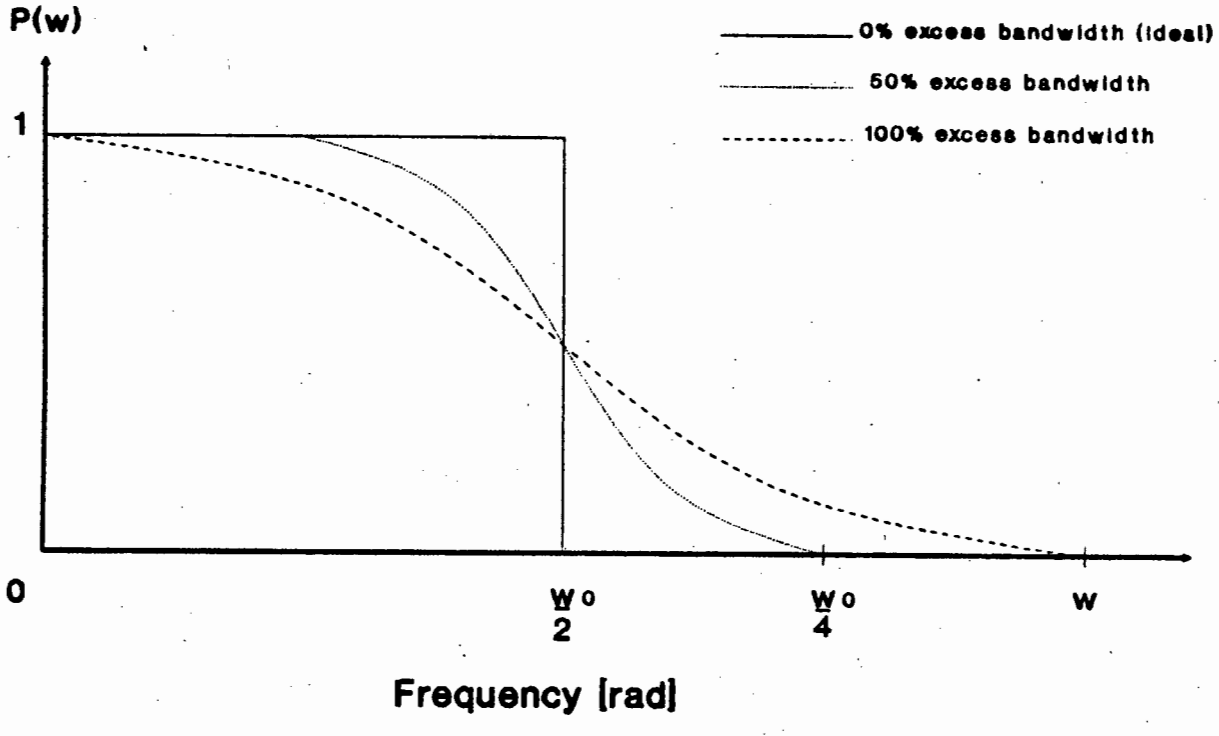


Figure 2.2 Frequency response of raised cosine filters

The corresponding impulse response of this filter is given by:

$$X(t) = \frac{\sin(Wt)/Wt)(\cos(\alpha Wt)}{(1 - (2\alpha Wt/\pi)^2} \qquad (2.2)$$

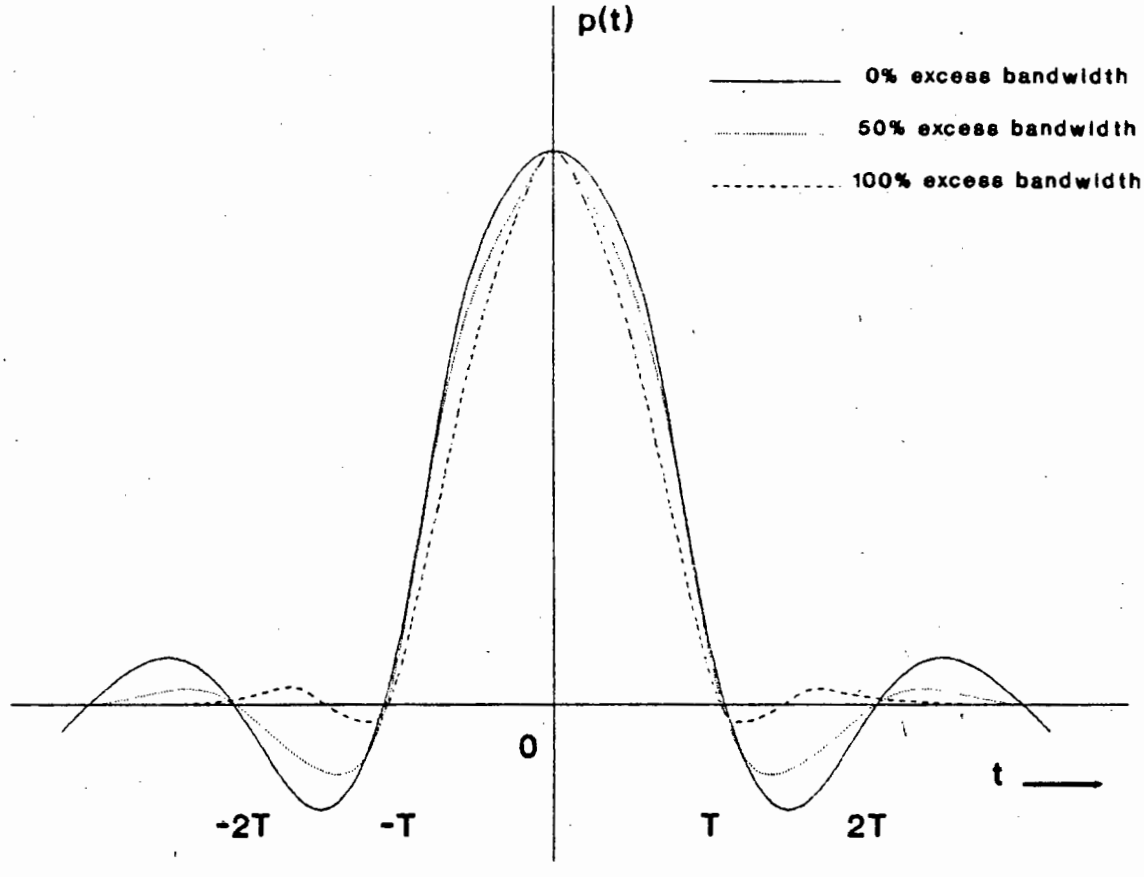


Figure 2.3 Impulse response of raised cosine filters

An $\alpha = 1$ raised-cosine filter is the most practical of these filters, being easily realizable and offering minimal overshoot of the pulse tails. It is obvious that the advantages of raised-cosine filters are obtained at the expense of increased bandwidth occupancy. This is why the spectral efficiency of most practical memoryless systems is 1 bps/Hz for the binary case.

2.2 THE GENERATION OF 16 QAM

There are various possible mechanisms for transmitting data over a channel. These can be divided into three groups: Amplitude modulation (AM), frequency modulation (FM) and phase modulation (PM). Frequency and phase modulation are closely related techniques and have the advantage that the transmitted waveforms have a constant envelope. This makes them ideal for power limited applications. Digital AM signals do not have a constant envelope but have other desirable features making them ideal for bandwidth limited applications. Since this project is a bandwidth limited application, FM and PM will not be discussed. 16 QAM, a special case of AM, is the modulation technique used in V.22bis modems and is used due to its good spectral efficiency which makes it ideal for a high- speed, bandwidth limited application. It will now be described using a step-by-step approach which will cover the following areas:

a) Amplitude modulation (AM)

b) M-ary Pulse Amplitude Modulation (PAM)

c) M-ary Quadrature Amplitude Modulation (QAM)

d) 16 QAM

2.2.1 Amplitude Modulation (AM)

The modulation and coherent demodulation of AM is illustrated in Fig 2.4. AM is generated by multiplying the amplitude of a sinusoidal carrier (b) with a modulating signal (a), also known as a baseband signal. The resulting spectrum will be a double-sided version of the baseband signal centered about the carrier frequency (c). The modulating process thus degrades the spectral efficiency by a factor of two. If the baseband signal has zero DC offset, i.e. no discrete spectral component at zero frequency, then the AM signal will have no carrier component. This would be called an AM suppressed-carrier signal. Conversely, by

adding a DC offset prior to modulation or by inserting the carrier after modulation, an AM large-carrier signal can be created.

An AM signal can be demodulated in two ways, namely envelope detection and coherent demodulation. In envelope detection, the envelope of the signal is extracted using a simple circuit. Although its simplicity makes it attractive, this technique yields suboptimum performance. Coherent demodulation is a more complex, but more efficient method. In coherent modulation, the received signal is multiplied again by the carrier frequency (d). The spectrum will then consist of a baseband component and a high frequency component centered about double the carrier frequency (e). By filtering off the high component part the baseband signal is recovered (f). One of the major disadvantages of this method is that the receiver must have access to the carrier signal. The carrier signal must be extracted from the received waveform and any error in the recovered carrier will cause a corresponding degradation of the entire system.

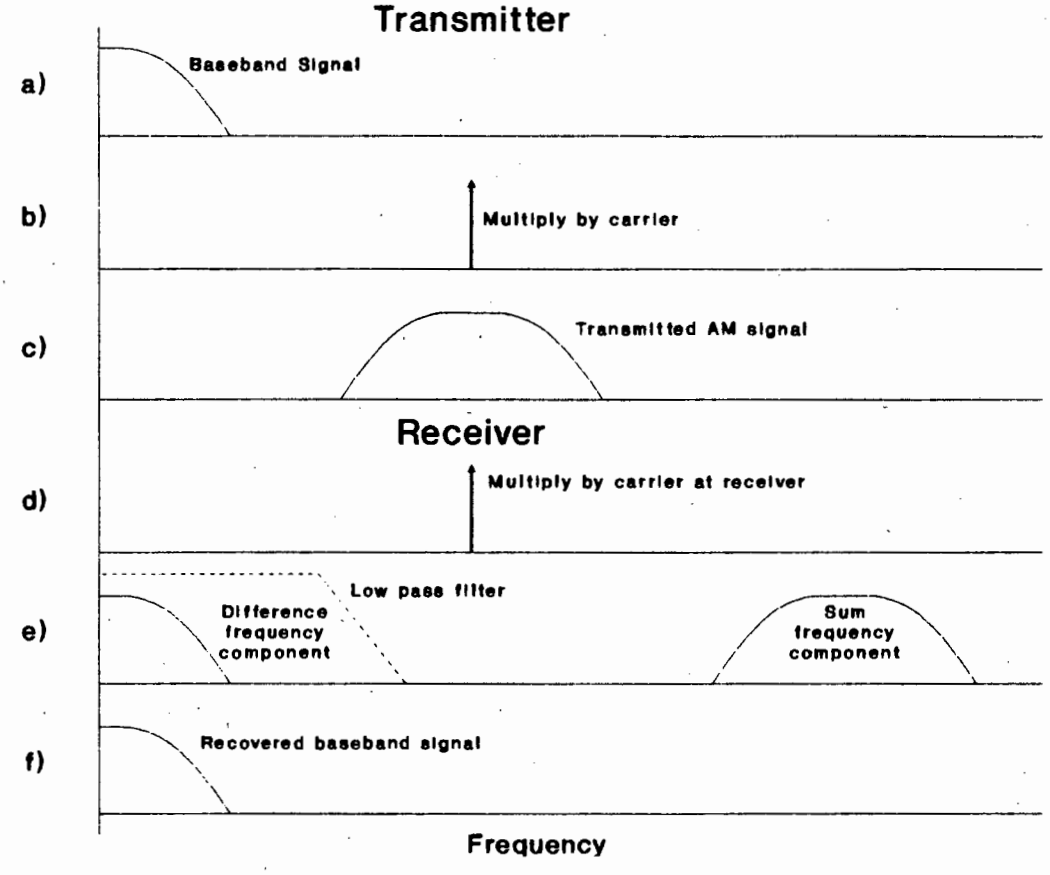


Figure 2.4 Modulation and coherent demodulation of AM signals.

2.2.2 M-ary Pulse Amplitude Modulation (PAM)

The simplest PAM system is the binary case. Here the baseband signal is simply a pulsed version of the two-level data stream which may be bandlimited using a Nyquist filter. M-ary PAM is generated using a baseband signal that has more than two levels. In the case of 4 PAM pairs of binary bits are combined to form a four-level data stream. The advantage of this is that the symbol rate is halved and so the spectral efficiency is doubled. The penalty incurred is that the levels are less widely separated, and so the signal to noise ratio will be degraded.

2.2.3 M-ary Quadrature amplitude modulation (QAM)

QAM is a technique which can be used to double the bandwidth efficiency of conventional AM. The crux of the technique is that two AM signals are transmitted in the same bandwidth, but the carriers of the two signals differ in phase by 90^{o}. This relationship allows the two signals to be separated from each other at the receiver. Consider the following example:

> Suppose a baseband signal g(t) is multiplied by a carrier signal sin(wt) to form the AM signal g(t)sin(wt). This is known as the in-phase or I channel. A second signal is then generated but the phase of the carrier is shifted by 90^{o}, so that it will have the form f(t)cos(wt). This is known as the quadrature or Q channel. The two signals are then added to produce a QAM signal which has the form:

$$q(t) = g(t)\sin(wt) + f(t)\cos(wt) \qquad (2.3)$$

We now consider the demodulation of the I channel. The QAM signal is multiplied by the I channel carrier sin(wt) to produce:

$$\begin{aligned} q(t)\sin(wt) &= (g(t)\sin(wt) + f(t)\cos(wt))\sin(wt) \\ &= g(t)\sin^2(wt) + f(t)\cos(wt)\sin(wt) \\ &= \tfrac{1}{2}g(t)(1 - \cos(2wt)) \\ &\quad + \tfrac{1}{2}f(t)(\sin(2wt) + \sin(0)) \\ &= \tfrac{1}{2}g(t) + \text{higher frequency terms} \end{aligned} \quad (2.4)$$

The process is completed by the addition of a low pass filter which will result in the recovered I channel being recovered. Similarly, the Q channel can be recovered by multiplying the transmitted signal by cos(wt). This process thus doubles the bandwidth efficiency of a normal AM signal and is possible because the two carrier signals sin(wt) and cos(wt), are orthogonal.

2.2.4 16 QAM

16 QAM is formed by using a combination of 4-ary AM and the QAM technique which has been described. Thus the data stream will be split up into two data streams at half the bit rate. Pairs of bits will then be combined in each to form two 4 PAM signals. These are then modulated onto the in-phase and quadrature carriers and added together. The bandwidth efficiency of 16 QAM is typically 2 bps/Hz, as 100% excess bandwidth raised cosine filters are usually used to bandlimit the signal. The constellation diagram, which shows the signal in terms of the orthogonal basis functions sin(wt) and cos(wt), is given in figure 2.5. A probability of error curve of 16 QAM is given in the comparison of QAM and QPRS (see section 3.4).

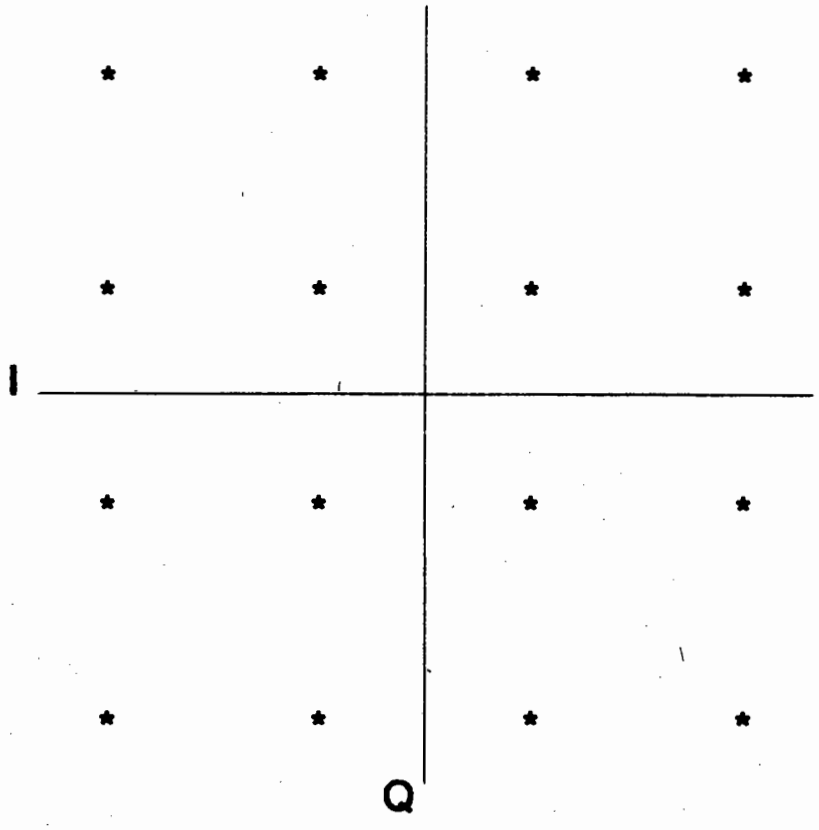


Figure 2.5 Constellation diagram of 16 QAM

REFERENCES

[2.1] Feher, K., **Digital Communications: Microwave Applications**, Prentice-Hall Inc., 1981, Chapter 7.

[2.2] Stremler, Ferrel G., **Introduction To Communication Systems, Second Edition**, Addison-Wesley Publishing Company, 1982, Pg 368.

3. CORRELATIVE LEVEL CODING (PARTIAL RESPONSE SIGNALLING)

3.1 INTRODUCTION TO PARTIAL RESPONSE SIGNALLING

The concept of correlative level coding was discovered in 1962 by Dr Adam Lender who developed many operational duobinary and other partial response systems. PRS has a higher spectral efficiency than memoryless systems making it attractive in bandwidth limited applications. A higher data throughput can be obtained for similar probability of error criteria.

PRS is based on Nyquist's second criterion which states that data can be correctly received as long as there is zero ISI at the transition points of the bandlimited waveform [3.1]. A controlled amount of ISI (usually 100 %) is purposely introduced by combining a number of pulses before transmission, using a digital transversal filter. This operation changes the spectrum of the data in such a way that its bandwidth occupancy is halved, which in turn allows a doubling in the transmission rate for a given bandwidth. Because these pulses are combined in a known way, the original data stream can still be correctly decoded at the receiver. An example, given by Lathi [3.1], will now be used to illustrate this point.

> Consider a system designed to transmit data at a rate of f_0 bits/second using polar signalling. When a "1" is transmitted by a full-width rectangular pulse p(t), the bandwidth is large enough so that the received pulse will rise to positive amplitude K, and when a negative pulse is transmitted by -p(t), the received pulse will rise to negative amplitude -K.
>
> If the transmitted pulse rate is now doubled, the transmitted pulse width is halved, and the received pulses cannot reach their full values. But if a "1" is followed by a "1", we have two half-width pulses in succession, making one full-width pulse. This causes the received pulse to reach full positive value K.

Similarly, if a "0" is followed by a "0", the received pulse reaches full negative value -K. But if a "1" is followed by a "0", or vice versa, the received pulse will stay close to zero. Thus, the received signal can be interpreted as shown in the following table:

Received Amplitude	Transmitted Digit
+K	1 (Previous digit also 1)
-K	0 (Previous digit also 0)
0	Complement of previous digit

Table 3.1 Decoding rule for PRS Example

Thus, data can be unambiguously received even when the data rate is doubled. The penalty incurred is that three levels are now possible at the receiver instead of two. Since the probability of error versus SNR performance is dependent on the spacing between the levels, there will be a corresponding degradation in system performance.

3.1.1 Classes of Partial Response Signalling

Using transversal filters, a wide variety of PRS formats can be realized. All offer increased spectral efficiency but have different spectral weightings and varying degrees of redundancy providing varying amounts of built-in error detection. A few of these are given in figure 3.1 [3.2]. **D** in the generating polynomials indicates a delay of one bit, $\mathbf{D^2}$ indicates a delay of two bit periods and so on.

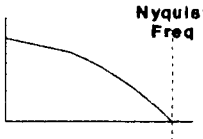
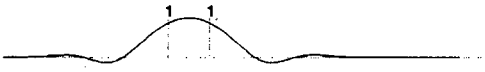
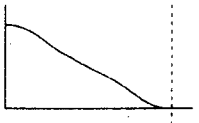
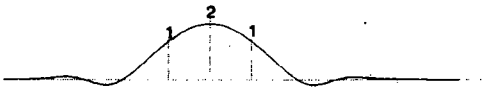
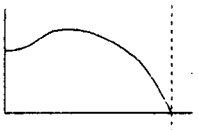
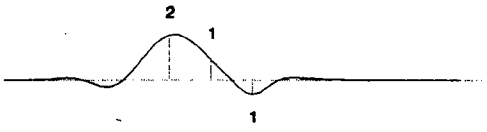
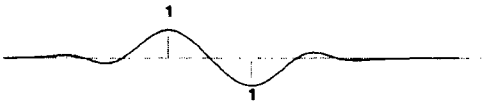
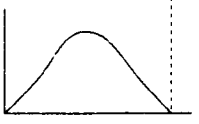
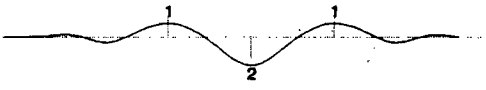
F(D)	FREQUENCY RESPONSE	IMPULSE RESPONSE	No. output levels for M input levels
Class 1 1 + D (Duobinary)			2M - 1
Class 2 $1 + 2D + D^2$			4M - 3
Class 3 $2 + D - D^2$			4M - 3
Class 4 1 - D (Modified Duobinary)			2M - 1
Class 5 $1 - 2D^2 + D^4$			4M - 3

Table 3.2 Classes of partial response signalling systems

It can be seen from the table how the different PRS formats result in different spectra. This is important when determining the format best suited to a particular application. Unfortunately, many of these coding schemes result in an excessive number of output levels, which results in a performance degradation. It is for this reason that class 1 and class 4 PRS are most commonly used as they produce only 2M-1 output levels, where M is the amount of input levels. Class 1 and 4 PRS are also known as duobinary and modified duobinary respectively. Both have a very simple implementation but an important difference is that modified duobinary PRS has a spectral null at DC which is often useful. Its uses would include transformer coupled circuits, DC powered cables, SSB modems, and carrier systems with carrier pilot tones.

3.1.2 Speed Tolerance

Speed tolerance can be defined as the sensitivity of the system to changes in the data rate. It is dependent primarily on the rate at which pulse tails die away. Figure 3.1 shows the speed tolerance of duobinary, modified duobinary and conventional binary signalling as a function of the roll-off parameter of a Nyquist raised-cosine filter [3.2]. It can be seen that duobinary signalling offers the best performance for transmission in the minimum Nyquist bandwidth (α = 0). For this reason it was selected for the 49 QPRS system in this project, as opposed to modified duobinary.

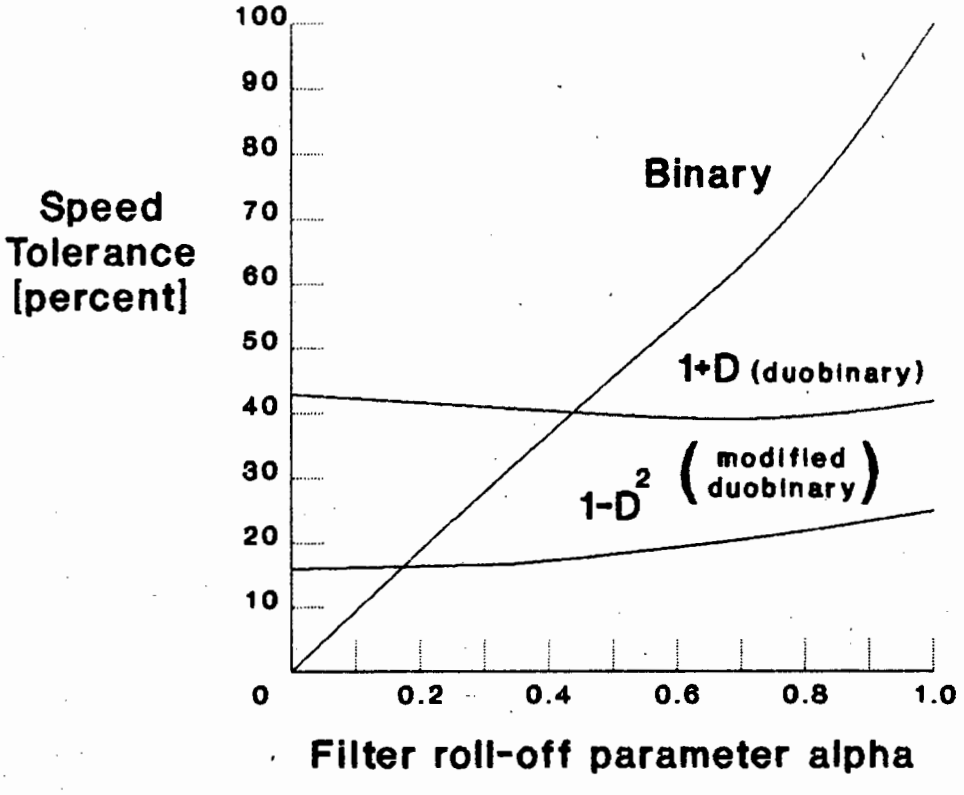


Figure 3.1 Speed tolerance as a function of the roll-off parameter of a Nyquist raised-cosine filter.

Notice that the speed tolerance of binary signalling is zero for α = 0, which supports the concept that conventional zero-memory schemes cannot achieve transmission in the minimum Nyquist bandwidth.

3.1.3 Duobinary PRS

Duobinary PRS, the class of PRS implemented in this project, is the most robust of the PRS formats in terms of error-rate performance, speed tolerance and ease of implementation. It is generated and interpreted according a set of rules which are now described.

Let the input data stream be x_k and let the duobinary output be y_k. Two successive binary input pulses are added so that [3.3]:

$$y_k = x_k + x_{k-1} \qquad (3.1)$$

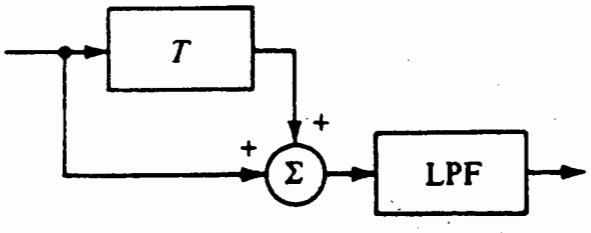


Figure 3.2 Transversal Filter for Duobinary Signalling

The resulting frequency transfer function is:

$$H(w) = (1 + e^{-jwT})_{\text{Low Pass}}$$

$$= \begin{cases} 2e^{-jwT/2} * \cos wT/2 & \text{for } w = \pi/T \\ 0 & \text{elsewhere} \end{cases} \qquad (3.2)$$

The corresponding impulse response is:

$$h(t) = \frac{4 \cos [(t - T/2)/T]}{T [1 - 4(t - T/2)^2/T^2]} \qquad (3.3)$$

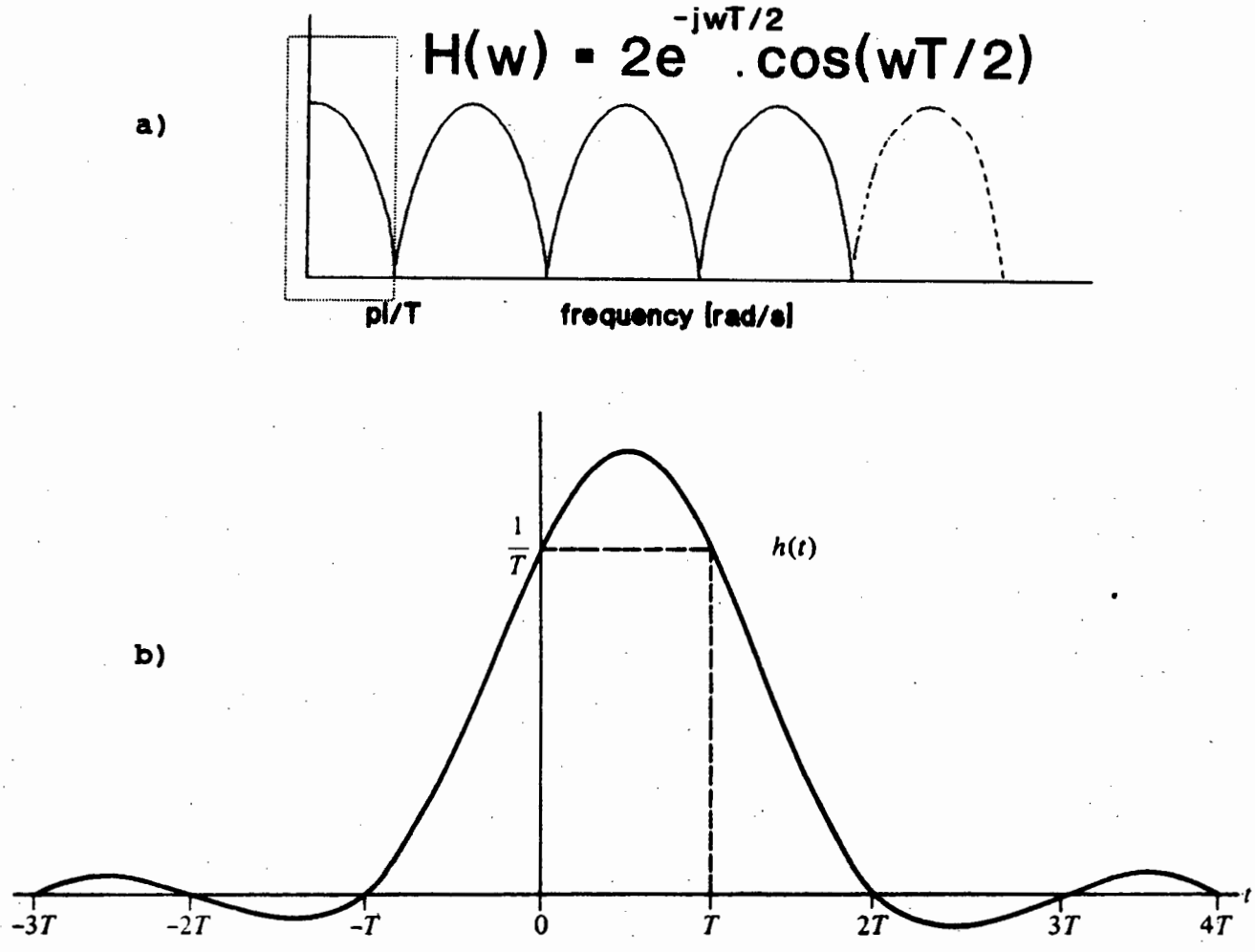


Figure 3.3 Class 1 PRS characteristics.
(a) Duobinary frequency transfer function. The box on the left indicates the lowpass filter operation which produces the baseband signal.
(b) Duobinary unit impulse response

From this it can be seen that the operation of adding two successive symbols provides the transmitted signal with a new spectral weighting. The width of the main lobe is halved, doubling the spectral efficiency and allowing signalling up to the Nyquist rate of 2 bps/Hz for the binary case.

At the receiver, the following decoding rule must now be followed:

$$x'_k = y_k - x'_{k-1} \qquad (3.4)$$

where x'_k is the decoded sequence

3.1.4 Precoding

A receiver using the decoding rule that has been described has to constantly refer to bits that have already been decoded in order to decode each incoming bit. Clearly, this will cause error propagation. This tendency can be eliminated by precoding the input data before transmission. In this procedure, the input data is precoded according to the rule:

$$b_k = (x_k - b_{k-1}) \bmod 2 \qquad (3.5)$$

where b_k is the new, precoded, input data stream.

The binary stream b_k, is now applied to the duobinary filter, yielding:

$$y_k = (x_k - b_{k-1}) \bmod 2 + b_{k-1} \qquad (3.6)$$

From this equation it can be deduced that if $x_k = 1$, then $y_k = 1$ regardless of the value of x_{k-1} and similarly, if $x_k = 0$, then $y_k = 0$ or 2. Thus it is no longer necessary to refer to the past history of the waveform, eliminating error propagation. The new decoding rule is:

$$x'_k = y_k \bmod 2 \qquad (3.7)$$

3.1.5 7-level duobinary PRS

Although the preceding discussion refers to the binary case in particular, it can be extended to M-ary signals. The coding rules for this project, the M = 4 case, are the same except that the modulo-2 operators are replaced by modulo-4 operators. The encoding process then produces a 7-level data stream. A block diagram of the entire coding and decoding process is given in figure 3.4, while an example of the process being applied to data is given in table 3.3. Although the precoder is initialized with zeros in the example, any value could have been used as the choice is arbitrary.

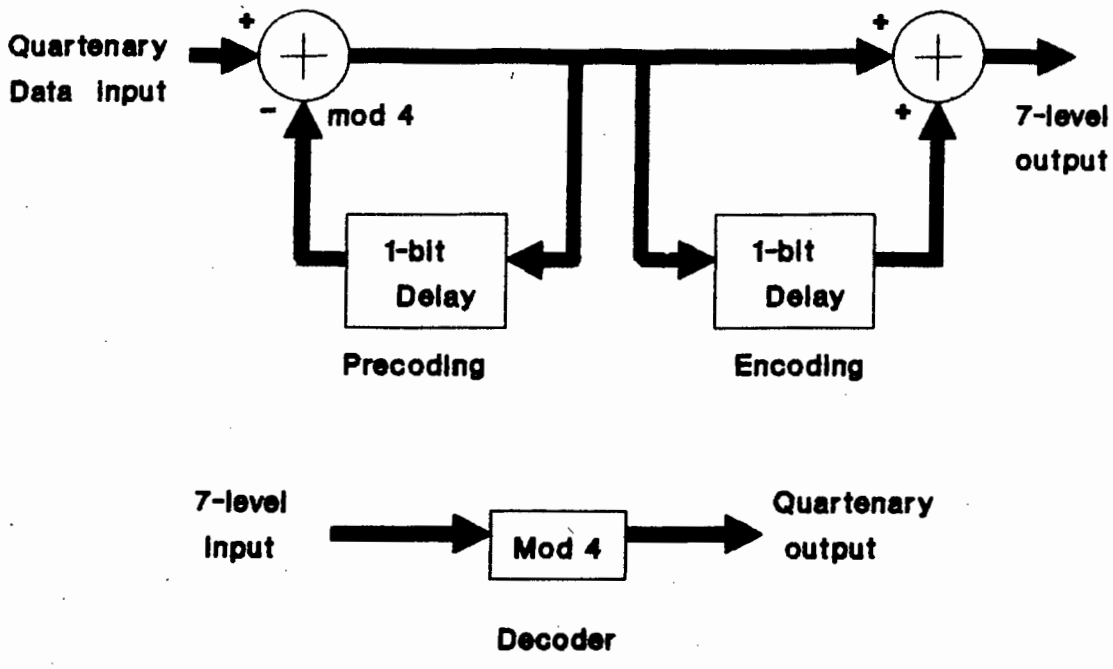


Figure 3.4 Block diagram of duobinary precoder, encoder and decoder.

Input symbols x_k	Precoded symbols b_k	Transmitted symbols y_k	Decoded symbols x'_k
	0		
1	1	1	1
0	3	4	0
2	3	6	2
3	0	3	3
3	3	3	3
0	1	4	0
2	1	2	2
1	0	1	1
2	2	2	2
3	1	3	3
0	3	4	0
1	2	5	1
1	3	5	1
3	0	3	3
0	0	0	0
2	2	2	2
0	2	4	0

Table 3.3 Example of M = 4 duobinary precoding, encoding and decoding.

At this point it is necessary to introduce the concept of an eye diagram. The eye diagram is a convenient way of observing imperfections that occur in modulation systems. Consider a waveform with regularly spaced sampling instants, where the waveform takes on one of a number of discrete levels at the sampling instants. If this waveform is viewed on an oscilloscope which is triggered with the sampling clock, the persistence of the screen will display superposed segments of the signal in such a way that the waveform takes on discrete levels at a certain time position. This is known as the eye diagram. The sharper the eye diagram, i.e. the bigger the eye opening, the more resistant the system will

be to noise, timing jitter and drift in the receiver sampling instant. The 7-level eye diagram of a bandlimited M=4 PRS signal is given in figure 3.5.

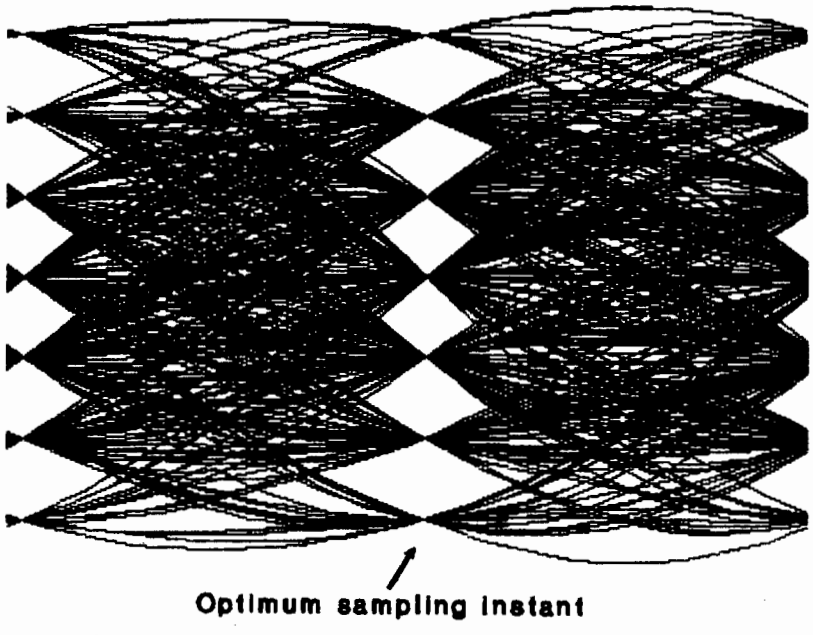


Figure 3.5 Eye diagram of 7-level class 1 PRS

3.2 ERROR PERFORMANCE OF PRS

3.2.1 Error detection using inherent redundancy

A unique characteristic of correlative coding is that it introduces inherent redundancy into the bitstream which can be used to detect errors. Consider the 3-level duobinary scheme:

a) If one of the outer levels is present at a particular sampling instant then the opposite outer level may not occur at the next sampling instant.

b) An outer level followed by the opposite outer level must be separated by an odd number of center samples.

c) An outer level followed by the same outer level must be separated by an even number of center samples.

A relatively simple system could be set up to monitor these violations. It must be pointed out that certain errors, or combinations of errors, may not cause violations of these rules. Nevertheless, the system can be used as a very convenient way of monitoring the error rate on a transmission link. Each of the PRS formats has a unique set of rules which can be utilized for error detection.

3.2.2 Bit-by-bit detection and decoding

The bit-by-bit detection and decoding technique, although suboptimum, is usually used in PRS because of its simplicity and ease of implementation. This method of detection, which is the method used in this project, entails sampling data at the point of maximum eye opening and decoding the sample without considering any other sample values. In the duobinary case, it simply requires a modulo-M operation. The reason that bit-by-bit detection is suboptimum is that the inherent redundancy of PRS is not exploited.

3.2.3 Optimal decoding (maximum-likelihood detection)

Maximum-likelihood sequence estimation is a technique which makes full use of redundancy in a sequence to correct a certain amount of errors. As already mentioned, the increase in the amount of signalling levels required for PRS will result in a performance degradation. As a result, the duobinary scheme requires approximately 3 dB more signal-to-noise ratio than the corresponding PAM scheme. Kobayashi [3.5] and Forney [3.6] showed that this degradation it due to the suboptimality of the bit-by-bit detection method. They showed that the maximum-likelihood detection algorithm could be used to recover almost all the loss in performance. The disadvantages of such a decoder are that it is relatively complex and a decoding delay is introduced [3.2]. The technique will not be discussed any further as it was not implemented in this thesis.

3.2.4 Probability of error in partial response signalling

There are two different methods for evaluating the probability of error, namely with and without precoding. The latter case will not be discussed as precoding is a very simple operation and is used in every practical PRS system to prevent error propagation. The probability of error of a precoded system is approximately: [3.2]

$$P_e \leq 2(1 - 1/m^M)Q(d/\sigma) \qquad (3.8)$$

where:

σ^2 is the noise variance at the decoder
d is the decision distance (half the level separation)
M is the amount of pulse samples
m is the amount of possible input levels (as in m-ary)
Q(x) is defined as $1/(2\pi)^{\frac{1}{2}} \int_{-\infty}^{x} \exp(-u^2/2)du$

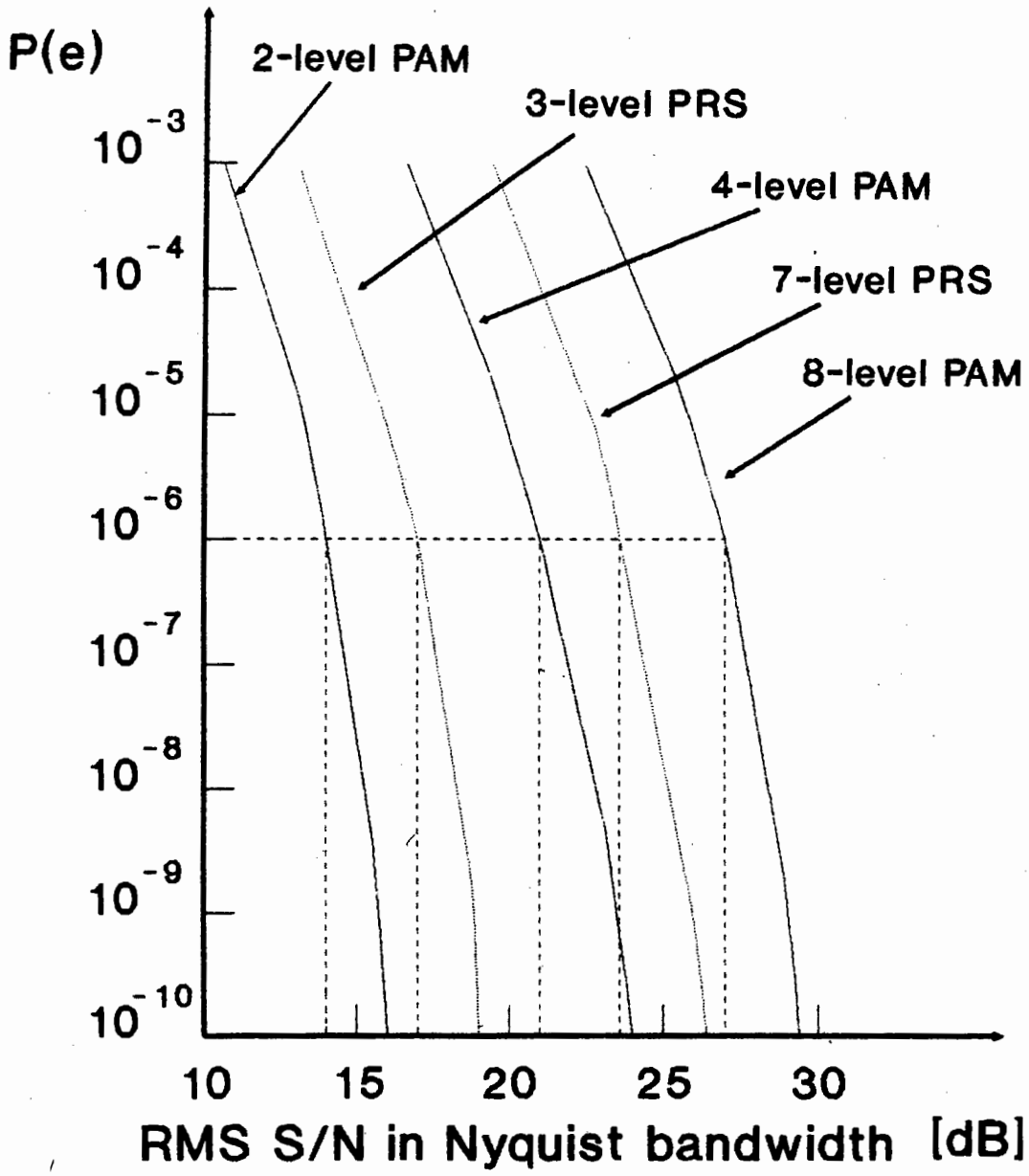


Figure 3.6 Probability of error curves of M-Level PAM and PRS [3.6]

From the P(e) curves it can be seen that PRS (bit-by-bit-detection) requires approximately 3 dB more signal to noise ratio to the corresponding PAM system for the same probability of error. Although PRS requires somewhat more signal to noise ratio, this is offset by the fact that its information carrying capacity is up to twice that of PAM, due to its high spectral efficiency. Bit error rate curves of QPRS are given in section 3.4, where QPRS systems are compared to QAM systems.

3.3 QUADRATURE PARTIAL RESPONSE SIGNALLING (QPRS)

3.3.1 Generation and demodulation

The generation of QPRS can be considered to be a special case of the QAM technique which has been described in section 2.2. QPRS is generated and demodulated as follows:

Transmitter operation:

a) The data stream that is to be transmitted is split into two streams each having half the bit rate (I and Q channels).

b) If an M-ary system is required (to further reduce bandwidth) then bits are grouped to form M-level symbols.

c) The data streams are applied to a PRS precoder.

d) The data streams are correlatively encoded using a transversal filter.

e) A square-root Nyquist low pass filter is used to bandlimit the data streams.

f) The I and Q channels are modulated onto in-phase and quadrature carriers.

g) The I and Q channels are then added resulting in a QPRS signal.

Receiver operation:

a) The received signal is multiplied by in-phase and quadrature carriers to produce the I and Q channels.

b) A square-root Nyquist lowpass filter (matched to the transmitter) then filters off unwanted components to produce the baseband signals.

c) The baseband signals are sampled yielding data streams of the correlatively coded values.

d) PRS decoding is performed to give the M-level data streams.

e) M-level to binary conversion is performed.

f) The two binary streams are combined to form the original data stream.

3.3.2 49 QPRS

49 QPRS is formed using the steps described in the preceding section. The M-level data stream will be a 4-level data stream in this case. The correlative coding used in this project converts the 4-level data to 7-level PRS data. 49 QPRS has a bandwidth efficiency of 4 bits/sec/Hz.

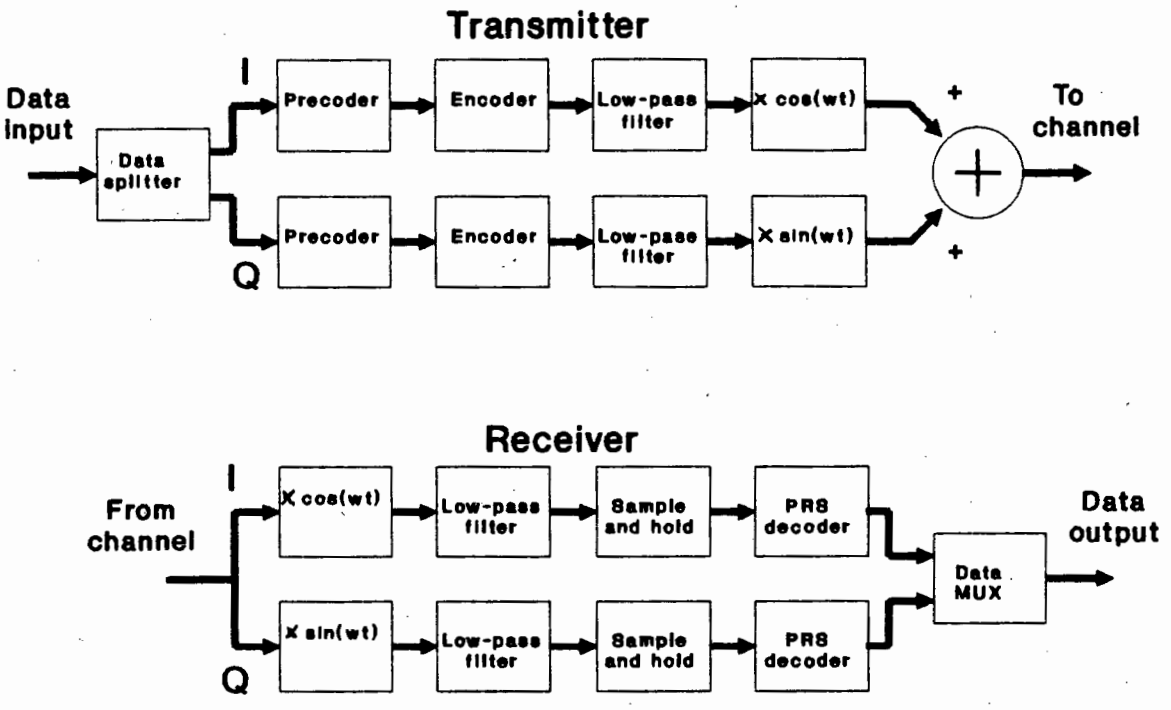


Figure 3.7 49 QPRS modulation block diagram

The constellation diagram of QPRS, given in figure 3.8, shows how the 7-level in-phase and quadrature channels are combined to form 49 constellation points, hence the name 49 QPRS.

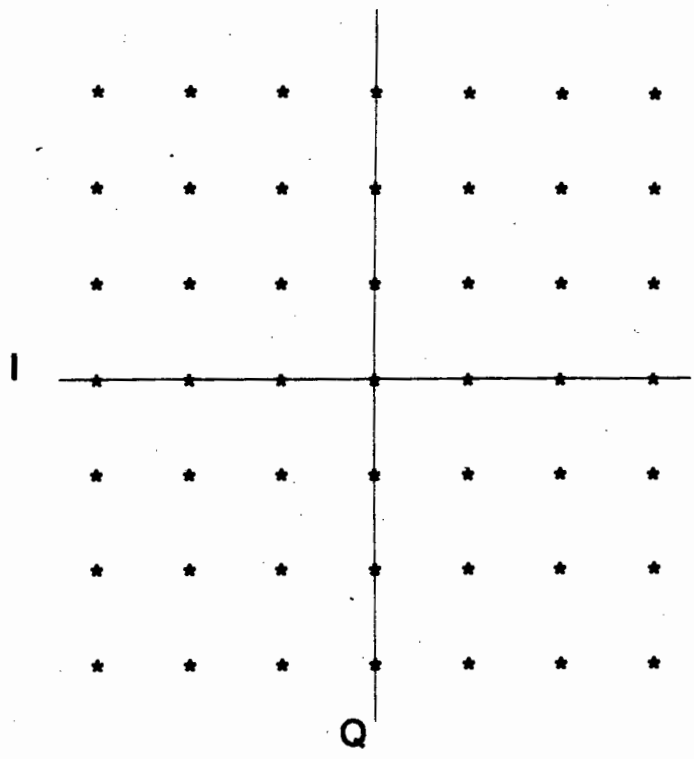


Figure 3.8 Constellation diagram of 49 QPRS

3.4 COMPARISON OF 16 QAM AND 49 QPRS

In this section, the differences and similarities between QAM and QPRS systems are highlighted. This comparison is relevant because the existing V.22bis scheme uses 16 QAM, while the proposed scheme uses 49 QPRS.

From the preceding sections, it can be seen that QPRS is essentially an extension of QAM. Both involve modulating two PAM type waveforms onto in-phase and quadrature carriers, and then adding the two together. The design complexity of QPRS is virtually the same as that of QAM, provided bit-by-bit detection is used. The only major difference lies in the coding of the data prior to modulation which allows bandlimiting at, or very close to, the Nyquist frequency. Bandlimiting in QAM systems must typically be carried out using excess bandwidths up to 100 %.

Memoryless systems typically require more signaling levels in order to achieve the same bandwidth efficiency as PRS systems. Error performance will always worsen as the amount of signaling levels is increased. A baseband modem with a spectral efficiency of 2 bits/s/Hz requires three levels in the case of PRS but four in the case of memoryless systems. When the technique is extended to M-ary systems, this advantage increases. This is shown in the following table [3.7]:

Spectral efficiency [bits/s/Hz]	Number of levels	
	Zero memory (alpha = 1)	Correlative coding
1	2	
2	4	3
3	8	
4	16	7
5	32	
6	64	15
7	128	
8	256	31

Table 3.4 Comparison of Correlative with Zero Memory (α=1) systems

This thesis deals with the 4 bits/s/Hz case, formed by the superposition of the in-phase and quadrature channels each transmitting at 2 bits/s/Hz. It thus requires a seven-level data stream to be transmitted. This greatly relaxes timing and SNR requirements to those corresponding to 16-level, zero-memory system. The following curves show the relative performance of QAM and QPRS (bit-by-bit detection):

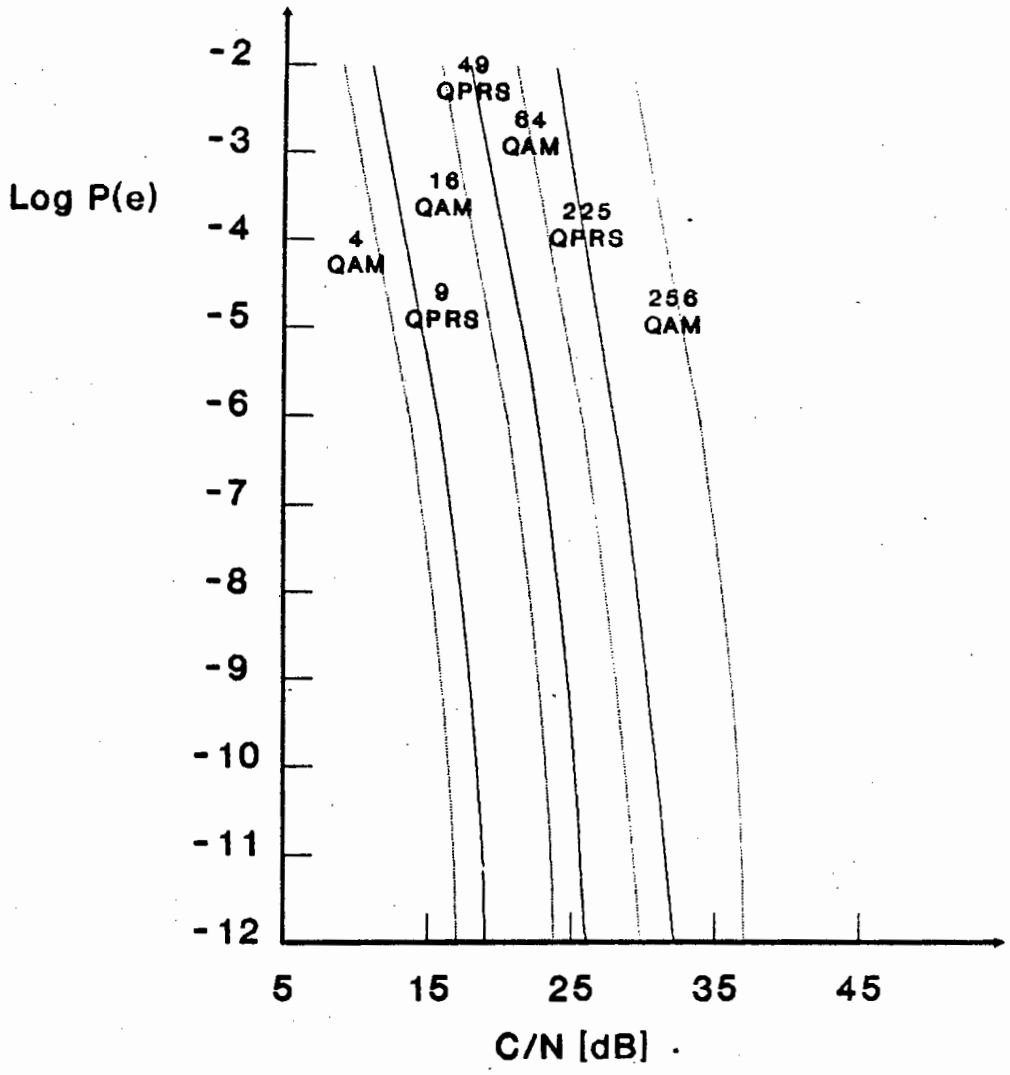


Figure 3.9 Probability of error curves for Multilevel QAM and QPRS modulation schemes [3.5].

The only reasonable methods of achieving 4800 bps in the available bandwidth for this project, if frequency division multiplexing is to be used, are 256 QAM and 49 QPRS (64 QAM does not have enough bandwidth efficiency). From the curves it can be seen that the performance of 256 QAM is far poorer than that of 49 QPRS, as it requires approximately 10 dB more carrier-to-noise ratio for the same probability of error. 49 QPRS on the other hand, requires only 2 to 3 dB more carrier-to-noise ratio than 16 QAM, which is the system currently employed to achieve 2400 bps transmission (CCITT V.22bis).

REFERENCES

[3.1] Lathi, B.P., **Modern Digital and Analog Communication Systems**, CBS College Publishing, 1983, Pg 158.

[3.2] Pasupathy, S. and Kabal, P., "Partial Response Signaling", IEEE Transactions on Communications, Volume com-23, no.9, September 1975.

[3.3] Stremler, Ferrel G., **Introduction To Communication Systems, Second Edition**, Addison-Wesley Publishing Company, 1982, Pg 539.

[3.4] Pasupathy, S., "Correlative Coding - A Bandwidth Efficient Signalling Scheme", IEEE Communications Society magazine, July 1977.

[3.5] Kobayashi, H., "Correlative level coding and maximum-likelihood decoding", IEEE Transactions on Information Theory, Vol. IT-17, September 1971, pp 586-594.

[3.6] Forney, G. D., Jr., "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference", IEEE Transactions on Information Theory, Vol. IT-18, May 1972, pp 363-378.

[3.7] Feher, K., **Digital Communications: Microwave Applications**, Prentice-Hall Inc., 1981, Chapter 7.

4. GENERAL CONCEPTS CONCERNING THE PROPOSED APPLICATION

4.1 CCITT RECOMMENDATION M.1020 TELEPHONE LINES

Ordinary voice grade telephone lines comply with CCITT recommendation M.1040. The quality of these lines is relatively poor and as a result they are only suitable for voice and low speed modem applications. In order to transmit data at speeds of 4800 bps and greater, special quality lines are provided which conform to CCITT recommendation M.1020. The modem scheme investigated in this thesis is intended to operate over M.1020 lines.

In order to provide special quality M.1020 circuits, cable pairs are specially selected and additional equalizers connected to correct the line parameters over the range 200 Hz to 3000 Hz [4.1]. These include the amplitude vs. frequency response, group delay vs. frequency response and harmonic distortion. The following graphs describe the limits on the amplitude and group delay response of an M.1020 telephone line. The actual response may not pass through the shaded area.

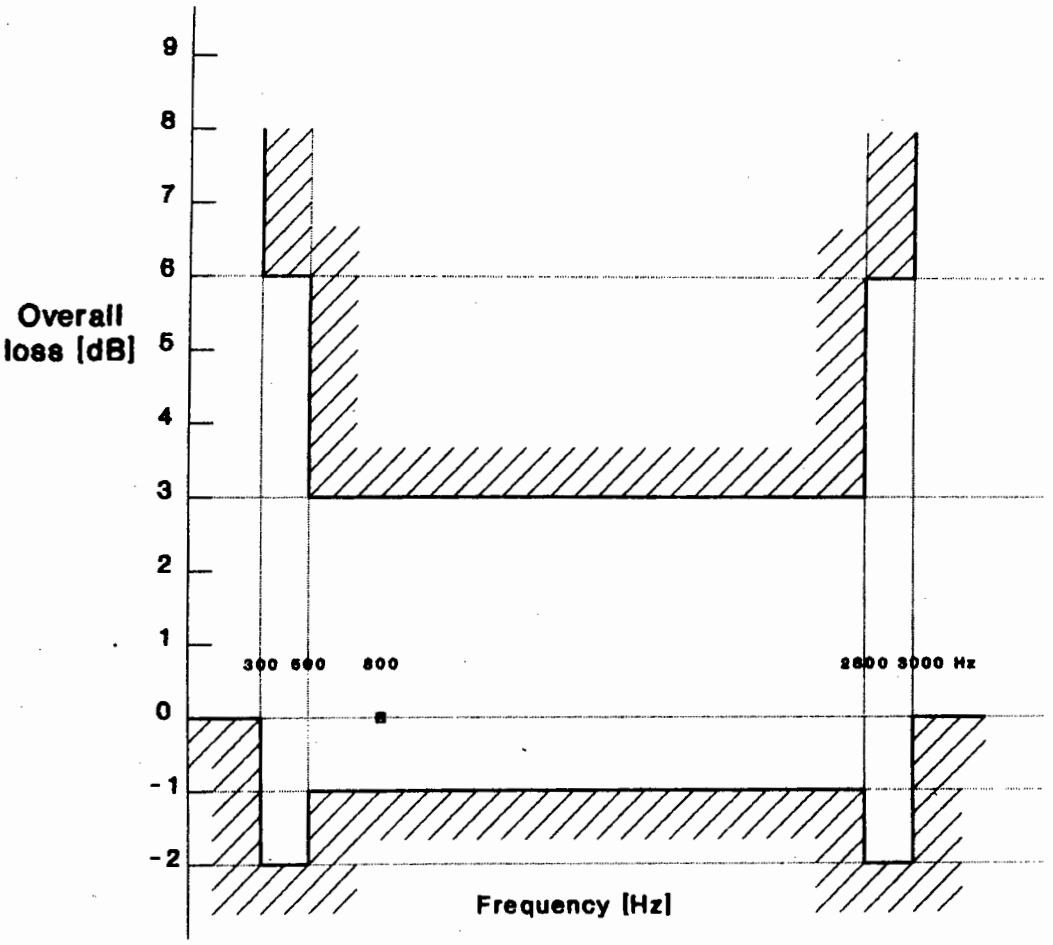


Figure 4.1 M.1020 limits for overall loss of the circuit relative to that at 800 Hz

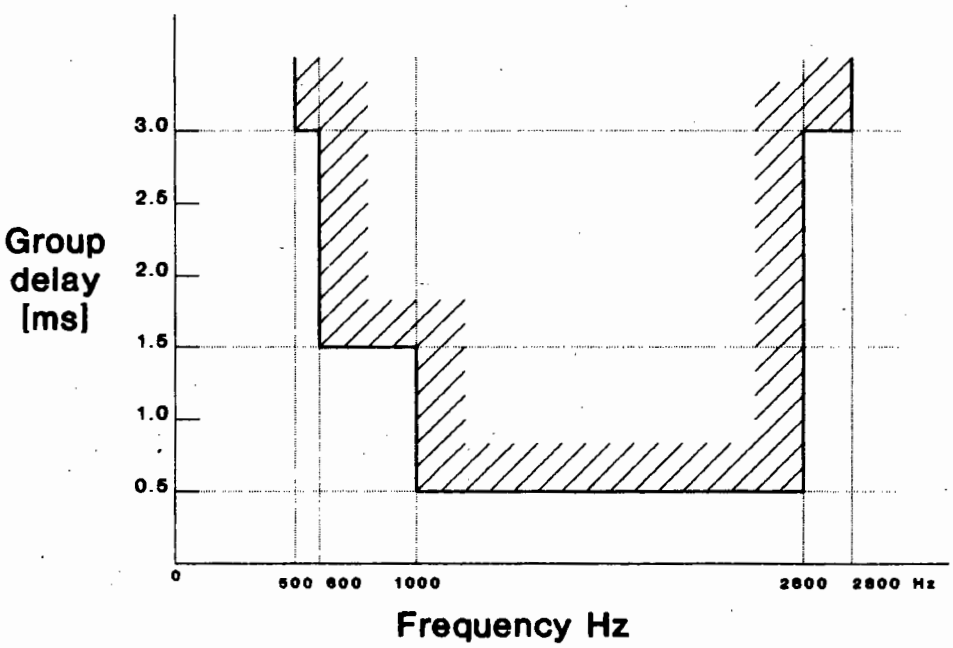


Figure 4.2 M.1020 limits for group delay relative to the minimum measured group delay in the 500-2800 Hz band.

From the figures it can be seen that the worst case line parameters may be extremely poor, especially group delay, which is only specified from 500 Hz to 2800 Hz. The actual parameters, however, are usually far better and high speed modems such as those conforming to V.27 (4800 bps) and V.29 (9600 bps) specifications are designed to operate on these circuits. If inadequate line parameters are encountered, modems usually operate at a fall-back rate so that transmission is still possible. Adaptive equalization can also be used to improve performance over a wide range of conditions.

In addition to those already given, M.1020 circuits also conform to requirements that include nominal overall loss, variation of overall loss with time, random noise, impulsive noise, phase jitter, quantizing noise, single tone interference and frequency error.

4.2 THE V.22BIS MODULATION SCHEME

The CCITT recommendation V.22bis modulation scheme will now be briefly described [4.2]. V.22bis uses a 2400 bps, full-duplex modulation scheme where data travelling in opposite directions is kept separate by frequency division multiplexing.

The modulation technique used is 16 QAM, which was described in detail in section 2.2. The transmitted signals are bandlimited using raised-cosine filtering. The filtering is shared equally between the transmitter and receiver so that square-root raised-cosine filtering is implemented at each side. The excess bandwidth used is 75 %. Guard tones are transmitted along with the data. A Guard tone of 550 Hz is transmitted with the low band, while a Guard tone of 1800 Hz is transmitted with the high band. The frequency spectra of the low and high band are given in figure 4.3.

At the receiver, notch filters at 550 Hz and 1800 Hz are used to remove the guard tones. If the high band is being received, a bandpass filter centered at 2400 Hz is used to reject any components that do not form part of the high band signal. Similarly, the low band is isolated using a bandpass filter centered at 1200 Hz. If the spectra in figure 4.3 are observed, it can be seen that the frequency separation between the two bands is only 150 Hz. As a result, the bandpass filters have a high roll-off factor, which in turn causes amplitude and group delay distortion near the cut-off frequency. In order to compensate for these distortions, an equalizer is also included in the system.

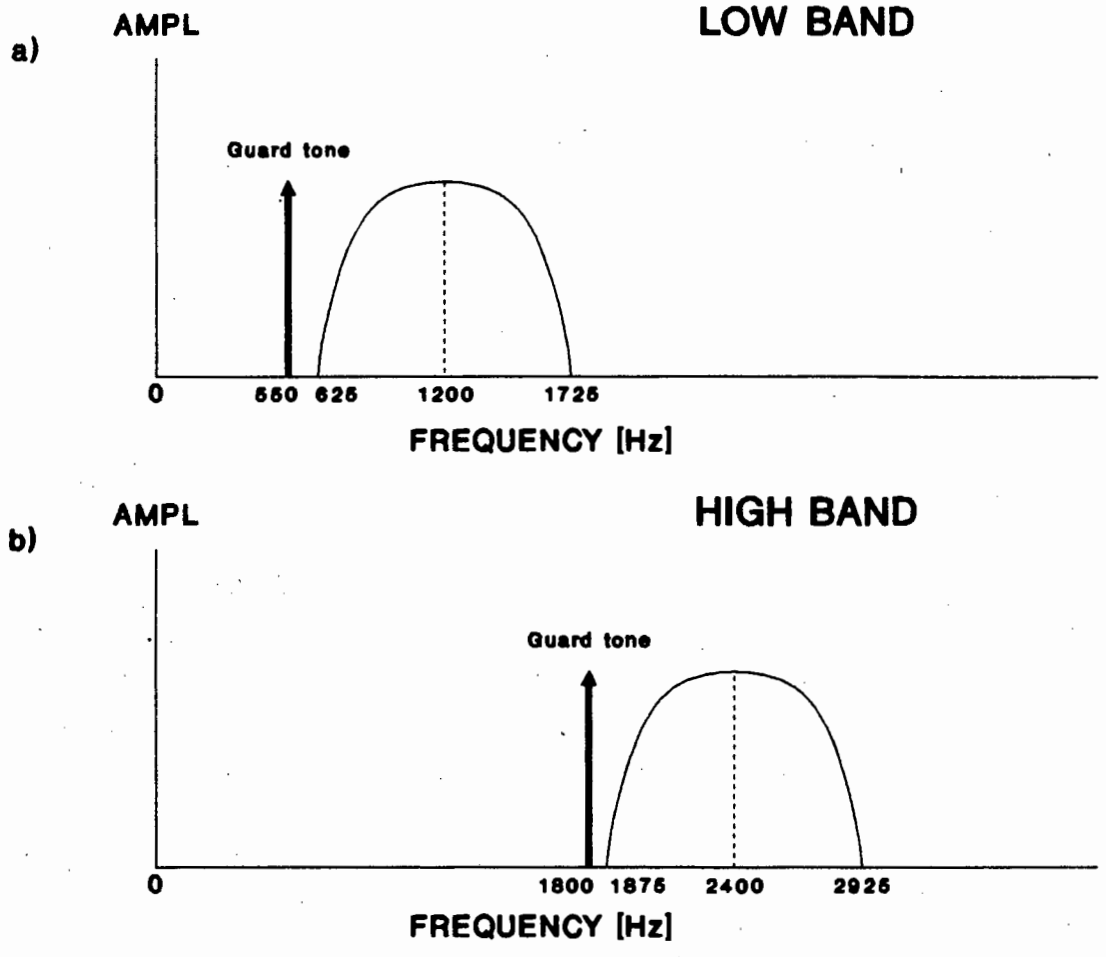


Figure 4.3 The V.22bis modulation scheme
a) Low band
b) High band

4.3 CROSSTALK IN FULL-DUPLEX DATA TRANSMISSION

There are various methods of transmitting data in both directions between two modems over telephone wires. The basic criterion is that the interference between the transmit and receive channels, known as crosstalk, should be kept to a minimum. Crosstalk acts in a similar way on the signal as noise does and can severely degrade the system. The term 'crosstalk' could also refer to interference between two separate telephone circuits. The discussion that follows, however, will only refer to crosstalk between two channels using the same telephone circuit.

In simplex transmission systems, crosstalk is completely avoided by using a separate circuit for the each data direction. Half-duplex schemes use only one circuit, and avoid crosstalk by transmitting in one direction at a time.

Full-duplex transmission allows transmission in both directions simultaneously over a single telephone channel. There are various schemes that achieve full-duplex transmission but before they are discussed, it is worth mentioning the function of the hybrid transformer.

The hybrid transformer effects a four wire to two wire conversion and is used in all standard telephones. The signal on the telephone line is the sum of the transmitted and receive signals. The hybrid uses a series of transformer windings to subtract the transmitted signal from the signal on the line to isolate the received signal. A problem is that it relies on impedance matching to the telephone wire, which should be 600Ω, but may deviate in practice. As a result of this limitation, a hybrid transformer will only isolate the received signal up to a point. On M.1020 telephone lines, the line impedance is conditioned, which increases the effectiveness of the hybrid. Nevertheless, a further method of isolating the received signal is usually required and some of the more important techniques will now be briefly mentioned.

4.3.1 Echo cancellation

Echo cancellation networks have been designed to isolate the received signal from the composite signal on the telephone line. Echo comes from both the far end and near end of the telephone line and a practical echo canceler will usually be adaptive. The use of echo cancellation has the advantage that it allows the whole usable bandwidth to be used by both channels. The disadvantage of echo cancellation is that the technique is considerably complex.

4.3.2 Time compression multiplexing

This is a full-duplex scheme in which the modems do not transmit at the same time. Instead, bursts of data are compressed and transmitted at slightly more than twice the average bit rate. Thus the scheme does not have to rely on good echo suppression at the receiver. Its drawback is that, as the range between the modems increases, there will be an increasing delay between the time that a burst is sent and the time that it is received. The receiving modem must account for this time before sending its burst of data. This is why modems employing TCM are usually range limited.

4.3.3 Frequency division multiplexing

This is a reliable and cheap scheme utilized by V.22 type modems, and is the scheme used in this project. The usable bandwidth is split into an upper and lower band, which are used by the transmit and receive channels. This is achieved by modulating the baseband signals onto different carrier frequencies. Crosstalk is effectively limited by using either a lowpass or a highpass filter, depending on which channel is being received. Care has to be taken that the signals are properly confined to their frequency bands. The disadvantage of FDM is that each channel can only use half of the usable bandwidth, with a corresponding decrease in data throughput.

4.4 MATCHED FILTERING AND SQUARE-ROOT RAISED-COSINE FILTERING

4.4.1 Matched filtering

The matched filter is widely used in telecommunications. The function performed by the matched filter is to maximize the signal-to-rms-noise-amplitude ratio at the decision-making instant of the receiver. In any sample-and-decide threshold detection system operating in the presence of white noise, the matched filter will yield optimum performance [4.3]. The important characteristic of the matched filter is that its impulse response is a time-reversed version of the pulse shape that it is being used to detect.

The impulse and frequency response of a matched filter are given by the formulas:

$$h(t) = k.f(Td-t) \qquad (4.1)$$

$$H(w) = k.F(-w)e^{-jwTd} \qquad (4.2)$$

where:

k is an arbitrary constant

Td is the decision-making instant

4.4.2 Square-root raised-cosine filtering

It has already been shown that a raised-cosine filter is a convenient method of bandlimiting the signal at the Nyquist frequency. However, in order to satisfy both Nyquist and matched filter criteria, a square-root raised-cosine filter, as opposed to a raised-cosine filter, should be used at both transmitter and receiver. The frequency response of this filter will be the square-root of a raised-cosine frequency response. The total frequency response of the system will be the product of the two square-root responses which will be the standard raised-cosine response. In this project, square-root raised-cosine filters were implemented digitally, and are further explained in section 6.2.

4.5 QUANTIZATION NOISE

In any system where an analog signal is stored or generated as a series of digitized samples, it is necessary to divide its vertical amplitude into a series of discrete levels, called quantization levels. The signal is then quantized by assigning it the quantization level values which are nearest to the sampled values. Once a signal is quantized, a certain amount of information is lost, so that the original signal can never be completely recovered from the quantized signal [4.4]. The difference between the sampled signal and the quantized signal is known as the quantization error. These errors, which will be random, are the cause of quantization noise. By increasing the number of quantization levels (i.e. by decreasing the quantization step size), the maximum error will be decreased, with a corresponding decrease in quantization noise. Quantization noise must be taken into account when selecting devices such as D/A's. The peak signal-to-quantization noise ratio can be expressed in decibels using the formula [4.5]:

$$\frac{S}{N}_{dB} = 4.8 + 20\log_{10}n \qquad (4.3)$$

where n is the number of quantization levels.

The following table shows the quantization noise that can be expected from various devices

Device	Number of bits	Number of levels	Quantization noise [dB]
Commonly used D/A's and A/D's	8	256	-53
	10	1024	-65
	12	4096	-77
	16	65536	-101
DSP56001 data bus	24	16777216	-149

Table 4.1 Quantization noise in digital systems

4.6 A TYPICAL MODEM CHIP - THE SC11066

The SC11066 is a complete 2400 bps modem VLSI IC containing all modem functions except adaptive equalization [4.6]. It is used in conjunction with an external controller to provide data rates compatible with Bell 103 and Bell 212A as well as CCITT V.21, V.22 and V.22bis standards.

The SC11066 includes:

a) A Full transmitter consisting of:
   Async to sync converter
   Scrambler
   Data encoder
   75 percent square root of raised cosine pulse shaper
   Quadrature modulator
   FSK (Bell 103 and CCITT V.21) modulator
   Hybrid

b) High band and low band filters

c) High band and low band compromise equalizers

d) V.22 notch filter (selectable at 550 or 1800 Hz)

e) Transmit smoothing filter

f) Programmable attenuator for transmit level adjust

g) DTMF, 550 Hz, 1800 Hz, 1300 Hz, and 2100 Hz tone generator.

h) Transmit clock circuit for synchronous operation

i) Pattern generator for generating fixed digital patterns in handshaking mode.

j) Receive section consisting of:
   64-step programmable gain controller (AGC)
   Energy detector at the output of the PGC
   Hilbert transformer
   Quadrature demodulator with low pass filters
   Baud timing recovery circuit
   FSK demodulator
   Sync to async converter

k) 8-bit analog to digital converter

l) Control and status registers

m) 8-bit microprocessor interface with interrupt and multiplexed address/data lines

n) Audio output with level adjustment

In this thesis project a 4800 bps scheme is investigated which, if required for production, would be integrated into an IC like the SC11066. Most of the functions mentioned would still be required and these are not investigated in this report. The new sections which would be necessary are as follows:

a) PRS data encoder.

b) Digital FIR (finite impulse response) square-root raised-cosine filter for pulse shaping with minimal excess bandwidth.

c) All digital quadrature modulator and demodulator.

d) Clock recovery schemes for both bands.

REFERENCES

[4.1] Department of Posts and telecommunications, Specification for ampitude/delay equalizer for the conditioning of voice grade circuits in accordance with CCITT Recommendation M.1020, Provision of ordinary quality and special quality data lines as gazetted on 1983/05/20.

[4.2] Folts, H. C. **McGraw-Hills compilation of Data Communications Standards**, McGraw-Hill Publications Company, New York, 1982.

[4.3] Lathi, B.P., **Modern Digital and Analog Communication Systems**, CBS College Publishing, 1983, Pg 503.

[4.4] Feher, K., **Digital Communications: Microwave Applications**, Prentice-Hall Inc., 1981, Pg 11.

[4.5] Stremler, Ferrel G., **Introduction To Communication Systems, Second Edition**, Addison-Wesley Publishing Company, 1982, Pg 511.

[4.6] "2400 Bit per Second Modem Analog Peripheral", Sierra Semiconductor corporation.

5. THE PROPOSED 49 QPRS APPLICATION

5.1 THE PROPOSED MODULATION SCHEME

5.1.1 The transmitted waveforms

The proposed modulation scheme is designed to be as close as possible to the V.22bis (16 QAM) scheme, which it is intended to upgrade. The carrier frequencies of 1200 Hz (low band) and 2400 Hz (high band), are the same as those used in the V.22bis standard. The major difference is that the 16 QAM modulation scheme of V.22bis is now replaced by 49 QPRS. The guard tones at 550 Hz and 1800 Hz in V.22bis are also changed. A 400 Hz pilot tone is transmitted with the high band while a 600 Hz pilot tone is transmitted with the low band. The function of these tones is explained in detail in section 5.2.

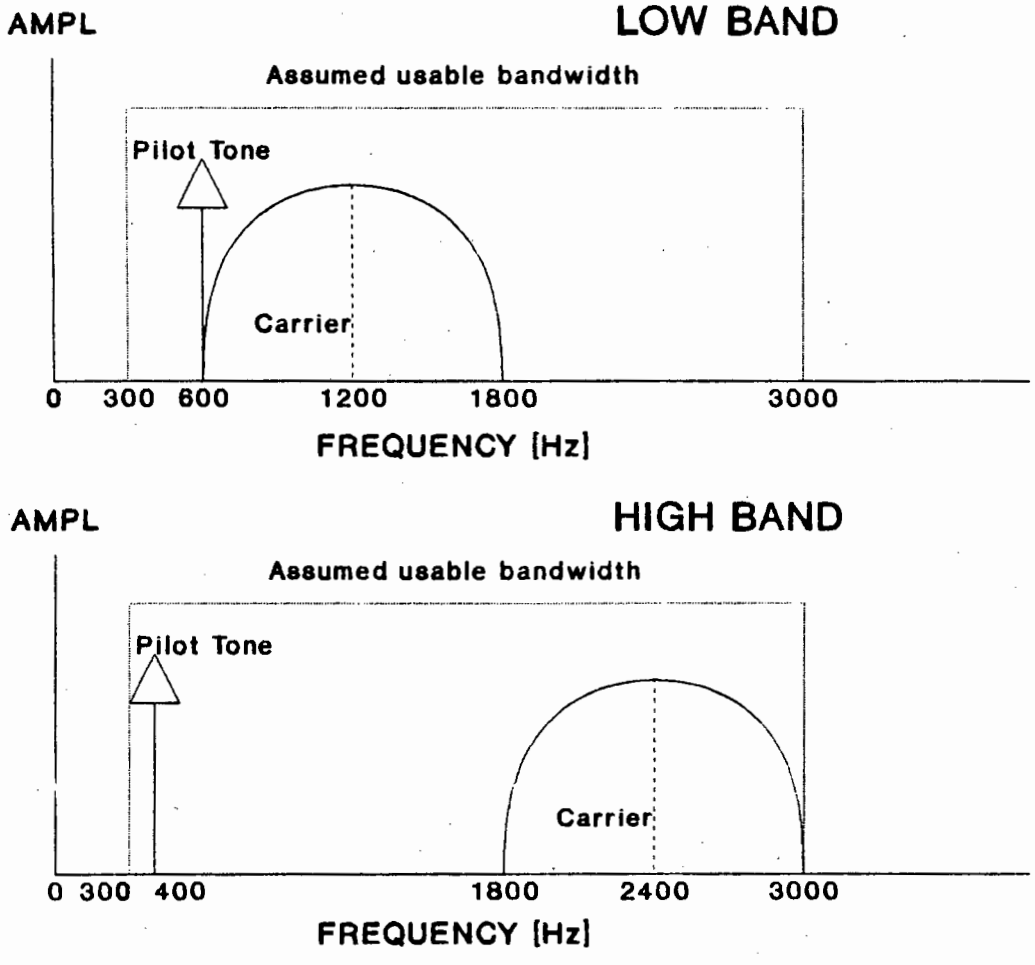


Figure 5.1 Frequency plan of proposed modem scheme

From the diagram it can be seen that practically all the available bandwidth is used. There is also no guard band between the two channels as both extend to 1800 Hz. Fortunately, there is a null at this point in the duobinary frequency response and this, along with the fact that digital filter techniques are used, allows a very sharp cut-off to be achieved. Good out-of-band suppression can also be achieved. The spectra that were obtained of the transmitted signals confirm these points. The resistance of the scheme to crosstalk caused by any frequency overlap was investigated by computer simulation and is discussed in chapter 9.

5.1.2 Demodulation of the transmitted signals

The scheme is designed so that it is not necessary to do any bandpass filtering on the received signal. This is because the baseband filter (matched to the transmitter filter) at the receiver will reject all the unwanted components, including echo from the signal being transmitted in the opposite direction. Figure 5.2 illustrates the recovery of the high and low band signals. The in-phase or quadrature channel will be recovered depending on the phase of the recovered carrier by which the received signal is multiplied. Once again, the null at the Nyquist frequency of PRS and the high order digital receiver filter allows critical filtering requirements to be met with relative ease.

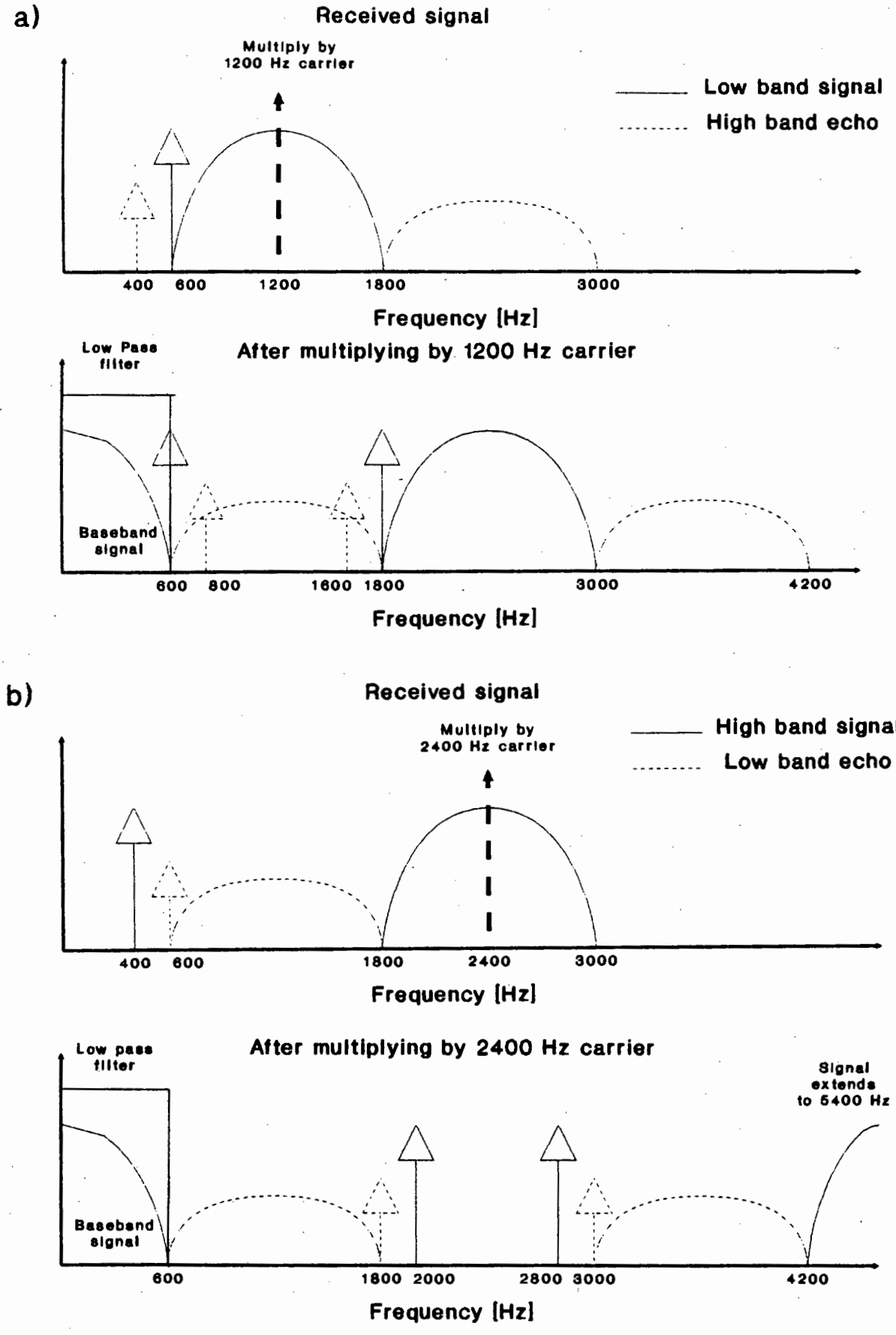


Figure 5.2 Demodulation of the 49 QPRS signals
a) Low band demodulation
b) High band demodulation

5.1.3 Zero-crossing Pilot Tone scheme

In the case of the low band the 600 Hz pilot tone will be situated exactly on the band edge and will not be removed, but will only be slightly attenuated, by the receiver filter. This, however, does not cause interference as a result of the zero-crossing scheme which is used. The new scheme allows a pilot tone to be transmitted along with the signal without requiring a guard band between the tone and the signal. The main advantage of this is that no extra bandwidth is required to transmit the pilot tone. The pilot tone is transmitted at the Nyquist frequency (600 Hz), which is very convenient as all clock frequencies are directly related to this frequency. The crux of the scheme is that the zero crossings of the pilot tone coincide exactly with the receiver sampling instants. The digital nature of the hardware ensures that this relationship is exact. Also, all Nyquist filtering is implemented by finite impulse response digital filters which have linear phase. As a result, these filters will also not affect the phase relationship between the pilot tone and the signal.

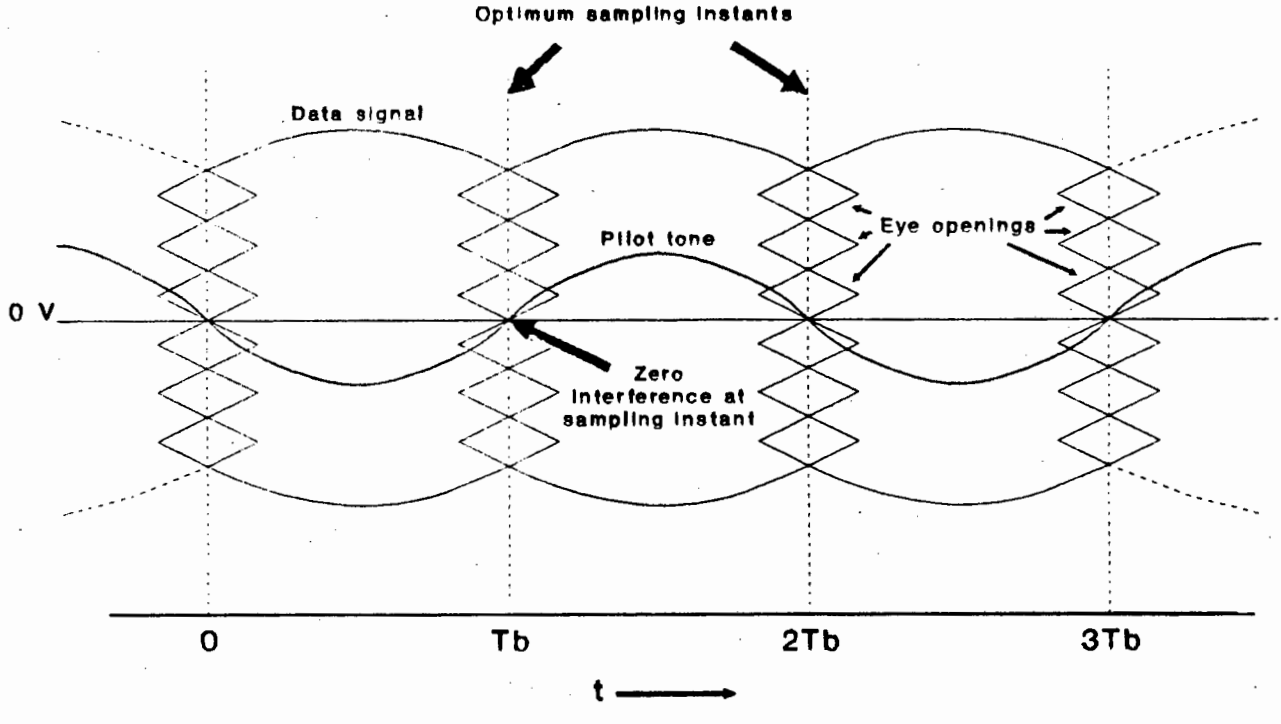


Figure 5.3 Zero-crossing pilot tone scheme for the low band

As the amplitude of the pilot tone is increased, there will be a corresponding increase in the overall power of the signal. However, since the maximum signal power on telephone lines is specified, it follows that the power in the data-carrying component of the signal will have to be decreased. Thus there will be some degradation in the probability of error vs. SNR performance. The P(e) vs. SNR curve given in section 3.4 will shift to the right by an amount equal to the power of the pilot tone. Figures 5.4 (a) and (b) show how the amplitude of the 49 QPRS signal is increased by the addition of a pilot tone.

Another effect caused by the addition of the pilot tone is a twisting of the eye openings. Due to the zero-crossing relationship between the data signal and the pilot tone which has been described, there will still be no interference at the sampling instants. However, the twisting of the eye openings causes the horizontal eye width to be slightly narrowed. This will be translated into a corresponding increase in timing sensitivity. Figure 5.4 (c) shows the demodulated baseband signal.

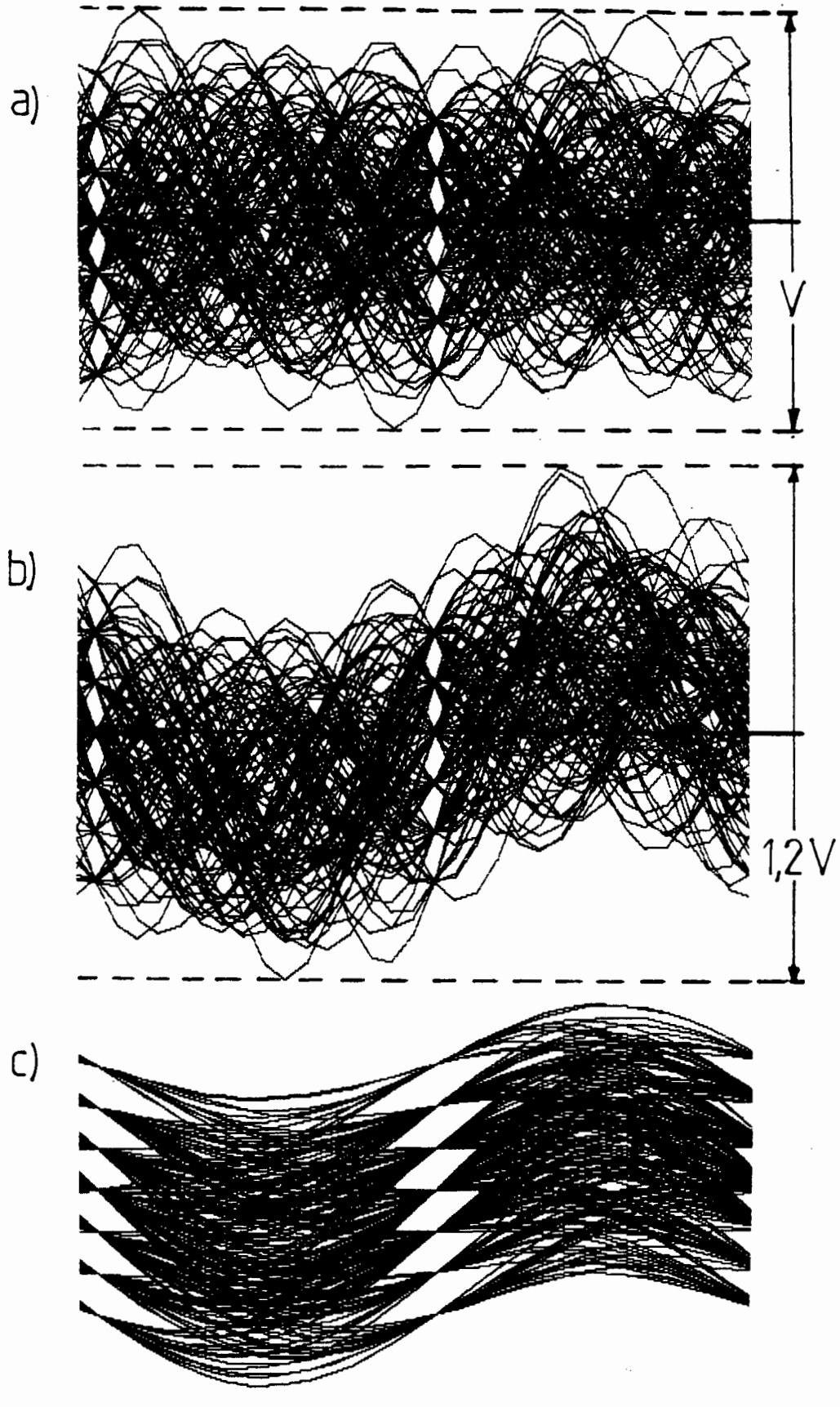


Figure 5.4 Effect of 600 Hz pilot tone on the low band signal (Computer simulations are shown)
a) 49 QPRS signal without pilot tone
b) 49 QPRS signal with pilot tone added
c) Demodulated in-phase channel showing twisting effect

5.2 CLOCKING OF THE MODEM AND CLOCK RECOVERY

There are various clock frequencies that have to be considered in the modem design. At the transmitter, there must be signals for sampling the binary data from the host system as well as a higher frequency clock for digitally generating each point of the transmitted waveform. The receiver must have access to three signals. The first is the sampling signal for sampling the received waveform. The second is the carrier signal which is used for coherent demodulation of the 49 QPRS signal. Finally, the symbol timing signal is necessary for sampling the demodulated waveform at the optimum sampling instant. The optimum sampling instant is the point of maximum vertical eye opening and should not be confused with the sampling of the received signal. A factor which must be taken into account when selecting the appropriate frequencies is the sampling theorem, as well as related concepts such as aliasing, the aperture effect and oversampling, which will be briefly explained.

5.2.1 The sampling theorem

The sampling theorem, which will not be proved here, is one of the most important theorems in communication theory. The theorem can be stated as follows [5.1]:

> A signal bandlimited to B Hz (i.e. its Fourier transform is zero for $|w| < 2\pi B$) is uniquely determined by its values at uniform intervals not greater than 1/2B seconds apart. Conversely a signal bandlimited to B Hz can be reconstructed from its samples taken uniformly at a rate not less than 2B samples per second.

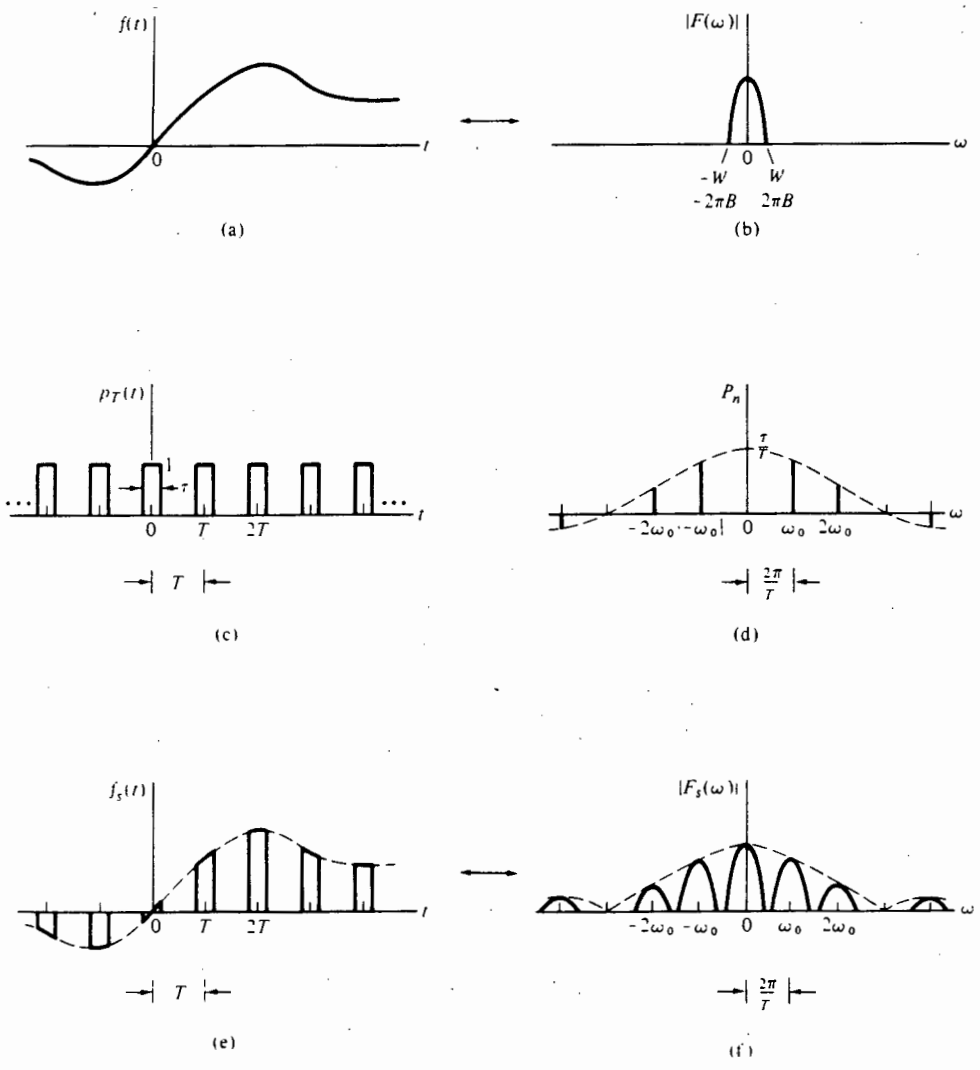


Figure 5.5 Sampling of a bandlimited signal
a) and b) The signal being sampled and its spectrum
c) and d) The sampling signal and its spectrum
e) and f) The sampled signal and its spectrum

Aliasing

From figure 5.5 it can be seen that the spectrum of the sampled signal consists of replicas of the original bandlimited signal spectrum. The spacing of these replicas is inversely proportional to the sampling frequency. In order to recover the original signal, these spectral replicas must be removed by a low pass filter. If the sampling frequency is less that the 2B samples per second limit specified by the sampling theorem, then the spectral replicas will overlap. Once this occurs the original signal can no longer be completely recovered.

Aperture Effect

The envelope of the spectrum of the sampled signal has a sinc(x) type shape which has a null frequency inversely proportional to the width of the sampling pulses. If the pulses are excessively wide, the innermost spectrum (the baseband spectrum) may be affected to the extent that the signal becomes distorted. There is usually a trade-off, as decreasing the pulse width also decreases the signal power, which is often undesirable.

Oversampling

Oversampling simply means that the signal is sampled at a higher frequency than theoretically necessary. This has important advantages. The higher the sampling frequency, the greater the distance between the spectral replicas. If the signal were sampled near the theoretical minimum sampling frequency, the spectral replicas would lie very close to each other. This would cause the low pass filter constraints to be very strict, and some group delay distortion is likely to occur around the cut-off frequency. Oversampling moves the spectral replicas further out, easing the filter constraints. If the width of the pulses is made equal to the sampling interval, then oversampling will also decrease the aperture effect, as the pulse width is decreased. Modern, digital equipment, such as compact disk players, commonly use four times oversampling, although this figure may vary.

5.2.2 Transmitter clock frequencies

There are two external clock frequencies to be considered at the transmitter. The first is the data clock (4800 Hz) and is used by the transmitter to sample binary data from the host system. The second is the clock to decide when the transmitter must generate each output sample. It was decided that there would be an integral relationship between these two frequencies as this simplifies timing.

The output sampling clock is subject to the constraints of the sampling theorem. Theoretically, a bandpass signal need not be sampled at a frequency greater than twice its highest

frequency, but only needs to be sampled at a frequency not less than double its bandwidth [5.2]. Sampling at a frequency lower than the frequency of the bandpass signal causes spectral replicas to appear between the signal and DC. As long as these do not overlap, the signal can still be recovered completely. Since the 49 QPRS signal is a bandpass signal with a bandwidth of 1200 Hz, it would seem that a frequency greater than 2400 Hz is necessary for sampling. The problem is that the spectral replicas would interfere with the frequency division multiplexing scheme being used. Also, the spectral replicas of the high band channel would occupy much of the region that is used by the low band.

The DSP56001 processor that was used to implement the transmitter was capable of a large overkill in sampling as it is extremely fast. It was decided that the maximum possible sampling frequency would be used. This is because there is a corresponding decrease in the analog filter requirements when removing spectral replicas as well as a decrease in aperture effect. It was decided that a sampling frequency of 38.4 kHz would be used for test purposes, which is 8 times the data input clock frequency of 4800 Hz.

In an actual system a clock would be supplied with the data to the modem by the host system. This clock would have a frequency of 4.8 kHz. The 38.4 kHz sampling signal could then be generated from the 4.8 kHz bit using a phase-locked-loop (PLL) with a divide-by-8 circuit in its feedback loop. In the prototype transmitter which was built, however, the host system is simply a pseudorandom sequence generator. For simplicity, the 38.4 kHz signal was generated using a clock generator (see section 7.1). This was then divided down to 4.8 kHz using a divide-by-8 circuit. The 4.8 kHz signal is then used to clock the PRBS generator.

5.2.3 Receiver clock frequencies and clock recovery

The proposed method of clock recovery is to use a single pilot tone for each channel, which is transmitted with the data stream. Pilot tones have the advantage that they can

easily be extracted at the receiver using PLL's. In the proposed scheme all the required signals, i.e. the sampling, carrier and symbol timing signals, can be derived from a single pilot tone. This is because the frequencies of these waveforms are arranged to be exact multiples of each other. The signals must be recovered from the transmitted waveforms without excessive jitter and should not have any phase ambiguity.

Once the 49 QPRS signal is sampled by the receiver it is immediately multiplied by the in-phase and quadrature carrier. In the case of the high band, the resulting waveform will contain spectral components that extend up to a frequency of 5.4 kHz. This was shown in section 5.1.2. The sampling theorem states that this signal must be represented by a sampling frequency of at least twice this frequency, i.e. 10.4 kHz. A sampling frequency of 19.2 kHz was chosen as it is the lowest even multiple of the bit rate above 10.4 kHz.

Low band clock recovery

A phase-locked-loop arrangement would be used to regenerate the required waveforms from the 600 Hz pilot tone. The VCO must be set to run at 19.2 kHz, the receiver sampling frequency, which is then divided by 16 to produce a 1200 Hz signal. This signal is used by the receiver to regenerate the in-phase and quadrature carriers and also indicates the symbol timing (the baud rate is 1200). A further divide-by-two circuit produces a 600 Hz waveform which is fed back to the phase detector. Starting at the phase detector and working backwards, it can be shown that the phase of all the regenerated signals will be unambiguous.

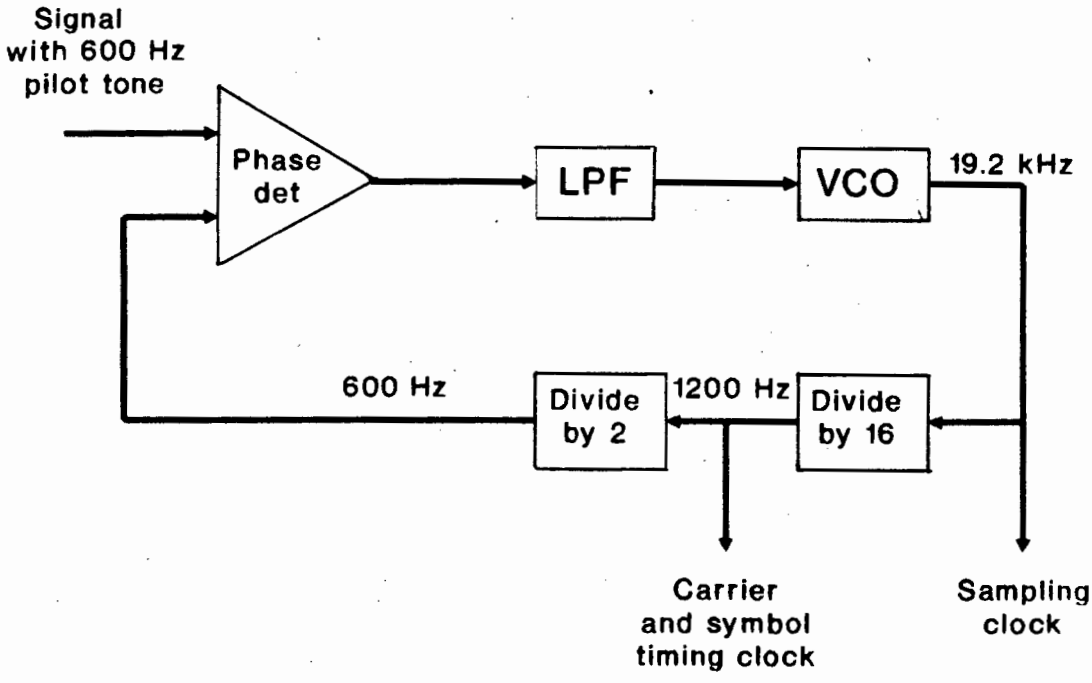


Figure 5.6 Clock recovery for the Low band signal

High band clock recovery

In this case all the required signals are generated from the 400 Hz pilot tone. There is no signal energy from either channel in the vicinity of 400 Hz and so a phase-lock loopwill be able to lock onto the signal without any difficulty.

As in the low band case, the VCO will run at 19.2 kHz, the receiver sampling frequency. This is then divided by 8 producing a 2400 Hz signal which is used by the receiver to regenerate the 2400 Hz quadrature carriers. The 1200 Hz symbol timing frequency is then generated by dividing the 2400 Hz signal by two. A further divide-by-3 operation brings the frequency down to 400 Hz which is fed back to the phase detector. Once again, the phase of all the signals will be unambiguous.

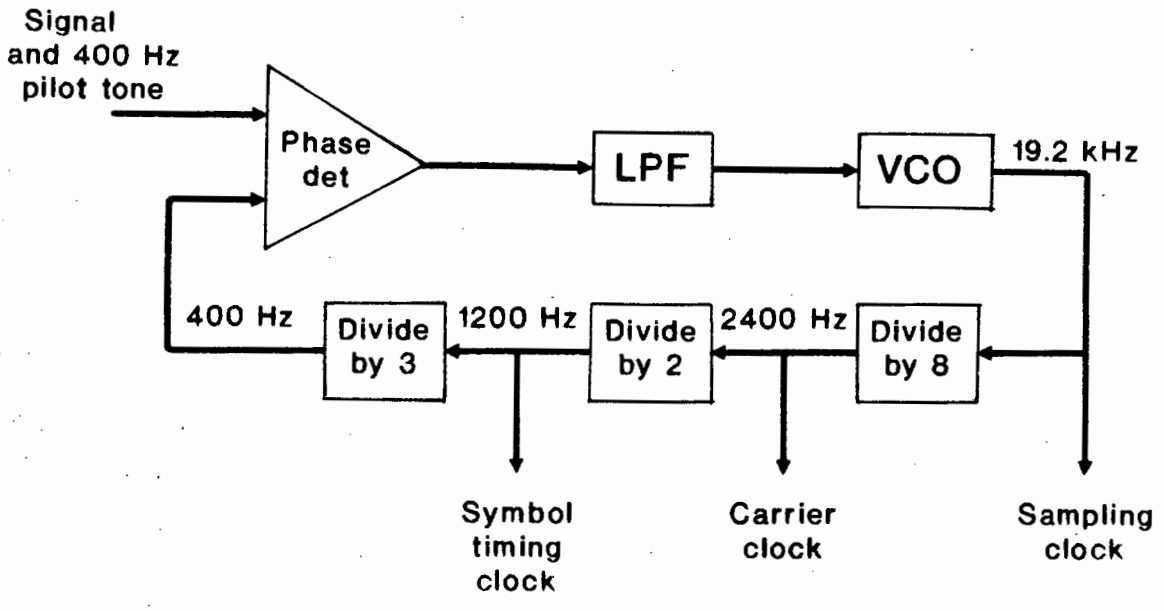


Figure 5.7 Clock recovery for the high band signal

5.3 Summary

The proposed scheme uses 49 QPRS to achieve 4800 bps full-duplex transmission in a usable bandwidth of 300 Hz to 3000 Hz. Like the V.22bis system, carrier frequencies of 1200 Hz and 2400 Hz are used for the two data directions. It is proposed that no bandpass filtering will be necessary on the received signals. Instead, the baseband Nyquist filter will serve the dual purpose of matched filtering and rejecting all unwanted frequency components. The Nyquist filter that is used will be a square-root raised-cosine filter. Pilot tones will be used for clock recovery. A pilot tone of 600 Hz is transmitted with the low band signal while a pilot tone of 400 Hz is transmitted with the high band signal. The pilot tones are arranged in such a way that all the necessary receiver clock frequencies can be received from a single pilot tone for each band.

REFERENCES

[5.1] Lathi, B.P., **Modern Digital and Analog Communication Systems**, CBS College Publishing, 1983, Pg 63.

[5.2] Stremler, Ferrel G., **Introduction To Communication Systems, Second Edition**, Addison-Wesley Publishing Company, 1982, Pg 124.

6. DIGITAL IMPLEMENTATION USING THE MOTOROLA DSP56001

6.1 THE DSP56001, ARCHITECTURE AND CAPABILITIES

6.1.1 The advantages of using the DSP56001

The DSP56001 is a high-speed, CMOS digital signal processor and is designed to execute many DSP algorithms in as few operations as possible [6.1]. The advantages of using digital signal processing over analog techniques include the following:

a) Fewer components necessary as complex functions can be implemented in software.

b) Stable, deterministic performance.

c) High noise immunity and power supply rejection.

d) Very close filter tolerances can be realized with no filter adjustments.

At the time of choosing the processor best suited to this project, the two most likely candidates were the Texas Instruments TMS32010 and the Motorola DSP56001. The DSP56001 was chosen because it was readily available mounted on a convenient PC plug-in card and also had the following further advantages relevant to this project:

a) A parallel architecture and a no-overhead, hardware DO loop. The DO instruction does not cause any additional time delay (this is not the case in the TMS320 series processors). This allows FIR filters to be implemented in the theoretical minimum number of instruction cycles (one instruction per tap weight). The parallel architecture allows an instruction prefetch, a 24*24 bit multiplication, a 56-bit addition, two data moves and two address pointer updates to be performed in a single instruction cycle.

b) Modulo addressing registers allow efficient implementation of circular buffers.

c) The processor operates at high speed (10.25 mips).

d) A built-in, four-quadrant sine-wave lookup table which is useful for implementing modulation.

6.1.2 DSP56001 architecture

A detailed discussion of the processor architecture will not be given here but a summary of major components relevant to this project is given. Much of the discussion will refer to the labels X and Y as the chip has a dual nature, i.e. the memory, buses, etc are divided into X and Y areas. The main components of the processor are as follows:

a) Data buses

b) Address buses

c) Data ALU

d) Address ALU

e) X data memory (with external expansion option)

f) Y data memory (with external expansion option)

g) Program controller

h) Program memory (with external expansion option)

i) Input and output consisting of:
- Memory expansion (Port A)
- General purpose I/O (Ports B and C)
- Host interface
- Serial communication interface (SCI)
- Synchronous serial interface (SSI)

Address registers (R0-R7) are 16 bits wide and may be used to directly access 65536 X-memory locations, 65536 Y-memory locations and 65536 program-memory locations. The contents of these registers may be pre- or post-updated according to the addressing mode selected.

Offset registers (N0-N7) are 16 bits wide and may contain offset values used to increment and decrement address registers in register update calculations. For example, an offset register can be used to step through a table at any given rate, such as 4 locations per step. In the transmitter program written for this project it was necessary to step through a lookup table at 32 locations per step and this was done using an offset register. Offset registers can also be used for general purpose storage.

Modifier registers (M0-M7) are 16 bits wide and define the addressing mode used in address calculations. In this project two addressing modes are used. The first is linear addressing which is the default mode. The second is modulo addressing. In the this mode, the M register will specify the modulus. This mode is used for circular buffers. For example, in order to create a circular buffer of length 16, a value of 15 will be loaded into the M0 register. The buffer will then be pointed to with the R0 address register.

X and Y Data memory

These 24-bit wide on-chip memory spaces each consist of two parts. The lowest 256 locations (locations 0-255) are random access memory (RAM). The next 256 locations (locations 256-511) are factory programmed read-only memory (ROM). The X data memory ROM is programmed with Mu-law and A-law companding tables, while the Y data memory is programmed with a full, four quadrant sine table. Both memory spaces are expandable off-chip to 65536 locations.

Program memory

The on-chip program memory consists of 512 locations of RAM. The interrupt vector addresses are located in the lowest 64 locations. The program memory may be expanded externally to 65536 locations.

Input/Output

The DSP56001 has very advanced I/O capabilities. The following aspects of its capability are relevant to this project:

> The expansion port (Port A) is the external data bus and is multiplexed to one of the three internal data buses. The bus can interface to a wide variety of devices and peripherals such as memory, A/D's, D/A's, etc. The timing of this bus is also programmable for interfacing to slower peripherals.
>
> Ports B and C can be programmed either as dedicated on-chip peripherals or general purpose I/O pins. In the latter case, a control register can be used to control the direction of the data flow. All the registers involved are memory mapped and read/write.

Host interface

The host interface is a full duplex, parallel port and can be connected directly to the bus of a host processor. In this project, the host processor is an IBM compatible PC which is used to program the DSP56001. Host processor communication with the host interface is accomplished using standard host processor data move instructions and addressing modes. Handshake flags are provided for polled or interrupt driven data transfers with the host processor. It is also possible for the host processor to issue vectored exception requests to the DSP56001.

Interrupts

The DSP56001 has two general purpose interrupts, IRQA and IRQB. These interrupts may be programmed to be level sensitive or negative edge triggered.

The signal groups of the processor

The input and out pins are made up into seven functional groups. These are the port A address and data buses, port A bus control, interrupt and mode control, power and clock, host interface or port B I/O, serial communication interface or port C I/O and the synchronous serial interface. The following diagram shows the processor and how the pins are made up into the various signal groups.

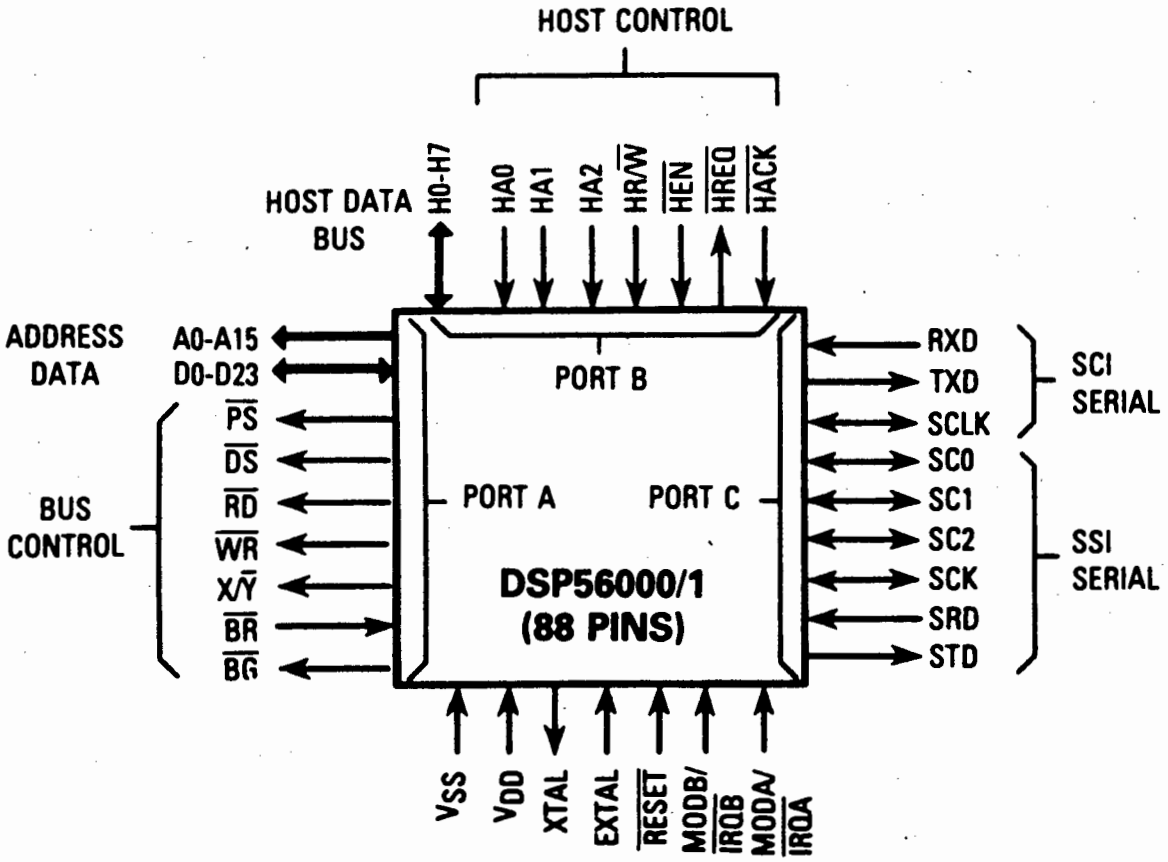


Figure 6.1 DSP56001 functional signal groups

6.2 MODEM FUNCTIONS IMPLEMENTED BY THE DSP56001

The digital 49 QPRS modem design consists of various functional blocks. At the transmitter these would include:

a) Splitting the incoming data into in-phase (I) and quadrature (Q) data streams

b) Precoding the data to avoid error propagation

c) Duobinary encoding of the precoded data

d) Nyquist type lowpass filtering

e) Modulation onto in-phase and quadrature carriers

f) The addition of the I and Q channels

g) The addition of pilot tones

h) Digital to analog conversion

i) Implementation of a lowpass smoothing filter

The receiver would include the following functional blocks:

a) Clock recovery

b) Analog to digital conversion

c) Multiplication by the recovered carriers

d) Nyquist type lowpass filtering

e) Duobinary decoding

f) Multiplexing of the I and Q data streams to form one data stream.

The proposed scheme attempts to incorporate as many of the modem functions as possible into the DSP56001. The only blocks not implemented by the processor will be the D/A conversion, the lowpass smoothing filter, the A/D conversion and the clock recovery scheme. Figure 6.2 shows a block diagram of all the functions implemented on the DSP56001.

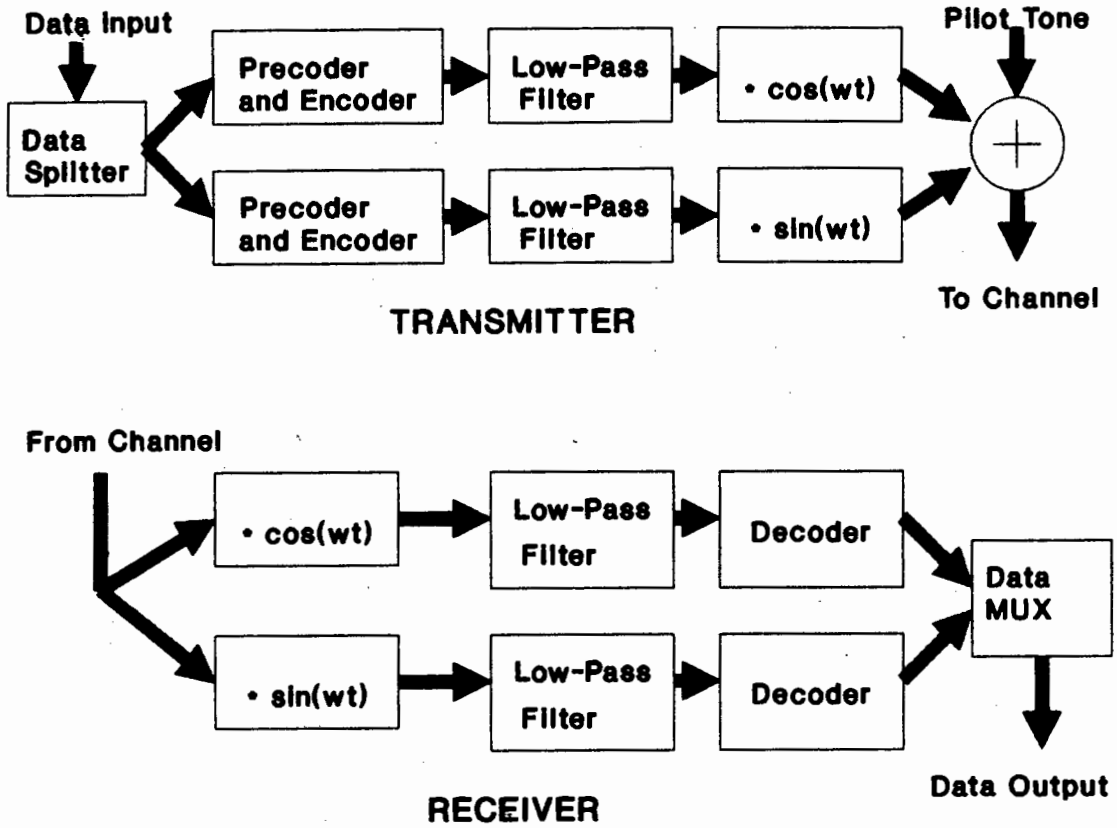


Figure 6.2 Modem functions implemented on the DSP56001

6.3 SQUARE ROOT RAISED COSINE DIGITAL FIR FILTERING

6.3.1 Finite impulse response (FIR) filters

There are many types of FIR filters, having different properties and different design. This brief discussion only concerns the type of FIR filter used in this project, which is a linear phase FIR filter. The impulse response of the required filter transfer function is digitally stored in a lookup table. The sampled waveform which is being filtered is simply convolved with the stored impulse response for each sample point. It can be shown that the filter will have exactly linear phase if its impulse response satisfies the condition [6.2]:

$$h(n) = h(N - 1 - n) \qquad (6.1)$$

where N is the length of the stored impulse response.

The linear phase characteristic makes these filters very attractive in applications such as this project where high roll-off filters are required for bandlimiting signal spectra. Conventional filters have a tendency to introduce large amounts of group delay as the roll-off factor increases and often need equalization to compensate for this (Like the filters used in V.22bis modems).

FIR filters have certain limitations. The main limitation are that the impulse response of a finite bandwidth filter is theoretically infinite. As a result, it must be truncated in order for it to be stored in a finite memory space. Processing speed limitations also limit the filter length that can be used. The truncation of the impulse response leads to the formation of sidelobes in the filter stop band. By proper selection of filter lengths and windowing functions, the level of these sidelobes can be reduced to an arbitrary level.

6.3.2 Square-root raised-cosine filter implementation

To meet Nyquist's and matched filter criteria a square-root raised-cosine filter is used at both transmitter and receiver. The overall response will then be that of a raised cosine filter. This filter is implemented digitally on the DSP56001 using a finite impulse response (FIR) design.

The sidelobes which result from the discontinuity caused by truncation of the impulse response are undesirable for this particular application. This is because the sidelobes will cause a frequency overlap of the two channels, thus degrading the crosstalk isolation offered by the FDM scheme. The effect of the discontinuity is greatly reduced by the correlation between the bits, which causes pulse tails to partially cancel. By increasing the excess bandwidth of the square-root raised-cosine filter, lower sidelobes can be obtained as the pulse tails die down faster, resulting in a smaller discontinuity. In this application, however, very low excess bandwidth is necessary as there is no guard band between the two channels. The solution is thus to use the longest lookup table that time constraints will allow, while maintaining a small excess bandwidth. The effect of the filter constraints on the effectiveness of the FDM scheme was investigated by computer simulation and is described in Chapter 9.

A numerical integration program was written to derive the impulse response of a square-root raised-cosine filter, as a closed form solution could not be found. The source code (TURBO PASCAL) of this program is given in Appendix E.

6.4 WINDOWING IN FIR FILTERS

Windowing functions have been widely used to reduce the level of sidelobes in FIR filters. The technique can be briefly described as the multiplication of the desired impulse response by a low frequency function that causes the original response to die away faster at the edges. The result is that truncation will cause a much smaller discontinuity in the impulse response, which in turn reduces sidelobe levels. This advantage is obtained at the expense of a slightly distorted main lobe (it will be widened). In this thesis the effect of a Hamming window is investigated, although there are many other windowing functions available. The Hamming window was selected because it offers good sidelobe suppression with only moderate distortion of the mainlobe. The Hamming widow function is given by the formula [6.2]:

$$W(n) = 0.54 - 0.46 * \cos(2\pi n/(N-1)) \qquad (6.2)$$

where:

N = amount of points being windowed

n refers to each point

The shape of the window is indicated in figure 6.3. The rectangular widow shown is analogous to simply truncating the lookup table.

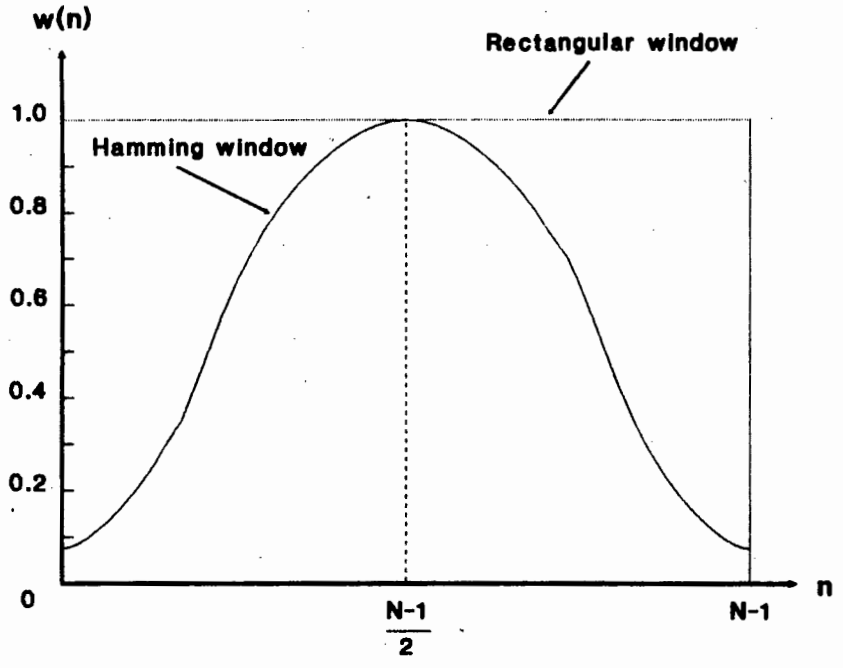


Figure 6.3 The Hamming window

6.5 SUMMARY

A digital implementation of the modem architecture, implemented on the Motorola DSP56001, is proposed. A digital implementation offers many advantages such as reduced complexity, increased reliability and noise immunity. The architecture of the DSP56001 is specifically designed for this type of application. Very little external hardware is needed as almost all the modem functions can be implemented in software on the DSP56001. Nyquist type bandlimiting of the 49 QPRS signals is implemented using square-root raised-cosine digital filters.

REFERENCES

[6.1] Motorola DSP56000/DSP56001 Digital Signal Processor Users Manual

[6.2] Oppenheim, Alan V., Schafer, Ronald W., **Digital Signal Processing**, Prentice-Hall International Incorporated, 1975, Pg 237 and Pg 242.

7. TRANSMITTER DESIGN

7.1 OVERVIEW OF THE TRANSMITTER DESIGN

The transmitter samples binary data from a data source and generates the 49 QPRS waveforms, as described in the previous chapters. It demonstrates the effectiveness with which the 49 QPRS modem design can be implemented on the DSP56001. A hardware switch selects the carrier frequency, which is 1200 Hz for the low band and 2400 Hz for the high band. The transmitter design can be broken down into three sections:

a) The DSP56001 assembler program:
This program, 'TRANSMIT.ASM', is the 'heart' of the transmitter. It implements most of the 49 QPRS modem functions, including data splitting, precoding, PRS encoding, Nyquist filtering and modulation onto carriers.

b) The Turbo Pascal host program:
This program, called 'TRANSMIT.PAS', is used to initialize and activate the DSP56001 via the host interface. The program loads the FIR filter lookup table from disk and downloads it to the DSP56001. If desired, a Hamming window can be applied to the lookup table before it is downloaded. The host program also generates and downloads the pilot tones to the DSP56001.

c) Hardware:
The main hardware components are the external memory, digital-to-analog converter (D/A) and the lowpass smoothing filter, which removes the steps in the D/A output. For test purposes, other hardware components were built. These include the clock generator and the PRBS generator.

Each of these sections will now be explained. The discussion will be as descriptive as possible, although in certain sections it is necessary to provide more detail.

7.2 TRANSMITTER HARDWARE DESIGN

The hardware consists of two sections. The first consists of the external memory of the DSP56001, the output latches and decoding logic. These components were wire-wrapped onto the DSP56001 card. The remaining hardware consists of the D/A, smoothing filter, clock generator and PRBS sequence generator. These components were initially built on breadboard and later transferred to printed-circuit board. Figure 7.1 shows a block diagram of the system, while a photograph of the hardware is shown in figure 7.2.

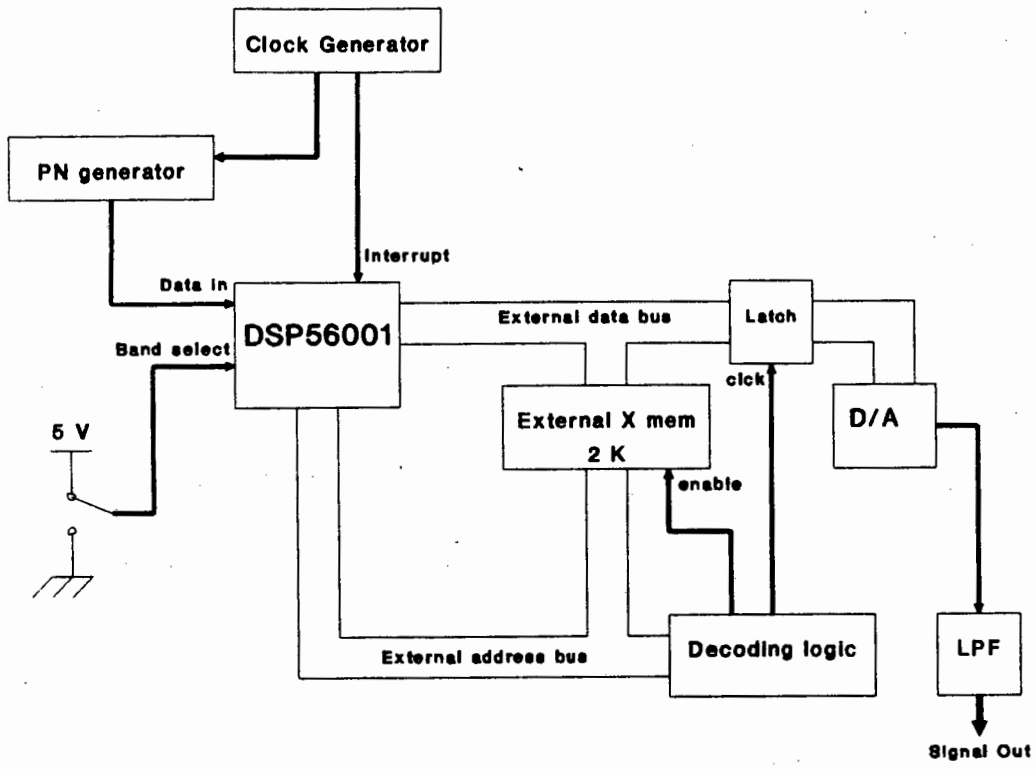


Figure 7.1 Block diagram of the transmitter hardware

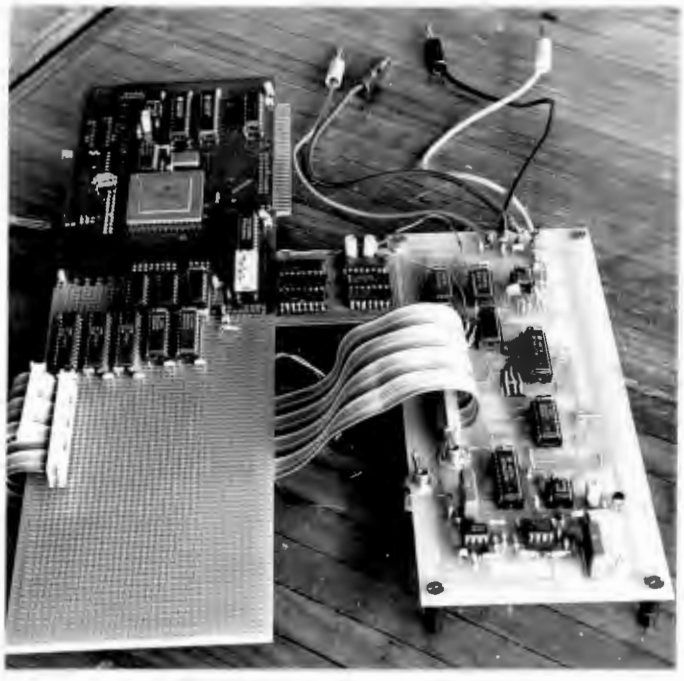


Figure 7.2 Photograph of the transmitter hardware

Each of the hardware components shown in figure 7.1 are now described. The circuit diagrams are then given.

7.2.1 Pseudorandom binary sequence (PRBS) generator

The PRBS generator is necessary for design purposes in order to simulate data transmission. It consists of a series of shift registers. The outputs of two of the shift registers form the inputs to an XOR gate. The output of the XOR gate is then fed back to the input of the first shift register. This arrangement generates a pseudorandom sequence of bits with a maximum length of $2^N - 1$, where N is the amount of shift registers used. The PRBS generator that was built can be switched to include either 3 or 24 shift registers. When 3 shift registers are used, a short sequence having a length of 7 bits is generated. This allows exact output values to be calculated so that the modem can be checked step by step. By including 24 shift registers, a much longer sequence is

generated. The long sequence allows eye diagrams and frequency spectra to be viewed effectively. A block diagram is given for clarity.

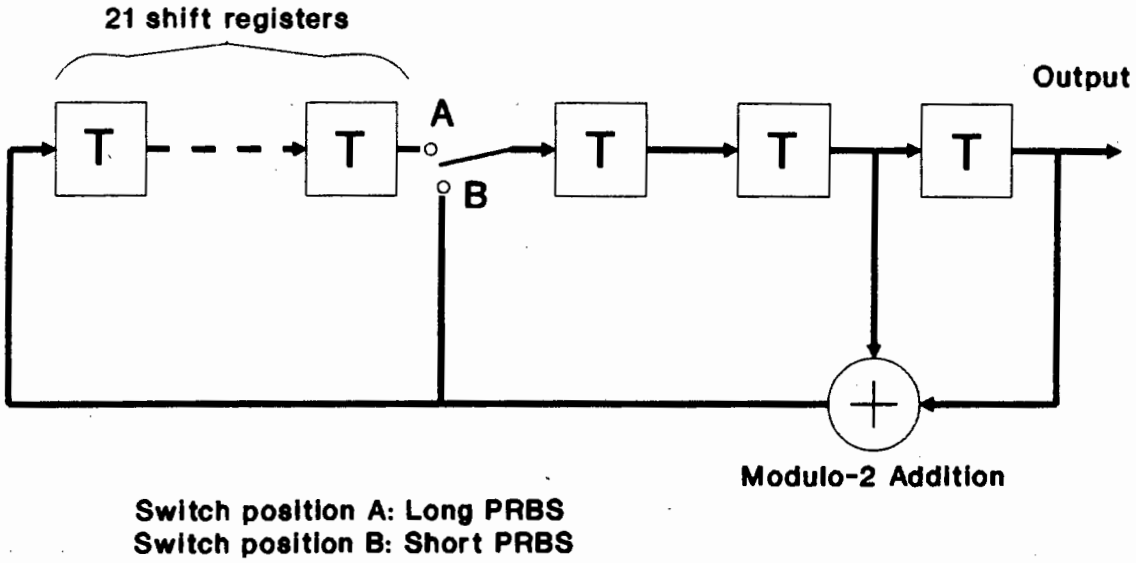


Figure 7.3 Block diagram of the PRBS generator

7.2.2 Clock generator

This section is only relevant for design purposes, as an operational modem would receive the clock with the data from the host system. The circuit diagram is given in figure 7.4. A relaxation oscillator is used [7.1]. This type of oscillator is very simple and the oscillation frequency is suitably stable for this application. The relaxation oscillator operates by charging a capacitor through the resistor R2, then discharging it rapidly when the voltage reaches the threshold defined by R4 and R3. The cycle then begins again. The oscillation frequency was set to 153.6 Hz. A divide-by-4 circuit, using two D-type flip-flops then brings the frequency down to the required 38.4 kHz. The divide-by-4 operation is used to 'clean up' the oscillator output. This 38.4 kHz clock drives the IRQA interrupt pin on the DSP56001, which will generate one output sample for each

clock cycle. The bit rate clock (4.8 kHz) is generated by dividing the 38.4 kHz clock by eight using three D-type flip-flops.

7.2.3 External Memory

The internal data memory of the DSP56001 is limited to 256 locations in the X memory space and 256 locations in the Y memory space. The speed of the processor is high enough to implement digital filters of far greater length than 256. In order to facilitate larger lookup tables, an external 2K memory bank was installed on the external data bus. Decoding logic was installed so that the memory could be accessed by reading or writing to any X bus location greater or equal to $1000. The external memory map thus extends from location $1000 to location $17FF and then wraps onto itself again i.e. location $1802 is the same as $1002. High-speed memory chips, HM65728's, were used. The access time of these devices is 35 ns, which is far less than the instruction cycle time of 98 ns. Since the memory chips are 8 bits wide and the data bus is 24 bits wide, three IC's were installed in parallel. The circuit diagram is given in figure 7.5. The diagram show both X and Y data memories. Y data memory was used when the receiver software was tested.

7.2.4 D/A converter

A 12-bit DAC AD7531 is used on the transmitter. The circuit diagram is shown in figure 7.6. When selecting the amount of bits necessary for the D/A conversion the criterion used was that the quantization noise introduced by the D/A should be far less than the noise introduced by other sources. The quantization noise produced by a 12-bit D/A is -77 dB and this value was considered adequate. The D/A operates in transparent mode, i.e. the D/A is not clocked, so that conversion is continuous. Two 8-bit 74LS374 latches are used to latch the 12-bit data from the DSP56001 to the D/A (only 4 bits of the second latch are used). Address lines A15 and A14 are used to latch the data as shown in the circuit

diagram. Data will be latched by first writing the data to location $8*** followed by an access to location $7***. Data lines D11 - D22 are latched.

7.2.5 Carrier set switch

A simple switch is used to select the carrier frequency. The output of the switch is connected to a general purpose I/O pin on the DSP56001. The DSP56001 continually polls this pin in order to decide which carrier frequency is being requested. This configuration is relevant only for test purposes. If the design is ever incorporated into an actual modem, the carrier frequency would depend on whether the modem initiated the data transfer (as in other V.22 type schemes).

7.2.6 Decoding Logic

the decoding logic is the logic used to interface between the address bus and the external hardware. It ensures that there is no bus contention. High speed CMOS logic was used throughout.

7.2.7 Smoothing lowpass filter

This smoothing lowpass filter is applied to the D/A output to remove the steps which appear in the output of the D/A. These steps appear as a result of the sample-and-hold operation of the D/A. In the frequency domain the steps appear as spectral replicas of the signal at multiples of the sampling rate. The sampling rate was selected so that the spectral replicas would appear at a relatively high frequency. As a result, the design of the smoothing filter is not critical. A sixth order Butterworth VCVS (voltage controlled voltage source) filter was selected as it has a very simple implementation and a good roll-off [7.1]. The circuit diagram is given in figure 7.7. It consists of three, 2nd order, filter sections. All have the same cutoff frequency of 1/(2.2RC), while the gain is increased in each stage to avoid dynamic range problems. The cutoff frequency

is 13 kHz which is high enough to ensure that the amplitude and group delay nonlinearities, which occur in the region of the cutoff frequency, will have negligible effect on the signal.

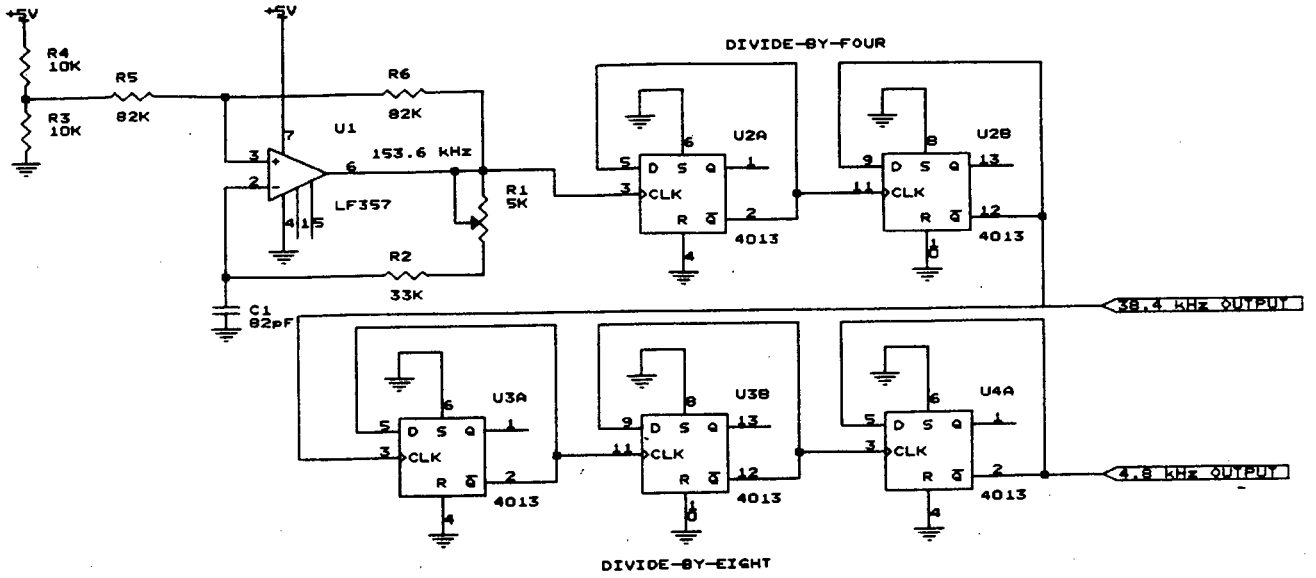


Figure 7.4 Clock generator circuit diagram

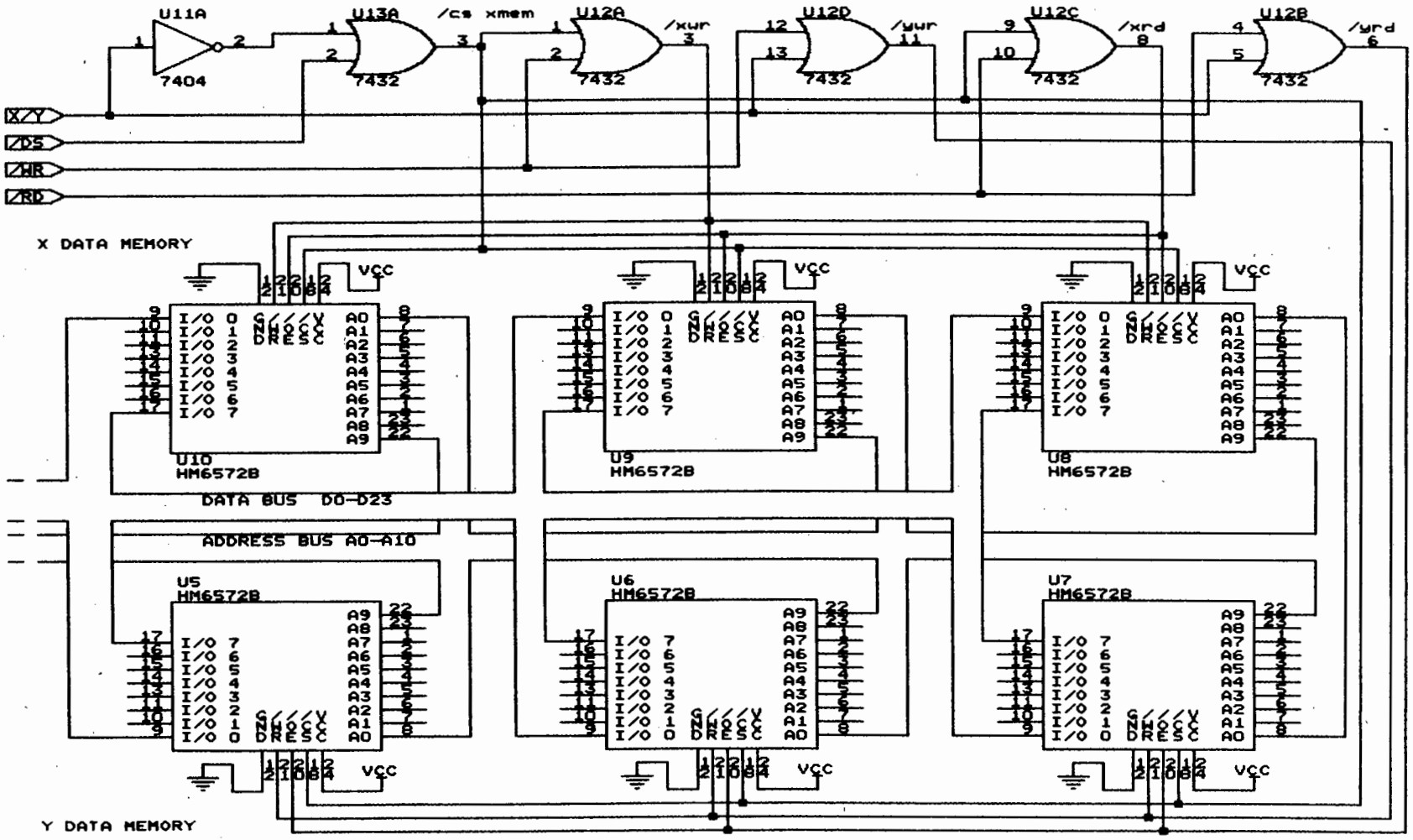


Figure 7.5 External memory circuit diagram

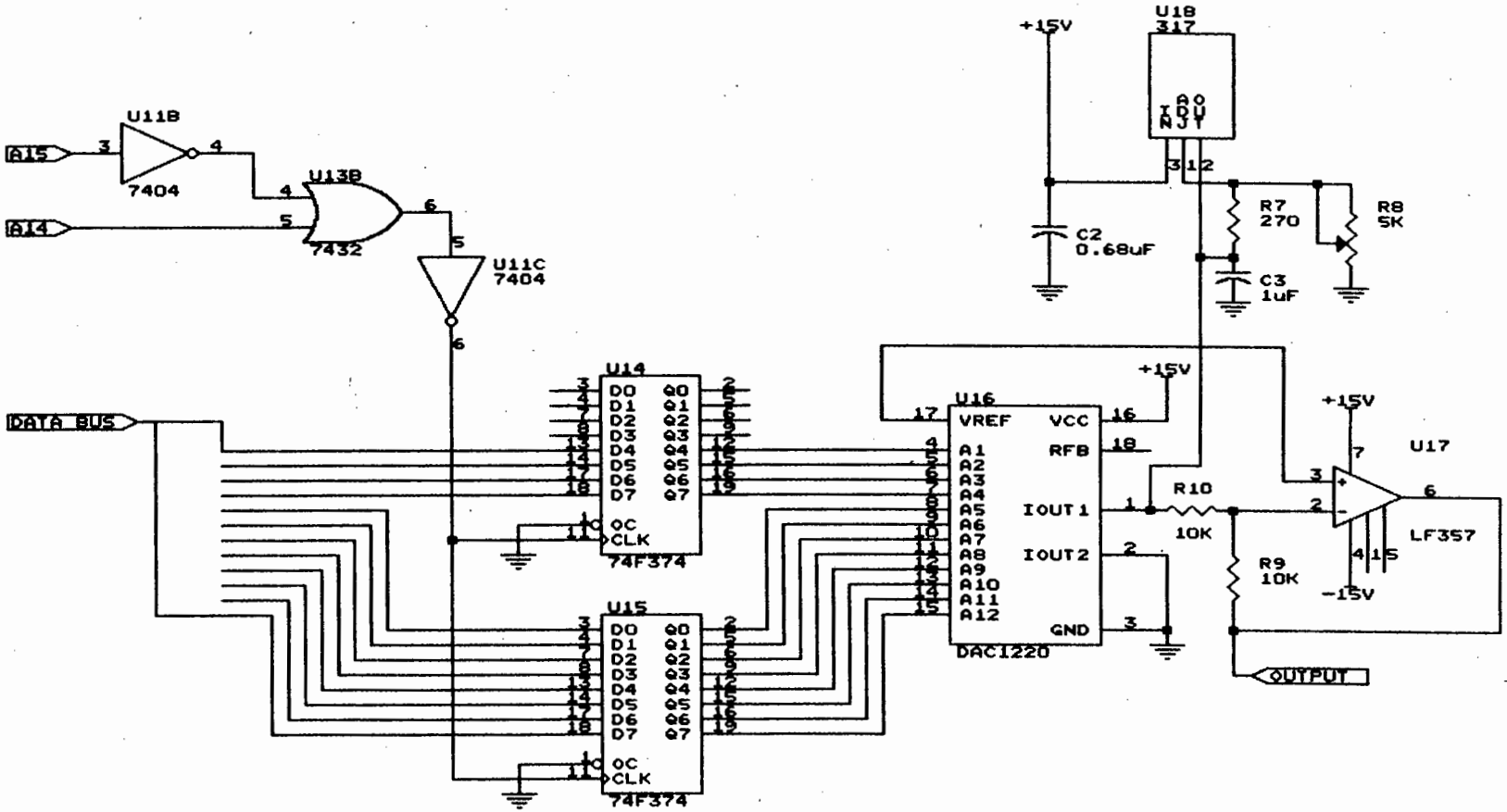


Figure 7.6 D/A converter and latch circuit diagram

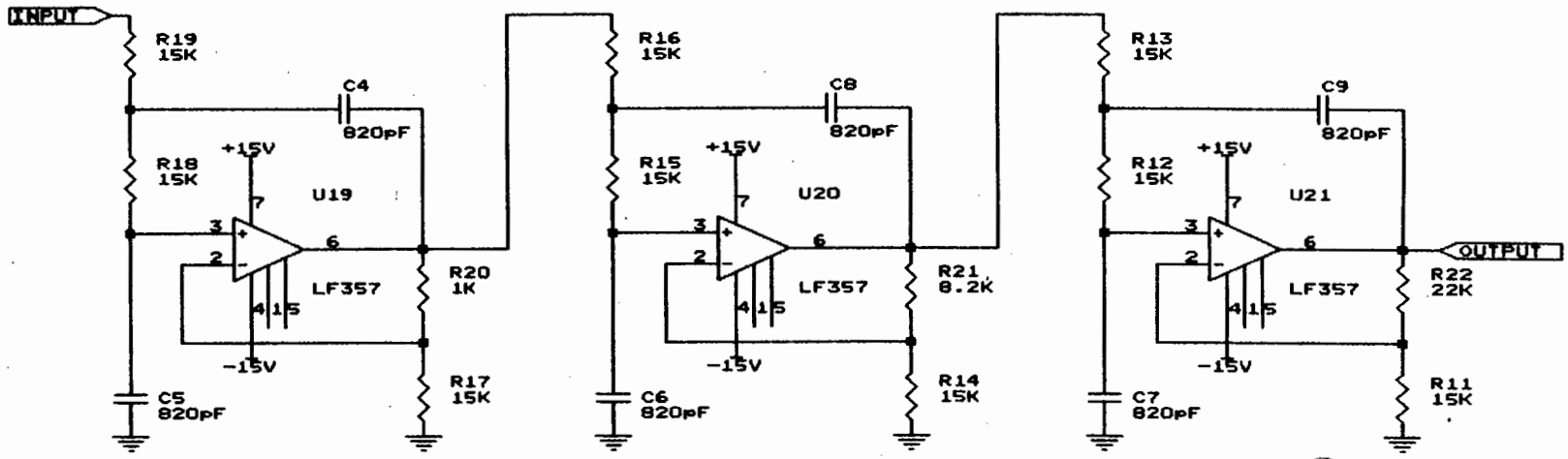


Figure 7.7 Smoothing lowpass filter circuit diagram

7.3 TURBO PASCAL HOST PROGRAM

The transmitter Turbo Pascal host program, 'TRANSMIT.PAS', is executed on the host PC. The source code is given in Appendix B, while a flow chart is given in figure 7.8. It performs the following functions:

a) Activates the DSP56001:
A utility procedure, called DSP_GO, was supplied by PERALEX in order to activate the DSP56001. This procedure activates the DSP56001 via the address at which the DSP56001 card is installed on the PC bus (in this case $2B0).

b) Transfers the FIR filter lookup table from the disk to the DSP56001:
The procedure 'Send_Lookup_table_to_DSP' loads the FIR filter lookup table from the file 'LOOKUP.DEC'. Two more utility procedures, supplied by PERALEX, called DSP_writeflag and DSP_writelongint, are used to download the lookup table to the DSP56001. The lookup table is also scaled according to a scaling factor set in the main program. Scaling the lookup table has the effect of scaling the 49 QPRS output, and it was used for this purpose. If desired, a subroutine can be invoked to apply a Hamming window to the lookup table before it is downloaded to the DSP56001.

c) Generates the pilot tones (400 Hz and 600 Hz) and downloads them to the DSP56001:
It is desirable to have control over the amplitude of the pilot tones. If the built-in sinewave lookup table of the DSP56001 were used for the generation of the tones, extra instructions would have been required for scaling, which would have introduced an unnecessary time penalty. Thus it was decided that the pilot tones would be generated and scaled by the host program and then downloaded to the DSP56001. A full cycle of each pilot tone is stored at a sampling frequency of 32 points per symbol period. Since the symbol rate is

1200 baud, 64 points of the 600 Hz pilot tone are generated, while 96 points of the 400 Hz pilot tone are generated. These are then downloaded to the DSP56001.

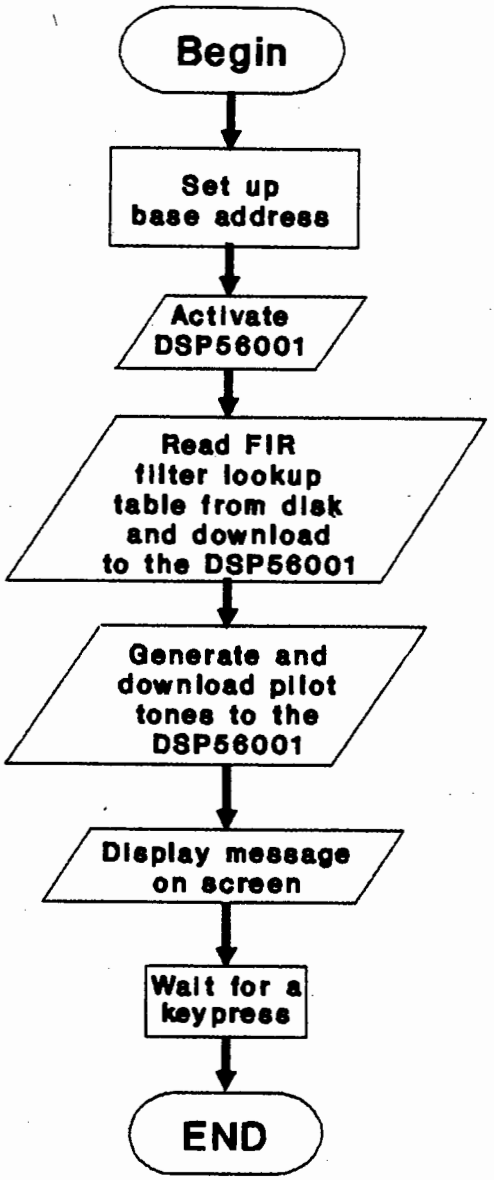


Figure 7.8 Flowchart of transmitter host program

7.4 DSP56001 ASSEMBLER PROGRAM

7.4.1 Introduction

This program was written so that the DSP56001 would operate as a real-time 49 QPRS transmitter using one of two carrier frequencies (1200 Hz and 2400 Hz). The timing of the generation of the output waveform is controlled via the IRQA interrupt of the DSP56001. A flowchart of the basic program flow is given figure 7.9, while flowcharts showing more detail are given later. The assembler source code is listed in Appendix B. The explanation of the assembler program will refer to a specific case of FIR lookup table parameters, although the program can easily be modified to use any other parameters. The system which is discussed uses a 768 point FIR lookup table. There are 32 points per symbol period and the table is 24 symbol periods wide.

The program can be broken down into two sections. These are the main program and the interrupt service routine. The basic functions of each will be discussed before moving on to a more detailed description of the program.

Main program:
The main program performs all the initialization routines and then enters a main loop where it waits for interrupts.

> The first step is to initialize all the relevant variables. The IRQA interrupt, general purpose I/O pins and the internal ROM (which contains the built in sinewave lookup table) are also initialized.
>
> The FIR filter lookup table and the pilot tones are downloaded via the Turbo Pascal host program.
>
> The program then enters its main loop where it waits for an interrupt. The only operation carried out in the main loop is an interrupt enable. If an interrupt request is received, the interrupt is disabled. Once the interrupt service routine is completed, the

interrupt will be re-enabled in the main loop and the program is ready to receive the next interrupt.

Interrupt service routine:

All the main computations are carried out in the interrupt service routine.

The first operation carried out by the routine is to disable the interrupt. The interrupt pending register is also disabled to prevent multiple interrupts from occurring.

The next operation carried out is to output the last data point that was generated. This ensures that the data output is properly timed as the amount of instructions required to reach this point after receiving an interrupt remains fixed.

This is followed by a decision whether to sample a bit from the PRBS generator. Since there are 32 samples per symbol period, and 4 bits make up one symbol, data must be sampled on every 8th interrupt.

The next output point is then generated and stored for output during the next interrupt service.

A check is made to decide whether new duobinary symbols are to be generated. This will occur on every 32nd interrupt as there are 32 output points per symbol period.

Finally, a check is made to establish whether a change in the carrier frequency is required.

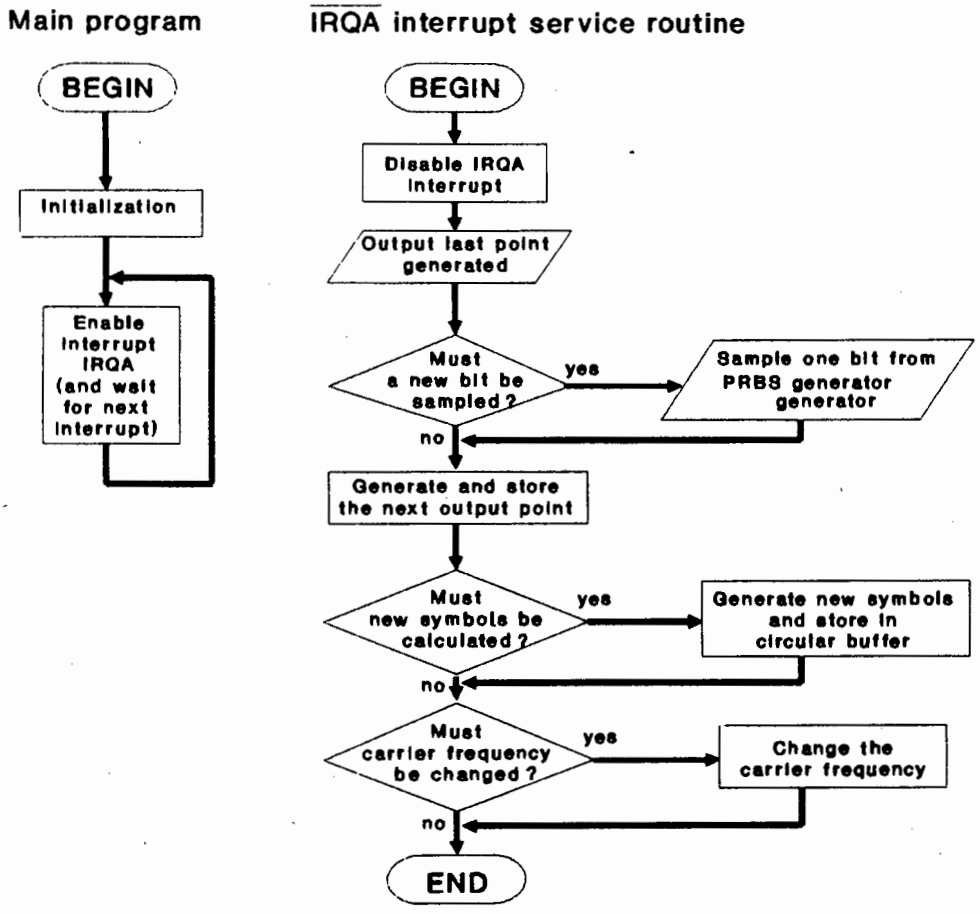


Figure 7.9 Flowchart of transmitter assembler program

The following sections of the program will now be expanded upon:

a) The address register assignments.

b) The operation of the FIR filter.

c) The timing of the data sampling and symbol calculation routines.

d) Duobinary precoding and encoding.

e) The setting of the carrier frequency.

7.4.2 Address register assignments

The assignments of the various address (R), offset (N) and modifier (M) registers are central to the operation of the program.

R0,M0,N0:
R0 is used to address the FIR filter lookup table. M0 is loaded with a value equal to one less than the length of the table. Its modulus function will then ensure that R0 is always mapped onto the lookup table. N0 is loaded with a value equal to the number of points per symbol as this is the offset required when updating R0 during a convolution.

R1:
This register is used as a general purpose counter to decide when bits must be sampled from the host system (PRBS generator) as well as when new symbols are to be calculated. Its operation is described in detail later.

R4:
R4 is used as a pointer to the circular buffer. The circular buffer contains the duobinary values and is convolved with the FIR filter lookup table at each step.

R5,M5:
The R5 register is used as a pointer to the pilot tones. The M5 register implements modulo addressing so that the pilot tone will continuously repeat itself. By initializing R5 and M5 with the start address and modulus of either pilot tone, the 400 Hz or 600 Hz pilot tone can be selected.

R6,M6,N6,R7,M7,N7:
These registers are used to address the built in sinewave lookup table in order to generate the in-phase and quadrature carriers. R6,M6 and N6 are used for the I channel while R7,M7 and N7 are used for the Q channel. R6 and R7 are initialized so that they point to positions on the table which are 90^{o} out of phase with each other. The modifier registers M6 and M7 are loaded with one less than the modulus of the table so that the addressing will

continuously repeat itself. The offset registers N6 and N7 control the offset added to R6 and R7 at each update and in this way they control the carrier frequency.

7.4.3 FIR filter table and circular buffer operation

The FIR low pass filter and the circular buffer work together to produce the output waveform from a stream of duobinary numbers. The net operation is that the lookup table is convolved with impulses of a finite section of the duobinary data stream to produce each output point. If the entire lookup table were to be convolved with the impulse train each time many of the operations would be redundant. This is because many of the multiplications would be multiplication by zero, due to the spaces between the impulses. Thus if the lookup table has 32 points per symbol width then it will only be necessary to convolve with every 32nd point on each pass. This operation is achieved simply by using the offset register N0. This saves a great deal of processing time which, in turn, allows far longer lookup tables to be used. The operation of the scheme is illustrated in figure 7.10. For simplicity, a lookup table only four symbols wide with four points per symbol is illustrated. The vertical bars indicate the storage locations of the lookup table.

For each output point, one four point convolution is executed. The points indicated by the arrows are convolved with the circular buffer for each convolution. The points in the lookup table that are used are shifted by one for each successive convolution, as shown. After four output points have been generated (i.e. four convolutions have been executed), a new doubinary number is entered into the circular buffer, overwriting the oldest value in the buffer. Data is written to the circular buffer via the addressing register R4 which is incremented before each write operation. When performing a convolution, R4 is decremented at each multiplication step so that the contents of the buffer are accessed from the most recent value to the oldest value. The process then repeats itself.

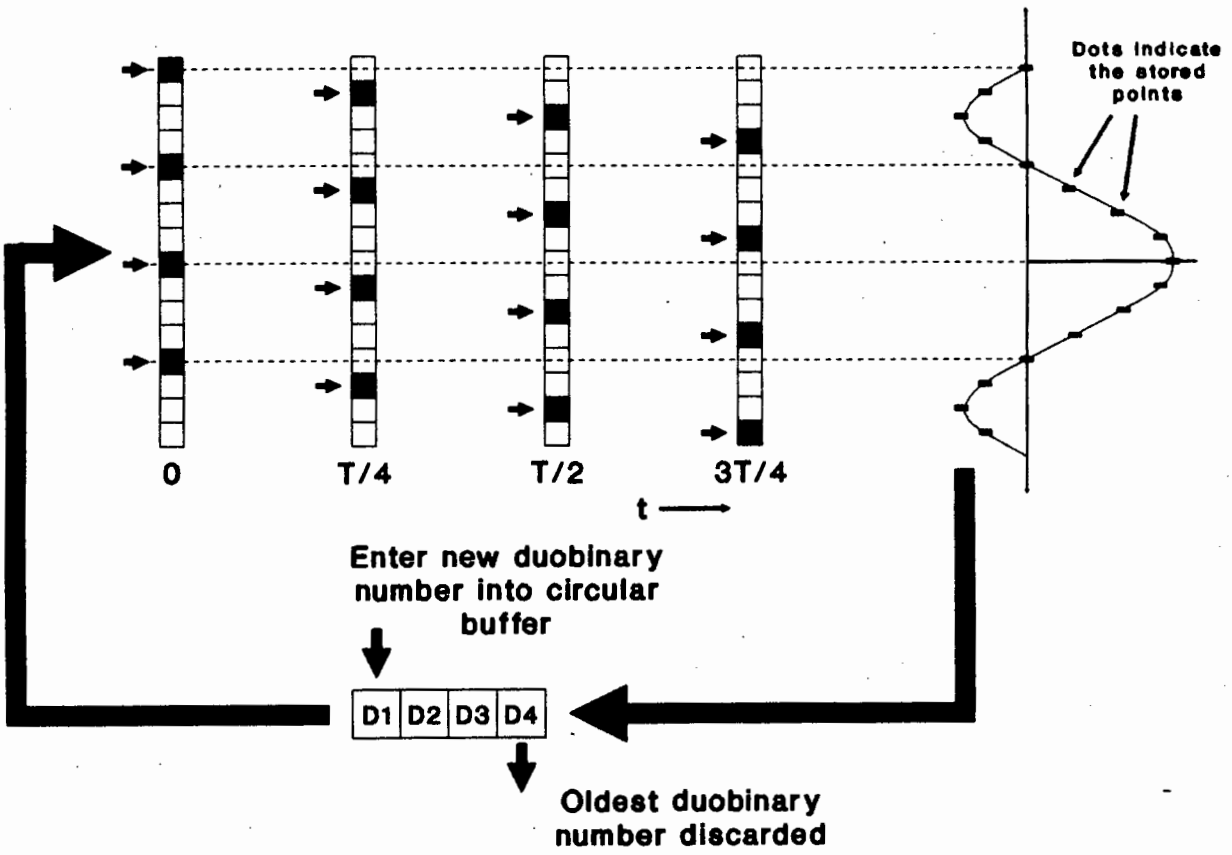


Figure 7.10 Operation of the transmitter FIR filter lookup table

The above discussion describes the filtering operations where there is only one circular buffer. In the actual system there are two circular buffers, one for the I channel and one for the Q channel. These buffers are actually combined and stored in one buffer of double their original length. This was done because the start addresses of circular buffers on the DSP56001 are confined to certain locations, depending on the buffer length. As a result, more external memory would have been required had the buffers not been combined into one. The filtering operation using this buffer is identical to the description that has been given except that the buffer operations occur in pairs.

Two output points are generated at a time. The first is multiplied by the in-phase carrier, while the second is multiplied by the quadrature carrier. The two results are then added to produce one point of the 49 QPRS waveform.

7.4.4 Timing for data sampling and symbol calculation

In this modem design, data is sampled from the host system in serial form. Each duobinary symbol, however, is made up of four bits, the I and Q channel symbols each being made up of two bits. One of the major motivations for using a signal processing chip was the reduction in hardware and so an external buffer which would allow four bits to be sampled at once was considered undesirable. Instead, bits are sampled one by one and stored in a four-bit internal buffer, called 'inbuf'.

As there are 32 interrupts per symbol period, and 4 bits per symbol, a bit must be sampled on every 8th interrupt. The actual calculation of the symbol values, which entails combining and encoding the binary input data to form duobinary data, must also be properly timed. This operation will be executed on every 32nd interrupt.

A software scheme which uses a general purpose counter (address register R1) to carry out the timing functions for data sampling and symbol calculation was implemented and is now described. A flowchart of the operations performed on the counter is given in figure 7.11. The operations shown in the flowchart will be executed once for every interrupt that is received.

R1 is incremented on every pass and reset on the 32nd pass. On every pass, the three least significant bit of R1 are checked. If they equal a value of three, then a data bit is sampled from the PRBS generator. This condition will be met on interrupt cycles 3, 11, 19 and 27. On the 32nd pass, new symbols are calculated. Care was taken to avoid the data sampling and symbol calculation routines occurring on the same interrupt cycle, as this would impose an additional time penalty. Data is sampled by reading the least significant bit of port C, which is configured as a general purpose I/O pin.

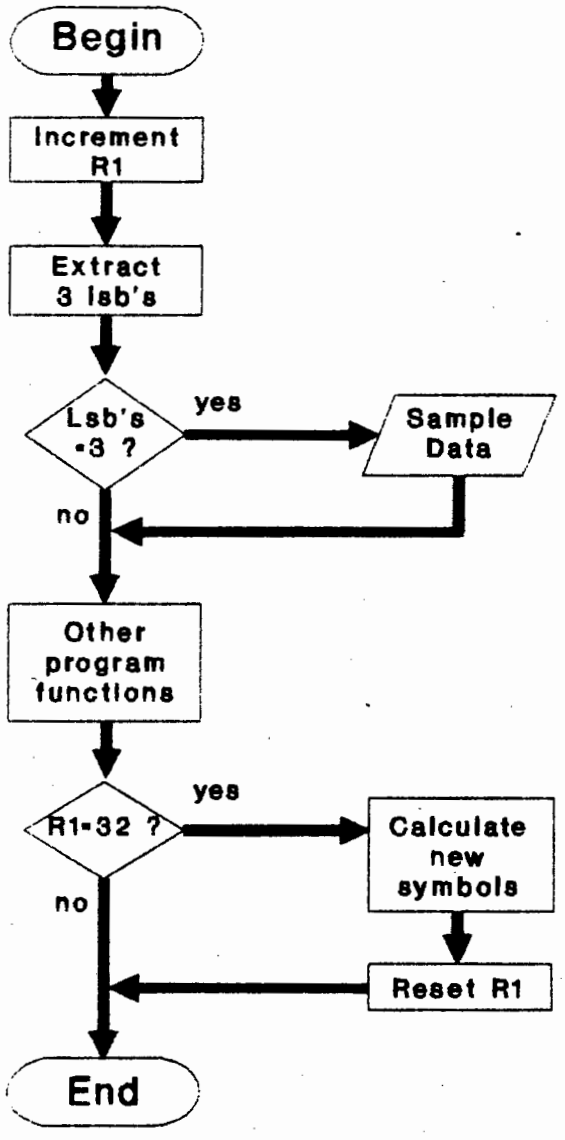


Figure 7.11 Flowchart of timing for data sampling and symbol calculation.

7.4.5 Duobinary precoding and encoding

The binary data that is sampled from the PRBS generator is stored in a four-bit buffer. When the symbol calculation routine is invoked, these four bits are split into two pairs of two bits each. These are encoded to form the duobinary symbols of the I and Q channels and are entered into the circular buffer. A flowchart of the subroutine that carries out the encoding is given in figure 7.12. 'Itemp' and 'Qtemp' are storage locations used to provide the 1-bit delay required for the precoding and encoding process. The

I-channel and Q-channel calculations are shown in parallel for clarity, although they are actually executed sequentially.

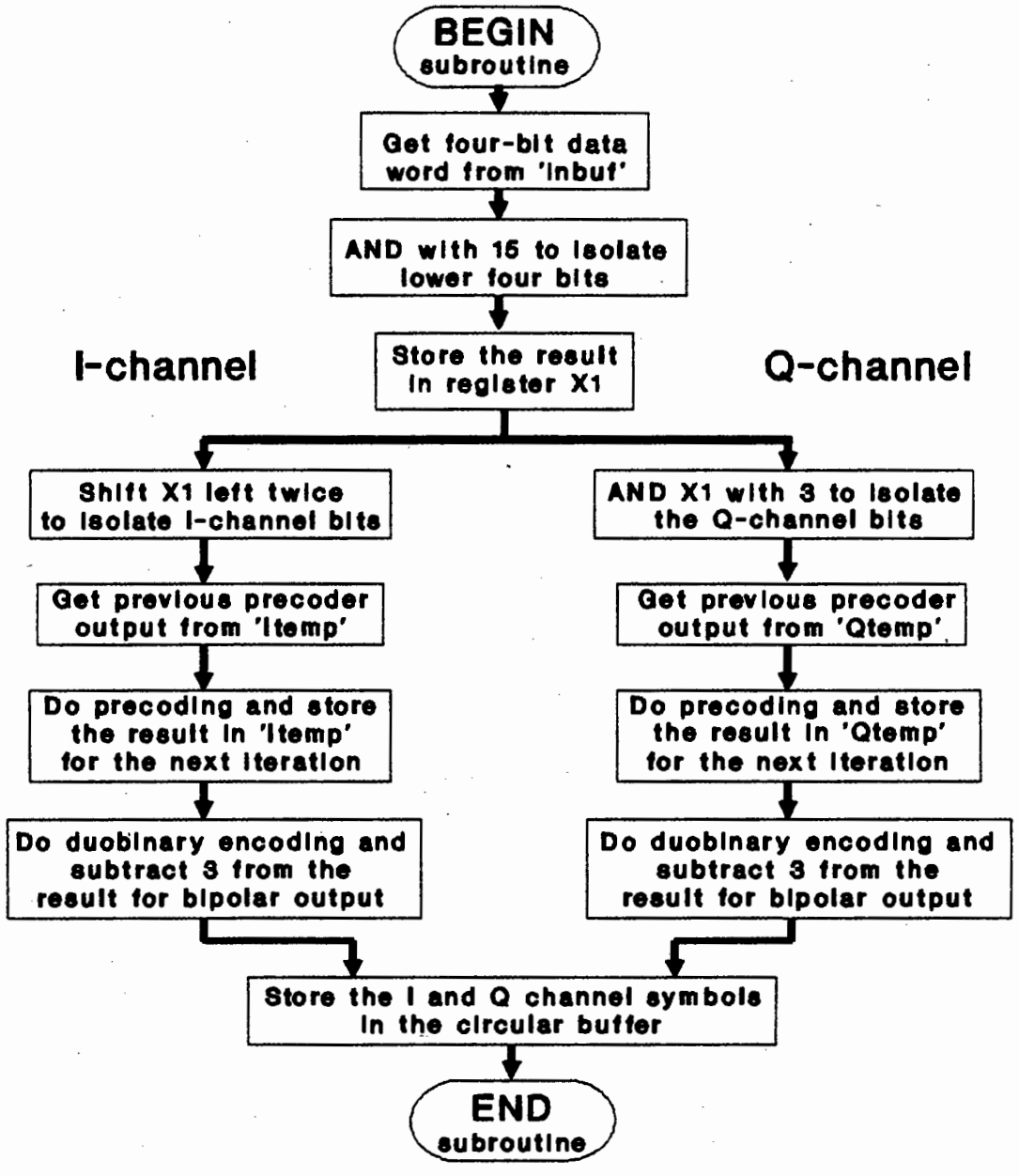


Figure 7.12 Flowchart of duobinary precoding and encoding routine

The actual coding processes require some clarification:

Before precoding, 4 is added so that the subtraction will always have a positive result. The modulo-4 operation required for precoding is done simply by ANDing with 3.

The result is added to the previously precoded number which results in a duobinary coded number between 0 and 6. A value of 3 is then subtracted from this number to give a bipolar symbol between the value of -3 and 3. This number is then entered into the circular buffer.

7.4.6 Carrier frequency setting

The carrier frequency (1200/2400 Hz) is set by selecting an external hardware switch which is then read via a pin on port C which is configured as a general purpose I/O pin. Although an operational modem may use a different system, this one was adequate for test purposes. During each interrupt service, the pin is checked. The carrier frequency selected by the voltage on the pin is compared with the current carrier setup, which is stored in the location 'CarryOld'. In order to save computational time, the pin is merely checked for a change so that the carrier need not be set up on every interrupt. If a change is detected, then a subroutine is used to make the required changes to the registers controlling the carrier frequency and the pilot tones.

The carrier frequency is reset by changing the offset registers N6 and N7. This alters the step size when using the built-in sinewave table to generate the I and Q carriers, thus changing the carrier frequency. The start addresses of the I and Q carriers (stored in R6 and R7) are also reset to maintain timing.

The pilot tones are reset by setting the R5 address register to the start address of the required pilot tone. The M5

modifier register must also be set to the modulus of the amount of storage locations used by the pilot tone table (64 locations for the 600 Hz tone or 96 locations for the 400 Hz tone).

7.5 DISCUSSION OF OUTPUT WAVEFORMS AND THEIR SPECTRA

In order to evaluate the performance of the transmitter, the eye diagrams of the 49 QPRS signals were viewed on an oscilloscope, while the frequency spectra were viewed on a spectrum analyzer.

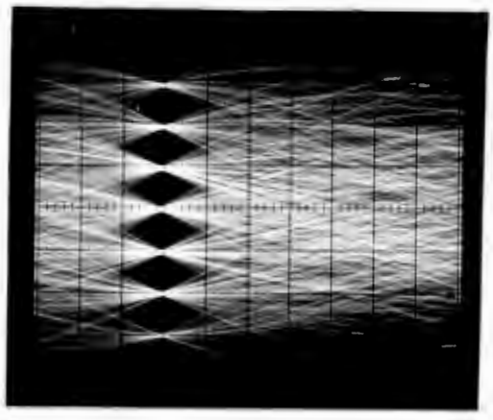
7.5.1 Eye diagram observations

Intersymbol interference caused by the hardware was observed by using raised-cosine lookup table, as opposed to a square-root raised-cosine lookup table. This is because a square-root raised-cosine filter is not a true Nyquist filter and has some inherent ISI which is only removed by the other square-root raised-cosine filter at the receiver.

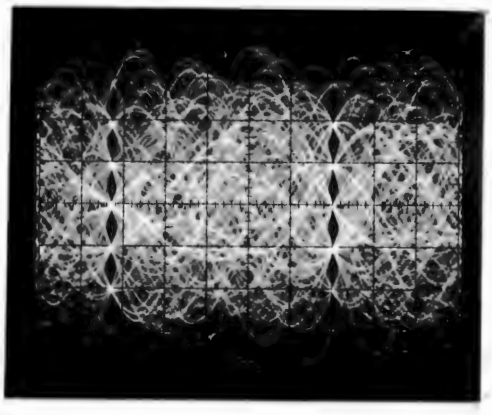
Using a raised-cosine filter, the eye diagrams were observed to be extremely sharp, with ISI at the sampling instants being virtually unobservable. This demonstrates the effectiveness of digital filter techniques. Photographs of the eye diagrams are shown in figure 7.13. The eyes do not appear perfectly sharp in the photographs due to the long film exposure time used.

When the excess bandwidth of the raised cosine filter was varied, there was no observable change in the eye diagram. Applying a Hamming window to the lookup table did also not result in any observable change to the eye diagram. Theoretically, applying a windowing function to the lookup table should have produced some ISI. This is because the resultant impulse response will no longer be that of a true Nyquist filter.

a)



b)



c)

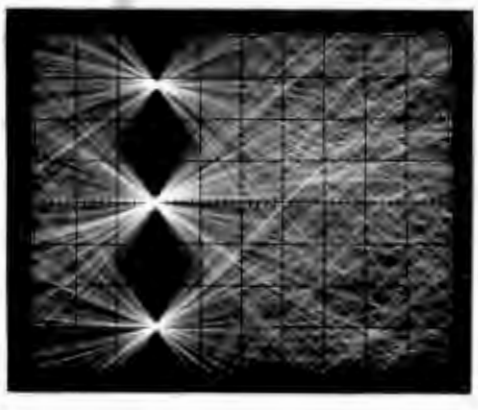


Figure 7.13 Photographs of transmitter eye diagrams
b) Low band 49 QPRS waveform
c) High band 49 QPRS waveform
d) Vertically expanded Low band 49 QPRS waveform

7.5.2 Frequency spectrum observations

The frequency spectra were exceptionally well defined. The theoretical duobinary shape could be observed in all the spectra, while very high filter roll-offs were obtained with satisfactory sidelobe suppression. The effect of varying the excess bandwidth, as well as the effect of using a Hamming

window was observed. The main criterion is that the out-of-band components of the low band channel must not cause excessive interference on the high band channel and vice versa. A more detailed investigation into this interference, known as crosstalk, is given in Chapter 9. Plots of the measured spectra are given in figure 7.14, followed by a discussion of the observations that can be made. In all the plots, the vertical scaling is 10 dB per division and the vertical marker is at 1800 Hz, which divides the two frequency bands.

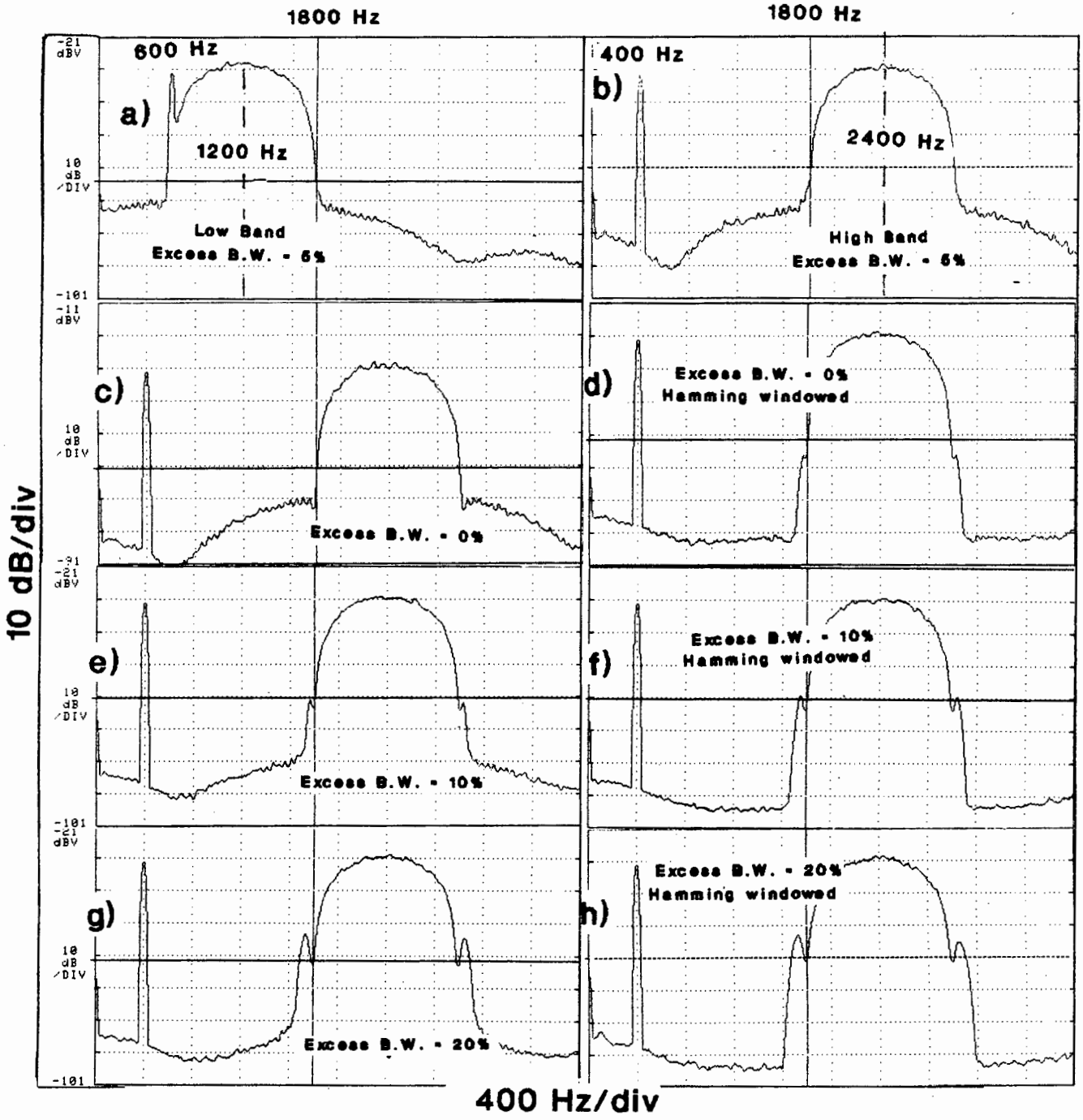


Figure 7.14 Measured transmitted power spectra

Figures (a) and (b) demonstrate typical low and high band spectra for the case where 5% excess bandwidth and no windowing is used. The tall spikes are the pilot tones. (600 Hz for the low band and 400 Hz for the high band). It can be seen that the first sidelobe is approximately 45 dB below the peak of the mainlobe which is excellent for a filter with such a fast roll-off. It can also be seen that the sidelobe levels of the low and high bands are approximately equal. For this reason only the high band is shown for the other cases of filter parameters.

Figures (c), (e) and (g) show the effect of varying the excess bandwidth on the frequency spectra using excess bandwidths of 0%, 10% and 20%. The sidelobe levels are decreased by approximately 10% in each case. This is because the pulse tails of the impulse response die away faster when the excess bandwidth is high, and thus the effects of truncation are reduced. The mainlobe, however, is widened as the excess bandwidth is increased. Thus there is a tradeoff. As the excess bandwidth is increased the sidelobe levels decrease but eventually the excess bandwidth itself will cause an overlap of the frequency bands. The effect of excess bandwidth on crosstalk is investigated further in chapter 9.

Figures (d), (f) and (h) show the effect of using a Hamming window on the FIR filter lookup table. In all cases the sidelobes are reduced at the expense of a slightly wider mainlobe. The sidelobe levels do not differ substantially in the three cases shown. From this it can be concluded that the effect of the window decreases as the excess bandwidth is increased. This is to be expected as the window reduces the discontinuity caused by truncation, which is already small when the excess bandwidth is large. In Chapter 9 the effect of windowing on the demodulated signal is investigated.

7.6 SUMMARY

A DSP56001-based 49 QPRS transmitter architecture has been described, comprising DSP56001 assembler software, hardware and a Turbo Pascal program which is used to initialize the DSP56001. All the complex modem functions, including duobinary encoding, Nyquist filtering, quadrature modulation and the generation of pilot tones are implemented in software on the DSP56001. The transmitter is able to transmit either the low or the high band signal, depending on the setting of an external hardware switch. The waveforms that were generated were exceptionally clean, and the eye diagrams showed that the inherent system degradation was practically nil. The spectra were equally well defined, with good out-of-band suppression being achieved. It was shown that by using a windowing function and/or increasing the excess bandwidth, lower sidelobes are obtained at the expense of a wider mainlobe.

REFERENCES

[7.1] Horowitz, P. and Hill, W., **The Art Of Electronics**, Cambridge University Press, 1985, Pg 159.

8. RECEIVER DESIGN

8.1 OVERVIEW OF RECEIVER DESIGN

As already mentioned, the receiver was not built as a hardware/software real-time combination like the transmitter. Instead, to limit the scope of the project, a software version of the receiver, 'RECEIVE.ASM', was written to demonstrate the implementation of the receiver on the DSP56001. The only hardware used was external X and Y data memory and its associated decoding logic, which was wire-wrapped to the DSP56001 card. The circuit diagram was given in Chapter 7.

Like the transmitter, a separate Turbo Pascal program, 'RECEIVE.PAS', was written to activate and initialize the DSP56001. It also controls the transfer of data between the disk and the DSP56001. A real-time version of the receiver would require analog-to-digital conversion and some adjustment to the input/output sections of the programs, but the remaining sections of the design would be unchanged.

The receiver operates on a sampled 49 QPRS waveform which is stored in the file 'DSPOUT.DEC' and returns the decoded output to another file, 'BIN_OUT.DEC'. The file of 49 QPRS samples was generated by altering the transmitter programs described in Chapter 8 so that binary bits are read from the file 'BIN_IN.DEC'. The 49 QPRS output is then stored in the file 'DSPOUT.DEC' to be read by the receiver program. By comparing the files 'BIN_IN.DEC' and 'BIN_OUT.DEC', the correct operation of the receiver program could be verified.

8.2 TURBO PASCAL HOST PROGRAM

The Turbo Pascal host program, 'RECEIVE.PAS', is executed on the host PC. A flowchart of the program is given in fig 8.1 to illustrate its operation more clearly, while the source code is given in Appendix C. The program performs the following functions:

a) Activates the DSP56001:
Like the transmitter, the utility program DSP_Go is used to activate the DSP56001.

b) Transfers the FIR filter table from the disk to the DSP56001:
The FIR filter lookup table is loaded from the file 'LOOKUP.DEC'. It is then scaled appropriately and downloaded to the DSP56001.

c) Reads the transmitted data from the file 'DSPOUT.DEC', which is stored on disk:
Reads a finite section of the sampled 49 QPRS waveform from the file 'DSPOUT.DEC'. The version given in the appendix skips every second point because the receiver sampling rate is 16 samples per symbol period, while the transmitter generates 32 points per symbol period.

d) Downloads the input data to the DSP56001 and reads the decoded bits from the DSP56001:
The input data is transferred to the DSP56001. For every 16 data points downloaded, a set of four decoded bits is uploaded from the chip. This data is then stored in the file 'BIN_OUT.DEC'.

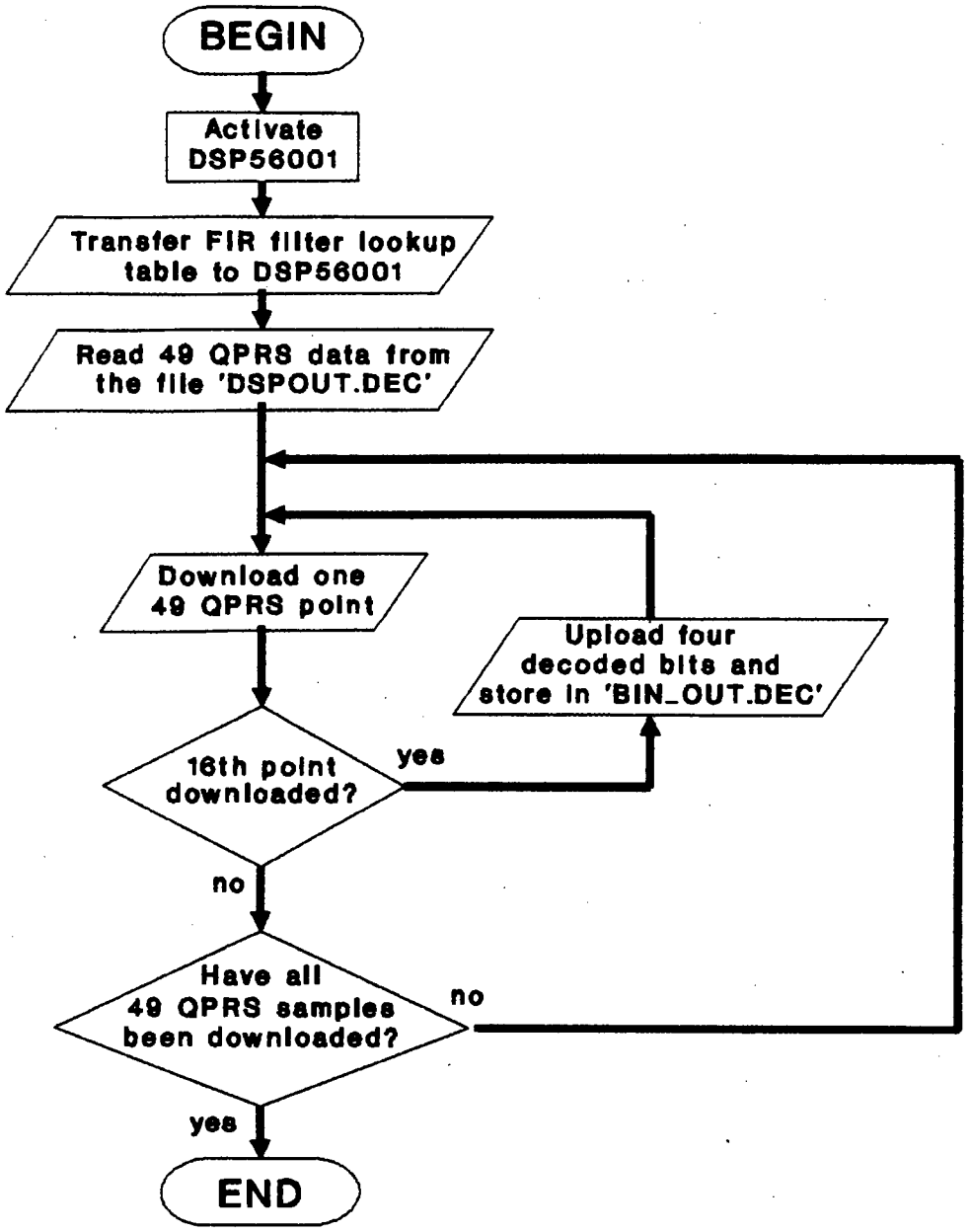


Figure 8.1 Flowchart of receiver host program

8.3 DSP56001 ASSEMBLER PROGRAM

8.3.1 Introduction

This program was written so that the DSP56001 would perform all the main 49 QPRS receiver functions except clock recovery and analog to digital conversion. The program can be set up to demodulate either the low band or the high band. A flowchart of the basic program functions is given in figure 8.2, while a listing of the source code is given in Appendix C. As already mentioned, all program I/O operates via the host interface and the Turbo Pascal program so that data is stored and read from disk.

The description that follows refers specifically to the case where the receiver filter used is a 384 point square-root raised-cosine FIR filter with 16 points per symbol period. By adjusting the first few lines of code (the equate statements), the program can be set to use any filter length. The operation of the receiver program has many similarities to the operation of the transmitter. Sections which are similar will not be explained in detail.

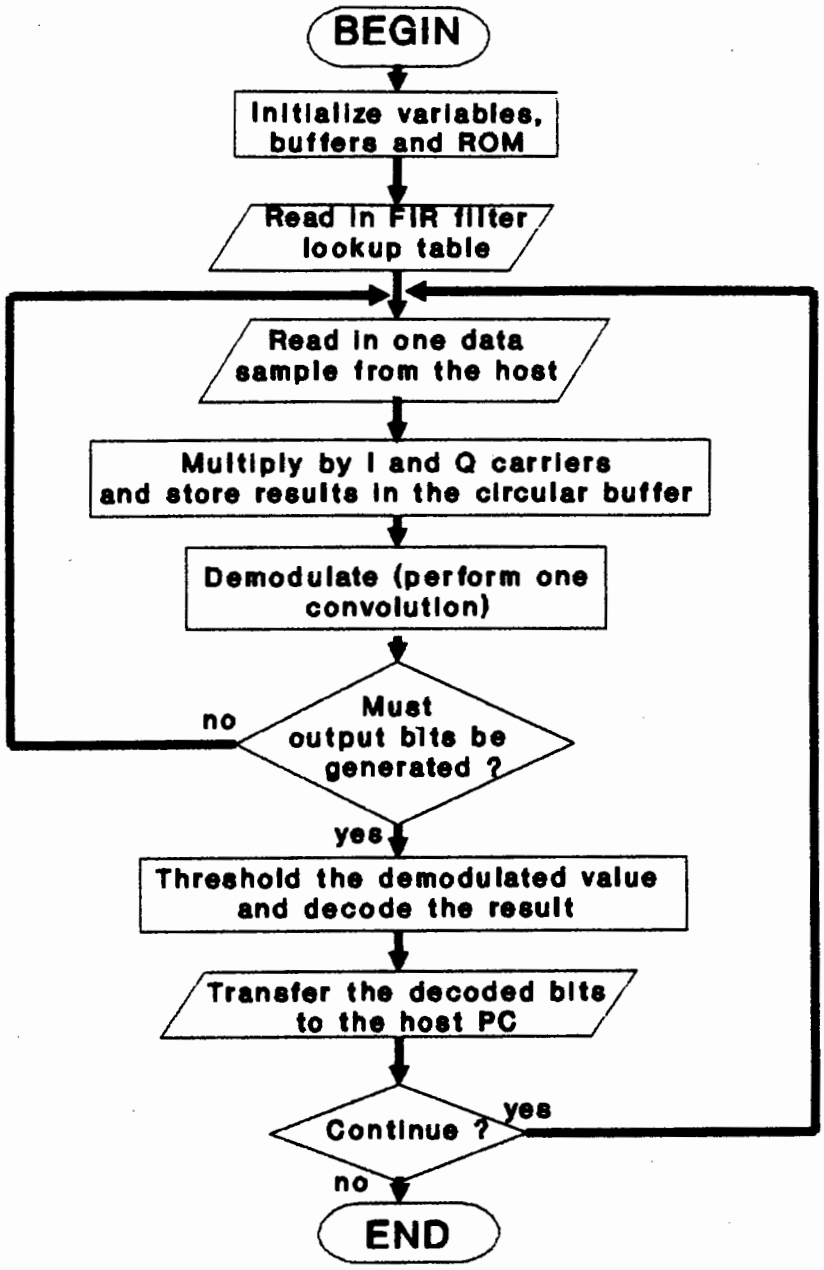


Figure 8.2 Flowchart of the receiver assembler program

The following aspects of the program will be expanded upon:

a) Address register assignments

b) FIR filter and circular buffer operation

c) Thresholding and decoding routine

8.3.2 Address register assignments

The assignments and functions of the addressing registers will be explained as their operation is central to the explanation of receiver operation. As in the transmitter, the discussion refers to the R (address), N (offset), and M (modifier) registers.

R0,M0,N0:
The functions of these registers are the same as in the transmitter.

R1:
This register is used as a general purpose counter. It ensures that on every 16th interrupt, the thresholding routine is invoked. The thresholding routine is actually combined with the decoding process. The output of the thresholding routine will be the recovered bits.

R4:
This register is a pointer to the circular buffer which holds the values generated when the 49 QPRS samples are multiplied by the in-phase and quadrature carriers. Like the transmitter, the I and Q channel values are stored in pairs in the same buffer.

R6,N6,M6,R7,N7,M7:
As in the transmitter, these registers are dedicated to the generation of the in-phase and quadrature carriers using the built-in sinewave lookup table. The R registers point to the lookup table and the N registers set the frequency, while the M registers are to used for modulo addressing.

8.3.3 FIR filter and circular buffer operation

The received data is filtered by a square-root raised-cosine FIR filter. The overall response of this filter and the transmitter will be that of a Nyquist, raised-cosine filter. The operation carried out is simply a convolution of the input data buffer with the stored FIR filter impulse response. In the program which was written, one convolution is carried out for every point that is input into the data

buffer. Like the transmitter, the operation is carried out on both the I and Q channels. The output of this filtering operation will be the recovered baseband signal.

Although the entire baseband waveform was recovered in the program which was written, this was done only for clarity (the baseband eye diagram could be viewed). In a real-time version, many of the operations would be redundant. This is because the decoded bits are derived solely from the points of maximum eye opening of the baseband waveform. Thus, only these points need to be recovered. If the receiver has a sampling rate of 16 points per symbol period, then the filter output need only be calculated on every 16th point. This would save a great deal of processing time.

8.3.4 Thresholding and decoding routine

The thresholding routine performs level slicing and duobinary decoding of the demodulated data simultaneously. This is possible because the bit-by-bit detection method is used (i.e. each decoding decision is based on a single demodulator output point). The threshold levels are set up in the initial equate statements in the program. The routine operates as follows:

a) The output bits are set to those corresponding to the input value being below the lowest threshold.

b) The input data is then compared to this threshold level.

c) If the input data has a value lower than the threshold, then the output bits are left unchanged. If the data has a higher value than the threshold, then the output bits are set to those corresponding to the input value being above this threshold and below the next highest threshold level.

Steps (b) and (c) are then repeated for each threshold level (except the 7th). It is not necessary to process the 7th level because if the input data is greater than the 6th threshold it can only be in the region above the 7th level, which corresponds to a unique decoded symbol value.

8.4 PROPOSED REAL-TIME ARCHITECTURE

8.4.1 Introduction

This section describes the hardware that would be necessary if the receiver were ever upgraded to function as a real-time 49 QPRS receiver. The required timing signals would be derived from the pilot tones using a separate clock recovery circuit for each band. an A/D would also be installed for data sampling. The receiver would be programmed to receive either the low band or the high band. A block diagram of a proposed real-time architecture is given in figure 8.3, followed by a brief discussion of its functions.

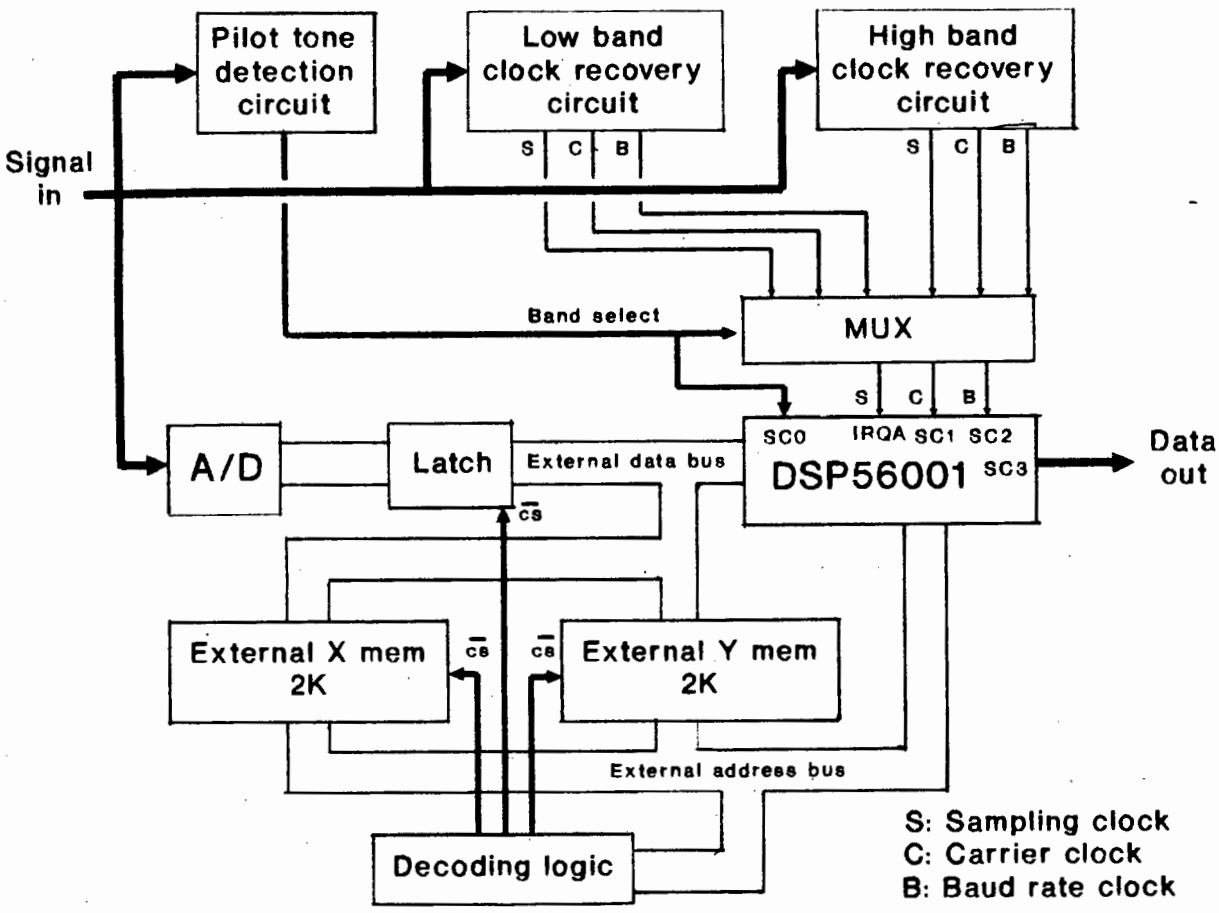


Figure 8.3 Proposed real-time receiver architecture

8.4.2 A/D and latch

The A/D must have a conversion frequency greater than 19.2 kHz, which is the recommended receiver sampling frequency. It is shown in Chapter 10 that the modem design has the ability to reject echoes in the order of 20 times greater

than the received signal. The A/D should have enough dynamic range to sample the received signal together with a large echo, if the full capabilities of the scheme are to be realized. It is recommended that the resolution of the A/D should be at least 12 bits, which would offer 72 dB of dynamic range. The latch is only necessary if the A/D does not have a tri-state function on its outputs.

8.4.3 Pilot tone detection circuit

This circuit, which could also have been included in the PLL's, would allow the modem to automatically receive either the low or high band. The detection of the 400 Hz pilot tone indicates that the high band is being received, while the detection of the 600 Hz pilot tone indicates that the low band is being received. The circuit would communicate with the DSP56001 via one of the general purpose inputs of the processor. The outputs of the two clock recovery circuits would be multiplexed so that the correct timing signals are received by the DSP56001. This would also be controlled by the band select signal.

8.4.4 Clock recovery circuits

The clock recovery circuits would use the pilot tones to regenerate the sampling frequency, carrier frequency and the baud rate (or symbol timing) clock. The latter two signals would be read by the DSP56001 via its general purpose I/O pins. The sampling clock would be connected to an interrupt pin as this clock forms the basis of the whole timing scheme. After receiving an interrupt, the I/O pins would be examined and if required, the internal carrier and symbol rate timing would be reset. Like the transmitter assembler program, the main program functions would take the form of an interrupt service routine.

8.4.5 External memory

The external memory stores the FIR filter lookup table as well as the circular buffer. The circular buffer stores one point for every input sample and will be of the same length

as the FIR filter lookup table. In the block diagram that is shown, the lookup table and circular buffer would be stored in the X and Y data memory respectively. This section of the hardware, together with some of the decoding logic, was actually built, as it was also necessary for the operation of the software version of the receiver.

8.4.6 Decoding logic

The function of this logic is to use the address lines to generate chip select signals for the peripherals. This ensures that there is no bus contention.

8.4.7 DSP56001 functions

The DSP56001 would perform the same functions as those performed in the software receiver version. These would include demodulation, Nyquist lowpass filtering, thresholding and duobinary decoding. It would then output the recovered bitstream via a pin on port C which is configured as a general purpose I/O pin

8.5 SUMMARY

A software-only version of the receiver has been described which was programmed on the DSP56001 in order to demonstrate how the bitstream would be recovered from the 49 QPRS signals. The receiver operates on data stored in a file, as opposed to real-time data. Like the transmitter program, the DSP56001 is initialized by a Turbo Pascal program, which is executed on the host PC. This program also coordinates the transfer of the input and output data between the disk and the DSP56001. A proposed real-time architecture of the receiver has also been described. The additional hardware that would be necessary was discussed.

9. INTERCHANNEL CROSSTALK SIMULATION

9.1 THE CAUSES OF INTERCHANNEL CROSSTALK

In the modem scheme that has been investigated in this thesis, frequency division multiplexing is employed as a means of separating the transmit and receive channels. However, due to the limited bandwidth available on M.1020 telephone lines, there is very little flexibility in deciding which carrier frequencies to use for the 49 QPRS signals. As was mentioned before, the proposed scheme uses the same carrier frequencies as the existing V.22bis system, but does not have any guard band separating the two signals.

The scheme relies on the receiver Nyquist filtering process to reject unwanted frequencies during demodulation as shown in section 5.1. The main unwanted frequency components will be the echo from the channel being transmitted by the modem, which must be separated from the received signal. The scheme would be foolproof if a rectangular lowpass filter were realizable. Unfortunately, this would require storage of an infinitely long impulse response. The impulse response of the filter must be truncated which causes sidelobes to appear in the filter frequency response. Some of the frequency components in these sidelobes of the low band channel will overlap with the high band channel and vice versa. As a result, it is not possible to completely isolate the signals from each other using highpass and lowpass filters, as in the V.22bis scheme. The effect of the truncation is also to cause a distortion of the filter response which leads to a certain amount of inherent intersymbol interference. The following are the main factors that will influence the effectiveness of the FDM scheme:

a) The effectiveness of the hybrid transformer:
Any isolation offered by the hybrid transformer will not have to be accounted for by the receiver filter. The M.1020 specification includes line impedance equalization. This means that it should be possible to obtain a fairly good impedance match to the telephone line. Thus it can be assumed that the hybrid will offer good isolation, although the actual amount of isolation may show large variations for different lines.

b) The amount of attenuation on the signal on the telephone line:
The difference in signal power between the received signal and the echo from the transmitted signal will be directly related to the transmission loss. Since the modem is intended for operation on specially conditioned telephone circuits, it can be expected that the loss will not be excessive. The M.1020 specification for transmission loss was given in section 4.1. The maximum loss is 3 dB in most of the usable bandwidth.

c) The FIR filter length:
The longer the stored impulse response, the less the effect of truncation. The width of the filter should thus be made as long as possible.

d) The amount of excess bandwidth used:
By increasing the excess bandwidth of the filter, the pulse tails will die away faster and lower sidelobes will be obtained as a result of truncation. There is a trade-off however, as the excess bandwidth will itself cause some frequencies to overlap. However, due to the null in the duobinary frequency response, which appears at the cut-off frequency, a moderate excess bandwidth can be expected to cause only a small overlap of frequencies.

e) Windowing of the lookup table:

Windowing techniques can be used to reduce the discontinuity at the truncation points. This would result in reduced FIR filter sidelobe levels. It is possible that this would result in less interchannel crosstalk. The windowed impulse response, however, will no longer be a true Nyquist-type impulse response, which would result in the modem having a certain amount of inherent ISI.

9.2 SIMULATION PROGRAM

The computer simulation intends to show more clearly how the factors which have been mentioned will effect interchannel crosstalk. The simulation program, which is written in Turbo Pascal, is given in Appendix D. A flowchart of the basic program structure is given in figure 9.1. The program is a partial translation of the DSP56001 program and uses the identical FIR filter lookup tables. The program will be described in broad terms only as the complexities of its operation are not as important as the simulation results. The operation of the program is as follows:

a) The FIR filter lookup table loaded from disk and a Hamming window is applied to it if desired.

b) The high and low band 49 QPRS signals are generated.

c) The two channels are then added to each other in varying proportions. The amplitude of the low band is kept constant while the amplitude of the high band is varied. This simulates the relative amplitudes of the received signal and the high band echo.

d) The low band channel is then demodulated to give the baseband eye diagram. The interference caused by spectral spill-over from the high band can then be observed.

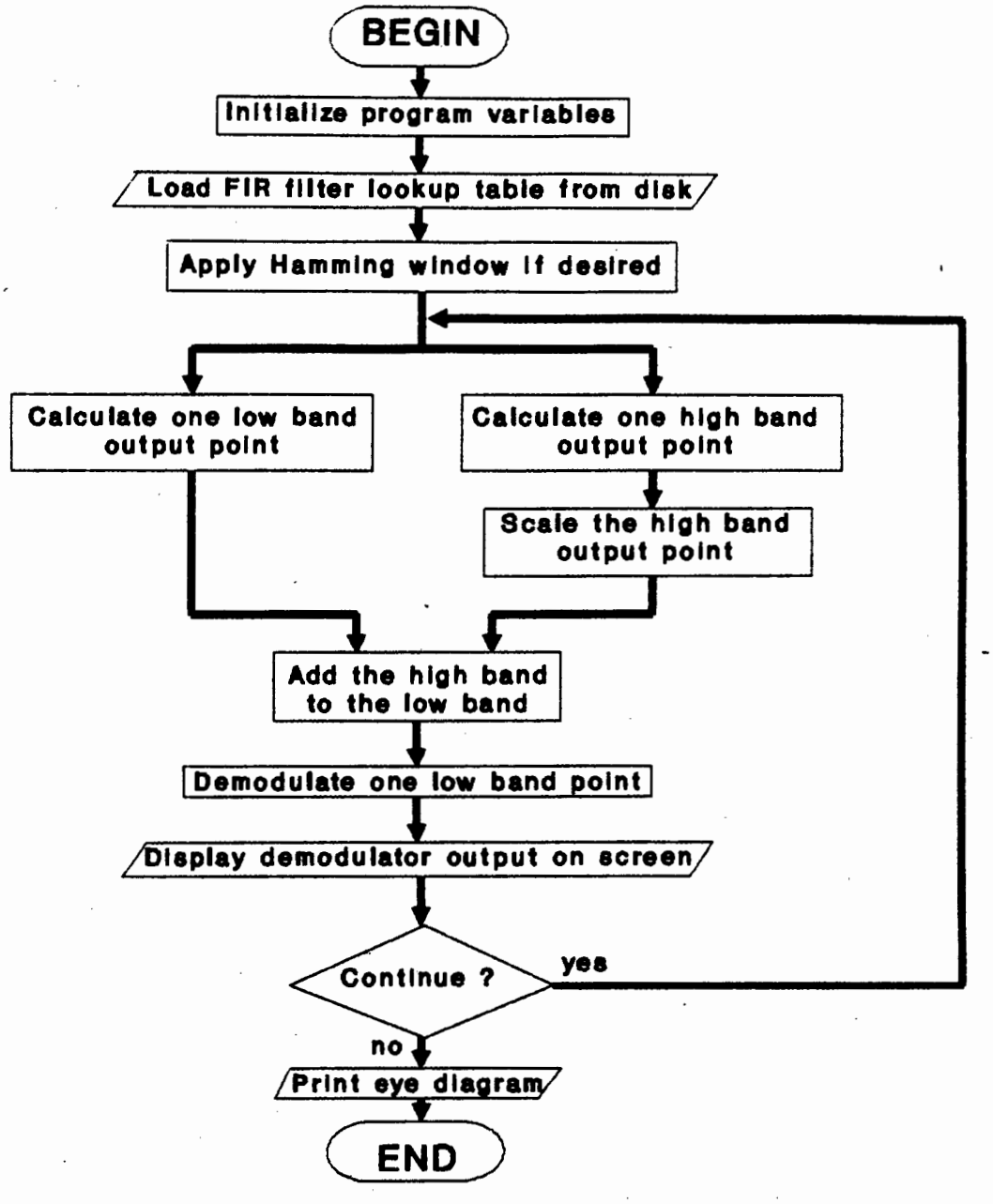


Figure 9.1 Flowchart of the crosstalk simulation program

In all eye diagrams shown in this chapter, the horizontal axis represents the time base, while the vertical axis represents amplitude. The vertical scale has been expanded so that only the central eyes are shown for clarity. Measurements of the vertical eye opening only, were taken. This is because the receiver symbol timing, which is related to the horizontal eye opening, is assumed to be perfect.

It should be pointed out that as the eye opening decreases, it becomes less well defined. It thus becomes more difficult to take accurate measurements. The measurements taken of vertical eye opening are thus approximations. The error, however, is not large enough to detract from the investigation. The effect could have been reduced by generating many more points in each simulation but the time taken for this would have been excessive.

The lookup tables used, which are loaded from disk when running the program, were generated beforehand using the numerical integration program which was written. 16 points per symbol are used throughout as this is enough to avoid any aliasing effects.

It is assumed that the interference caused by the high band on the low band is the same as the interference caused by the low band on the high band. This is because their spectra have identical shapes, even though they are centered about different carrier frequencies. For this reason both cases are not investigated. In order to further validate this assumption it was checked by running a simulation of both cases using the same parameters. The resultant eye diagrams are shown in figure 9.2. It can be observed that the amount of interference is similar in both cases. The high band is demodulated simply by changing the carrier frequency in the demodulation routine. The FIR filter used is 16 symbol periods wide and has 10% excess bandwidth.

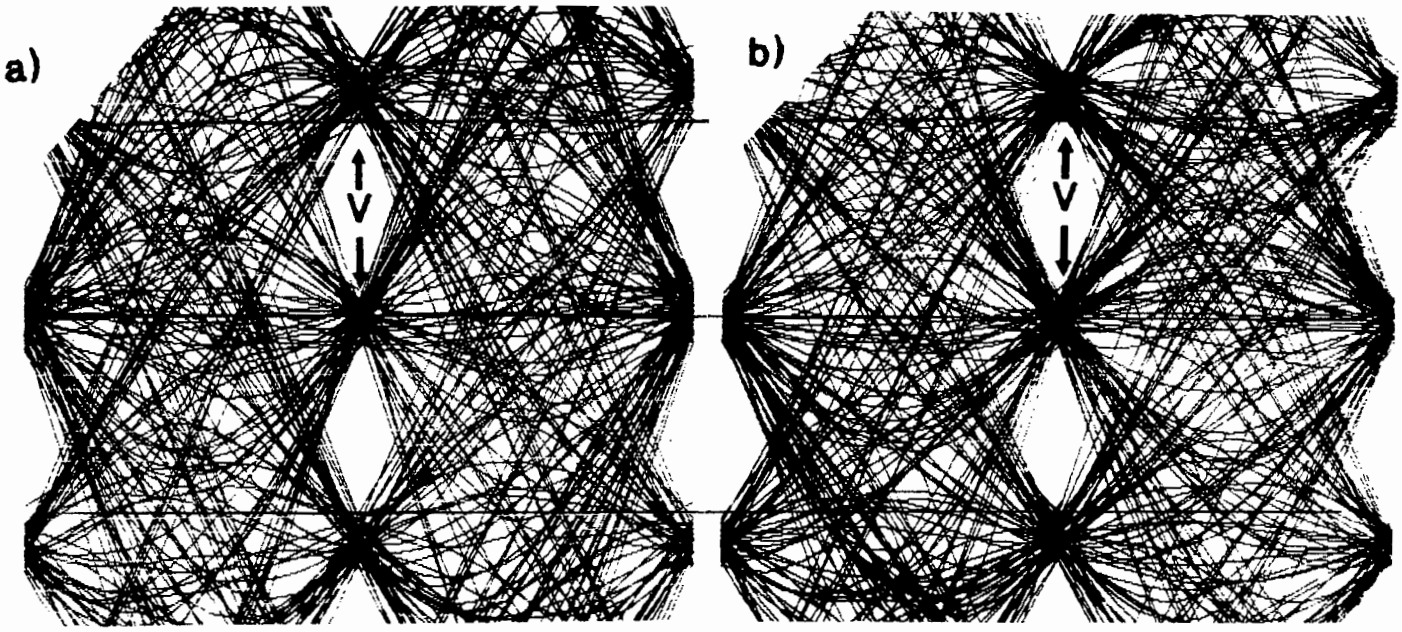


Figure 9.2 Crosstalk caused by the high band on the low band and vice versa. The demodulated signal is as follows:
a) Low band where the received signal is low band + (high band * 2). (6 dB echo)
b) High band where the received signal is high band + (low band * 2). (6 dB echo)

Although there are many combinations of parameters that could have been investigated, the scope of the simulation was limited to show only the most important trends. A problem with the simulation is that if a system using the maximum possible length lookup table that the DSP56001 timing constraints will allow is used, the simulations take an extremely long time to execute. This is why shorter lookup tables were used wherever possible.

9.3 SIMULATION RESULTS

9.3.1 Effect of interchannel isolation

The eye diagrams shown illustrate the effect of the isolation between the channels, which will depend on the effectiveness of the hybrid and on transmission loss. The transmitter which was built, used a sampling rate of 32 samples per symbol period and was able to use lookup tables up to 24 symbols wide. If the sampling rate is halved to 16 points per symbol period, then there will be enough time available to use lookup tables up to 48 symbols wide. This figure was taken as the maximum for the system, although by optimizing the code, it may be possible to use slightly longer lookup tables. Performance will improve as the lookup table length increases, as shown in the next section. A 10% excess bandwidth square-root raised-cosine filter having a width of 48 symbols was used to obtain the eye diagrams in figure 9.3.

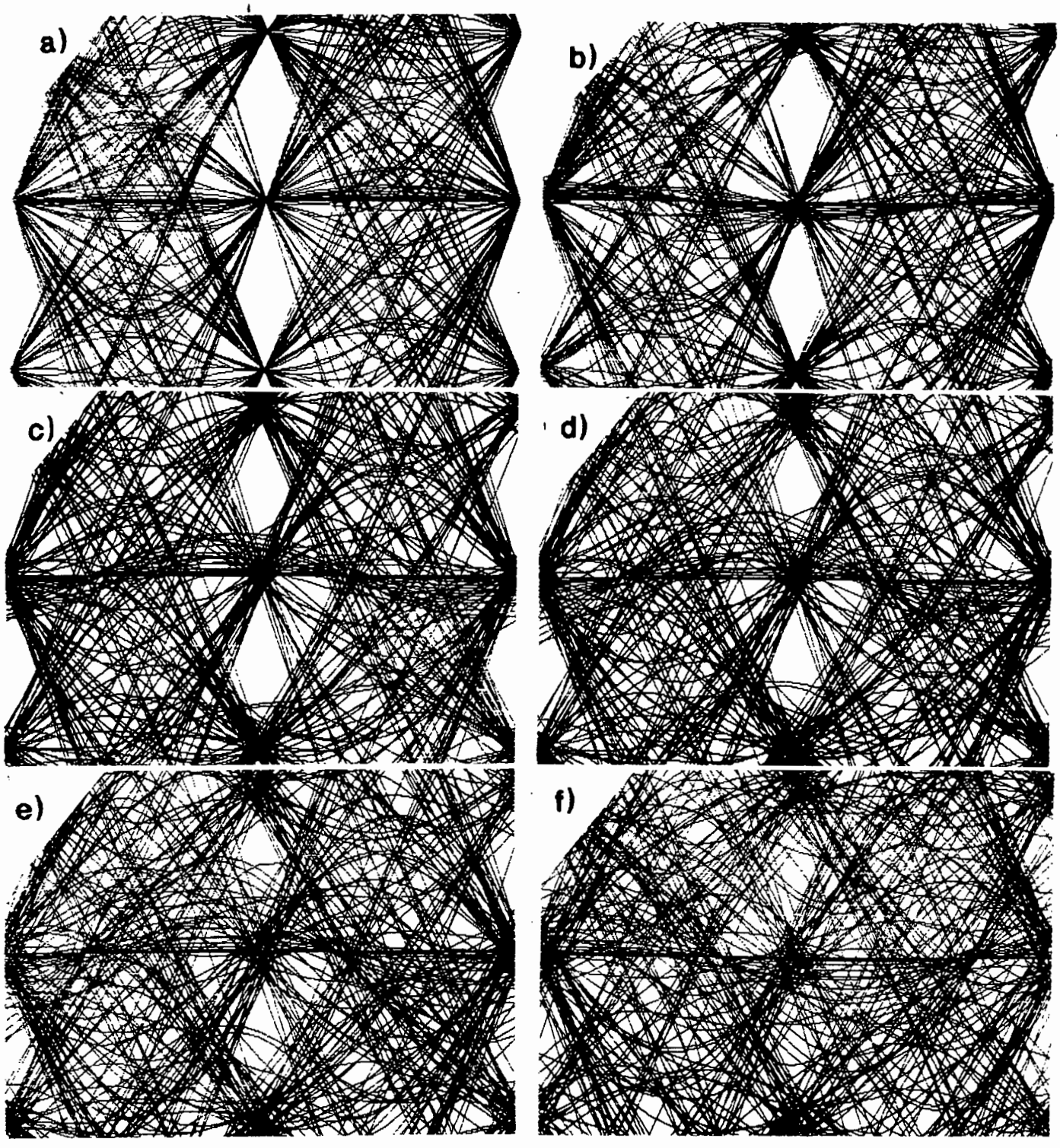


Figure 9.3 Demodulator output eye diagrams for varying degrees of isolation. 10% excess bandwidth and an impulse response width of 16 symbol periods is used. The received signals are as follows, where the decibel figure in brackets indicates the power in the high band relative to the low band:

a) Low band only
b) Low band + (High band * 4) (12 dB)
c) Low band + (High band * 8) (18 dB)
d) Low band + (High band * 12) (22 dB)
e) Low band + (High band * 16) (24 dB)
f) Low band + (High band * 20) (26 dB)

From these eye diagrams, the following graph was plotted which indicates the vertical eye opening versus the amplitude of the added high band signal:

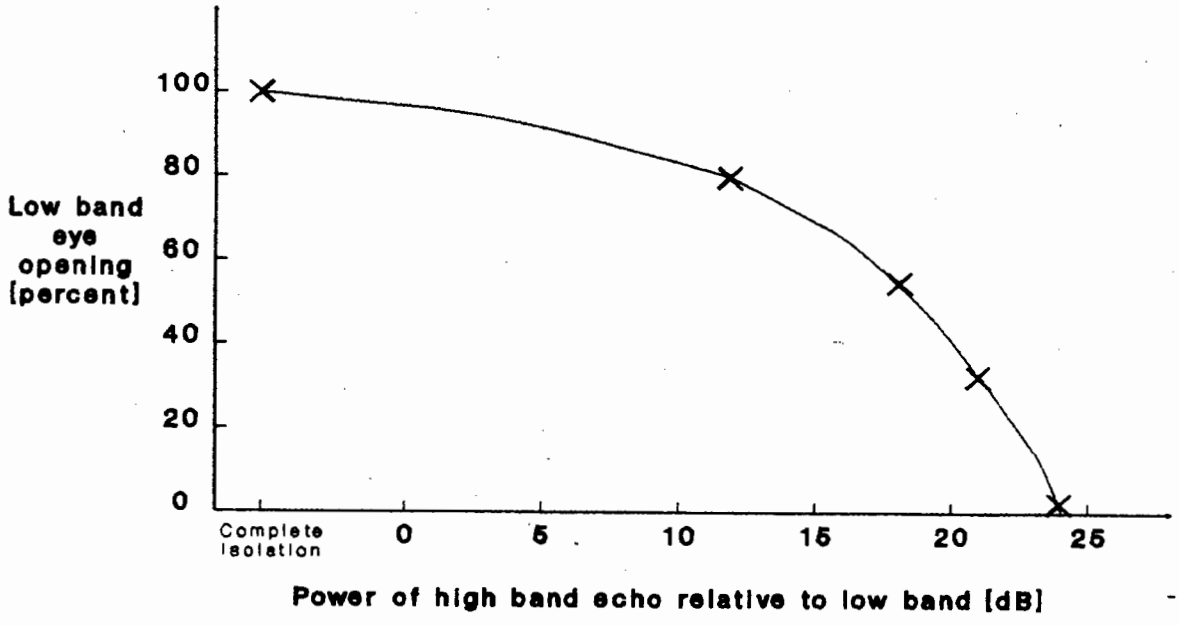


Figure 9.4 Graph of vertical eye opening versus amplitude of added high band signal.

From the graph it can be seen that the scheme has the ability to reject echoes that have a signal power of approximately 24 dB above to the received signal. It would, however, be impractical to operate the system at this limit. This is because noise effects would have to be accounted for, as well as timing jitter and inaccuracy in the detection thresholds of the receiver. It is probably more realistic to say that echoes up to about 10 dB above the received signal level will not cause a serious loss in performance. The amount of echo will rely heavily on the fine tuning of the hybrid transformer. It should be possible to tune the hybrid fairly accurately as M.1020 telephones have equalized impedance and are leased to the customer (The hybrid could be set up for that particular line).

9.3.2 Dependence on FIR filter length

In this simulation, the amplitude of the high band was kept at four times the amplitude of the low band. A 10% excess bandwidth filter was used and the width of the lookup table

was varied in each case. The eye diagrams given in figure 9.5 show the effect of varying the length of a 10% excess bandwidth filter lookup table. The received signal is the low band + (high band * 4). Thus the high band is 12 dB above the low band. The diagrams show that the length of the lookup table, in terms of symbol periods, will have a very large effect on the performance of the scheme.

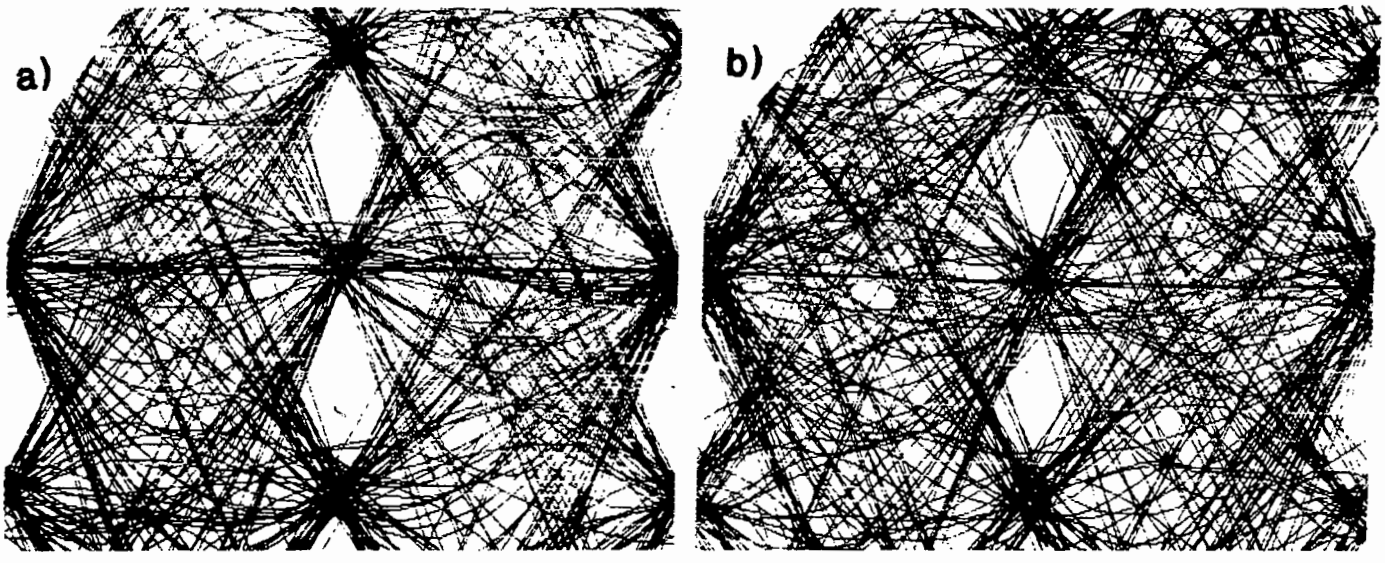


Figure 9.5 The effect of the FIR filter lookup table length on crosstalk. The width of the lookup table is:
a) 32 symbols
b) 16 symbols

When a lookup table with a width of 8 symbol periods is used the eye is completely closed. Thus the trend of increasing performance as lookup table length increases has been demonstrated.

9.3.3 Dependence on excess bandwidth

Figure 9.6 shows the effect of the amount of excess bandwidth used on the demodulated waveform. In all cases, the amplitude of the high band was kept constant at double the amplitude of the low band (or 6 dB above the low band). A lookup table length of 16 symbol periods was used in the simulations.

From the eye diagrams, the following graph of vertical eye opening versus excess bandwidth was plotted.

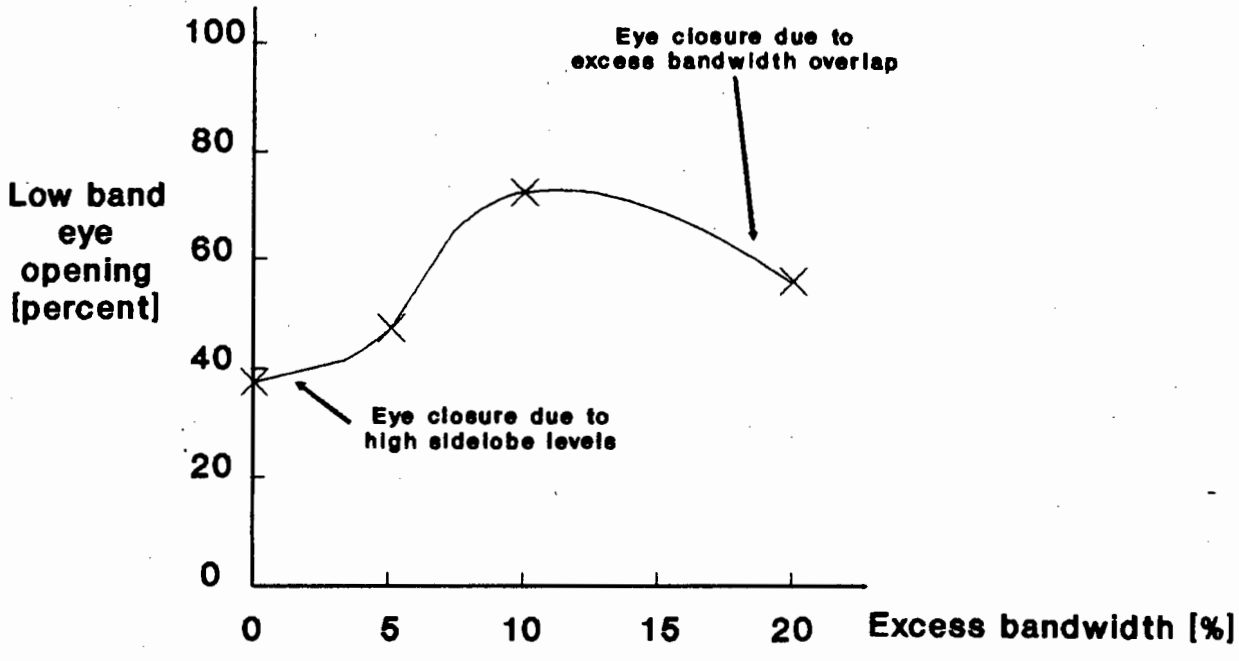


Figure 9.7 Graph of vertical eye opening versus excess bandwidth

It can be seen that for low excess bandwidth the degradation of the signal is relatively high. This is due to the high sidelobe levels caused by truncation of the impulse response at a point where the pulse tail has not died away sufficiently. As the excess bandwidth increases the signal

improves as the negative effect of truncation decreases. For higher excess bandwidths, the degradation increases again. This is caused by the excess bandwidth itself causing an overlap of the signals. Thus the optimum amount of excess bandwidths about 10% in the above case. The effects of excess bandwidth on the transmitted signal spectra were shown in chapter 7.

9.3.4 The effect of windowing the impulse response

It was shown in Chapter 7 that the Hamming window could be used to reduce the sidelobe levels at the expense of a slightly widened mainlobe. In this simulation, the effects of the Hamming window on crosstalk is investigated. 10% excess bandwidth is used throughout.

A number of deductions can be made from the eye diagrams in figure 9.8:

From (a) and (b), which show the effect of the Hamming window where there is complete isolation between the two bands, it can be seen that the Hamming window has not degraded the signal. Even though the Hamming window distorts the Nyquist type impulse response, the reduced distortion due to lower truncation of the impulse response has resulted in a somewhat sharper eye diagram.

From (c) and (d) it can be seen that using the Hamming window has greatly reduced the amount of crosstalk. This is due to the fact that the effect of truncation is decreased, leading to lower sidelobe levels.

From (e) and (f) it can be seen that the window has far less effect on crosstalk when a longer lookup table is used. This is because in a longer lookup table the pulse tails are far smaller at the truncation points, an so the effect of truncation is already small prior to windowing.

9.4 Summary

This investigation of interchannel crosstalk has highlighted some of the important points affecting the performance of the proposed modem scheme.

It has been shown that crosstalk will get progressively worse as the interchannel isolation decreases. The effectiveness of the scheme will thus depend heavily on the fine tuning of the hybrid transformer. A graph has been plotted which shows the eye closure caused by increasing echo. It is difficult to predict an absolute maximum amount of echo that will allow the system to operate efficiently as this will be determined by other parameters such as noise, timing jitter, etc. Nevertheless, it appears that the scheme has the ability to reject large amounts of echo and would be

degraded very little for echo less than about 10 dB above the received signal.

There is a direct correspondence between the length of lookup table used and the performance of the system. Thus the longest lookup table length that timing and storage constraints will allow should be used.

The use of a windowing function will only improve preformance substantially if a suboptimal filter is used. When a more optimal filter is used, the effect is a very slight widening of the vertical eye opening. In can thus be concluded that windowing offers a very small advantage, but due to its trivial implementation, it is probably worth using one.

10. CONCLUSION

This chapter will begin with a short summary of the entire thesis. In section 10.2, the results of the thesis project are discussed. Finally, recommendations for future work are made in section 10.3.

10.1 CHAPTER SUMMARY

In Chapters 2 and 3, the background theory to this thesis project was given. This included a discussion of conventional, memoryless systems, followed by an introduction to PRS. It was shown that PRS can be used to obtain a higher data throughput than memoryless schemes, at the expense of a somewhat degraded error performance.

A few general concepts relating to this project were given in Chapter 4. These included discussions of M.1020 lines, the V.22bis modulation scheme, matched filtering and quantization noise.

Chapter 5 described the proposed 49 QPRS modulation scheme for achieving 4800 bps full-duplex transmission on an M.1020 line.

In Chapter 6, the advantages of using digital signal processing technology for the modem implementation were outlined. The implementation of the scheme on a Motorola DSP56001 DSP chip was then discussed.

In Chapter 7, the transmitter design and construction was described, along with a discussion of the output waveforms and frequency spectra.

Chapter 8 described the design of a DSP56001-based receiver. The software of the receiver was implemented, while a proposed real-time design was described.

Finally, in Chapter 9, crosstalk between the two channels was investigated by computer simulation and the results evaluated.

10.2 DISCUSSION OF RESULTS

Most of the results obtained in thesis are based on the transmitter output waveforms and the computer simulation of interchannel crosstalk.

Some of the advantages of a DSP-based architecture have been demonstrated in this project. The design was reliable and, since it is digital, does not require any tuning.

The transmitted waveforms were observed to be extremely clean. Intersymbol interference at the instants of maximum vertical eye opening was unobservable for both low and high bands. This indicates that the performance of the transmitter was optimum, as it does not have any inherent degradation.

The spectra of the transmitted signals were also observed to be precisely defined. Very sharp bandlimiting was achieved with excellent suppression of sidelobes. The parameters of the FIR filter lookup table were varied which produced interesting results. It was shown that as the raised-cosine excess bandwidth is reduced the sidelobe levels are increased. The sidelobe levels ranged from approximately -45 dB (0% excess bandwidth) to approximately -57 dB (20% excess bandwidth). It was also shown that a Hamming window, which is a typical windowing function for this type of application, could be applied to the lookup table to obtain lower sidelobe levels at the expense of a slightly wider mainlobe. The advantages of widowing are reduced as more excess bandwidth is used. The decrease in sidelobe levels ranged from 17 dB (using 0% excess bandwidth) to about 2 dB (using 20% excess bandwidth).

The successful operation of the receiver software which was written suggests that the receiver could be implemented on the DSP56001 just as effectively as the transmitter.

The computer simulation of interchannel crosstalk showed that the scheme has the ability to reject large echoes of

the transmitted signal, even though no bandpass filtering is used. In noiseless conditions with perfect clock recovery, the receiver can reject echoes up 24 dB above the received signal. These conditions are unrealistic in practice however, and it would be more realistic to say that echoes in the order of 10 dB or less above the received signal will have negligible effect on the performance of the system.

It was also shown that the amount of excess bandwidth used in the square-root raised-cosine filters will have a large effect on crosstalk. If the excess bandwidth used is small, the large discontinuity in the truncated impulse response leads to high sidelobe levels. The resultant spectral overlap between the two channels causes a large amount of crosstalk. Conversely, if too much excess bandwidth is used, sidelobe levels will be low but the excess bandwidth itself will cause crosstalk. In the particular simulation which was carried out, eye closure was 62% for 0% excess bandwidth, 28% for 10% excess bandwidth, and 43% for 20% excess bandwidth. Thus there is an optimum amount of excess bandwidth that must be used (in this case around 10%).

It was also shown that as lookup table length is increased, the performance of the system will improve. This result is obvious, as a desired impulse response can be better approximated by a longer lookup table.

The effect of using a Hamming window on crosstalk was also investigated. It was shown that when a short, suboptimal lookup table is used, the Hamming window greatly increased

performance. In the case of a 5% excess bandwidth, 8 symbol periods wide lookup table, where an echo of 12 dB above the received signal is encountered, the Hamming widow improved the vertical eye opening from approximately 20% to approximately 70%. When a more optimal lookup table was used, however, the Hamming window had no visible effect.

It is possible that the digital implementation of 49 QPRS could be used to double the data throughput of the 16 QAM systems currently in use in V.22bis modems.

10.3 RECOMMENDATIONS FOR FUTURE WORK

Due to the complexity of the project, it was considered beyond the requirements of a half-thesis to complete the whole modem as a transmitter/receiver combination. The results obtained are extremely promising, although by no means complete.

The next logical step would be to complete the modem prototype in order to gain a more quantitative indication of the system performance. This would include bit-error-rate tests. The following components would have to be implemented to complete the modem:

a) An A/D would have to be installed on the receiver which would be configured to operate in real-time.

b) The clock recovery circuits would have to be built and tuned to minimize timing jitter.

c) An adaptive equalizer would also have to be implemented, if the system is to be optimized.

The implementation of maximum-likelihood detection using the Viterbi algorithm should also be investigated. A hardware implementation would be considerably complex, but a software implementation using another DSP chip may solve this problem. If successfully implemented, the system would tolerate up to 3 dB more noise power, for a given probability of error.

The use of data compression could also be investigated. A typical compression ratio of 4:1 would give 19.2 kbps full-duplex operation.

APPENDIX

APPENDIX A - COMSIG 90 PAPER

THE DEVELOPMENT OF NOVEL MODEM STRUCTURES TO ENHANCE THE THROUGHPUT OF DIAL UP TELEPHONE LINES

RUSSEL HORWITZ, TIM D. COURTENAY

and ROBIN M. BRAUN, Member IEEE.

University of Cape Town

ABSTRACT

A popular approach to designing high-speed modems for use on voice-grade telephone channels is to use a high order modulation scheme combined with a digital architecture. This paper describes the development of a modem structure which uses the DSP56001 signal processor to implement 49 quadrature partial response signalling (49 QPRS). The choice of 49 QPRS is justified by its high spectral efficiency and reasonable error rate performance. This system is being developed for two applications, which are described in the paper.

INTRODUCTION

The speed of data transfer over conventional telephone lines is limited by various factors. The predominant limitations are:

1. Channel group delay and magnitude response.
2. Noise introduced by the channel.
3. Noise introduced by the hardware.
4. Hardware complexity and cost.

The modems discussed in this paper are designed for transmission over CCITT standard M.1020 voice grade telephone lines [1]. Two modem designs are described which use a digital implementation of partial response signalling to achieve high data rates at relatively low cost and complexity, while keeping noise introduced by the hardware to a minimum. Section 1 describes the 49 QPRS modulation scheme. This is compared to 16 QAM, which is commonly used for high-speed, bandwidth efficient modem applications. In Section 2, the digital implementation of QPRS is described. The transmitted eye diagram is shown. Sections 3 and 4 describe the features unique to each of the two designs. Time waveforms and power spectra are shown to illustrate their operation.

1. 49 QUADRATURE PARTIAL RESPONSE SIGNALLING

Partial response signalling (PRS) is a bandwidth efficient modulation technique which allows transmission at the Nyquist rate [2]. A controlled amount of intersymbol interference is introduced, causing a spectral shaping which allows a very high roll-off Nyquist filter to be implemented with relative ease. The modems described in this paper use class 1 PRS and employ the following coding rules:

precoding: $b_k = (x_k - b_{k-1}) \bmod 4$

encoding: $y_k = b_k + b_{k-1}$

decoding: $x'_k = y_k \bmod 4$

where x_k is the original data in the form of a 2 bit word (4 level)

b_k is the precoded sequence (4 level)

y_k is the transmitted sequence (7 level)

x'_k is the decoded data (4 level).

The low-pass filtered waveform of the 7-level sequence y_k (for one channel) is shown in Figure 1. The signal value at the eye locations is that of the transmitted y_k sequence. This scheme is used in conjunction with quadrature modulation to produce the 49 QPRS signal. The bandwidth efficiency of this signal is 4 bits/sec/Hz. Figure 2 shows the constellation diagram.

Figure 3 shows performance curves for various modulation schemes. Notice that 49 QPRS has only 2 dB worse error performance than 16 QAM, although 16 QAM requires up to double the bandwidth for the same bit rate, depending on the spectral shaping used. 256 QAM, however, has similar spectral efficiency as 49 QPRS but requires a 10 dB increase in Carrier to Noise ratio for the same error rate.

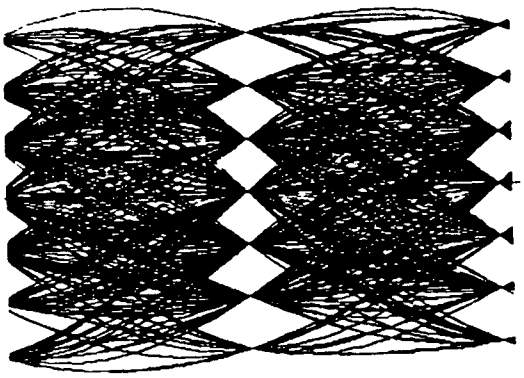


FIGURE 1 - PRS 7-level baseband eye diagram (1 channel shown).

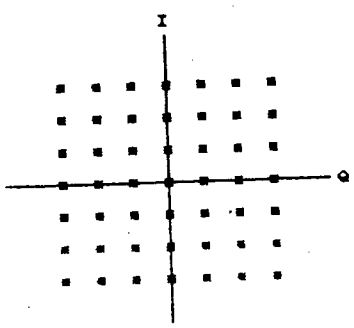


FIGURE 2 - Constellation diagram for 49 QPRS.

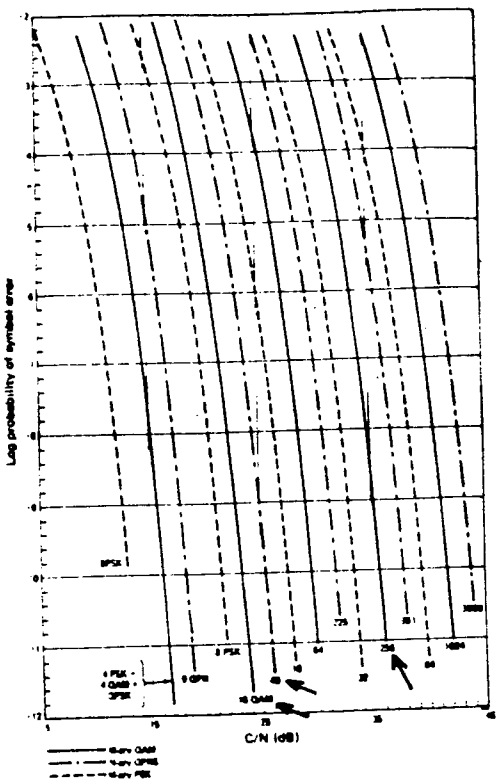


FIGURE 3 - Probability of error performance curves of M-ary PSK, M-ary QAM and N-ary QPR modulation systems [3].

2 DIGITAL IMPLEMENTATION

The Motorola DSP56001 digital signal processor is used to implement all encoding, low-pass filtering, modulation, demodulation, decoding and the addition of pilot tones. Figure 4 shows the sequence of events relating to the transmitter and the receiver. The important advantages of this processor are [4]:

1. A parallel architecture and a no-overhead, hardware DO loop allow FIR filters to be implemented in the theoretical minimum number of execution cycles (one per tap weight).
2. Modulo addressing registers allow efficient implementation of circular buffers.
3. The high speed at which the processor operates (10.25 mips).
4. A built in sine-wave lookup table which is useful for implementing modulation

The above approach allows a complex scheme to be implemented with relative ease. As a result, the time and frequency response correlate very well with theoretical predictions.

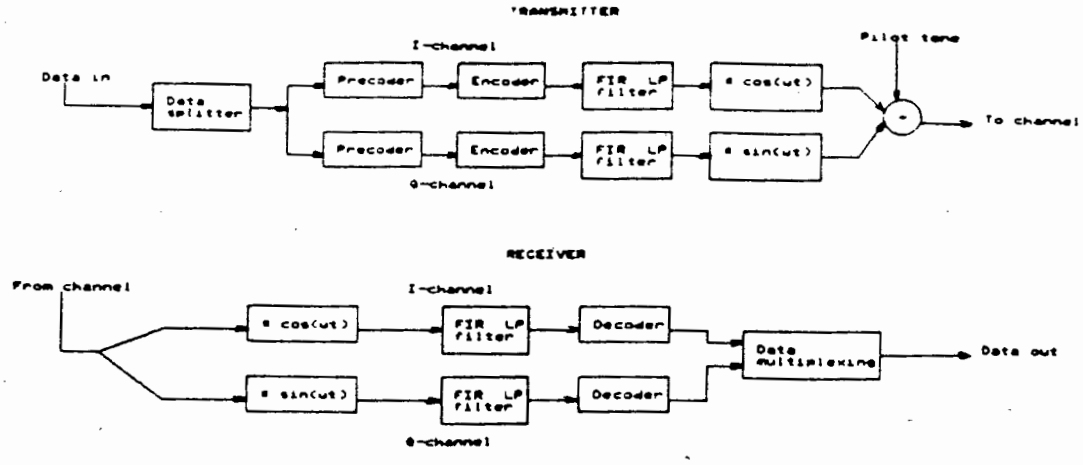


FIGURE 4 - Modem functions implemented by the DSP56001 microprocessor.

To effect a filter pair that complies with both matched filter and Nyquist criteria, square-root raised cosine FIR filters are used at the transmitter and receiver [5]. The frequency response of this filter is given by :

$$X(w) = \begin{cases} T/2\,(1 + \cos\, w/2W) & \text{for } w \quad 2W \quad (1) \\ = 0 \quad \text{elsewhere} \end{cases}$$

The FIR filter tap weights, which form the impulse response, were obtained using numerical integration. Sufficient oversampling ensures that spectral replicas can be easily rejected and that the aperture effect [6] does not degrade the signal.

3. APPLICATION A

The first modem design aims to upgrade V.22bis, which is a 2400 bit/sec, full-duplex, frequency division multiplexed modulation scheme, to 4800 bits/sec full duplex operation [2]. As in V.22bis, the carrier frequencies for the transmit and receive channels are at 1200 and 2400 Hz. Pilot tones are transmitted at 400 and 600 Hz with the 2400 and 1200 Hz channels respectively. The system assumes a usable bandwidth of 400 to 3000 Hz.

The frequency of the carriers, symbol timing and pilot tones are all directly related. This allows the carriers and symbol timing waveforms to be unambiguously derived from the pilot tones, which are recovered with phase-locked loops at the receiver. Although the 600 Hz pilot tone is on the band edge of the 1200 Hz channel, it does not interfere with the data as it inserted in such a way that its zero crossings coincide with the eyes. The 400 Hz tone is widely separated in frequency from the 2400 Hz channel and is easily filtered out at the receiver. The measured spectra of the two channels are given in Figure 5.

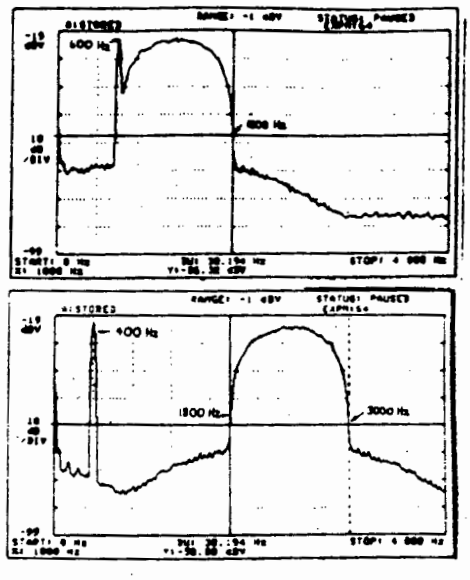


FIGURE 5 a) - 1200 Hz channel with 600 Hz pilot tone.
b) - 2400 Hz channel with 400 Hz pilot tone.
(4 % excess root raised cosine bandwidth)

4. APPLICATION B

The second modem design intends to replace a V.29-type modem, which has variable data rates of 2400, 4800 and 9600 bits/sec and operates over conditioned leased lines, with one intended to function over M.1020 voice-grade lines. Note that there is no bit rate improvement, but the reduced spectral width now conforms to M.1020 constraints. The variable data rates are achieved as follows:

Data Rate	Modulation scheme
9600	49 QPRS
4800	9 QPRS
2400	3 PRS (In-phase channel only)

In the 4800 bits per second case, a one bit word is used. The coding equations remain the same except that the mod 4 operation is changed to mod 2. A further reduction in the bit rate to 2400 bits per second is achieved by using the in-phase channel only. The time waveforms for these cases are given in Figure 6. Figure 7 shows the signal spectrum for the case of 9600 bits/sec. The carrier frequency is 1800 Hz while the pilot tone is inserted at 600 Hz. The pilot tone is employed for the same reason as in application A.

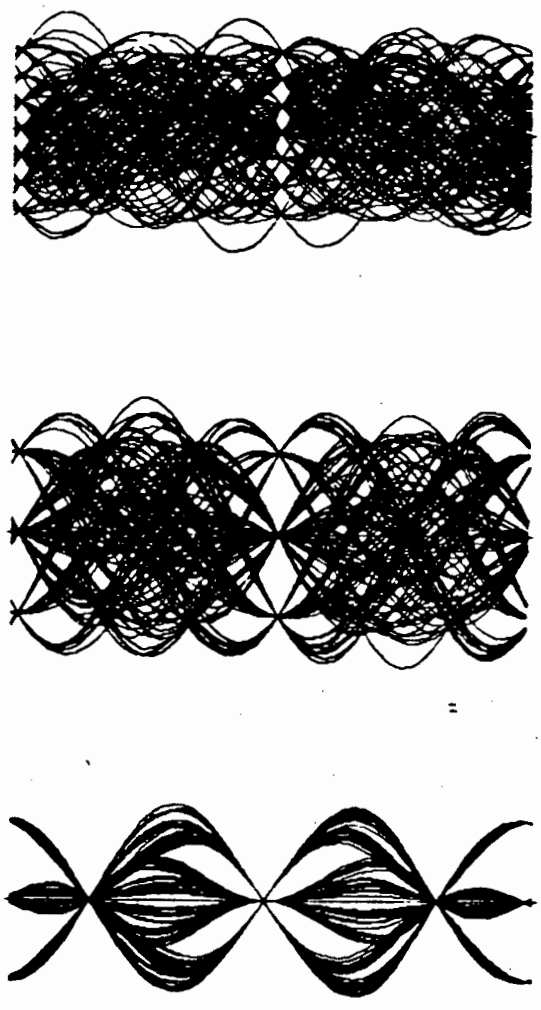


FIGURE 6 - Time waveforms of the three modulation schemes:
a) - 9600 bps
b) - 4800 bps
c) - 2400 bps.

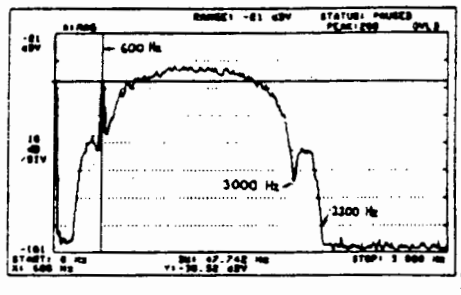


FIGURE 7 - Spectrum of the 49 QPRS waveform (9600 bps). (30 % excess root raised cosine bandwidth)

CONCLUSION

A modem design exhibiting features such as high bandwidth efficiency, low complexity and digital reliability has been described. Two specific applications for which this design is particularly well suited have been described. Future work will include comprehensive signal-to-noise versus bit error rate testing.

REFERENCES

1. Lender A., "The Duobinary Technique for High-speed Data Transmission", IEEE Trans. Commun. Electron., vol. 82, pp. 214-218, May 1963.

2. CCITT Recommendation M.1020, Department of Posts and Telecommunications.

3. Feher K., "Advanced Digital Communications, Systems and Signal Processing Techniques", Prentice-Hall Inc., New Jersey, 1987, pg. 328.

4. Motorola DSP56001 Users Guide.

5. Stremler F. G., "Introduction to Communication Systems, 2nd Edition", Addison-Wesley, Reading, Massachusetts, 1982, pg. 367-370.

6. Taub H. and Schilling D.L., "Principles of Communication Systems", McGraw-Hill, New York,1971.

APPENDIX B - TRANSMITTER PROGRAMS

TRANSMITTER HOST PROGRAM

```
'TRANSMIT.PAS'
program Transmit;
{-------------------------------------------------------------
This program initializes and activates the DSP56001 so that
it functions as a real time transmitter. The assembler
program, 'TRANSMIT.ASM' must be downloaded to the DSP56001
prior to the running of this program.
-------------------------------------------------------------}
uses
 dos, crt,dsplib2;

var
    num24                 :longint; {variable which is used to
                                     download 24-bit values to
                                     the DSP56001}
    baseaddress,timeout :word;     {adress of DSP56001 on the
                                     PC bus}
    Lookupsize            :integer; {the size of the FIR
                                     filter lookup
                                     table which is downloaded
                                     to the DSP56001}
    scale                 :longint; {scaling factor for the
                                     FIR lookup table
                                     for controlling the
                                     amplitude of the
                                     signal}


procedure Send_Lookup_table_to_DSP;
{*************************************************************
 Load lookup table from LOOKUP.DEC and write it to DSP56001
*************************************************************}
var
  infile  :text;      {file variable}
  n       :integer;   {counter}
  lookup  :real;      {stores points of the lookup table}
  num24   :longint;

begin
  writeln('Transfering lookup table from disk to DSP..');
  assign(infile,'\turbo\russ\LOOKUP.DEC');
  reset(infile);
  for n := 1 to Lookupsize do
    begin
      readln(infile,lookup);
      DSP_writeflag(baseaddress,timeout);
      num24 := round(lookup*scale);
      DSP_writelongint(baseaddress,num24);
    end;
  close(infile);
end;
```

```
procedure Send_Pilot_tones_to_DSP;
(*********************************************************)
(* Generate and send pilot tones to DSP.                 *)
(*********************************************************)
var
  n:integer;
  tone:real;

begin
  writeln('Transfering pilot tones to DSP..');

  {Transfer 600 Hz pilot tone to the DSP56001}
   for n := 0 to 63 do
    begin

      {Generate value}
      tone := sin(2*pi*n/64);

      {Scale and download generated value}
      DSP_writeflag(baseaddress,timeout);
      num24 := round(tone*200000);
      DSP_writelongint(baseaddress,num24);
    end;

  {Transfer 400 Hz pilot tone to the DSP56001}
  for n := 0 to 95 do
    begin

      {Generate value}
      tone := sin(2*pi*n/96);

      {Scale and download generated value}
      DSP_writeflag(baseaddress,timeout);
      num24 := round(tone*200000);
      DSP_writelongint(baseaddress,num24);
    end;
end;


BEGIN
  BaseAddress:=$2b0;
  scale := 100000;
  DSP_Go(BaseAddress);
  timeout:=100;
  Lookupsize := 768;        {Loads a 768 point lookup table}
  Send_Lookup_table_to_DSP;
  Send_Pilot_tones_to_DSP;
  writeln('should be working now');
  Repeat until keypressed;
END.
```

TRANSMITTER ASSEMBLER PROGRAM

```
'TRANSMIT.ASM'
;This program implements the transmitter using either of the
;carrier frequencies, which are hardware selectable.

 opt    fc,mu,s,w,mex
          org     p:$0000
          jmp     begin1
          org     p:$0008      ;set up IRQA interrupt
          jsr     irqa
          org     p:$0100


begin1

contc     equ     $1fc         ;port c control -  bits 0 and
                               ;1 used for I/O
dirc      equ     $1fc         ;port c direction word
ipren     equ     $0007        ;enable irqa ipr word

lkpsize   equ     768          ;size of lookup table
lkpmod    equ     767          ;mod of kookup table(lkpsize-
                               ;1)
cycles    equ     32           ;points per symbol
bufsize   equ     48           ;size of circular buffers
                               ;(lkpsize/cycles)
bufmod    equ     47           ;mod of circular buffers
                               ;(lkpsize/cycles-1)
lkpstart  equ     $1000        ;y:starting address for
                               ;lookup table
bufstart  equ     $00          ;x:starting address for I
                               ;circular buffer
icarryst  equ     $100         ;y:I start of built in
                               ;carrier lookup table
qcarryst  equ     $140         ;y:Q start of built in
                               ;carrier lookup table
carrymod  equ     255          ;mod of built in carrier
                               ;lookup table
offset12  equ     8            ;offset register for 1200 Hz
                               ;carrier
offset24  equ     16           ;offset register for 2400 Hz
                               ;carrier

plt12st   equ     $1600        ;x:start address of 1200 Hz
                               ;pilot tone
plt12sz   equ     64
plt12mod  equ     63           ;modulus of 1200 Hz pilot
                               ;tone
plt24st   equ     $1700        ;x:start address of 2400 Hz
                               ;pilot tone
plt24sz   equ     96           ;length of pilot tone table
plt24mod  equ     95           ;modulus of 2400 Hz pilot
                               ;tone

carrynew  equ     $fa          ;stores the new carrier
                               ;frequency
carryold  equ     $fb          ;stores the current carrier
                               ;frequency
tempout   equ     $fc          ;temporary store for output
```

```
                                    ;helps timing
inbuf    equ      $fd               ;x:Adress of data input
                                    ;buffer
itemp    equ      $fe               ;x:temporary storage location
                                    ;for I precoder
qtemp    equ      $ff               ;x:temporary storage location
                                    ;for Q precoder


;READ IN LOOKUP TABLE
         move   #lkpstart,r0        ;set up pointers
         move   #lkpmod,m0
         move   #cycles,n0

         do     #lkpsize,tablein
lkpin    btst   #0,x:$ffe9
         jcc    lkpin
         move   x:$ffeb,x0
         move   x0,x:(r0)+
tablein

;READ IN 1200 Hz PILOT TONE
         move   #plt12st,r5
         move   #plt12mod,m5
         do     #plt12sz,pl12in
in1200   btst   #0,x:$ffe9
         jcc    in1200
         move   x:$ffeb,x0
         move   x0,x:(r5)+
pl12in

;READ IN 2400 Hz PILOT TONE
         move   #plt24st,r5
         move   #plt24mod,m5
         do     #plt24sz,pl24in
in2400   btst   #0,x:$ffe9
         jcc    in2400
         move   x:$ffeb,x0
         move   x0,x:(r5)+
pl24in

;START IN 1200 HZ CARRIER MODE
         move   #0,a0
         move   a0,x:carryold
         move   a0,x:carrynew
         jsr    set1200

;SET UP CARRIER LOOKUP TABLE
         movec  #$6,omr             ;enable roms
         move   #icarryst,r6        ;set up pointers
         move   #qcarryst,r7
         move   #carrymod,m6
         move   #carrymod,m7

;INITIALIZE DATA INPUT BUFFER
         move #0,x0
         move x0,x:inbuf          ;clear the input buffer
```

```
;SET UP AND CLEAR CIRCULAR BUFFER
          move    #bufstart,r4
          move    #bufmod,m4
          move    #0,y0
          do      #bufsize,iclear
          move    y0,y:(r4)+
iclear

;INITIALIZE PRECODERS
          move    #0,x0
          move    x0,x:itemp          ;zero in I temporary
                                      ;storage
          move    x0,x:qtemp          ;zero in Q temporary
                                      ;storage

;INITIALIZE PORT C FOR DATA INPUT AND CARRIER FREQUENCY
;SETTING
         movep    #contc,x:$ffe1      ;assign I/O pins
         movep    #dirc,x:$ffe3       ;assign data direction

;SET UP AND ENABLE IRQA INTERRUPT
         movep    #ipren,x:$ffff
         move     #$00,sr


;MAIN PROGRAM LOOP
main     move     #ipren,a0           ;enable IRQA interrupt
         move     a0,x:$ffff
         jmp      main


;INTERRUPT SERVICE ROUTINE TO GENERATE ONE POINT

irqa     movep  #0,x:$ffff            ;disable interrupt and
                                      ;pending register

         ;OUTPUT PREVIOUSLY GENERATED POINT
         move   x:tempout,a1          ;get value from memory
         move   a1,y:$8400            ;output data to the
                                      ;latch
         move   a1,y:$7400            ;provide latching signal

         ;CHECK IF DATA MUST BE SAMPLED
         clr  a       x:(r1)+,x0     ;bogus move to increment
                                      ;counter
         move r1,x1                   ;get counter
         move #7,a1
         and  x1,a                    ;extract 3 msb's,
                                      ;result in a1
         move #6,b0                   ;move 6 to b0 for
                                      ;comparison
         move b0,x1
         cmp  x1,a                    ;check if 8 cycles done
         jseq bitin                   ;get bit and check
                                      ;carrier frequency
```

```
;DO CONVOLUTION
;The a and b accumulators are for the I and Q
;channels respectively
clr     a                           ;  clear I
                                    ;and Q accumulators
clr     b               y:(r4)-,y1 ;  Q channel
move    x:(r0)+n0,x0    y:(r4)-,y0 ;I channel
                                    ;and lookup table

do      #bufmod,onescan
mac     x0,y0,a    y:(r4)-,y0
mac     x0,y1,b    x:(r0)+n0,x0       y:(r4)-,y1
onescan
mac     x0,y0,a             ;do last
                            ;multiplies and increment
mac     x0,y1,b       x:(r0)+,x0  ;table start
                            ;address (bogus move)



;MULTIPLY BY CARRIERS
move    a0,x0                           ;I channel in
                                        ;x0
move    y:(r6)+n6,y0                    ;I carrier in
                                        ;y0
mpy     x0,y0,a        y:(r7)+n7,y0     ;multiply, Q
                                        ;carrier in y0
move    b0,x0                           ;Q channel in
                                        ;x0
mac     x0,y0,a        x:(r5)+,b1 ;multiply and
                                  ;add to I channel
                                  ;get pilot tone

move    b1,x1
add     x1,a                ;add pilot tone

;STORE OUTPUT FOR NEXT INTERRUPT
move    a1,x:tempout

jsr     symchk              ;check if new symbols
                            ;must be calculated

clr      a      x:$ffe5,y0  ;check if carrier
                            ;frequency must be
                            ;changed
move     #2,a1
and      y0,a           ;isolate carrier set bit
asr      a              ;bit now in position 0 of
                        ;a1
move     a1,x:carrynew  ;store updated carrier
                        ;setup
move     x:carryold,x1
move     a1,x:carryold  ;replace old carrier
cmp      x1,a           ;compare new and old setups
jsne     carrychg       ;make required changes if a
                        ;change is requested
nop

rti                     ;end of interrupt service
                        ;routine
```

```
;CHECK IF SYMBOLS MUST BE CALCULATED
symchk    move r1,x1          ;get counter
          clr  a
          move #32,a1         ;new symbols on 32nd cycle
          cmp  x1,a           ;compare
          jseq symcalc        ;calculate new symbols if
                              ;neccessary
          nop
          rts

;** DATA SAMPLING AND CARRIER CHECK SUBROUTINE***
bitin     move    x:$ffe5,y0  ;input new data
          clr     a
          move    #1,a1
          and     y0,a
          move    a1,a0       ;separate out LSB from
                              ;carrier control bit
          move    x:inbuf,b0  ;get old data
          addl    a,b
          move    #$f,a0
          move    a0,x1       ;4 in x1
          move    b0,b1       ;buffer in b1
          and     x1,b
          move    b1,x:inbuf  ;put new value back in memory
          rts


;*** SYMBOL CALCULATION AND RESET SUBROUTINE ***
symcalc   move   #00,r1
          move   x:inbuf,x1     ;get value of buffer
          move   #lkpstart,r0   ;initialize lookup table
                                ;pointers
          move   y:(r4)+,x0

      ;I CHANNEL CALCULATION
          move   #$c,a1         ;isolate I channel bits
          and    x1,a
          asr    a
          asr    a              ;a1 contains the mod 4
                                ;number
          move   #4,b0
          move   b0,y1
          add    y1,a           ;add 4

          ;SUBTRACT PREVIOUSLY PRECODED VALUE
          move    x:itemp,x0    ;get previous value from
                                ;temporary storage
          sub     x0,a          ;do subtraction

          ;TAKE MODULO-4
          move    #3,b0
          move    b0,y1
          and     y1,a
          move    a1,x:itemp    ;put precoded number into
                                ;temporary storage
```

```
            ;DO DUOBINARY ENCODING
            add     x0,a
            sub     y1,a                ;level shift by -3
            move    a1,y:(r4)+          ;input a new number into
                                        ;buffer

         ;Q CHANNEL CALCULATION
            move    #$3,a1              ;Isolate Q channel bits
            and     x1,a
            move    #4,b0
            move    b0,y1
            add     y1,a                ;add 4

            ;SUBTRACT PREVIOUSLY PRECODED VALUE
            move    x:qtemp,x0          ;get previous value from
                                        ;temporary storage
            sub     x0,a                ;do subtraction

            ;TAKE MODULO-4
            move    #3,b0
            move    b0,y1
            and     y1,a
            move    a1,x:qtemp          ;put precoded number into
                                        ;temporary storage

            ;DO DUOBINARY ENCODING
            add     x0,a
            sub     y1,a                ;level shift by -3
            move    a1,y:(r4)           ;input into buffer don't
                                        ;increment pointer
            rts

;SUBROUTINES TO CHANGE CARRIER FREQUENCIES
carrychg    clr     a
            move    x:carrynew,x1       ;store new carrier setting
            move    #1,a1               ;check which carrier is
                                        ;requested
            cmp     x1,a
            jsne    set1200
            nop
            jseq    set2400
            nop
            jsr     reset
            nop
            rts

set1200     move    #plt12st,r5         ;set low band carrier and
                                        ;pilot tones
            move    #plt12mod,m5
            move    #offset12,n6
            move    #offset12,n7
            rts

set2400     move    #plt24st,r5         ;set high band carrier
                                        ;and pilot tones
            move    #plt24mod,m5
            move    #offset24,n6
            move    #offset24,n7
            rts
```

```
reset   move    #lkpstart,r0        ;reset lookup table
                                    ;pointer
        move    #0,r1               ;reset timing counter
        move    #icarryst,r6        ;reset carrier start
                                    ;addresses
        move    #qcarryst,r7

        rts
        end
```

APPENDIX C - RECEIVER PROGRAMS

RECEIVER HOST PROGRAM

```
'RECEIVE.PAS'
program receive;
(*************************************************************
This program activates and initializes the DSP56001 so that
it operates as a 49QPRS transmitter. The transmitter does
not run in real time but uses data that has been written to
disk. The recovered bitstream is also written back to disk.
The program is controlled by the host interface interrupt.
This particular version of the program uses a sampling rate
of 16 samples per symbol (this wold correspond to 19.2 kHz
in a real version), although it could be adjusted to use a
different sampling frequency.
*************************************************************}
uses
 dos, crt,dsplib2;

type
  InArray = Array[1..4096] of real;

var
  num24        :longint;   {used for input and output of 24-
                            bit numbers}
  baseaddress :word;       {address of DSP56001 on PC bus}
  timeout      :word;      {used to prevent program jamming}
  Lookupsize   :integer;   {size of the FIR filter lookup
                            table}
  points       :longint;   {number of points sampled off
                            disk}
  Data         :InArray;   {stores the input data}
  symbols      :integer;   {number of uotput symbols being
                            decoded}
  cycle        :integer;   {number of sample points per
                            symbol}


procedure Send_Lookup_table_to_DSP;
{*************************************************************
Loads the lookup table from disk and writes it to the
DSP56001. The lookup table must be stored in the file
'LOOKUP.DEC'. and write it to DSP.
*************************************************************}
var
  infile:text;
  n:integer;
  lookup:real;
```

```
begin
  writeln('Transfering lookup table from disk to DSP..');
  assign(infile,'LOOKUP.DEC');
  reset(infile);
  for n := 1 to Lookupsize do
    begin
      readln(infile,lookup);
      DSP_writeflag(baseaddress,timeout);
      num24 := round(lookup*200000);
      DSP_writelongint(baseaddress,num24);
    end;
  close(infile);
end;


procedure Read_data_from_dspout;
{*************************************************************
Reads the input data from a file on disk which must be
called 'DSPOUT.DEC'. This data is stored in the array
Data[n]. In the case of this program, every second
transmitted point is sampled by the receiver and the dummy
variable, temp, is used to skip over every other point.
*************************************************************}

var
  n      :integer;   {counter}
  infile :text;      {file variable}
  temp   :real;      {dummy variable}
  total  :integer;   {total amount of points to be input}

begin
  writeln('Reading data from DSPOUT.DEC..'); {display
                                    message on screen}
  assign(infile,'dspout.DEC');
  reset(infile);
  total := symbols*cycle*2;
  for n := 1 to total do
    begin
      readln(infile,Data[n]);
      readln(infile,junk);
    end;
  close(infile);
end;
```

```
procedure Transmit_and_receive_data;
{*************************************************************
This subroutine down loads the input samples to the DSP56001
and receives the decoded bits (four at a time), from the
chip. The decoded bits are received on every 16th pass. This
corresponds to a sampling rate of 16 points per symbol. The
decoded bits are written to the file 'BINOUT.DEC'.
*************************************************************}
var
  infile,outfile   :text;      {file variables}
  n,m              :integer;   {counters}
  input24,output24 :longint;   {24-bit variables used for
                                input and output}

begin
  writeln('Transmitting and receiving data from DSP..');
  assign(outfile,'BINOUT.DEC');
  rewrite(outfile);
  m := 0;
  for n := 1 to (symbols*cycle) do   {symbols * cycle is the
                                      total amount of
                                      points}
    begin
      input24 := round(Data[n]);                {download one
                                                 input point}
      DSP_Writeflag(baseaddress,timeout);
      DSP_writelongint(baseaddress,input24);

      if (m mod 16) = 0 then            {do this routine on
                                         every 16th pass}
        begin

          DSP_Readflag(baseaddress,timeout); {recieve I
                                              channel bits}
          DSP_Readlongint(baseaddress,output24);
          writeln(outfile,'I1: ',output24);
          DSP_Readflag(baseaddress,timeout);
          DSP_Readlongint(baseaddress,output24);
          writeln(outfile,'I2: ',output24);

          DSP_Readflag(baseaddress,timeout); {receive Q
                                              channel bits}
          DSP_Readlongint(baseaddress,output24);
          writeln(outfile,'Q1: ',output24);
          DSP_Readflag(baseaddress,timeout);
          DSP_Readlongint(baseaddress,output24);
          writeln(outfile,'Q2: ',output24);

        end;

      m := m + 1;  {increment counter}
    end;
    close(outfile);
end;
```

```
BEGIN
  BaseAddress:=$2b0;              {activate DSP56001}
  DSP_Go(BaseAddress);
  timeout:=100;

  lookupsize := 384;              {set up lookup table
                                   parameters}
  symbols := 64;
  cycle   := 16;

  Send_Lookup_table_to_DSP;       {download lookup table}

  Read_data_from_dspout;          {read input samples from
                                   disk}

  Transmit_and_receive_data;      {transmit input samples and
                                   receive decoded bits}
END.
```

RECEIVER ASSEMBLER PROGRAM

```
'RECEIVE.ASM'
;This program implements the receiver software. 49 QPRS data
;is downloaded via the host interface. This data is then
;multiplied by the I and Q carriers. This is followed by FIR
;lowpass filtering which results in the I and Q baseband
;waveforms. These waveforms are threshold detected at the
;symbol timing instants and decoded according to the
;duobinary rules. The decoded data is then uploaded via the
;host interface. The program can be set to demodulate either
;the low or high band

          opt     fc,mu,s,w,mex
          org     p:$0000
          jmp     begin1

          org     p:$0100

begin1

thresh1   equ       1500000       ;set threshold levels
thresh2   equ       2100000
thresh3   equ       2700000
thresh4   equ       3300000
thresh5   equ       3900000
thresh6   equ       4500000
levelsh   equ       3000000       ;to level shift the input of
                                  ;the threhold detector
points    equ       1024          ;number of points sampled
cycle     equ       16            ;points per symbol
lkpsize   equ       384           ;size of lookup table
lkpmod    equ       383           ;mod of kookup table(lkpsize-
                                  ;1)
lkpstart  equ       $1000         ;y:starting address for
                                  ;lookup table
bufstart  equ       $1000         ;x:starting address for
                                  ;circular buffer
bufsize   equ       768           ;set size of circular buffer
bufmod    equ       767
icarryst  equ       $100          ;y:I start of built in
                                  ;carrier lookup table
qcarryst  equ       $140          ;y:Q start of built in
                                  ;carrier lookup table
carrymod  equ       255           ;mod of built in carrier
                                  ;lookup table
offset12  equ       16            ;offset register for 1200 Hz
                                  ;carrier
offset24  equ       32            ;offset register for 2400 Hz
                                  ;carrier
itempout  equ       $0            ;temporary storage locations
                                  ;for decoded data
qtempout  equ       $1
```

```
;READ IN LOOKUP TABLE
          move    #lkpstart,r0        ;set up pointers
          move    #lkpmod,m0
          do      #lkpsize,tablein
lkpin     btst    #0,x:$ffe9          ;read in lookup table
          jcc     lkpin
          move    x:$ffeb,x0
          move    x0,x:(r0)+
tablein

;SET UP AND CLEAR CIRCULAR BUFFER
          move    #bufstart,r4         ;set up pointers
          move    #bufmod,m4
          move    #0,b0
          do      #bufsize,clean       ;clear buffer
          move    b0,y:(r4)+
clean

;SET UP CARRIERS
          movec   #$6,omr              ;enable roms
          move    #carrymod,m6
          move    #carrymod,m7
          jsr     resetcar
          jsr     set1200              ;set to demodulate low
band

;SET UP COUNTER AND TEMPORARY STORAGE
          move    #0,r1                ;set counter
          move    #15,m1               ;counter must be reset
                                       ;on every
                                       ;16th increment
          move    #0,a0
          move    a0,x:itempout        ;clear temporary output
                                       ;storage locations
          move    a0,x:qtempout


;MAIN PROGRAM LOOP
          do      #points,again

datain    btst    #0,x:$ffe9           ;read in one data point
          jcc     datain
          move    x:$ffeb,b0

         ;MULTIPLY BY CARRIERS
          move    b0,x0               ;Data in x0
          move    y:(r6)+n6,y0        ;I carrier in y0
          mpy     x0,y0,a             ;multiply
          move    a1,y:(r4)-          ;put data in buffer

          move    y:(r7)+n7,y0        ;Q carrier in y0
          mpy     x0,y0,a             ;multiply
          move    a1,y:(r4)           ;put data in buffer
```

```
            ;DO FIR FILTER CONVOLUTION
            ;accumulators a and b are used for the I and Q
            ;channels respectively
            clr     a                       ;clear I accumulator
            clr     b

            move    x:(r0)+,x0      y:(r4)+,y0  ;I channel
                                                ;and lookup table
            move    y:(r4)+,y1
            do      #bufmod,onescan
            mac     x0,y0,a         y:(r4)+,y0
            mac     x0,y1,b         x:(r0)+,x0      y:(r4)+,y1
onescan
            mac     x0,y0,a             ;do last multiplies
            mac     x0,y1,b
            move    a1,x:itempout       ;store results
            move    b1,x:qtempout

            ;CHECK IF OUTPUT MUST BE CALCULATED
            clr     a
            move    r1,y0
            move    #0,a1               ;output on every 16th
                                        ;pass
            cmp     y0,a                ;(r1 is 0 on every 16th
                                        ;pass)
            jseq    iqout

            move    x:(r1)+,y0          ;bogus move to increment
                                        ;counter

            move    y:(r4)-,y0          ;bogus move to decrement
                                        ;circular
                                        ;buffer pointer
                                        ;again


;PROCESS THE I AND Q CHANNELS
iqout       move    x:itempout,x1       ;get I channel baseband
                                        ;point
            jsr     decode              ;decode it
            move    x:qtempout,x1       ;get Q channel baseband
                                        ;point
            jsr     decode              ;decode it
            nop
            rts


;OUTPUT ROUTINE
;value being decoded is in x1
decode      clr     a
            move    #levelsh,a1          ;level shift
            add     x1,a                 ;result in a
            jsr     thrshold             ;threshold detect
```

```
          ;UPLOAD TWO DECODED BITS VIA THE HOST INTERFACE
output1   btst   #1,x:$ffe9
          jcc    output1
          move   b1,x:$ffeb          ;output 1 bit
output2   btst   #1,x:$ffe9
          jcc    output2
          move   b0,x:$ffeb          ;output 1 bit
          rts

;THRESHOLDING ROUTINE
;Data must be in a1 (a0 cleared). O/P is put in b1 and b0.

thrshold  move  #0,b1                ;set output to that
                                     ;corresponding to
          move  #0,b0                ;lowest level
          move  #thresh1,x0          ;get threshold
          cmp   x0,a                 ;compare
          jlt   second               ;If it was lower than
                                     ;the threshold
          move  #0,b1                ;do not change the
                                     ;output, else set
          move  #1,b0                ;output to correspond to
                                     ;next level
second    move  #thresh2,x0          ;repeat as above
          cmp   x0,a
          jlt   third
          move  #1,b1
          move  #0,b0
third     move  #thresh3,x0          ;repeat as above
          cmp   x0,a
          jlt   fourth
          move  #1,b1
          move  #1,b0
fourth    move  #thresh4,x0          ;repeat as above
          cmp   x0,a
          jlt   fifth
          move  #0,b1
          move  #0,b0
fifth     move  #thresh5,x0          ;repeat as above
          cmp   x0,a
          jlt   sixth
          move  #0,b1
          move  #1,b0
sixth     move  #thresh6,x0          ;repeat as above
          cmp   x0,a
          jlt   seventh
          move  #1,b1
          move  #0,b0
seventh   nop
          nop
          rts


;RESET CARRIER TABLES
resetcar  move  #icarryst,r6         ;reset pointers for
                                     ;carriers
          move  #qcarryst,r7
          rts
```

```
;SET UP 1200 Hz CARRIER MODE
set1200  move    #offset12,n6
         move    #offset12,n7
         rts

;SET UP 2400 Hz CARRIER MODE
set2400  move    #offset24,n6
         move    #offset24,n7
         rts
         end
```

APPENDIX D - CROSSTALK SIMULATION PROGRAM

```
'SIMULATE.PAS'
PROGRAM SIMULATE;
{-------------------------------------------------------------
This program simulates crosstalk caused by the slight
overlap between the low and high band channels. This program
allows the effects of varying the FIR filter parameters on
crosstalk to be investigated. The effects of varying degrees
of inherent isolation, which ae determined by the hybrid
transformer and the transmission loss, are also simulated.
The program generates both the low band and the high band
channels. The high band is then scaled and added to the low
band after which the low band is demodulated. An eye diagram
of the demodulated signal is plotted on the screen and is
printed once the simulation is complete. By observing the
eye openings, the performance of the system can be evauated.
-------------------------------------------------------------}

uses crt,dos,graph,Gkernel,Gdriver,printer;

type
  registers = array[1..64] of integer;  {used to store
                                         circuular buffers}
  BigArray = array[1..2048] of real;    {used to store the
                                         FIR filter lookup
                                         table}

var
  ITxbuf1200,QTxbuf1200       :registers;  {Circular buffer
                                            for the low
                                            band}
  ITxbuf2400,QTxbuf2400       :registers;  {Circular buffer
                                            for the high
                                            band}
  TXout1200                   :real;       {The low band
                                            signal}
  RXoutput                    :real;       {the demodulated
                                            signal}
  Iout1200,Qout1200           :real;       {Baseband signals
                                            for the I and Q
                                            low band
                                            channels}
  Irandom1200,Qrandom1200     :integer;    {used as delays
                                            for low band
                                            duobinary
                                            encoding}
  TXout2400                   :real;       {The high band
                                            signal}
  Iout2400,Qout2400           :real;       {Baseband signals
                                            for the I and Q
                                            high band
                                            channels}
  Irandom2400,Qrandom2400     :integer;    {used as delays
                                            for high band
                                            duobinary
                                            encoding}
  TXtemp1200,Txtemp2400,RXtemp:real;       {for joining
                                            points when
                                            plotting
```

```
                                                    eye diagrams}
  TxBufpoint1200,RxBufPoint   :integer;     {pointers to the
                                             circular
                                             buffers}
  TxBufPoint2400              :integer;
  TxLookuppoint1200           :integer;     {Pointers to the
                                             transmitter
                                             lookup table}
  TxLookupPoint2400           :integer;
  Lookup                      :BigArray;    {Stores the FIR
                                             filter lookup
                                             table values}
  RxBuf                       :BigArray;    {Circular buffer
                                             holding the
                                             receiver input
                                             data}
  n,m                         :integer;     {general purpose
                                             counters}
  numPoints                   :longint;     {number of points
                                             generated}
  Crosstalk                   :real;        {scaling factor
                                             for the high
                                             band}
  Hpoints                     :integer;     {controls
                                             horizontal sweep
                                             of eye diagram
                                             plot}
  carrier                     :real;        {stores the
                                             immediate
                                             value of the
                                             carrier}
  length                      :integer;     {The length of
                                             the lookup
                                             table}
  cycle                       :integer;     {points per
                                             symbol period}
  span                        :integer;     {width of lookup
                                             table in symbol
                                             periods}
  Filename                    :string[20]; {filename of
                                             lookup table}

Procedure Load_lookup_table_from_disk;
{-------------------------------------------------------------
Loads the lookup table from a file. The first three values
read are the variables length, cycle and span.
-------------------------------------------------------------}
var
  n:integer;          {counter}
  infile:text;        {file variable}
```

```
begin
  assign(infile,Filename);
  reset(infile);
  readln(infile,length);
  readln(infile,cycle);
  readln(infile,span);
  for n := 1 to length do Readln(infile,Lookup[n]);
  close(infile);
end;


procedure Make_hardcopy;
{-------------------------------------------------------
Plots the eye diagram on a printer.
-------------------------------------------------------}
begin
  HardCopy(false,1);
end;


Function DuobinaryNumber(var RandomNumber1:integer):integer;
{-------------------------------------------------------
This function returns a 7-level duobinary sequence of
numbers, one at a time, for each call (-3,-2,-1,0,+1,+2,+3).
The Turbo Pascal random number generator is used. There is
no precoding, as we are only interested in eye diagrams, and
so error propagation does not play any role. The variables
'RandomNumber1' and 'RandomNumber2' are used to provide the
delay for duobinary encoding.
-------------------------------------------------------}
var
  RandomNumber2   :integer;

begin
  RandomNumber2 := random(4);
  DuobinaryNumber := RandomNumber1 + RandomNumber2 - 3;
  RandomNumber1 := RandomNumber2;
end;


procedure Initialize;
{-------------------------------------------------------
This procedure performs the neccessary initialization of the
program variables and arrays.
-------------------------------------------------------}

begin
  clrscr;
  Load_lookup_table_from_disk;
  Randseed  := 8;
  Hpoints   := 0;
  Rxtemp    := 175;    {This allows the first point to be
                        joined correctly when plotting the
                        eye diagram}
  NumPoints := 0;
```

```
  { Initialize low band variables }
  Irandom1200 := 0;
  Qrandom1200 := 0;
  TxBufPoint1200 := 1;
  RxBufPoint := 1;
  TxLookupPoint1200 := 1;
  for n := 1 to span do
    begin
      ITxBuf1200[n] := 0;
      QTxBuf1200[n] := 0;
    end;

  { Initialize high band variables }
  Irandom2400 := 0;
  Qrandom2400 := 0;
  TxBufPoint2400 := 1;
  TxLookupPoint2400 := 1;
  for n := 1 to span do
    begin
      ITxBuf2400[n] := 0;
      QTxBuf2400[n] := 0;
    end;
  for n := 1 to length do RxBuf[n] := 0;
end;


procedure Display(TXout:real;var
TxTemp:real;Hpoints:longint; Vscale,Voffset:real);
{-------------------------------------------------------
Plots the eye diagram on the screen. Vscale and Voffset are
the vertical scaling factor and the vertical offset
respectively. TxTemp is a delay which is used to join
points.
-------------------------------------------------------}

var
  Xval1,Xval2,Yval :integer; {X,Y values to be plotted}

begin
  Yval := round(TXout*Vscale+Voffset);
  Xval1 := trunc(Hpoints*320/cycle + 40);
  Xval2 := trunc((Hpoints+1)*320/cycle + 40);
  DrawLine(Xval1,TXtemp,Xval2,Yval);
  TXtemp := Yval;
end;
```

```
procedure calculate_transmitter_output(var TXout:real;
  var BufPoint,LookupPoint:integer;var Ibuf,Qbuf:registers;
  var Irandom,Qrandom:integer;ModulationFreq:real);
{-------------------------------------------------------
Calculate one transmitter output point. The value of
'ModulationFreq' determines the modulation frequency used.
This procedure is used to generate both the high band and
low band channels.
-------------------------------------------------------}
var
  Iout,Qout:real;  {The baseband I and Q waveforms}
  b:integer;       {counter}

{ Calculate baseband I channel using the FIR filter}
begin
  Iout := 0;
    for b := 0 to (span-1) do
      begin
        Iout := Iout + IBuf[(BufPoint - b + span) mod span +
                1] * lookup[LookupPoint + (b * cycle) mod
                (length + 1)];
      end;

{ Calculate baseband Q channel using the FIR filter}
  Qout := 0;
    for b := 0 to (span-1) do
      begin
        Qout := Qout + QBuf[(BufPoint - b + span) mod span +
                1] * lookup[LookupPoint + (b * cycle) mod
                (length + 1)];
      end;

{ Calculate the value of the carrier }
  carrier := pi * (LookupPoint - 1) / cycle;

{ Modulate the I and Q channels onto I and Q carriers and
add }
  Txout := Iout * cos(carrier * modulationFreq)
             + Qout * sin(carrier * modulationFreq);

{ Update pointers to the lookup table and circular buffers }
    LookupPoint := (LookupPoint mod cycle) + 1;
    If LookupPoint = 1 then
      begin
        BufPoint := (BufPoint mod (span)) + 1;
        IBuf[BufPoint] := DuobinaryNumber(Irandom);
        QBuf[BufPoint] := DuobinaryNumber(Qrandom);
      end;
end;
```

```
Procedure Calculate_receiver_output(Txout1200:real;var
RxOutput:real;ModulationFreq:real);
{-------------------------------------------------------
This procedure calculates one receiver output point. The
value of 'ModulationFreq' determines which channel is
demodulated.
-------------------------------------------------------}
Var
  n:integer;

begin
  { Multiply by the carrier }
  Txout1200 := Txout1200 * cos(carrier * ModulationFreq);
  RxOutput := 0;

  { Enter value into the receiver buffer }
  Rxbuf[RxBufPoint] := Txout1200;
  RxBufPoint := RxBufPoint mod length + 1;

  { Do FIR filtering operation }
  for n := 1 to length do
    RxOutput := RxOutput + Lookup[n]
                * RxBuf[(RxBufPoint - n  + length) mod
length + 1];
end;

Procedure HammingWindow;
{-------------------------------------------------------
This procedure applies a Hamming window to the FIR filter
lookup table
-------------------------------------------------------}
var
  n:integer;   {counter}

begin
  for n := 0 to (length-1) do
    Lookup[n+1] := Lookup[n+1] * (0.54 - 0.46 *
                    cos(2*pi*n/(length-1)));
end;


{*********************************************************
-------------------- MAIN PROGRAM -----------------------
*********************************************************}

begin
  { Display message on printer }
  writeln(lst,'INTERCHANNEL CROSSTALK SIMULATION');
  Write('Enter filename of lookup table: ');
  Readln(filename);
  Write('Enter crosstalk factor: ');
  Readln(Crosstalk);

    Initialize;
    InitGraphic;
    Drawborder;

    { Apply the Hamming window }
    HammingWindow;
```

```
repeat
 { Calculate low band signal }
 calculate_transmitter_output(TXout1200,TxBufPoint1200,
 TxLooupPoint1200,Txbuf1200,QTxbuf1200,Irandom1200,
 Qrandom1200,2);

  { Calculate high band signal }

 calculate_transmitter_output(TXout2400,TxBufPoint2400,
 TxLookupPoint2400,Txbuf2400,QTxbuf2400,Irandom2400,
 Qrandom2400,4);

  { Scale high band and add to the low band }
  TxOut1200 := TxOut1200 + Crosstalk * TxOut2400;

  { Demodulate the low band }
  Calculate_receiver_output(Txout1200,RxOutput,2);
  Display(RxOutput,RxTemp,Hpoints,8,175);

  { Update counters }
  Numpoints := numpoints + 1;
  Hpoints := (Hpoints + 1) mod (cycle*2); {superimpose 2
                                           bit periods
                                           at a time}
until keypressed or (numpoints = 10000);

{ Write information to the printer and print the eye
 diagram }
writeln(lst,'HAMMING WINDOW');
writeln(lst,'Crosstalk: ',Crosstalk,
'   Width: ',span,' symbols','  file: ',Filename);
    Make_Hardcopy;
LeaveGraphic;
end.
```

APPENDIX E - FIR FILTER LOOKUP TABLE GENERATION PROGRAMS

NUMERICAL INTEGRATION PROGRAM

```
'SIMPSON.PAS'
program Simpson;
{-------------------------------------------------------------
This program is used to generate square-root raised-cosine
implulse responses to be used in the FIR filters. In order
to do this the real inverse Fourier transform of the
frequency response must be found. The formula to implemented
is:

        ⌠+T/2
        |  cos(wt) √(Raised cosine frequency response) dw
        ⌡-T/2

The raised-cosine response was given in Chapter 2.
Unfortunately, the above formula requires an integral to
which no closed-form solution could be found. This program
uses a numerical integration rule, called Simpson's rule, to
perform the integral. In Simpson's rule, the curve is
divided into successive pieces using a step size h. Sections
of width 2h are then approximated by a parabola through its
ends and midpoint. The areas under the parabolic arcs are
then added to give Simpson's rule. Simpson's rule states
that the area under a curve = h/3 * [y(0) + 4y(1) + 2y(2) +
4y(3) + 2y(4) + .... + 2y(n-2) + 4y(n-1) + yn]

Since the impulse response will be a duoble-sided even
function, the program only generates half of the impulse
response. This saves execution time. A separate program
'WRAP.EXE' is then used to make the impulse response double
sided. This program can be set up to generate any amount of
lookup tables and can then be left running until it has
completed the task. The various tables will be stored in
files specified by the user.
-------------------------------------------------------------}
uses
  crt,dos;

type
  point = array[0..1024] of real;

var
  n,m,i,j:           longint; {general purpose counters}
  twofour,zeroone:   longint; {These are used to generate the
                               2,4,2,4,2... sequence in the
                               Simpson's rule formula}
  Firstval,Lastval: real;     {The first and last points are
                               calculated separately}
  sum               :real;    {accumulator for the summation}
  t                 :real;    {counts out each time interval}
  H                 :real;    {step size}
  Y                 :real;    {stores the frequency response
                               points}
  freq              :real;    {counts the frequency
                               intervals}
  points            :point;   {stores the impulse response}
  outfile           :text;    {file variable}
```

```
procedure
Do_1int(lookupsize,symbols,cycle,numpoints:integer;alpha:
real);
{-------------------------------------------------------
This procedure does one integral over the entire frequency
response in order to generate one point of the impulse
response.
-------------------------------------------------------}
begin
  zeroone := 0;
  sum := 0;
  For n := 1 to numpoints-2 do
    begin
      zeroone := (zeroone + 1) mod 2;
      twofour := zeroone*2 + 2;       {sequence will count
                                        as 4,2,4,2...}

      {Implement square-root-raised cosine formula}
      if abs(freq) <= (1-alpha) then Y := sqrt(0.5) *
                          cos(freq * t)
      else if ((1-alpha) < abs(freq)) and ((1+alpha) >
                                abs(freq)) then
        Y := sqrt(0.25 * (1-sin(pi * (abs(freq)-
               1)/2/alpha))) * cos(freq * t)
      else Y := 0;

      {Do accumulation and increment frequency}
      sum := sum + Y*twofour;
      Freq := Freq + H;
    end;

    {multiply by H/3 factor}
    sum := sum * H/3 ;
end;


Procedure
Calculate_lookup_table(LookupSize,symbols,cycle,numpoints:
integer;alpha:real;Filename:string);
{-------------------------------------------------------------
This procedure will calculate a single lookup table using
the variables which are passed in the calling statement.
-------------------------------------------------------------}

begin;
  clrscr;

  {Initialize the array}
  for m := 0 to 1024 do points[m] := 0;

  assign(outfile,Filename);
  rewrite(outfile);
  writeln('Size of half of lookup table: ',LookupSize);
  writeln('Width in symbols: ',symbols);
  writeln('Points per symbol: ',cycle);
  Writeln('Number of integration points: ',numpoints);
  writeln('Filter alpha (0-1): ',alpha);
```

```
  {Set up first and last values of the double-sided impulse
   response}
  Firstval := -2;
  Lastval := 2;

  t := 0;

  {Calculate H}
  H := (lastval-firstval)/(numpoints-1);

  {Do calculation}
  For m := 0 to lookupsize do
    begin
      GOTOXY(5,10);
      writeln('Calculating point number: ',m);
      t := m/cycle * pi;
      Freq :=Firstval + H;
      Do_1int(Lookupsize,symbols,cycle,numpoints,alpha);
      points[lookupsize-m] := sum;
    end;

    {Write the result to a file. The first three numbers
     will be the variables lookupsize, cycle and symbols}
    writeln(outfile,lookupsize);
    writeln(outfile,cycle);
    writeln(outfile,symbols);
   for m := 0 to lookupsize do
    writeln(outfile,points[m]);
  close(outfile);
end;


{*********************************************************
                      MAIN PROGRAM

The main program allows a number of lookup tables to
calculated sequentialy.A few examples are shown. The
procedure Calculate_lookup_table(LookupSize,
symbols,cycle,numpoints,alpha,Filename) is called. The
variables have the following meanings:

   Lookupsize: The amount of points in the single-sided
               lookup table
   symbols:    The width in symbol periods of the single-
               sided lookup table
   cycle:      Points per symbol width
   numpoints:  Number of integration points used per
               integral
   alpha:      Excess bandwidth
   Filename:   The file in which the result is stored

*********************************************************}

BEGIN
  Calculate_lookup_table(128,8,16,300,0.2,'LOOKUP1.PAS');
  Calculate_lookup_table(128,8,16,300,0.3,'LOOKUP2.PAS');
  Calculate_lookup_table(256,16,16,300,0,'LOOKUP3.PAS');
  Calculate_lookup_table(256,16,16,300,0.05,'LOOKUP4.PAS');
END.
```

PROGRAM TO GENERATE DOUBLE-SIDED LOOKUP TABLE

```
'WRAP.PAS'
program wrap;
{-------------------------------------------------------
This program converts a single-sided lookup table generated
by 'SIMPSON.PAS' into a double-sided lookup table. The
program assumes that the input file has a *.PAS extention
and will write a new file with a '*.DEC' extention.
-------------------------------------------------------}
uses
  dos,crt;

type
  bigarray = array[1..4096] of real;

var
  point             :bigarray; {stores the lookup table}
  infile,outfile    :text;     {file variables}
  numpoints         :integer;  {number of points}
  filestring1:string[12];      {filname with PAS extension}
  filestring2:string[12];      {filname with DEC extension}
  n:integer;

BEGIN
  write('Enter filename: ');
  read(filestring1);
  filestring2 := filestring1 + '.DEC';
  filestring1 := filestring1 + '.PAS';

  assign(infile,filestring1);
  assign(outfile,filestring2);
  reset(infile);
  rewrite(outfile);

  readln(infile,numpoints);
  readln(infile);
  readln(infile);

  for n := 1 to (numpoints+1) do readln(infile,point[n]);
  for n := 1 to (numpoints+1) do writeln(outfile,point[n]);
  for n := numpoints downto 2 do writeln(outfile,point[n]);

  close(infile);
  close(outfile);
END.
```

APPENDIX F - PROGRAM FOR VIEWING 49 QPRS DATA STORED IN A FILE

```
'EYE.PAS'
program eye;
{-------------------------------------------------------------
This program was written to allow a file of 49 QPRS samples
to be displayed on the screen in the form of an eye diagram.
It was used extensively in the development of both the
transmitter and receiver software.
-------------------------------------------------------------}
uses
  dos,crt,gdriver,gkernel;

type
  loadfile=array[1..4096] of real;

var
  point              :loadfile;   {stores the points}
  filestring         :string[12]; {filename of the file to be
                                   plotted}
  filesize           :integer;    {number of points to be
                                   plotted}
  points_per_symbol  :integer;    {number of points per
                                   symbol period}

procedure LoadPoints;
{-------------------------------------------------------------
Loads points off the disk from file. Assumes *.DEC
extension.
-------------------------------------------------------------}
var
  infile             :text;     {file variable}

begin
  FileString := FileString + '.DEC';
  assign(infile,FileString);
  reset(infile);
  writeln('Loading points from disk..');

  filesize := 0;
  while not Eof(infile) do
    begin
      filesize := filesize + 1;   {automatically records
                                   number of points}
      readln(infile,point[filesize]);
    end;
  close(infile);
end;
```

```
procedure Plot_eye_diagram;
{-------------------------------------------------------------
This procedure plots the eye diagram.
-------------------------------------------------------------}
var
  pointnumber :longint; {counter which refers to which point
                         is being plotted}
  wrapnumber  :longint; {counter which wraps curve onto
                         itself for eye diagram}
  Hscale      :longint; {horizontal scaling factor}
  x1,x2       :longint; {x-coordinates of horizontal lines
                         to be drawn}
  quit,go_on  :boolean; {to get out of loop}
  Voffset     :integer; {vertical offset of the eye diagram}
  Vscale      :real   ; {vertical scaling factor}


begin
  initgraphic;
  quit := false;
  Voffset := 170;
  Vscale  := 40;
  repeat
    clearscreen;
    Hscale := round(320 / points_per_symbol);
    DrawStraight(1,100,Voffset);
    for pointnumber := 1 to (Filesize-1) do
      begin
        wrapnumber := (pointnumber-1) mod round(2 *
                          points_per_symbol);
        x1 :=round(wrapnumber*Hscale);
        x2 :=round((wrapnumber+1)*Hscale);
        drawstraight(x1,x2,round(point[pointnumber]*Vscale +
                     Voffset));
      end;
```

```
      {The following section of code allows a number of
      options to be selected}
      Repeat
        go_on := false;
        case upcase(ReadKey) of
             #13:begin          {Quit if ENTER is pressed}
                   Quit := true;
                   go_on := true;
                 end;
             'B':begin
                   Vscale := Vscale * 3;  {'B' to scale up}
                   go_on := true;
                 end;
             'S':begin
                   Vscale := Vscale / 3;  {'S' to scale down}
                   go_on := true;
                 end;
             #32:go_on := true;
             'H':HardCopy(false,1);   {'H' to print the eye
                                       diagram}
        end;
      until Go_on = true;
  Until Quit = true;
  LeaveGraphic;
end;

BEGIN
  write('Enter filename: ');
  readln(FileString);
  write('Enter points per symbol: ');
  readln(points_per_symbol);
  LoadPoints;
  Plot_eye_Diagram;
END.
```