DESIGN, ANALYSIS, AND IMPLEMENTATION OF A DSP-BASED MODEM FOR CODE-PHASE-SHIFT KEYING

by

ROBERT G. LINK

Ph.D.(Physics), The University of British Columbia, 1989

A THESIS DRAFT SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF ELECTRICAL ENGINEERING

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical Engineering

The University of British Columbia
Vancouver, Canada

Date Oct. 15, 1996

Abstract

This thesis presents the design, analysis, and implementation of a radio-frequency modem which employs the Code-Phase-Shift Keying (CPSK) method of Direct-Sequence Spread-Spectrum (DS-SS) signaling. A correlation receiver is designed for a DSP based implementation. The received RF signal is brought to a digitally represented base-band signal by product demodulation at the nominal RF frequency, followed by low-pass filtering and dual-channel 8 bit sampling at 2 MSPS. Base-band signal processing is done on a quad TI TMS320C40 general purpose DSP. The SS spreading factor is user selectable from a range of 7 to 127; with corresponding data rates in the range 100 to 9.4 KBPS.

The herein designed tracking and acquisition algorithms are adapted from those of conventional DS-SS systems; while the carrier-phase tracking problem is solved by a new method hereby called Phase-Invariant-Reception (PIR).

Extensive bit-error-rate (BER) measurements have been made in Additive White Gaussian Noise (AWGN) and in the presence of single tone interference. All measured BER vs SNR or BER vs JSR curves are compared to those of an ideal optimal receiver. The implementation loss, with respect to the optimal receiver, for single-channel data flow in AWGN is approximately 1.8 dB in power efficiency when the stationary receiver and transmitter use local oscillators of the same nominal frequency. The additional BER power efficiency loss as a function of the difference in frequency of the latter two oscillators, is obtained exactly analytically and is confirmed experimentally.

The main contributions to the suboptimal BER performance are due to approximating the optimal correlator with an analog filter plus digital correlator, to tracking the timing slippage, and to using PIR instead of optimal coherent reception. The performance of the analog filter/digital correlator combination is obtained approximately analytically, and found to be independent of the signal-to-noise ratio. The BER degradation incurred by the tracking is derived analytically and found to be negligible for the timing error induced by the typical present day TTL clock oscillators, unless the SNR is very high. The theoretical BER performance of PIR is obtained exactly analytically, and found to approach that of coherent reception as the SNR increases.

In addition, measured data for the mean time-to-acquire and the mean time-to-lose-lock in AWGN is presented; along with an outline on how these curves could be obtained analytically, and their implications for packet transmission.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I wish to thank my research supervisor Dr. Victor C.M. Leung for the direction he gave me, and for the leadership that he gave to the WLAN project, of which this work forms part.

I thank the professors from whom I have learned about the theory of modern day telecommunication systems — these are Prof. Samir Kallel, Prof. Takis Mathiopoulos, and Prof. Victor C.M. Leung. I also thank Professor Robert W. Donaldson for advice on my program and for having me on two earlier research projects, before letting me go to the WLAN project.

A very special thanks go to our scientific engineer Hansen Wang for designing a portion of the modem hardware, for doing the PCB layout, for assistance with the modem testing, and for very valuable help in general. Thanks also go to Amiee Chan for assisting Hansen and I with the RF portion of the hardware design and construction.

This work was supported by the Science Council of British Columbia through a Technology B.C. Grant.

Chapter 1 Introduction

1.1 Spread-Spectrum Communications: Motivation and Applications

A spread spectrum communication system is defined to be one for which the transmitted signal occupies a bandwidth much larger than the minimum bandwidth necessary to send the information [1]. Every information signal modulates a spreading signal, called the code signal, which is independent of the data. At the receiver, recovery of the original signal is accomplished by despreading; whereby the received spread signal is correlated with a synchronized replica of the spreading signal. The signal thus recovered is then processed by the usual techniques for communication signal reception. The techniques for spreading are direct sequencing, frequency hopping, time hopping, and hybrids of these.

The use of spread-spectrum techniques originated in the development, by the military, of communication systems highly immune to intentional interference by a jammer. The idea is that if many orthogonal signal coordinates are available to a communication link, and if only a small subset of these coordinates are used at any one time, a jammer who cannot determine the signal subset currently in use would be severely handicapped. Because the error performance of the system is a function of the received signal to noise ratio, against infinite power gaussian noise, increasing the bandwidth does not improve performance. However, when the noise comes from a jammer with fixed finite power, and with uncertainty as to where in the signal space the transmission is presently located, the error performance is significantly improved.

The energy density of a spread signal can be made very low because the signal power is on average spread uniformly over the enlarged number of signalling coordinates. This makes the signal very difficult to detect; and in fact, to anyone who does not posses a synchronized replica of the spreading signal, the spread signal will appear to be random noise.

These properties of high interference rejection and low probability of interception have been exploited to give rise to the multiple access technique called code-division multiple access (CDMA) wherein each user of the communication channel employs a unique spreading signal to locate herself in the common transmission band. In fact, the method has become the basis for the development of new cellular radio systems for personal communications

networks. The spreading signal is based on a pseudorandom sequence called a key. Each radio uses its own unique key for receiving transmissions, and each radio can transmit waveforms with the key corresponding to some other radio. The different keys correspond to sequences designed to have low cross-correlation and low autocorrelation properties.

The low cross-correlation of two different keys insures that a receiver locked on to a signal will experience relatively little interference from any other signal of the same channel based on a different key. Similarly, the low autocorrelation of a key will mitigate multipath effects, because a signal due to a multipath component which arrives with a delay with respect to the signal to which the receiver is synchronized, will be strongly attenuated.

Direct sequencing is discussed in detail in the next chapter. In the frequency hopping technique, the spreading is done by changing the carrier frequency of the transmitted signal, at a rate called the hop rate, according to the code sequence. With the time hopping technique, the code sequence is used to key the transmitter on and off.

1.2 Objective and Outline of This Thesis

The objective of this thesis is to present the design, analysis, and implementation of a new radio-frequency (RF) direct-sequence spread-spectrum (DS-SS) modem which can enhance both the power efficiency and the bandwidth efficiency of the present day conventional DS-SS systems. It uses a technique called Code-Phase-Shift Keying (CPSK), which is first proposed in [2].

The presentation is organized as follows: In Chapter 2 an introduction is given to the code-phase-shift keying method of direct-sequence spread-spectrum signaling. The remaining chapters describe work by the author which constitutes original contribution.

Chapter 3 documents both the design and the implementation on a digital signal processing (DSP) platform, of the modem. A correlation receiver is designed, and the problem of synchronizing the receiver to the baseband signal is solved. The synchronization to the bit intervals is achieved by an acquisition scheme which is a generalization of the simplest acquisition scheme for conventional DS-SS. A timing recovery algorithm to maintain the bit interval synchronization, called tracking, has been realized in a hardware efficient manner by a novel design.

Optimal correlating has been approximated by a combination of analog filtering and digital correlating; and in Chapter 4, an analysis of the signal-to-noise ratio (SNR) degradation

incurred by this approximation has been estimated analytically. This chapter also contains theoretical analysis of the bit-error-rate (BER) in additive white gaussian noise (AWGN) performance of the herein designed signal tracking scheme. The implementation loss associated with the tracking scheme has been analytically estimated and found to be negligible for BERs less than 10^{-6} for the timing slippages induced by typical TTL clock oscillators. To close Chapter 4, measured BER performance in AWGN, of the receiver with the carrier-wave synchronization imposed externally, is presented. These measurements were performed to test the design, and to obtain an experimental confirmation of the analytic performance estimate thus far.

The synchronization to the phase of the carrier wave problem is solved by a new method which is hereby named Phase-Invariant-Reception (PIR). This is presented in Chapter 5 along with a theoretical performance analysis and experimental results for the BER in AWGN. The degree to which PIR approximates optimal reception has been obtained exactly analytically as a function of the frequency difference between the carrier-wave and the local oscillator (LO) of the receiver.

Chapter 6 gives theoretical analysis and experimental results on system performance in the presence of single-tone interference. A numerical expression for the BER performance of the coherent CPSK receiver has been obtained in AWGN and single tone interference. This is compared to the measured performance of the CPSK receiver employing phase-invariant reception. All claims to receiver performance have been verified, to within experimental uncertainty, by the bit error rate measurements.

Chapter 7 presents experimental results on the acquisition and tracking performance of the receiver, and outlines the mathematical problems which need to be solved in order to predict these results theoretically. Mean time-to-acquire and mean time-to-lose-lock in AWGN data is reported, along with the implications for transmission data packet format.

Chapter 8 finishes with the conclusions and discussion of future development.

Chapter 2 Introduction to DS-SS and CPSK

2.1 Standard Direct-Sequence Spread-Spectrum with Binary-Phase-Shift Keying

At the modulator of a generic direct-sequence system, the information signal, of data rate R_d, is multiplied by a code signal with symbol rate R_c, called the code chip rate. The ratio $G = R_c/R_d$ is equal to the factor by which the signal transmission bandwidth is spread. It is usually much greater than unity, and is called the processing gain.

Since multiplication in the time domain transforms to convolution in the frequency domain, provided the information signal is relatively narrow-band, the product signal will have approximately the bandwidth of the spreading signal. At the demodulator, the received signal is multiplied by a synchronized replica of the code signal; thus collapsing the desired signal to its original bandwidth, while spreading any undesired signal in the same way that the transmitter spread the desired signal originally. The signal is subsequently passed through a bandpass filter, whose passband corresponds to the spectrum of the information signal, resulting in a high rejection of the interfering signal. The difference in output and input signal-to-noise ratios, in dBs, for a narrow band interferer is equal to the processing gain in dB. To see this in detail, refer to Figure 1 and consider the following analysis for the conventional coherent DS-BPSK-SS system. More details can be found in [3].

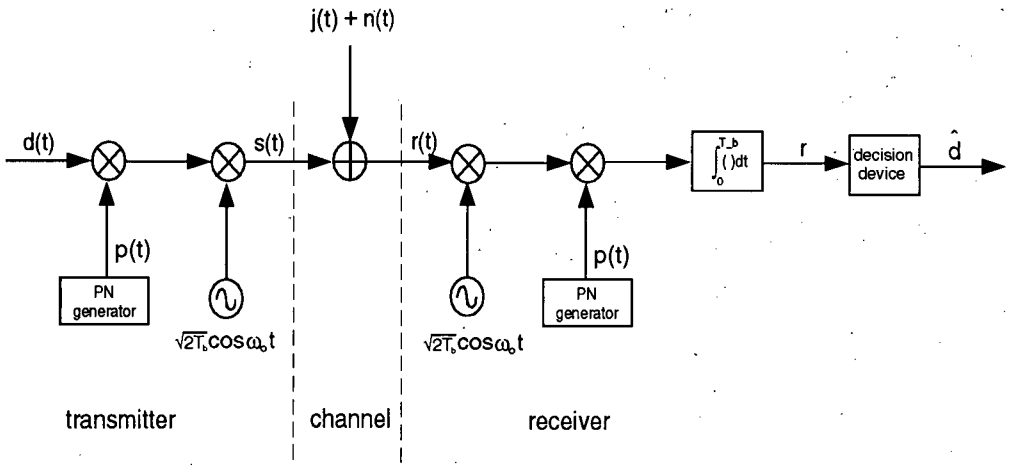

Figure 1 Conventional DS-BPSK-SS System

The information sequence to be sent over the channel is represented by an antipodal pulse stream, $d(t)$, where a pulse value of $+1$ represents a 0 bit, a pulse value of -1 represents a 1 bit, and the bit duration is T_b. This information signal is spread by multiplication with the antipodal chip stream $p(t)$, with chip duration $T_c = T_b/G$. $p(t)$ represents a periodic pseudorandom sequence called a pseudonoise (PN) sequence; the generation of which is discussed in Appendix A. It is required that $p(t)$ be synchronized with $d(t)$, so that each bit period accommodates precisely G chips. Finally, the spread signal modulates the carrier, to produce the transmitted signal:

$$s(t) = \sqrt{2P}d(t)p(t)\cos\omega_o t; \tag{2.1}$$

where $P = E_b/T_b$ is the transmitter power, E_b is the energy of the signal representing one bit, and ω_o is the carrier frequency in radians.

With the receiver exactly synchronized to the bit interval, and to the carrier phase, the delay for the communication link can be taken to be zero, and the simplified block diagram, omitting the synchronization modules, of the system is as in Figure 1. We wish to consider the error performance of the idealized receiver for a channel which adds noise and interference to the transmitted signal.

The channel output is

$$r(t) = s(t) + j(t) + n(t); \tag{2.2}$$

where $j(t)$ is narrow-band interference, centered at the carrier, of total average power J_{av}, and $n(t)$ is additive white gaussian noise of power spectral density N_o. Denoting the spread signal bandwidth as W, the value of the power spectral density of an equivalent wide-band interference is $J_o = J_{av}/W$.

The received signal is despread, by correlating against $p(t)$, and demodulated by correlating against $\sqrt{2/T_b}\cos\omega_o$. Thus, after every T_b seconds, the BPSK detector outputs

$$r = d\sqrt{E_b} + J + N; \tag{2.3}$$

where d $(= \pm 1)$ is the data bit for the T_b second interval, and the interference and noise components are respectively:

$$J = \sqrt{\frac{2}{T_b}}\int_0^{T_b} p(t)j(t)\cos\omega_o t\, dt \tag{2.4}$$

$$N = \sqrt{\frac{2}{T_b}} \int_0^{T_b} p(t)n(t) \cos \omega_o t \, dt. \tag{2.5}$$

The white noise has zero mean and variance $E[N^2] = N_o/2$; while the interference has zero mean, and in the limit of zero bandwidth, variance $E[J^2] = J_o/2$.

The BPSK decision rule is to choose $\hat{d} = +1$ if $r > 0$, and choose $\hat{d} = -1$ if $r \leq 0$. Assuming that $d = \pm 1$ are transmitted with equal probability, one may take, without loss of generality, $d = -1$; to compute the probability of bit error P_ϵ.

$$P_\epsilon = \Pr\{r > 0 | d = -1\} = \Pr\{J + N > \sqrt{E_b}\} \tag{2.6}$$

With the statistics of the interference gaussian and independent of the thermal noise statistics, the bit error probability is given by

$$P_\epsilon = \frac{1}{2} \text{erfc} \left(\sqrt{\frac{E_b}{N_o + J_o}} \right); \tag{2.7}$$

where erfc is the complementary error function.

Therefore; as previously mentioned, the spreading does not improve the error rate performance of Coherent BPSK in white noise. However, the effects of the narrow-band interferer are reduced by a factor equal to the processing gain.

To implement Forward Error Correction (FEC) encoding in conventional DS-SS, the data stream is first encoded to a higher rate symbol stream by block or by convolutional encoding; which is then transmitted by the same transmitter as in Figure 1. The receiver structure for coherent reception is also the same except that the integration is now over the symbol period; and the resulting stream of decision variables is used by a more general decision device to form an estimate of the original data stream.

To increase the data rate without increasing the required transmission bandwidth, while maintaining the high interference rejection capability, one can use *M*-ary modulation, in which a signaling alphabet of M different code sequences is used to transmit alphabet symbols representing multiple bit sequences. One such method, which offers the performance advantages of conventional *M*-ary DS-SS systems; but also many implementation advantages over other *M*-ary DS-SS systems, is the CPSK method described in the next section.

2.2 Code-Phase-Shift Keying DS-BPSK-SS

In the CPSK method, each of the $M = 2^k$, where the word length k is the number of bits per symbol, signaling waveforms is obtained by a different phase shift (an integer number of chips) of a single PN maximal length code sequence $p(t)$. It's autocorrelation function is

$$R(\tau) = \sum_{i=0}^{i=G-1} p(iT_c)p((i+\tau)T_c) = \begin{cases} G, & \text{if } \tau = 0, G, 2G, \ldots; \\ -1, & \text{otherwise.} \end{cases} \tag{2.8}$$

The transmitter, shown in Figure 2, groups the data into k-bit data symbols of duration T_s. Each of these are represented by an integer m, $0 \leq m \leq M-1$, which is used to select the signaling waveform $p_m(t) = p(t - m_c T_c)$; which is the phase shift by $m_c = m(G+1)/M$ chips of the non-shifted PN sequence $p(t)$. The final up-conversion for transmission at carrier frequency ω_o results in transmitter output

$$s(t) = \sqrt{2P}\, p_m(t) \cos \omega_o t; \tag{2.9}$$

where $P = E_s/T_s$ is the transmitter power, and E_s is the energy of the signal representing a symbol (a PN sequence).

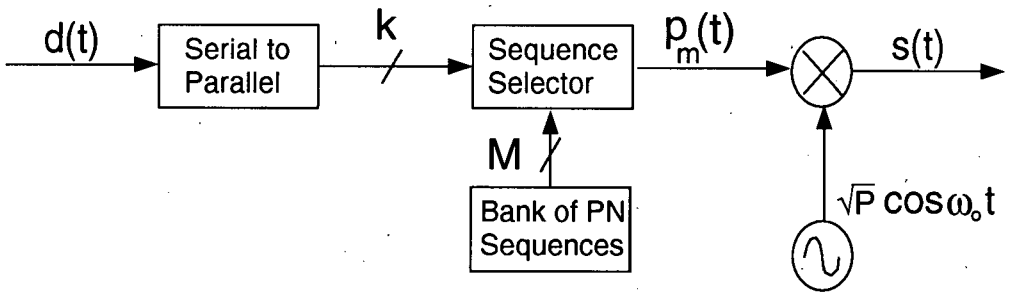

Figure 2 CPSK Transmitter

For now, assume synchronization, so that the CPSK correlation receiver for coherent reception is as in Figure 3. The received signal, $r(t) = s(t) + j(t) + n(t)$, is fed to a bank of M correlators, one for each alphabet member PN sequence $p_m(t)$. When the $m-$th symbol is sent, the output of the $m-$th correlator, after every symbol period, is

$$r_m = \sqrt{E_s} + J_m + N_m; \tag{2.10}$$

while the output of the other correlators, r_i for $i \neq m$, are

$$r_i = \frac{-1}{G}\sqrt{E_s} + J_i + N_i. \tag{2.11}$$

The noise and interference terms are ($\forall i \mid 0 \leq i \leq M-1$):

$$J_i = \sqrt{\frac{2}{T_s}} \int_0^{T_s} p_i(t) j(t) \cos \omega_o t \, dt \tag{2.12}$$

$$N_i = \sqrt{\frac{2}{T_s}} \int_0^{T_s} p_i(t) n(t) \cos \omega_o t \, dt. \tag{2.13}$$

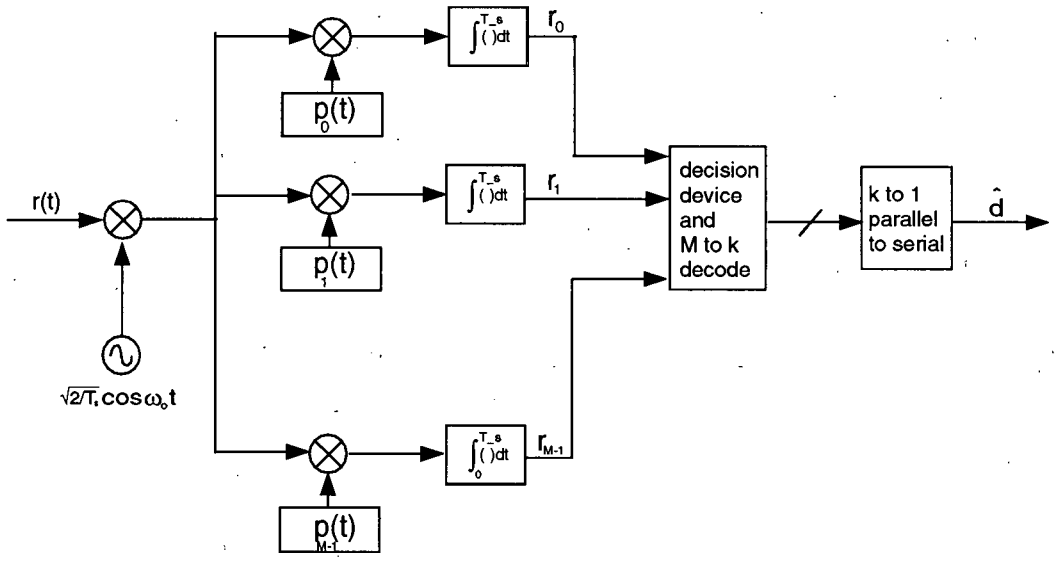

Figure 3 Synchronized CPSK Receiver

The decision device decides in favor of the symbol corresponding to the correlator with the largest output. When the interference is a single tone jammer at the carrier wave frequency,

$$j(t) = A\cos(\omega_o t + \phi), \tag{2.14}$$

the terms J_i are equal for all i, and therefore do not affect the decision. Thus, the effects of a carrier-wave jammer are completely mitigated by CPSK signaling. When the single-tone interference is not precisely at the carrier-wave frequency, the decision is affected, and the noise performance of the receiver is degraded (but still much better than for conventional DS-SS). The analysis of this more complicated situation is left for Chapter 6.

The white noise terms are independent, zero-mean, gaussian random variables with variances $E[N_i^2] = N_o/2$; and therefore, when the spreading factor is much greater than unity, $G >> 1$, the probability of symbol error in AWGN, P_ε, is precisely that for coherent reception of M orthogonal signals [4]:

$$P_\varepsilon(M, E_s/N_o) = 1 - \int_{-\infty}^{+\infty} f(\alpha - \sqrt{E_s}) \left(\int_{-\infty}^{\alpha} f(\beta)\, d\beta \right)^{(M-1)} d\alpha; \tag{2.15}$$

where

$$f(\alpha) = \frac{1}{\sqrt{\pi N_o}} \exp\left(\frac{-\alpha^2}{N_o}\right). \tag{2.16}$$

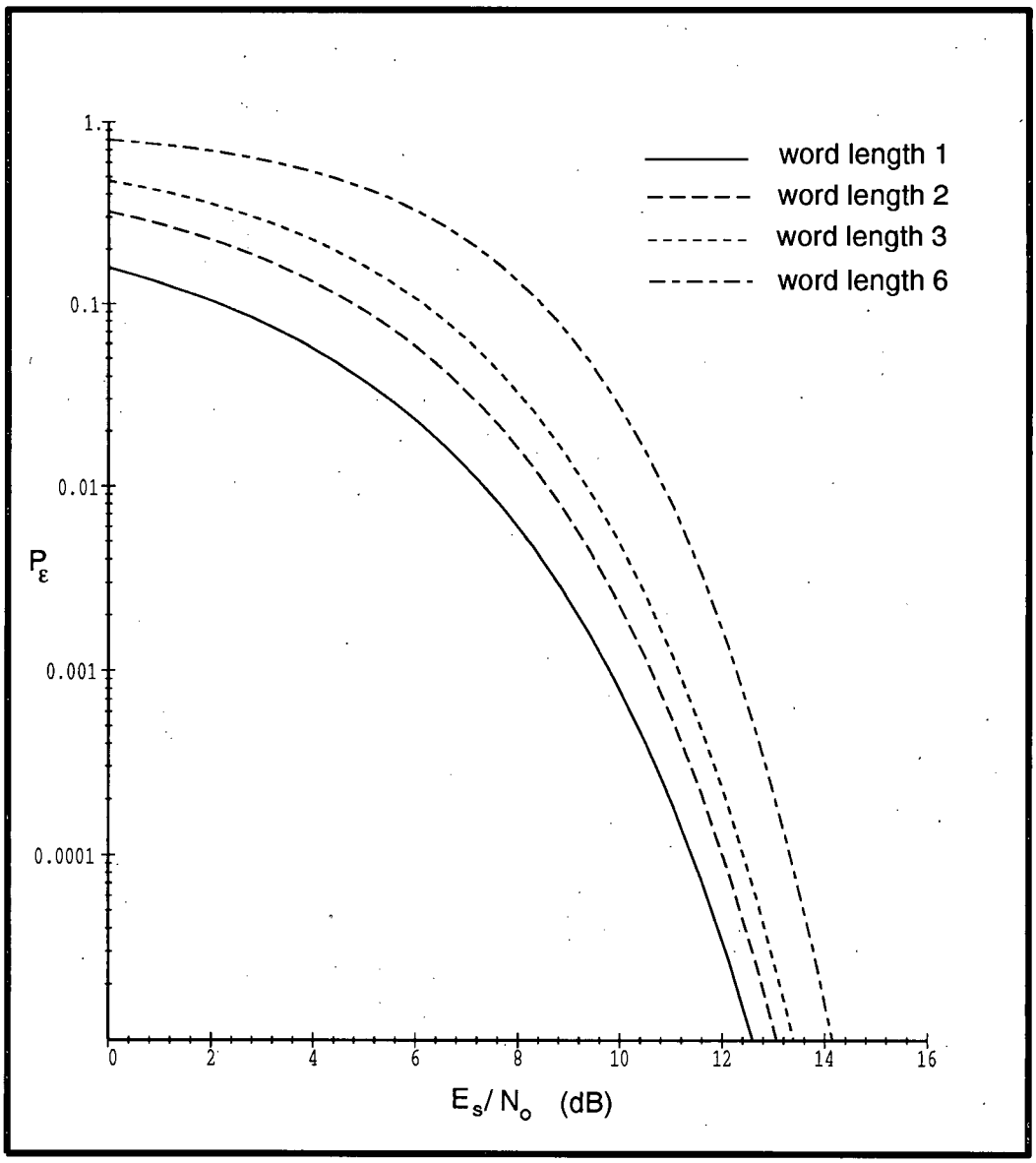

Figure 4 Probability of Symbol Error vs. SNR for CPSK with Coherent Reception in AWGN

Figure 4 shows a plot of the probability of symbol error as a function of symbol signal-to-noise ratio (SNR $= E_s/N_o$) for several values of M. Using the fact that an M-ary symbol represents $k = \log_2 M$ bits, and that the energy of a bit is $E_b = E_s/k$; the symbol error probability as a function of symbol SNR can be converted to a bit error probability as a function of bit SNR. The BER (equal to the bit error probability), at a given bit signal-to-noise ratio decreases with increasing word length k. The power efficiency of a communication system operating at a certain BER is defined as the SNR required to attain the specified BER. The power efficiency of optimal coherent M-ary signaling for word length 1 is 3 dB poorer, for word length 2 is slightly poorer, and for word length 3 is better than that of coherent BPSK (at any BER). At word length 6, M-ary signaling outperforms BPSK by 3.5 dB at a BER of 10^{-5}. In other words, at word length 6, in thermal noise, CPSK outperforms conventional DS-SS by 3.5 dB.

The system discussed thus far is an idealization whose performance represents the upper limit which an implementation could realize. Synchronization of the receiver to the phase of the carrier wave, and to the time of the bit and chip intervals has been assumed for coherent reception. Hereafter, a set of synchronization modules are designed, and optimal correlators are approximated, to realize a complete receiver which approximates optimal coherent reception.

Chapter 3 The Modem Implementation on a DSP Platform

3.1 General Architecture and Implementation

The development is aimed towards the construction of a wireless, high processing gain, high data throughput modem which transmits at radio frequency in an ISM band. (ISM bands are the industrial, scientific and medical bands in the gigahertz range over which unlicensed spread spectrum systems are employed.) See Figure 5 for the high level block diagram illustrating the general architecture. The signal processing unique to CPSK is done at the digital stage, which is implemented on a general purpose DSP. To test this system, we have developed a prototype which operates at IF (chosen to be 140 MHz) thus allowing the performance to be evaluated by measurements taken with an RF channel simulator inserted into the IF link. An eventual RF modem could be built and on-air tested by integrating the present system with IF/RF up/down converters and RF transceivers.

The general purpose DSP used is the QPC/C40B, built by Loughborough Sound Images, which is comprised of four TI TIM-40 modules, each of which hosts a TI TMS320C40 and 96 kilowords of SRAM [5]. Of the four C40s; three are dedicated to the receiver, and one is dedicated to the transmitter.

An optimizing C compiler along with a substantial run-time library has been purchased from 3L Software of Edinburgh [6]. However, the TMS320C40 C compiler does not produce satisfactory code for many of the time critical operations; as it does not seem to be aware of all of the hardware — in particular, the circular addressing modes of the C40. Therefore, it was found necessary to code many of the software modules directly in TMS assembler. The code listings, (either in C40 C or assembler, depending on the module), are given in Appendix D.

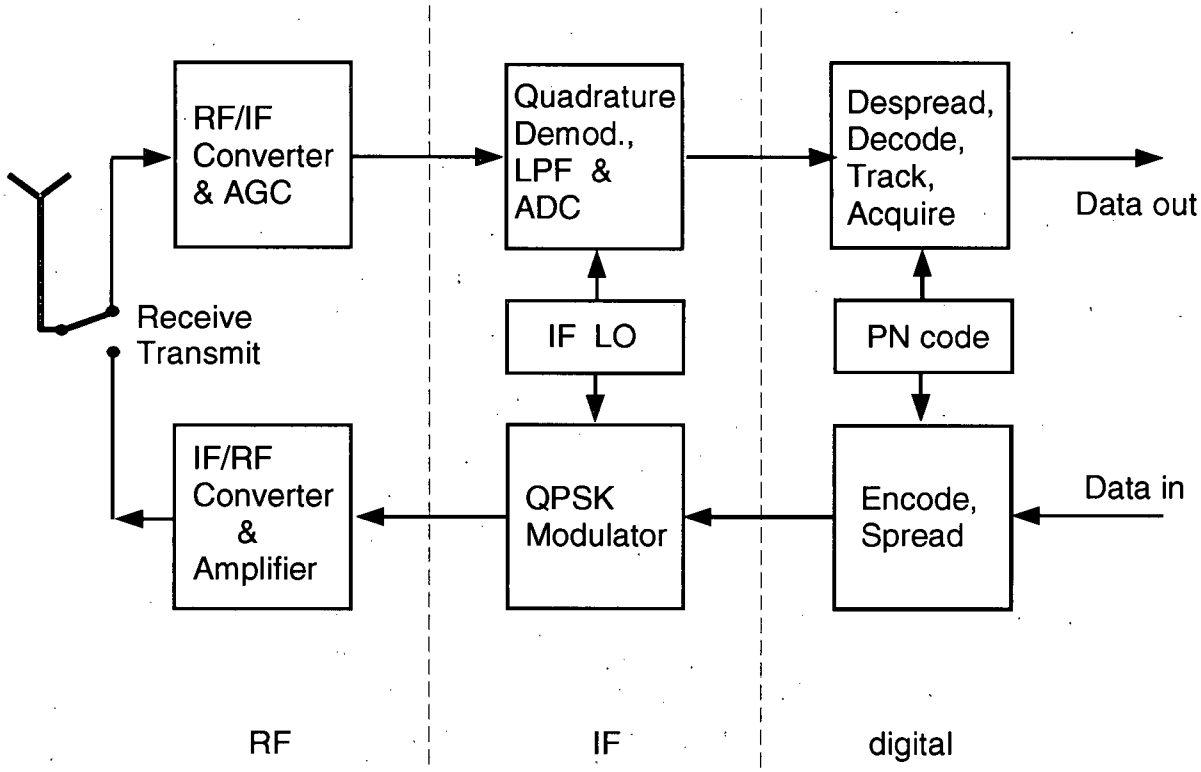

Figure 5 Wireless Spread-Spectrum Modem Block Diagram

3.2 The C40 Based IF Transmitter

The digital modulator implementing the code-phase-shift keying, direct-sequence spread-spectrum signaling technique, as described in the previous chapter, uses a C40 to produce an asynchronous chip stream from the data file to be transmitted. During the transmission, the software is responsible for maintaining the C40 output communication port FIFO non-empty; while an interface reads from the C40 fifo and clocks the data into a level-shifter (converts TTL to bipolar) and amplifier. The resulting chip stream is then BPSK modulated onto the IF carrier by Mini-Circuits MIQY-140M quadrature modulator. This is followed by a bandpass LC filter (in-house constructed by the author due to the unavailability of an off-the-shelf unit) to remove the signal replicas produced at harmonics of the carrier by the modulator. See Figure 6 for the block diagram of the C40 based IF transmitter. The indicated modules are built on a PCB copperboard, and connected to the C40 by a high density, 0.025" pitch ribbon cable. A description and a circuit schematic for each of these modules is given in Appendix B.

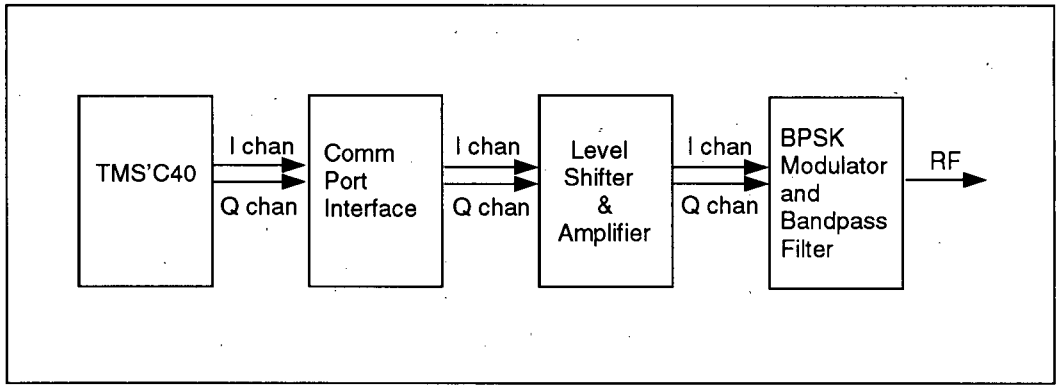

Figure 6 C40 Based IF Transmitter Block Diagram

3.3 The IF Receiver

The incoming IF signal is product demodulated and low-pass filtered so as to recover the baseband signal. The BPSK product demodulation is done with Mini-Circuits MIQY-140D quadrature demodulator. The ensuing low-pass filter not only removes the second and higher harmonics of the IF signal resulting from the product operation; but has a bandwidth chosen low enough to maximize the output signal-to-noise ratio of the BPSK signal demodulator. This is discussed in detail in the next chapter.

The baseband signal is sampled at twice the chipping rate (or four time the chipping rate with a two-sample preaccumulation) by a dual-channel 8–bit ADC. The samples are synchronously written to a bank of four FIFO memories, and asynchronously read from the FIFOs and written to the DSP by four FIFO/TMS320C40 communication ports. As explained more fully in the next section, for each channel of data (I or Q), the tracking algorithm requires the same channel of data delayed by one data symbol — resulting in a total of four data streams to be processed by the digital demodulator.

Figure 7 shows the high-level block diagram of the IF receiver. It was prototyped by the author and built on a full size PC AT printed circuit board as documented in [7]. Each of the modules indicated in Figure 7 is described in detail, with circuit schematics, in Appendix C.

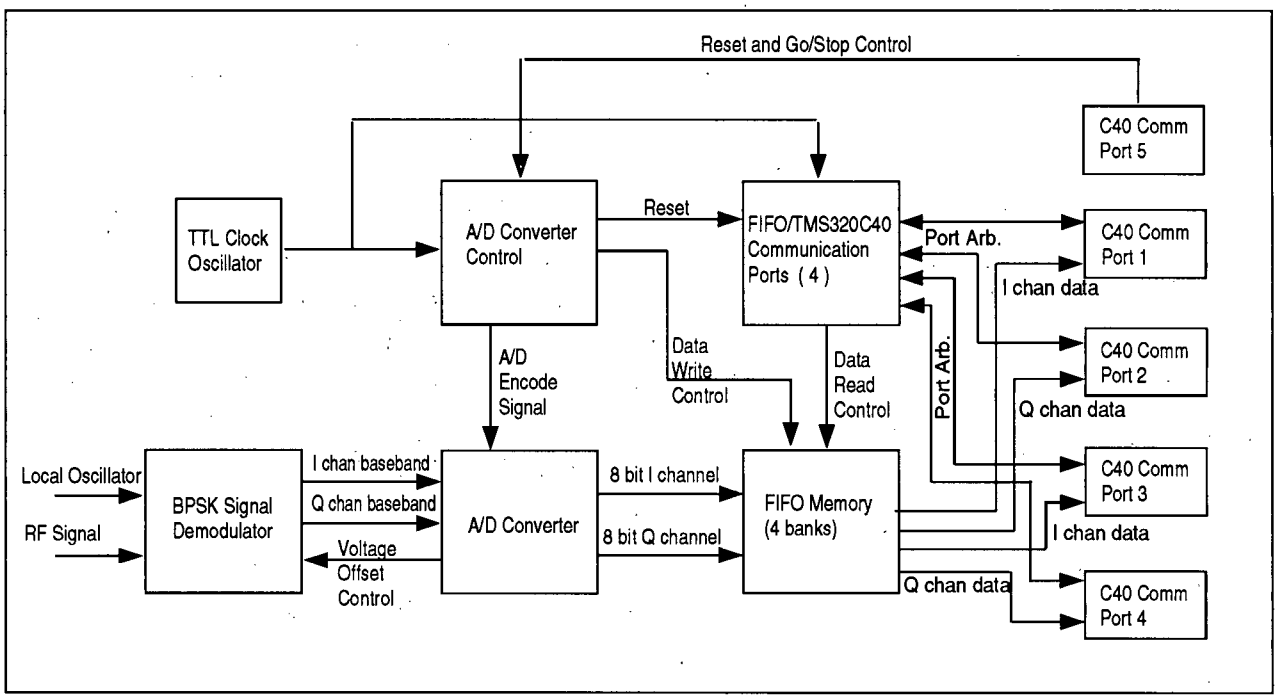

Figure 7 IF Receiver Block Diagram

3.4 The Digital Demodulator

There are two primary methods of implementing a despreader for the standard DS/SS system: the code-matched filter and the serial correlator-accumulator technique. This is also the case in the CPSK signaling scheme, wherein the receiver employs a bank of correlators or matched filters — one for each of the possible phase shifts.

In the code-matched filter technique, the entire reference PN sequence and each of its phase shifts corresponding to a data symbol are stored in separate length G registers; while the incoming signal samples are stored in another length G shift register. As the incoming signal moves chip by chip down the signal register, it will give a large positive correlation with one signal register when it contains a data symbol. In this method, no separate sequence acquisition is required, as the signal is acquired during the first complete symbol received. However, implementing this algorithm in software results in a prohibitively slow receiver because the cross-correlation between the signal sequence and each phase shifted PN sequence is computed once per chip. With a serial correlator-accumulator, one has an entire symbol period in which to compute the required cross correlations; and therefore we choose this technique, as it is appropriate for a DSP based implementation.

Figure 8 shows a block diagram of the digital receiver. The term m-correlator means a correlator which correlates against the PN sequence, $p_m(t)$, representing the m-th data symbol.

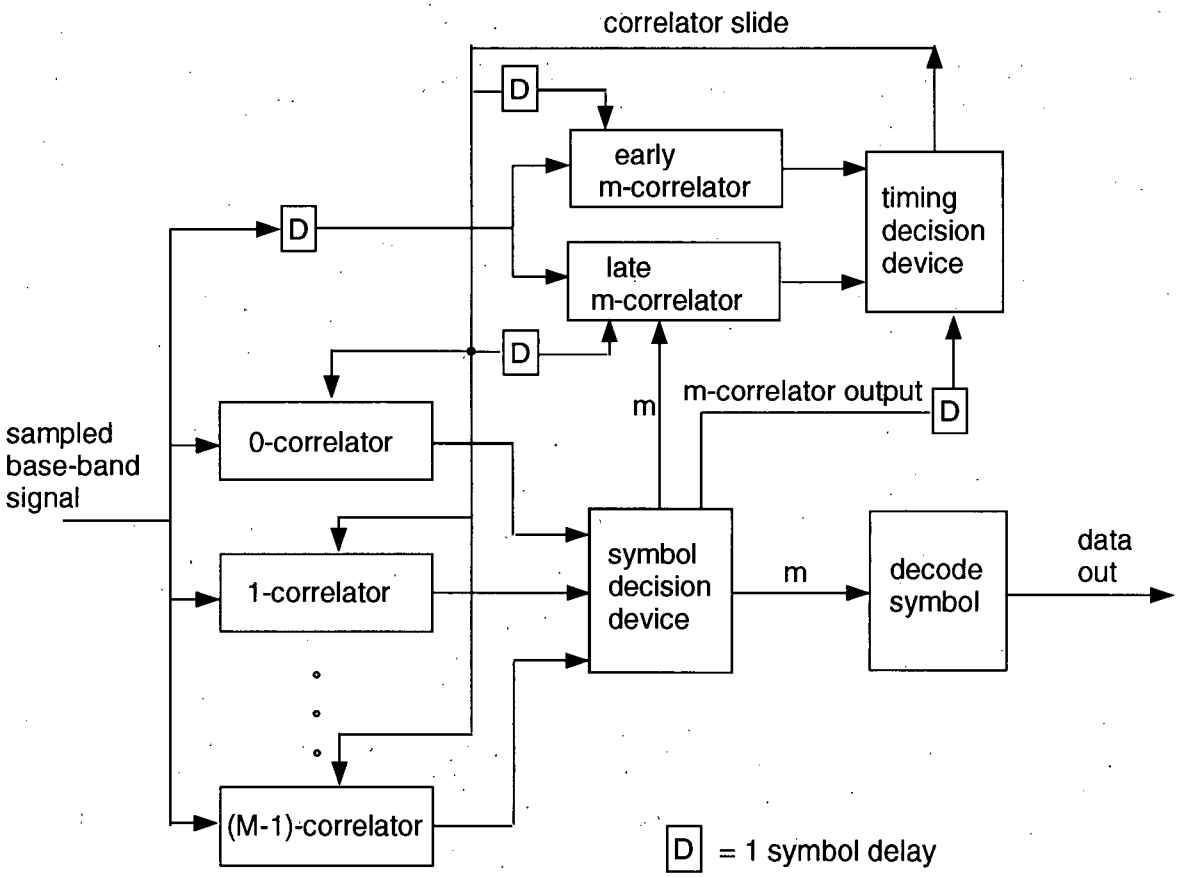

Figure 8 Digital Demodulator Block Diagram

Each correlator must process at the sampling rate (two times the chipping rate, R_c); whereas the symbol decision device must process at the correlators output rate (the data symbol rate times the number of symbols). With a fixed amount of processing power available for correlation at a given PN sequence length, the maximum chipping rate which the system can handle is inversely proportional to the number of data symbols (the number of correlators); whereas the data rate is proportional to the symbol word length, k, times the chipping rate. Because the number of data symbols is 2^k, the data rate, R_d, is given by

$$R_d = K\frac{k}{2^k}; \tag{3.1}$$

for some constant K. Thus, the maximum data rate is obtained for word lengths 1 and 2.

Based on this finite processing power performance consideration, we choose to implement the word length 1 and word length 2 versions on the DSP. By adding more processing power, one could maintain the chipping rate constant as one increases the data symbol alphabet length. This would give a data rate proportional to the word length. This could be done by adding more processors to the DSP, or by implementing the correlators in hardware, and using the DSP for processing the correlator bank output. The latter approach is taken in [8]; where a RAKE receiver for conventional DS-SS is designed.

The three processors dedicated to the digital demodulator share the tasks indicated in Figure 8 as follows: Processor A implements the symbol decision device, the timing decision device, and the symbol decoder. Processor B implements the bank of punctual correlators. Processor D implements the early and late pair of correlators required for tracking. (Processor C is dedicated to transmission). The one symbol delay required for the input sample stream to the early/late correlator pair is implemented in hardware by a FIFO on the IF receiver. Processor A is the master processor which controls the two slave processors implementing correlators. It reads from, and configures the correlators; reading and updating the configuration at the data symbol rate.

3.5 The Tracking and Acquisition Algorithms

The system is designed for packet transmission. After initialization, the receiver enters it's acquisition mode, to perform the initial PN code synchronization, and remains in this mode indefinitely until it successfully acquires a signal. It then enters it's tracking mode; during which it continually fine adjusts the symbol interval synchronization, and decodes and stores the incoming data. It remains in this mode until it loses lock — either because of excessive noise or because the signal transmission has ended. It then passes the data, stored in the DSP's on board memory, to the host computer. The program may then be restarted.

Each data packet must have a preamble, of sufficient length (discussed later), consisting of a string of the zeroth data symbol. This is followed by a special data sequence to flag the end of the preamble and the beginning of the original data. The data is also terminated by the special sequence. The data passed to the host by the modem will consist of the portion of the preamble remaining after the receiver achieved acquisition, the encapsulated data packet, and perhaps several bits decoded from noise before the receiver lost lock. It is left to the host to extract the encapsulated data packet.

In the acquisition mode, the data symbol correlators are precessed by a shift of two samples, (one chip) between every trial symbol read. With reference to Figure 8, the symbol decision device chooses the correlator with the largest output and passes that output to the timing device, the timing device outputs the two-sample correlator precession, and the symbol decode is not invoked. The early/late pair of correlators compute a one sample early and a one sample late (called slides) version of the correlation (with precessed correlator) corresponding to the symbol which gave the largest correlation on the trial precession. When a data symbol correlator output crosses a first threshold, the timing device compares the early and late correlator outputs with the on-time correlator output, and the maximum of the three is checked against a second higher threshold. If it is crossed, the timing device adjusts the precessed (shifted) correlators according to the one sample slide adjustment and on the next trial symbol read, checks if the largest correlation crosses a third threshold. If the acquisition is confirmed, it switches to tracking mode; if the acquisition is not confirmed, it stays in acquisition mode — increments the precession and repeats the tests. In this way, the signal can be acquired in the presence of a timing error less than or equal to the maximum timing error that the tracking module can handle. See Figure 9 for the timing decision device's algorithm for acquisition.

The CPSK signaling method is well suited to this acquisition scheme because all M data symbol correlators contribute to the search on every trial. Therefore, to precess through an entire PN sequence length G; takes only G/M trials. In other words, the acquisition time for the CPSK scheme is cut down from the standard acquisition time for conventional DS-SS by a factor of M.

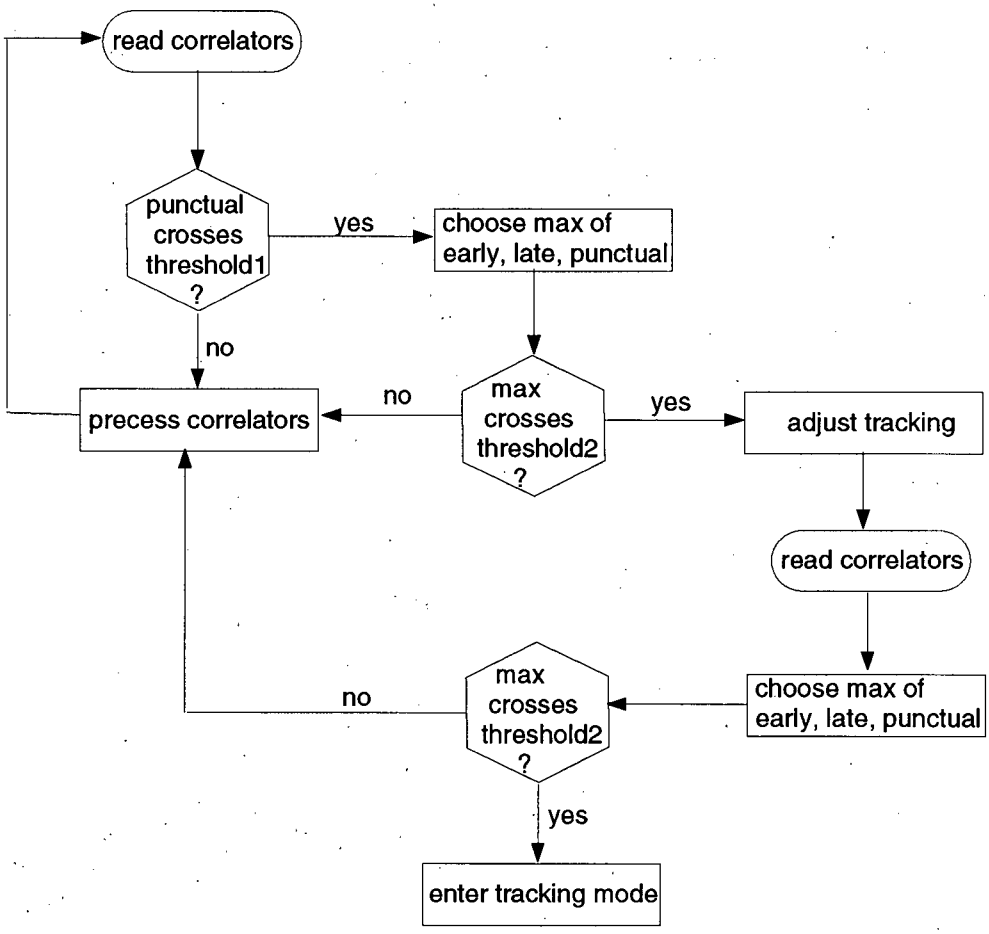

Figure 9 Timing Decision Algorithm in Acquisition Mode

In tracking mode (refer to Figure 8), the symbol decision device chooses the correlator with the largest output and passes the symbol to the symbol decoder. This symbol also determines which shift of the PN sequence the early/late pair of correlators use for correlating against the one symbol delayed stream. If the early or the late correlator give a larger output than the punctual correlator, and the timing decision device has its flag raised for tracking adjustment in the corresponding direction, all the correlators are precessed by the plus one or minus one sample slide. If the flag was not up, it gets raised. If the punctual correlator gave the largest output, the flag is lowered. In other words, the correlators must indicate that the tracking must be adjusted in a particular direction twice in a row before the tracking adjustment is made. See Figure 10 for the timing device's tracking algorithm. It follows a similar algorithm to check for tracking loss..

The timing decision device checks for loss of lock by raising a flag if the maximum of the punctual, early, or late correlator output falls below a threshold. If the flag was already up, the tracking is declared to be lost. If the maximum of the punctual, early, or late correlator exceeds the threshold, the flag is lowered. That is, the maximum of the punctual, early, or late correlator outputs must fall below the threshold on two consecutive data symbols before the tracking is declared to be lost — at which point the reception is aborted, and the stored data is then passed to the host.

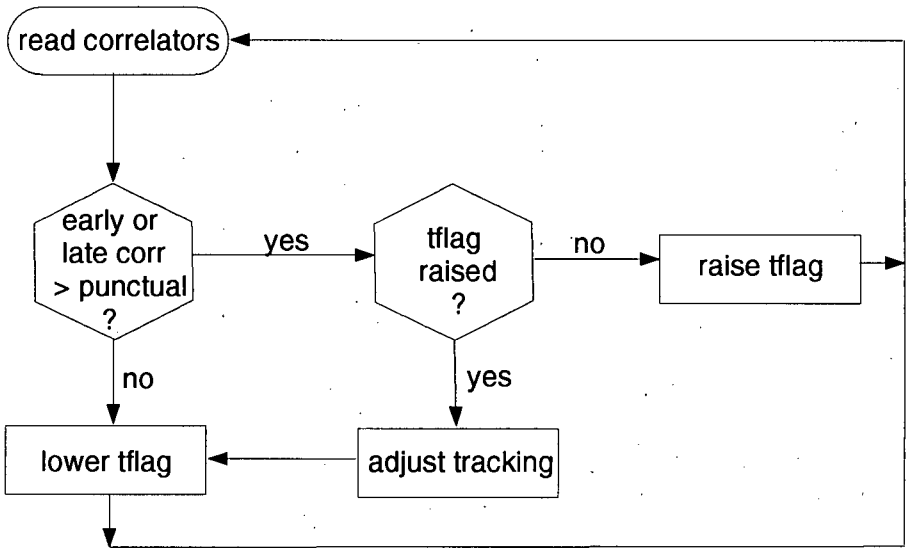

Figure 10 Timing Decision Algorithm for Tracking

3.6 DSP On Board Results

Before inserting the IF link, the DSP software was tested on-board by having the digital transmitter communicate directly with the digital receiver. Besides providing a test of the software functionality, this allows determination of the maximum sampling rate which the receiver can handle. Also, timing error can be simulated in real time by adding or deleting samples from the transmitted data stream. The receiver acquires and decodes the otherwise noiseless transmission successfully for a timing slippage as high as 1 sample for every 4 data bits. Because there are $2G$ samples per data bit, this translates to a maximum timing slippage of $12.5/G$ percent.

Table 1 shows the maximum sampling rates in MSPS and the corresponding data rate in KBPS for spreading factors of $G = 2^l - 1$, and word lengths 1 and 2, for single channel and complex channel processing.

shift	PN seq	k=1, single chan.		k=2, single chan		k=1, I+Q chan.	
reg len l	len G	MSPS	KBPS	MSPS	KBPS	MSPS	KBPS
3	7	1.45	104	0.80	114	0.70	100
4	15	1.96	65.3	1.02	68.0	0.92	61.3
5	31	2.33	37.6	1.16	37.4	1.06	34.2
6	63	2.57	20.4	1.24	19.7	1.15	18.3
7	127	2.70	10.6	1.28	10.1	1.19	9.4
8	255	2.77	5.4	1.30	5.1	1.22	4.8
9	511	2.81	2.7	1.32	2.6	1.24	2.4
10	1023	2.83	1.4	1.32	1.3	1.24	1.2

Table 1 Maximum sampling rates and the corresponding data rates

The maximum chipping rate is determined by the speed at which the receiver DSP modules can process. For word lengths of 1 or 2, and with G larger than the threshold value of 63, the inner kernel of the code implementing a correlator is the bottleneck. For G smaller than the threshold value, the bottleneck becomes the "intelligent" process which accepts the correlator outputs. We have not determined how much this threshold value of the spreading factor increases for word lengths higher than 2.

3.7 Experimental Setup

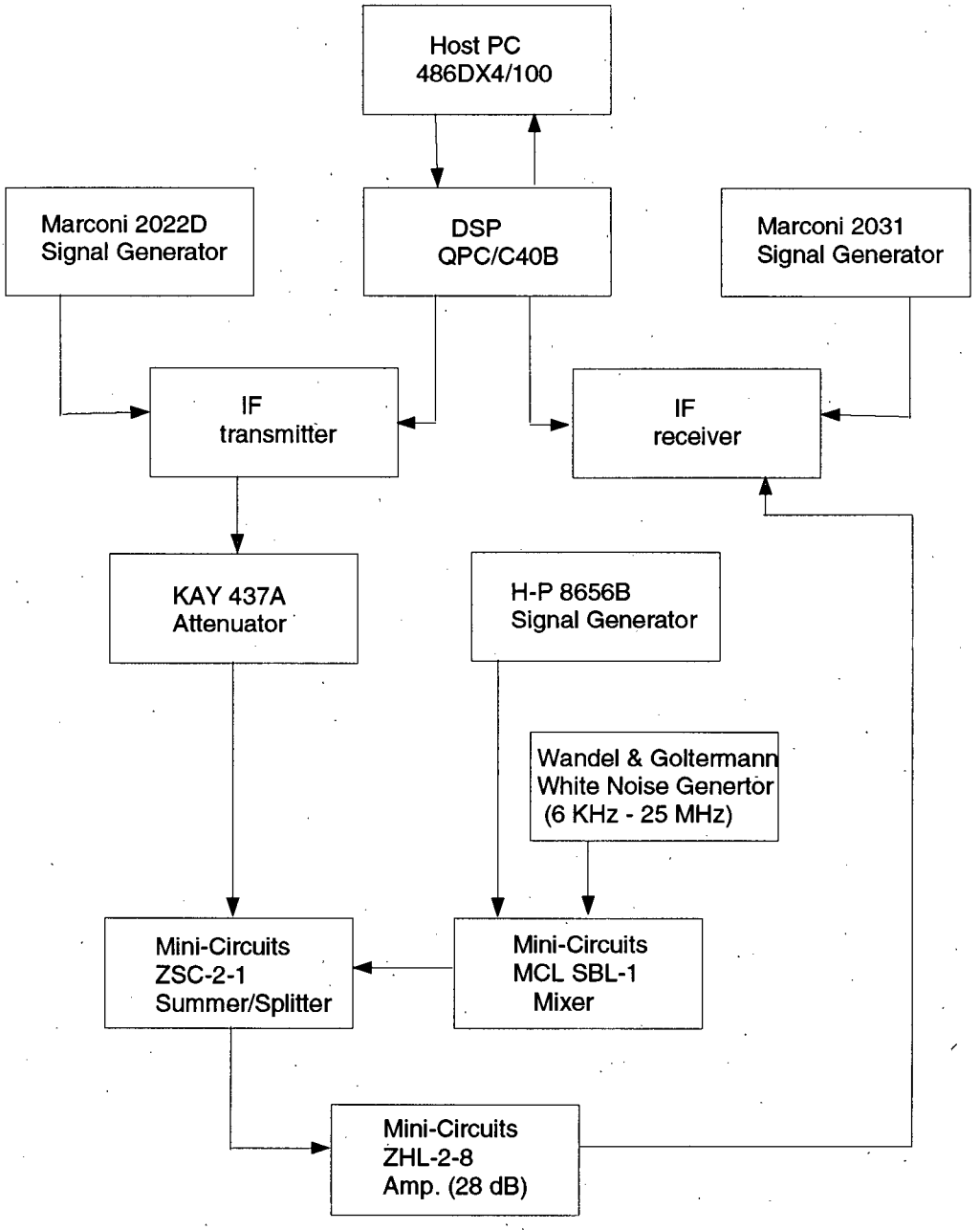

Figure 11 Experimental Setup for Measuring BER in AWGN

The DSP is a full-sized PC AT card residing in a 16 bit expansion slot of the host 486 PC. It communicates with the IF transmitter and the IF receiver via the C40's 8–bit high speed parallel communication ports.

The Marconi 2022D signal generator provides the 140 MHz LO for the IF transmitter; while the Marconi 2031 signal generator provides the 140 MHz LO for the IF receiver. It should be stressed that the IF transmitter and the IF receiver are each driven by their own TTL clock oscillator as well; rendering them independent.

The IF transmitter produces a 2.9 dBm double-sideband-suppressed-carrier signal centered at 140 MHz, which is further attenuated by 30 to 50 dB, by the KAY 437A step attenuator. This signal is summed with the AWGN by Mini-Circuit's two-way, 0 degrees, ZSC-2–1 Summer/Splitter.

The AWGN is produced at baseband by the Wandel & Goltermann White Noise Generator; and is mixed up to the frequency range of the data transmission by Mini-Circuit's MCL SBL-1 frequency mixer, whose LO is provided by the Hewlett-Packard 8656B signal generator.

The total signal (data plus noise), is brought back up in strength by 28 dB at the IF receiver front-end. The signal-to-noise power measurements were made at the IF receiver front-end with the Tektronix 497P Spectrum Analyser. The BER measurements were made by having the transmitter send a simple pattern of data bits (therefore, essentially random pattern of chips) and having the receiver operate like a BER analyser. It triggers on the first non-zero symbol, and counts the times that the decoded symbol stream does not match the pattern.

Chapter 4 Steady-State Performance of the Coherent Receiver

4.1 Approximating The Optimal Correlator

In the DSP based implementation of the CPSK receiver, the baseband signal is sampled at twice the chipping rate, and the $2G$ samples representing a data symbol are correlated by an add and accumulate operation. Thus the integration in Figure 3 is approximated by a summation, and the optimal correlator is roughly approximated. To improve the approximation, we low-pass filter the baseband signal, $y(t)$, before sampling it; and therefore, approximate the ideal analog correlator of Figure 12 by the combination of analog filter and digital correlator in Figure 13.

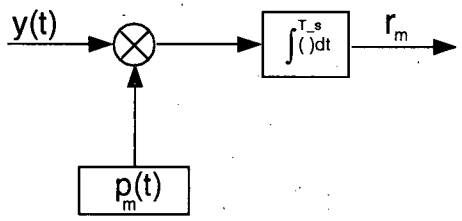

Figure 12 Analog Correlator

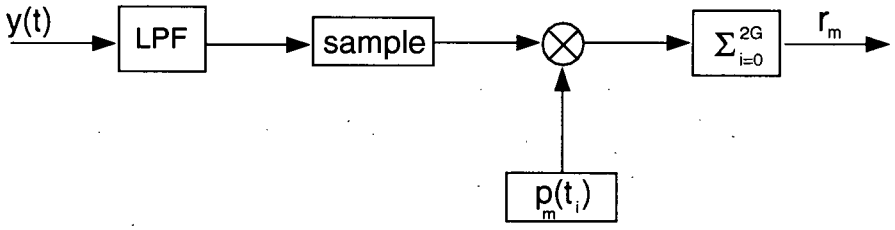

Figure 13 Analog Filter with Digital Correlator

The resolution of the tracking scheme is one sampling interval, so that the position in the chipping interval of the pair of samples for one chip is a random variable. The purpose of this chapter is to show how the cut-off frequency of the low-pass filter (which filters the chips) can be chosen so as to maximize the output signal-to-noise ratio of the digital correlator for a PN sequence of chips.

The output signal-to-noise ratio of a filter or correlator, SNR_o, at time T, is defined as the ratio of the instantaneous power of the output signal, $r_o(T)$, to the average power in output noise $n_o(t)$:

$$\mathrm{SNR}_o = \frac{|r_o(T)|^2}{\mathcal{N}_o}. \tag{4.1}$$

A matched filter or correlator, optimizes this ratio to the value for coherent reception:

$$\mathrm{SNR}_{o,opt} = \frac{2E_s}{N_o}. \tag{4.2}$$

We take the low-pass filter of Figure 13 to be first order RC; thus, with transfer function given by

$$H(f) = \frac{1}{1 + jf/f_c}; \tag{4.3}$$

where the 3–dB bandwidth of the filter, f_c, is given by

$$f_c = \frac{1}{2\pi RC}. \tag{4.4}$$

We require the output SNR of the digital correlator for the reception of $2G$ samples of the LPF output correlated against the PN sequence, $p_m(t_i)$, which takes values ± 1. Start by calculating the output noise power $\mathcal{N}_o$.

The ith sample of the low-pass filtered noise is

$$n_i = \int_{-\infty}^{+\infty} h(\tau) n(t_i - \tau)\, d\tau; \tag{4.5}$$

where $h(\tau)$ is the impulse response of the LPF. Since the noise output is $\sum_{i=0}^{2G} p_m(t_i) n_i$, the average output noise power, obtained by taking the expectation of the noise output squared, is

$$\mathcal{N}_o = \sum_{i=0}^{2G} \sum_{j=0}^{2G} p_m(t_i) p_m(t_j) E[n_i n_j]. \tag{4.6}$$

Using (4.5), the expectation $E[n(t_i)n(t_j)] = \frac{1}{2} N_o \delta(t_i - t_j)$, and the fact that $h(\tau)$ is the fourier transform of $H(f)$; leads to

$$E[n_i n_j] = \frac{\pi}{2} N_o f_c e^{-2\pi f_c |t_j - t_i|}. \tag{4.7}$$

Since the cutoff frequency, f_c, is the same order of magnitude as T_c^{-1}, and for $i \neq j$, $|t_j - t_i| \geq T_c/2$, the terms for which $i = j$ are dominant. Furthermore, most of the terms

in the sum (4.6) cancel for $i \neq j$; therefore, the sum (4.6) is approximately given by the sum of the $i = j$ terms:

$$\mathcal{N}_o = \pi G N_o f_c. \tag{4.8}$$

To calculate the output signal power of the filter/correlator, we first need the sampled response of the LPF to a square pulse chip of duration T_c and amplitude A, as shown in Figure 14. The time origin is indicated by the circle, and the two sampling instances are indicated by the crosses. Let τ be the time, as measured from the origin, of the first sampling instance. The second sampling instance occurs at time $\tau + T_c/2$. Let δt be the time interval from the rising edge of the square pulse to the time origin. The tracking scheme (next chapter) keeps the sampling times synchronized to the chip intervals, to within 1 sampling interval, by maintaining maximum signal power. Therefore τ takes a random value uniformly distributed in the interval $[0, \frac{1}{2}T_c]$.

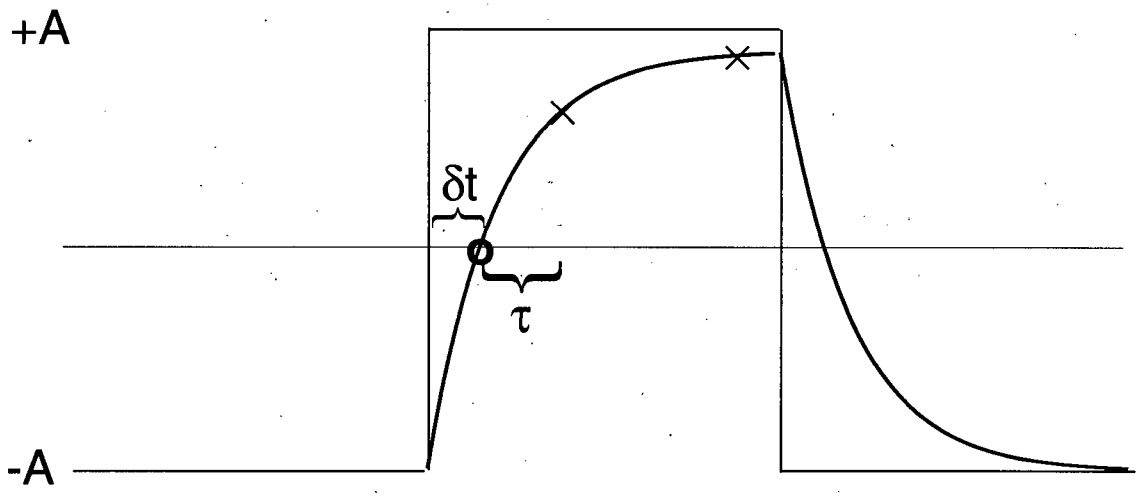

Figure 14 LPF Response To a Square Pulse Chip

δt is determined by demanding that the well-known exponential solution to the first order RC circuit go from $-A$ to 0 in time δt:

$$0 = -A + 2A\left(1 - e^{-\delta t/RC}\right); \tag{4.9}$$

which solves, using (4.4), for δt to

$$\delta t = \frac{\ln 2}{2\pi f_c}. \tag{4.10}$$

The sampled response, $s_c(\tau)$ of the LPF to the chip, is the sum of the LPF output at the two sampling times:

$$s_c(\tau) = \begin{cases} 2A - A(1+e^{-b})e^{-\omega_c \tau}, & \text{for } \tau \in [0, \frac{1}{2}T_c - \delta t]; \\ A(e^{b} - e^{-b} - 1)e^{-\omega_c \tau}, & \text{for } \tau \in [\frac{1}{2}T_c - \delta t, \frac{1}{2}T_c]; \end{cases} \tag{4.11}$$

where b is defined to be $b = \pi f_c T_c$. To obtain an approximate expression for $r_o(T)$, the response of the correlator to a synchronized (to within a sampling interval) PN sequence, we make use of (4.11) along with some of the general properties of maximal length PN sequences. We will assume that the LPF response to chips which are preceded by a chip of the same polarity is $\pm A$. In other words, that the LPF output essentially reaches the voltage of the input when the input is held constant for longer than a chip duration. Once the LPF cut-off frequency is found, it can be checked that this assumption is consistent with the solution.

One quarter of the chips of a maximal length PN sequence (of length G) are included in run lengths of 1; and there are $G/4$ runs of length greater than 1. Therefore, since $3G/4$ chips are members of a run length greater than 1, and $G/4$ chips begin such a run length, there are $G/2$ chips preceded by a chip of the same polarity; and therefore the sampled signal strength of $G/2$ chips is approximately $\pm 2A$. The $G/4$ chips that immediately follow a greater than length 1 run length have the signal strength magnitude depicted in Figure 15 and given exactly by (4.11). The $G/4$ chips that immediately follow a run length of 1 have a signal strength magnitude approximately given by (4.11). Therefore, since multiplication by the synchronized PN sequence gives all the samples positive polarity, the output of the digital correlator is approximately

$$r_o(T) = \frac{1}{2}G(s_c(\tau) + 2A). \tag{4.12}$$

It remains to average the output SNR of equation (4.1) over the random variable τ.

From (4.2) it follows immediately that

$$\mathrm{SNR}_{o,opt} = \frac{2A^2 T_s}{N_o}. \tag{4.13}$$

So that with (4.12) and (4.8) substituted into (4.1), we have the ratio of the average output signal-to-noise ratio of the digital correlator, $\overline{\mathrm{SNR}_o}$, to the output signal-to-noise ratio of an ideal analog correlator, $\mathrm{SNR}_{o,opt}$:

$$\frac{\overline{\mathrm{SNR}_o}}{\mathrm{SNR}_{o,opt}} = \frac{\overline{\left(\frac{1}{2}(2 + s_c(\tau))\right)^2}}{2b}; \tag{4.14}$$

where the indicated average is with respect to τ, and $s_c(\tau)$ is given by (4.11) with A set to unity.

The average of $s_c(\tau)$ with respect to τ is

$$\overline{s}_c = 2\left(1 - \frac{\ln 2}{b}\right) + \frac{e^{-2b}}{b}; \tag{4.15}$$

and the average of $s_c^2(\tau)$ with respect to τ is

$$\overline{s_c^2} = 4\left(1 - \frac{ln2}{b}\right) + \frac{1}{2b}\left(10e^{-2b} + 14e^{-3b} - e^{-4b} - 4\right). \tag{4.16}$$

Substitution of these last 2 equations into (4.14) yields the ratio of SNR's as a function of $b = \pi f_c T_c$ for the received data symbol.

The optimization of (4.14) with respect to b is not well defined because the signal strength A was assigned to one half of the chips, when really these strengths should be subject to the optimization as well. Instead we optimize, with respect to b, the ratio of the average output SNR of the LPF due to the chip of Figure 14 to the output SNR of a filter matched to the chip. This ratio is given by:

$$\frac{\overline{\mathrm{SNR}}_{o,chip}}{\mathrm{SNR}_{o,mat}} = \frac{\overline{(s_c(\tau))^2}}{2b}; \tag{4.17}$$

and is plotted in Figure 15. Optimizing the ratio (4.17) with respect to b yields an optimal value for the chip of

$$b = 2.28; \tag{4.18}$$

which, when substituted into (4.14) for the symbol, yields

$$\frac{\overline{\mathrm{SNR}}_o}{\mathrm{SNR}_{o,opt}} = 0.63 = -2.0 \text{ dB}. \tag{4.19}$$

The optimal value of b, as given by (4.18), corresponds to the first order RC LPF cut-off frequency $f_c = 0.73T_c^{-1}$.

The extremely complicated complete optimization problem has here been simplified to something tractable by the approximations made above. Thus both the optimum value of the LPF cut-off frequency, and the performance degradation of -2.0 dB incurred by approximating the matched filter by the LPF followed by the digital correlator, should only be viewed as reasonable estimates. Experimentation with the hardware of the implementation

has lead us to the approximate optimal value of $f_c = 0.5T_c^{-1}$ for which we have found an SNR degradation of only 0.8 ± 0.4 dB. By doubling the sampling rate and performing a two-sample preaccumulation, we find an SNR degradation of only 0.45 ± 0.4 dB.

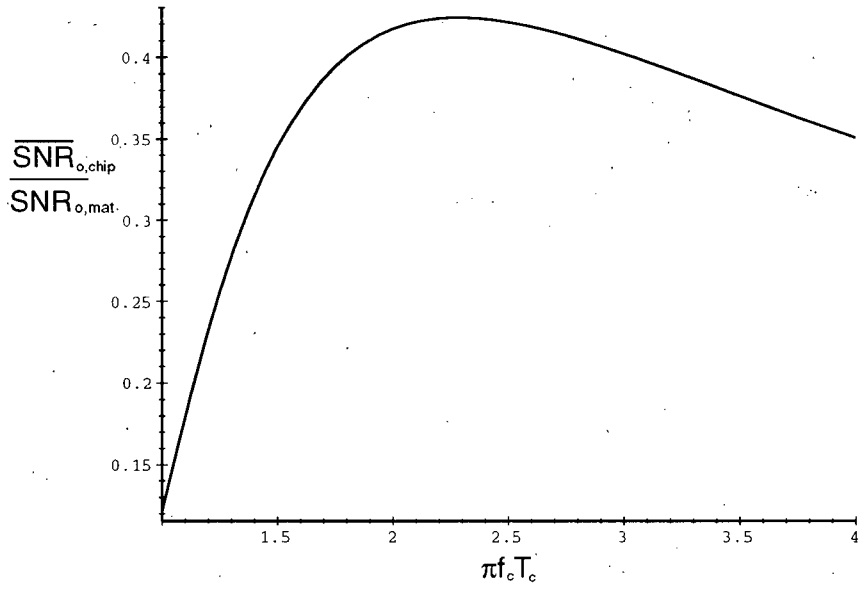

Figure 15 Signal-to-Noise Ratio for a Sampled Low-Pass Filtered Chip as a Function of LPF Cut-off Frequency

4.2 BER Performance In The Presence of Timing Slippage

In this chapter, we evaluate the BER in AWGN performance degradation for coherent reception, due to the fact that the receiver is not always exactly on track, even to within the one sample timing resolution. Recall from chapter 3 that at every data symbol interval, the output signal strength of the correlator corresponding to the data symbol being received is compared to the output signal strength of the corresponding correlator synchronized to the signal delayed by one sampling interval, and to the output signal strength of the corresponding correlator synchronized to the signal advanced by one sampling interval. If the delayed or advanced correlator has a higher output than the on-time correlator, the sample stream is shifted with respect to the on-time correlator appropriately.

Let S_0 denote the state in which the receiver is synchronized to within the one sample resolution; S_1 denote the state in which the receiver synchronization is off by one sample; and S_2 denote the state in which the receiver synchronization is off by more than one sample.

The receiver tests for state S_2 by checking if the signal strengths of the on-time, delayed, and advanced correlators all fall below a threshold. If it tests positive for state S_2, the receiver leaves its tracking and data decoding mode and returns to the state in which it is searching for a new transmission. When the receiver is in state S_2, the probability of exiting the tracking/data decoding mode is very high, so that the probability of state S_2 is very low; and we approximate the problem by assuming that the probability of state S_2 is zero. We have checked this approximation, in the absence of timing slippage, by including state S_2 in the analysis, and have obtained the same result as that of the following calculation.

Therefore, we approximate the receiver when it is in tracking/data decoding mode by the two-state Markov chain depicted in Figure 16. The solution for the two-state chain state

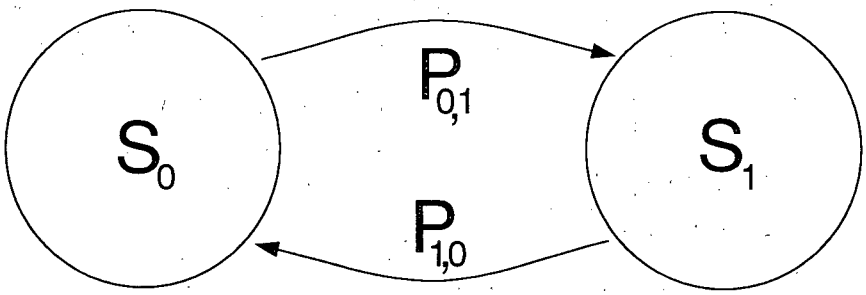

Figure 16 Two-State Markov Chain

probabilities, P_0, P_1, as functions of the state-transition probabilities $P_{i,j}$, can be obtained from the detailed balance equations $P_0 P_{0,1} = P_1 P_{1,0}$.

$$\begin{aligned} P_0 &= \frac{1}{1 + P_{0,1}/P_{1,0}} \\ P_1 &= \frac{1}{1 + P_{1,0}/P_{0,1}} \end{aligned} \tag{4.20}$$

Let BER_0 and BER_1 denote the bit-error rates when the receiver is in state S_0 and S_1, respectively. Then the average bit-error rate is given by

$$\mathrm{BER} = \mathrm{BER}_0 P_0 + \mathrm{BER}_1 P_1. \tag{4.21}$$

We take the word length 1 case, so that BER_0 is the ideal BER given by (2.15) for the $M = 2$ case:

$$\mathrm{BER}_0 = P_\epsilon = \frac{1}{2}\mathrm{erfc}\left(\sqrt{\frac{E_b}{2N_o}}\right). \tag{4.22}$$

The correlator output is down by a factor of approximately 3 dB when the receiver is 1/2 chip off-track; therefore BER_1 is given by the expression for BER_0 with 6 dB less SNR. It remains to determine the state-transition probabilities.

The transition probability from state S_0 to state S_1, $P_{0,1}$, has two components. One is due to the AWGN which can cause the output signal strength in either the early or the late correlator to be greater than that of the on-time correlator, even though the receiver is actually on track. The other is due to a drift of the chip time interval of the transmitted stream relative to the sampling times of the receiver. The latter, for stationary transmitter and receiver, is due to a frequency difference of the transmitter's and receiver's TTL clock oscillator. Let $N_{1/2}$ denote the number of data bits received in the mean time, $t_{1/2}$, that it takes the receiver's clock to drift by one sampling interval relative to the transmitter's clock. The component of $P_{0,1}$ due to the clock drift is equal to $1/N_{1/2}$.

Let Δ denote the percent frequency difference between the two clock oscillators:

$$\Delta = \frac{|f_r - f_t|}{f_r}; \tag{4.23}$$

where f_r and f_t are the receiver and transmitter clock frequencies, respectively. Then

$$N_{1/2} = \frac{1}{2G\Delta}. \tag{4.24}$$

To determine the component of $P_{0,1}$ due to the AWGN, we need the probability, $p_{0,1}$, that the output, r_0, of the on-time correlator corresponding to the received data symbol is less than the output, $r_{1/2}$, of the corresponding correlator which is displaced by 1/2 a chipping interval (one sampling interval) from the received data symbol. To obtain this probability, it is convenient to consider the embedding G-dimensional vector signal space with the orthonormal basis consisting of the normalized maximal length PN sequence and it's $G-1$ code-phase-shifted versions. Let $\vec{s}_0$, $\vec{s}_{1/2}$, and $\vec{s}_1$ denote the on-time, 1/2 chip displaced, and 1 chip displaced signals, respectively. Since all the vector signals have length

$\sqrt{E_s}$, $\vec{s}_o \cdot \vec{s}_1 = 0$, and $\vec{s}_0 \cdot \vec{s}_{1/2} = \vec{s}_1 \cdot \vec{s}_{1/2} = E_s/2$, the signal space in the $\vec{s}_0, \vec{s}_1$ plane is as in Figure 17.

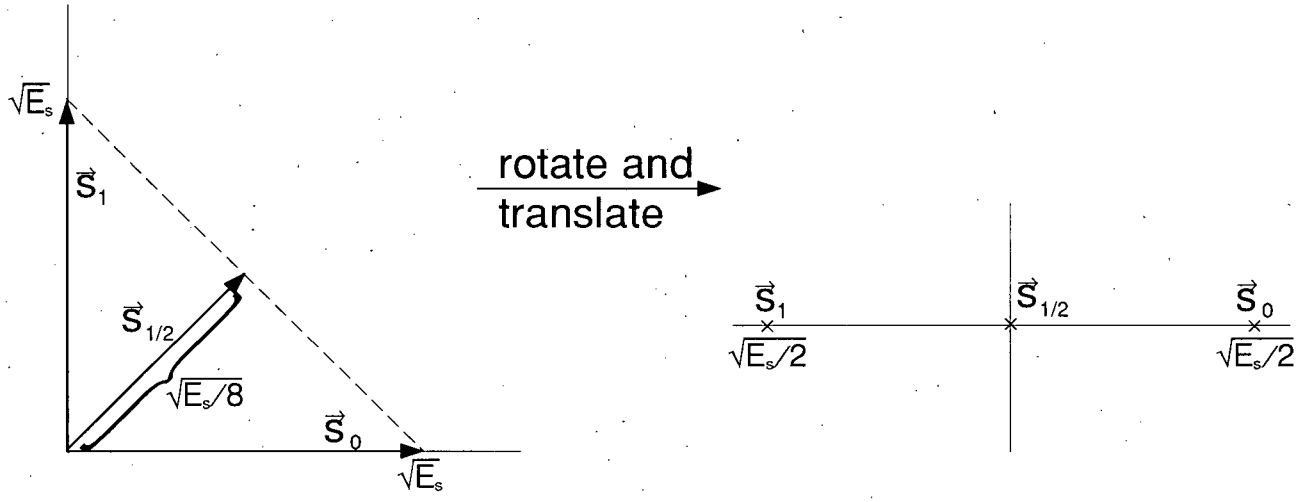

Figure 17 Signal Space for Tracking Performance Calculation

For coherent reception, Figure 17 makes it clear that given that $\vec{s}_0$ is sent, the probability that the vector received is closer to $\vec{s}_{1/2}$ is

$$p_{0,1} = \frac{1}{2}\mathrm{erfc}\left(\sqrt{\frac{E_s}{8N_o}}\right). \tag{4.25}$$

The way that the state-transition probability $P_{0,1}$ is related to $p_{0,1}$ is dependent on the tracking algorithm. Recall that there are two components:

$$P_{0,1} = 2(p_{0,1})^2 + (N_{1/2})^{-1}. \tag{4.26}$$

In the term due to the AWGN, the square is taken because the test for tracking in a particular direction must pass twice in a row on consecutive data bits before the tracking adjustment is made; and the factor of 2 is due to the fact that a tracking error can be made in either direction.

Given that we are in state S_1, to return to state S_0, the output of the correlator synchronized to the signal (an early or late correlator) must exceed the output of the correlator which is half a chip displaced (the correlator which is the on-time correlator when the receiver is on track). The probability of this occurring is $1 - p_{0,1}$; and it must occur twice in a row before the tracker acts, therefore

$$P_{1,0} = (1 - p_{0,1})^2. \tag{4.27}$$

Equations (5.1) through (5.8) give the complete solution to the problem of determining the BER in AWGN with timing error. Figure 18 shows a plot of the BER as a function of the SNR for single-channel, word length 1, coherent reception with a particular choice of parameters. The ideal bit error rate is given by (5.3) and is plotted for comparison. With tracking in the presence of zero timing slippage there is a BER degradation corresponding to 0.35 dB at low SNRs up to about 13 dB, that increases to .5 dB at an SNR of 16 dB. With finite timing slippage, the performance depends on the spreading factor. The curve for the $G = 63$ PN sequence is shown for the case of timing slippage $\Delta = 5 \times 10^{-6}$, and for the case of a timing slippage of $\Delta = 5 \times 10^{-7}$, which is typical between a pair of TTL clock oscillators. Over the relevant SNR range for data transmission, and the above spreading factor, it is not until the BER of 6×10^{-6} for $\Delta = 5 \times 10^{-6}$, and the BER of 2×10^{-7} for $\Delta = 5 \times 10^{-7}$, that the timing slippage induces a performance degradation of 1 dB. At low SNR, the performance degradation, although practically negligible, is due entirely to the tracking errors caused by the noise. At high SNR the performance degradation is due almost entirely to the timing slippage.

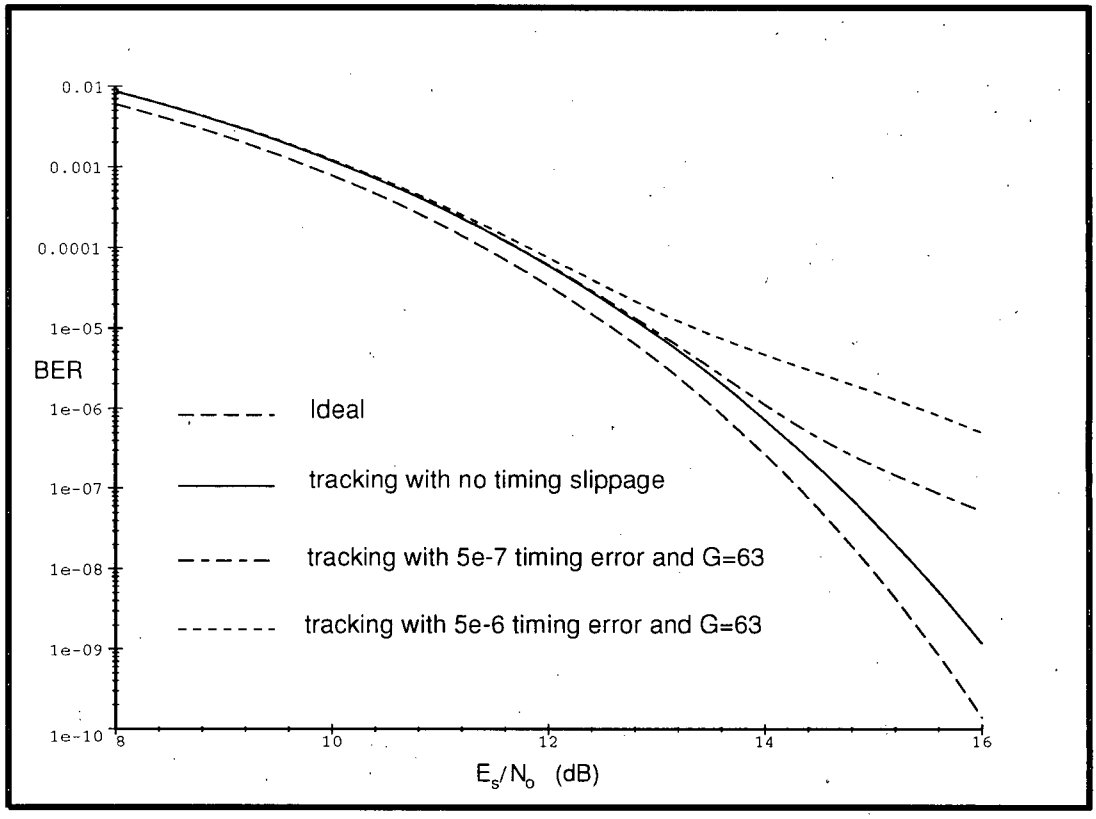

Figure 18 BER Performance in AWGN with Tracking

4.3 Experimental BER in AWGN Performance with Carrier-Wave Synchronization Imposed Externally

The experimental results reported in this chapter were obtained by making a two-way, zero-degree split of the 140MHz Marconi 2031 signal generator output to provide the IF receiver and IF transmitter with the same LO. In addition to providing a test of the acquisition, tracking, and decoding algorithms, the reason for doing this is that it provides the BER performance limit for the present system upgraded to solve the carrier-wave synchronization problem (which we do in the next chapter). The approximation to optimal coherent reception thus far is quantified, allowing us to identify the additional performance degradation from optimal incurred by our solution to the carrier-wave synchronization problem.

We have carefully checked that the BER performance of the word length 1, and word length 2, CPSK transmissions does not depend on the spreading factor G. Each point on the graph of Figure 19 represents the results of 25 measurements; 5 measurements at each

of the 5 spreading factors $G = 7, 15, 31, 63, 127$. For comparison, the BER versus SNR curves for theoretical, optimal reception of M-ary signals for $M = 2$, and $M = 4$ are given on the same graph.

In the BER range from 10^{-3} to 5×10^{-5} the implementation loss is 0.8 dB; of which we identify 0.35 dB as due to the effect of noise on tracking, and 0.45 dB as due to approximating the optimal correlator. As the BER's decrease from 5×10^{-5} to 10^{-7} the loss increases by another 0.2 dB; which we ascribe to a combination of intrinsic receiver noise and timing slippage. The experimental uncertainty on the SNR is approximately ± 0.4 dB overall. The increments between the experimental SNR values have negligible experimental uncertainty. In other words, the shape of the experimental curves have very little uncertainty; but their horizontal placement is uncertain by 0.4 dB.

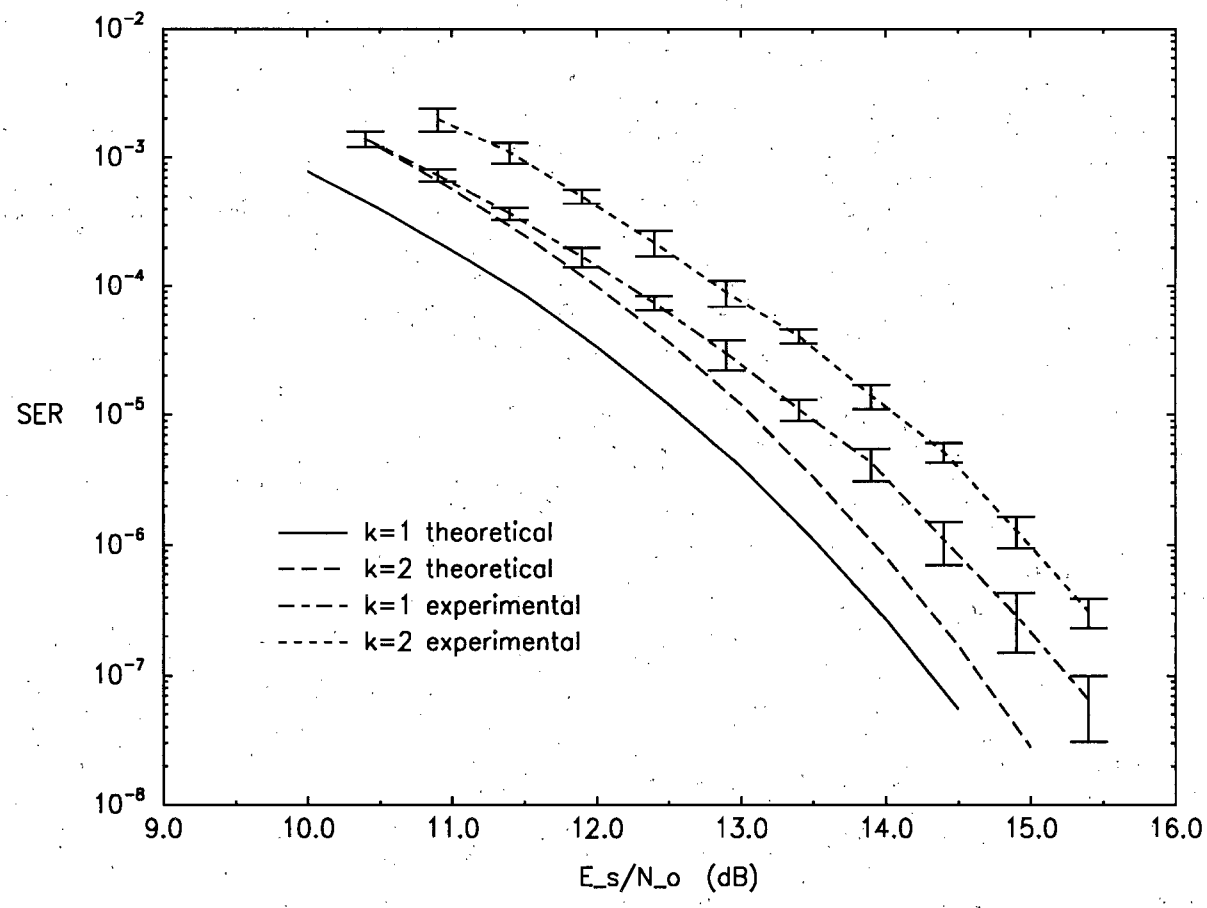

Figure 19 Bit Error Rate vs Signal-to-Noise Ratio for M=2 and M=4 CPSK with Carrier-Wave Synchronization Imposed Externally

Chapter 5 Solving the Carrier-Wave Synchronization Problem by PIR for CPSK

5.1 Phase-Invariant Reception

Phase-Invariant Reception is a technique for demodulating a complex baseband data stream which does not require an estimate of the degree to which the I and Q channels have been rotated into one-another by the down-conversion from RF. It requires that the phase angle between the received complex RF signal and the local oscillator of the receiver be fairly constant over the duration of a data symbol. We first assume it to be exactly constant, and in a later section quantify the SNR degradation when it varies with time.

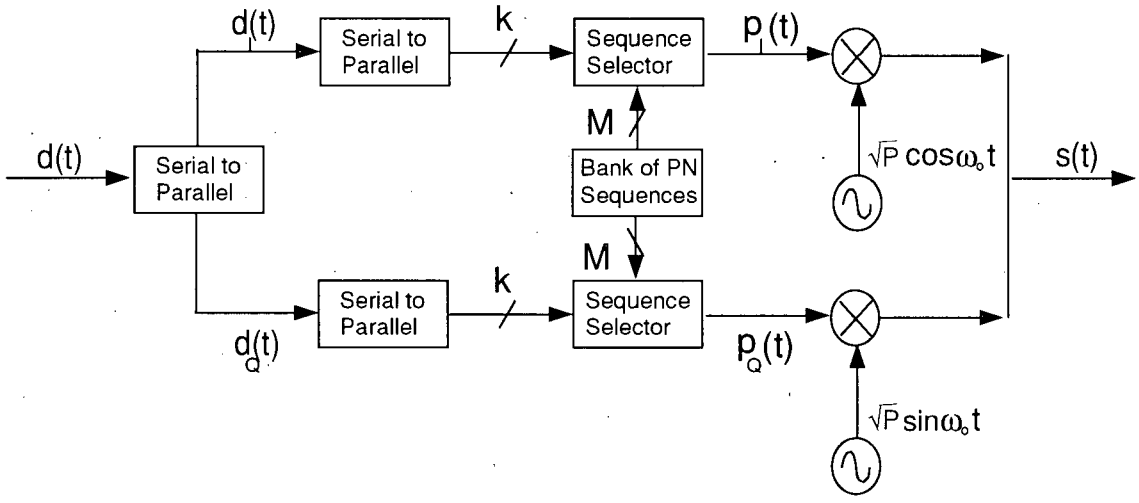

Figure 20 Quadrature CPSK Transmitter

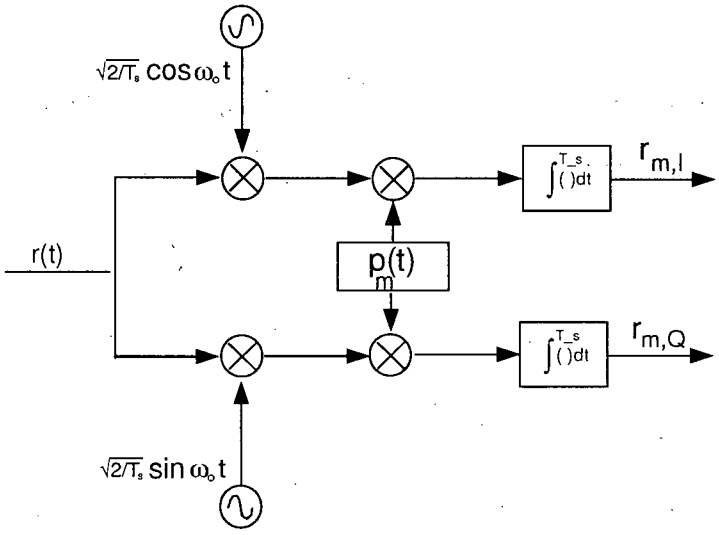

Figure 21 Quadrature Demodulation followed by Complex Correlation

Quadrature RF transmission for CPSK is realized with the transmitter of Figure 20; which is basically two copies, in quadrature, of the transmitter in Figure 2. The corresponding receiver is as in Figure 3, except that the RF demodulation produces a complex signal (with I and Q components) which is fed to a bank of correlators, each of which having two identical arms. Figure 21 shows the complex signal resulting from quadrature RF demodulation, and the ensuing complex correlator.

The signal received at the demodulator now has both I and Q components:

$$s(t) = \sqrt{2P}p_I(t)\cos(\omega_o t + \theta) + \sqrt{2P}p_Q(t)\sin(\omega_o t + \theta); \tag{5.1}$$

where p_I and p_Q are the PN sequences representing the I and Q channel data symbols respectively, and P is the power of the transmission dedicated to a single channel. θ is the phase angle difference between the carrier and the local oscillator of the receiver.

The I and Q channel outputs of the quadrature demodulator, neglecting the double frequency components which get filtered off anyway, are obtained by multiplication by $\sqrt{2/T_s}\cos\omega_o t$, and $\sqrt{2/T_s}\sin\omega_o t$, respectively:

$$I(t) = \frac{\sqrt{E_s}}{T_s}\big(p_I(t)\cos\theta + p_Q(t)\sin\theta\big);$$

$$Q(t) = \frac{\sqrt{E_s}}{T_s}\big(-p_I(t)\sin\theta + p_Q(t)\cos\theta\big). \tag{5.2}$$

Then the complex output, $\vec{r}_m = (r_{m,I}, r_{m,Q})$, of the m'th correlator is formed by multiplication by $p_m(t)$, followed integration over the symbol period:

$$r_{m,I} = \frac{\sqrt{E_s}}{G}\big(c_{m,I}\cos\theta + c_{m,Q}\sin\theta\big);$$

$$r_{m,Q} = \frac{\sqrt{E_s}}{G}\big(-c_{m,I}\sin\theta + c_{m,Q}\cos\theta\big); \tag{5.3}$$

where $c_{m,I}$ ($c_{m,Q}$) is the correlation between the m'th PN sequence and the PN sequence which represents the I (Q) channel data symbol. For the synchronized system, $c_{m,I}$ equals G or -1, depending on whether the I channel carries the m'th data symbol or not, (and similarly for $c_{m,Q}$).

For the time being, consider the word length 1 case, and form the following two quantities which are invariant with respect to the phase rotation θ:

$$\begin{aligned} \mathrm{h} &= \frac{1}{2}\big(r_{0,I}^2 + r_{0,Q}^2 - \big(r_{1,I}^2 + r_{1,Q}^2\big)\big); \\ \mathrm{g} &= -r_{0,I}r_{1,Q} + r_{0,Q}r_{1,I}. \end{aligned} \tag{5.4}$$

2h is the magnitude squared of the 0 correlator minus the magnitude squared of the 1 correlator; while g is the dot product of the 0 correlator with the rotation by 90 degrees of the 1 correlator. It is easy to check that when the (I,Q) dibit is received in the absence of interference and noise, that the invariants take the values shown in the Table 2. Therefore,

(I,Q)	g	h	h-g	g+h
(0,0)	0	+k	+k	+k
(0,1)	-k	0	+k	-k
(1,0)	+k	0	-k	+k
(1,1)	0	-k	-k	-k

Table 2 Decision Variables for Phase-Invariant Decoding

g$-$h and g$+$h are the appropriate decision variables for the I and Q channel bits, respectively.

For higher word lengths, one way of generalizing this scheme would be to first choose the two correlator outputs of the largest squared magnitude, and then apply the described scheme to decide on the I and Q channel data symbols. For single-channel (real) data transmission, the same data symbol would be sent over the I and Q channel, and the decision variable

would simply be h. In this case, the generalization to higher word lengths is immediate: the decision variables are the magnitudes squared of the complex correlator outputs.

The local oscillator of the receiver's IF to baseband demodulator runs freely at its nominal frequency ω_o. With the conventional direct-sequence spread-spectrum demodulator with a free local oscillator, it is necessary to track the phase error because the decision variable for the two data symbols is based on the output of one complex correlator. The two data symbols (corresponding to bit 0 or bit 1) are 180 degree shifts in the I-Q plane of one another. PIR is possible for the CPSK method because different data symbols correspond to different complex correlators.

In the next two sections we derive the BER performance of this technique in AWGN. To show that the decision variables continue to be independent of the phase rotation, it is convenient to represent the noise by the narrow-band representation:

$$n(t) = n_I(t)\cos(\omega_o t + \theta) - n_Q(t)\sin(\omega_o t + \theta). \tag{5.5}$$

The I and Q channel quadrature demodulator outputs become

$$I(t) = \frac{1}{\sqrt{T_s}}\left(\sqrt{\frac{E_s}{T_s}}p_I(t) + \frac{1}{\sqrt{2}}n_I(t)\right)\cos\theta + \left(\sqrt{\frac{E_s}{T_s}}p_Q(t) - \frac{1}{\sqrt{2}}n_Q(t)\right)\sin\theta;$$

$$Q(t) = \frac{-1}{\sqrt{T_s}}\left(\sqrt{\frac{E_s}{T_s}}p_I(t) + \frac{1}{\sqrt{2}}n_I(t)\right)\sin\theta + \left(\sqrt{\frac{E_s}{T_s}}p_Q(t) - \frac{1}{\sqrt{2}}n_Q(t)\right)\cos\theta. \tag{5.6}$$

This form makes it evident that if the decision variables are independent of θ in the absence of noise (which they are), then they are independent of θ in the presence of noise as well.

5.2 BER For Real Data over the Complex Channel with AWGN — Theoretical Analysis and Experimental Results

By real data over the complex channel, we mean quadrature transmission with the same data symbol in each channel. We will first calculate the BER in AWGN for the $M = 2$ case. From this we will obtain a bound on the SER (data symbol error rate) for arbitrary M by means of the union bound for probability. Recall that for Phase-Invariant Reception, the decision variables are the magnitudes of the complex correlators.

In the last section it was shown that for PIR, that the phase angle θ can be taken to be zero; so that for spreading factor, G, much greater than unity, the reception of the zeroth symbol results in the correlator output:

$$\begin{aligned} r_{0,I} &= \sqrt{E_s} + N_{0,I} \\ r_{0,Q} &= \sqrt{E_s} + N_{0,Q} \\ r_{1,I} &= N_{1,I} \\ r_{1,Q} &= N_{1,Q}; \end{aligned} \tag{5.7}$$

where $E_s = E_b/2$, and the noise components are:

$$\begin{aligned} N_{i,I} &= \sqrt{\frac{2}{T_b}} \int_0^{T_b} p_i(t) n(t) \cos \omega_o t \, dt \\ N_{i,Q} &= \sqrt{\frac{2}{T_b}} \int_0^{T_b} p_i(t) n(t) \sin \omega_o t \, dt. \end{aligned} \tag{5.8}$$

These are independent gaussian random variables with zero mean and deviation $\sigma^2 = N_o/2$.

For equally likely transmission of a zero or a one, the probability of error is equal to the probability of error conditioned on the reception of a zero:

$$\begin{aligned} P_\epsilon(2, E_b/N_o) &= \Pr\{|\vec{r}_0| < |\vec{r}_1|\} \\ &= \int_0^\infty f_{|\vec{r}_1|-|\vec{r}_0|}(\alpha) \, d\alpha; \end{aligned} \tag{5.9}$$

where $|\vec{r}_0| = \sqrt{r_{0,I}^2 + r_{0,Q}^2}$, $|\vec{r}_1| = \sqrt{r_{1,I}^2 + r_{1,Q}^2}$, and $f_{|\vec{r}_1|-|\vec{r}_0|}$ is the probability density function for $|\vec{r}_1| - |\vec{r}_0|$.

By going to polar coordinates:

$$\begin{aligned} r_{i,I} &= |\vec{r}_i| \cos \theta_i, \\ r_{i,Q} &= |\vec{r}_i| \sin \theta_i; \end{aligned} \tag{5.10}$$

the square-law transformation for random variables, which gives the probability density for $|\vec{r}_i|$ in terms of the probability density for $\vec{r}_i$, is obtained:

$$f_{|\vec{r}_i|}(\alpha) = \begin{cases} \int_0^{2\pi} \alpha f_{\vec{r}_i}(\alpha \cos\theta, \alpha \sin\theta) \, d\theta, & \text{for } \alpha \geq 0; \\ 0, & \text{for } \alpha < 0. \end{cases} \tag{5.11}$$

Since the probability density functions for $\vec{r}_i$ are:

$$\begin{aligned} f_{\vec{r}_0}(\alpha_1, \alpha_2) &= \frac{1}{\pi N_o} e^{-((\alpha_1 - \sqrt{E_s})^2 + (\alpha_2 - \sqrt{E_s})^2)/N_o}; \\ f_{\vec{r}_1}(\alpha_1, \alpha_2) &= \frac{1}{\pi N_o} e^{-(\alpha_1^2 + \alpha_2^2)/N_o}; \end{aligned} \tag{5.12}$$

the probability densities for $|\vec{r}_i|$ are:

$$\begin{aligned} f_{|\vec{r}_0|}(\alpha) &= \frac{1}{\pi N_o}\int_0^{2\pi} \alpha e^{-((\alpha\cos\theta-\sqrt{E_s})^2+(\alpha\sin\theta-\sqrt{E_s})^2)/N_o}\,d\theta; \\ f_{|\vec{r}_1|}(\alpha) &= \frac{2\alpha}{N_o}e^{-\alpha^2/N_o}. \end{aligned} \tag{5.13}$$

The general result for the difference of two independent random variables, in the present notation, is

$$f_{|\vec{r}_1|-|\vec{r}_0|}(\alpha) = \int_0^{+\infty} f_{|\vec{r}_1|}(\alpha+\beta) f_{|\vec{r}_0|}(\beta)\,d\beta. \tag{5.14}$$

The probability of error is given by (5.13) substituted into (5.14), substituted into (5.9). The resulting triple integral can be reduced to the following single integral:

$$\begin{aligned} P_e(2, E_b/N_o) = \frac{e^{-2E_b/N_o}}{4\pi}\int_0^{2\pi} d\theta\Bigg(1 \\ + \sqrt{\frac{\pi E_b}{2N_o}}e^{E_b(cos\theta+\sin\theta)^2/(2N_o)}\mathrm{erfc}\left(-\sqrt{\frac{E_b}{2N_o}}(\cos\theta+\sin\theta)\right)\Bigg). \end{aligned} \tag{5.15}$$

We have evaluated this integral numerically with Maple V, and plotted the BER ($= P_e$) has a function of the SNR (E_b/N_o). The theoretical curve is shown in Figure 22. For comparison, the curve for coherent reception (the best performance theoretically possible) of two orthogonal signals is given on the same graph. At zero frequency difference, (df = 0), between the carrier wave and the local oscillator of the receiver; at a BER of 10^{-3}, PIR is 1.1 dB less efficient than coherent reception; at a BER of 10^{-8}, PIR is 0.5 dB less efficient than coherent reception. In the limit of infinite SNR, the BER performance in noise of PIR approaches that of coherent reception. The reason that PIR does not do as well as coherent reception, is that the PIR decision variables are quadratics of the decision variables for coherent reception.

The measured curve for df = 0 is shown as well; exhibiting an implementation loss, with respect to theoretical PIR, of 0.8 dB at the BER of 10^{-3}, and a loss of 1.1 dB at the BER of 10^{-6}. The total implementation loss with respect to coherent reception is approximately constant at 1.8 dB for any BER. The BER measurements were obtained by the experimental setup described in chapter 3. Each point represents 20 measurements, all taken with a spreading factor of 63.

In the next section, the performance degradation of a function of df is obtained analytically. Figure 22 also shows the experimental curve at the value of df found to give a signal-to-noise degradation of 1 dB.

The union bound gives an upper limit for the BER, in AWGN, of M-ary orthogonal signaling in terms of the BER for the $M = 2$ case. The symbol error rate, P_ϵ, when all M signals are equally likely to be sent is

$$P_\epsilon(M, E_s/N_o) \leq (M-1)P_\epsilon(2, E_s/N_o). \tag{5.16}$$

In the case of coherent reception, the union bound upper limit becomes equality to within a percent for SERs lower than 10^{-4}. Also, in this SER range, the SER vs. SNR curves (see Figure 4) become practically straight lines (on the linear-log plot). Therefore, for this range of SERs, the power efficiency loss of PIR relative to coherent reception is no greater than the relative loss in the word length 1 case.

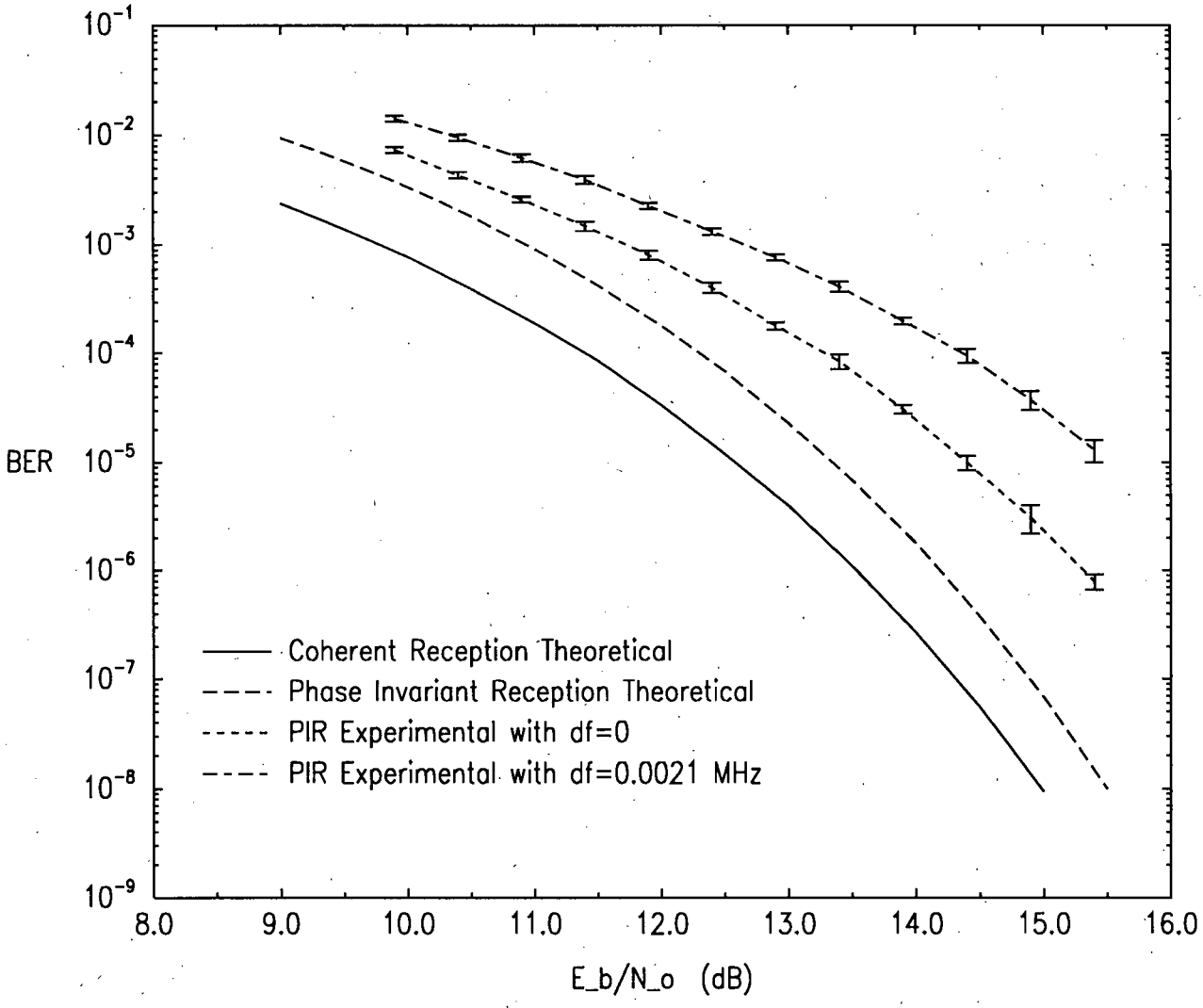

Figure 22 BER vs. Signal to Noise Ratio for PIR with Real Data over Complex Channel, Word Length 1

5.3 BER Performance of PIR for Time Varying Phase

In this section we examine the effective loss in signal-to-noise ratio as a function of the difference in frequency between the carrier-wave and the local oscillator of the receiver for real data transmission.

The quadrature demodulator output is given by (5.2), except that now θ is a function of time. Let $d\omega$ denote the difference in angular frequency between the carrier-wave and

the local oscillator of the demodulator. With $\theta = d\omega\, t$, the I and Q channel outputs of the quadrature demodulator are:

$$\begin{aligned} I(t) &= \frac{\sqrt{E_s}}{T_s} p_m(t)(\cos d\omega t + \sin d\omega t) \\ Q(t) &= \frac{\sqrt{E_s}}{T_s} p_m(t)(\cos d\omega t - \sin d\omega t;) \end{aligned} \tag{5.17}$$

where $p_m(t)$ is the symbol being received.

Let δ denote the percentage of the chipping frequency by which the carrier-wave and local oscillator differ in frequency:

$$d\omega = \delta\omega_c = \frac{2\pi\delta}{T_c}. \tag{5.18}$$

Then, provided that we have PN code alignment, with the integral of the correlator approximated by a sum over chipping times, the output of the mth correlator due to the information-bearing part of the incoming signal is:

$$\begin{aligned} r_{m,I} &= \frac{\sqrt{E_s}}{G} \sum_{i=0}^{G-1} (\cos(2\pi i\delta) + \sin(2\pi i\delta)) \\ r_{m,Q} &= \frac{\sqrt{E_s}}{G} \sum_{i=0}^{G-1} (\cos(2\pi i\delta) - \sin(2\pi i\delta)). \end{aligned} \tag{5.19}$$

Therefore, the magnitude squared of the mth correlator (equal to the effective bit energy) is

$$r_{m,I}^2 + r_{m,Q}^2 = E_b k(G,\delta); \tag{5.20}$$

where

$$k(G,\delta) = \frac{1}{G^2} \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} \cos(2\pi(i-j)\delta). \tag{5.21}$$

$k(G,\delta)$ is the ratio of the signal-to-noise ratio that the decision device uses, SNR_o, to the signal-to-noise ratio of the signal at the receiver front end, SNR.

$$k(G,\delta) = \frac{\mathrm{SNR}_o}{\mathrm{SNR}}. \tag{5.22}$$

Figure 23 shows a plot of k (in dB) as a function of δ for the case $G = 63$. The curve shows that the output SNR is down from the received SNR by 1 dB at the value of delta equal to 4×10^{-3}. We have confirmed this using fslove of Maple V. At a chipping rate of 0.5 MHz (the rate supported by the implementation), this corresponds to a frequency difference of

df=0.0021 MHz. We have also measured the BER at this frequency difference between the stationary transmitter's LO and stationary receiver's LO. The experimental curve is shown in Figure 22 of the previous section. To within experimental uncertainty, the BER degradation is indeed 1 dB.

The frequency difference corresponding to a 1 dB degradation is 15 ppm. Commercial oscillators are available at 140 MHz that are accurate to within 15 ppm. For example, the Raltron VCOHF series 6700.

The degradation of the SNR with increasing δ here presented is not unique to Phase-Invariant Reception. Other systems we have found [9] which demodulate the carrier with a local oscillator at the nominal carrier frequency (ie. not a phase-locked loop), estimate the phase rotation θ, and rotate the complex demodulator output so as to remove the phase rotation from the decision variables. However, all such systems which only update the phase rotation estimate at the data rate, and not at the chipping rate, (these include the implementations [9] and [10]), suffer precisely the effective SNR loss herein calculated.

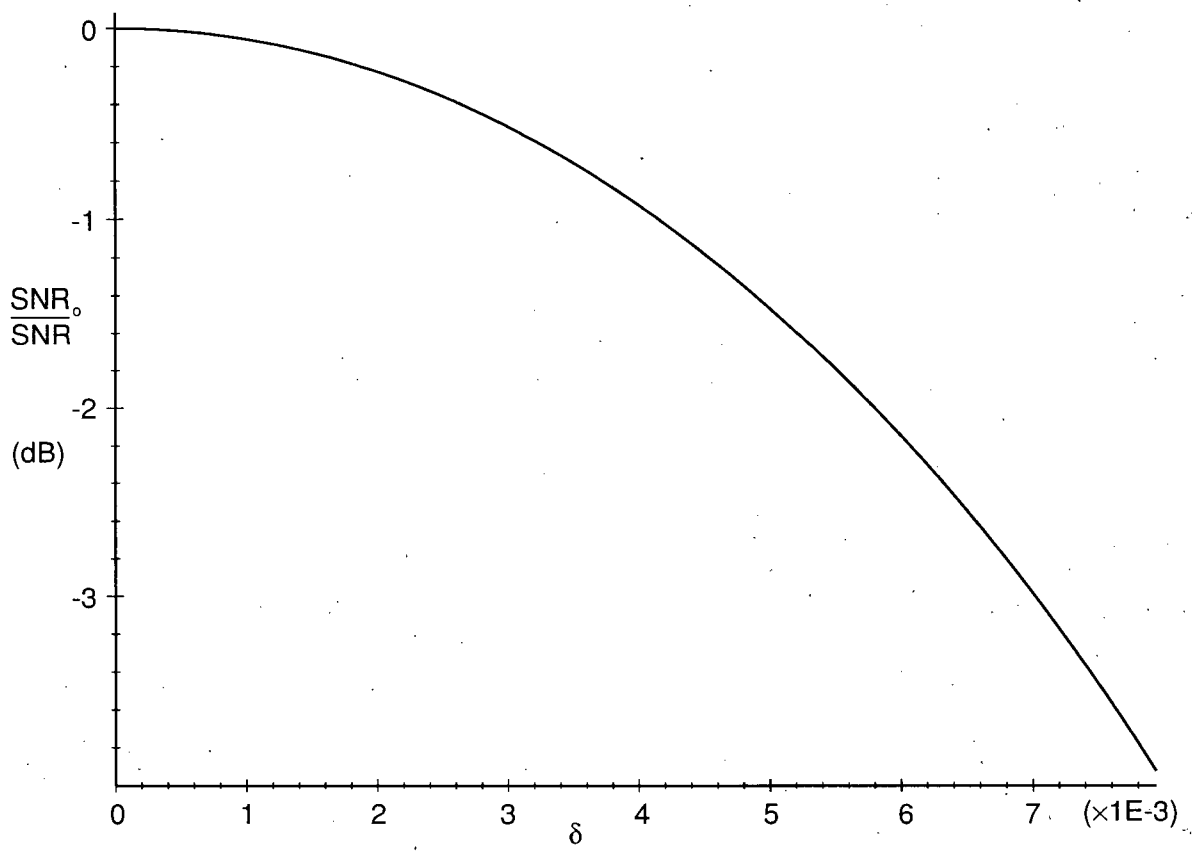

Figure 23 Output SNR As a Function of Rx and Tx LO Frequency Difference $\delta = \mathrm{df}/\mathrm{f}_c$

5.4 BER For Complex Data Flow with AWGN

As of this writing, we have not obtained an analytical expression for the BER in AWGN for PIR with independent I and Q channel data streams. Thus we rely on the experimental curve, and compare it to the theoretical curve for coherent reception for the case in which each channel carries one of two possible orthogonal signals. We refer to the data encoding for this scenario as complex word length 1 encoding. First we outline the as of yet to be solved problem for the theoretical BER.

Suppose the dibit (0,0) is transmitted. Referring to the decoding table of figure (23), one sees that it is correctly decoded iff $\mathrm{g} - \mathrm{h} \leq 0$ and $\mathrm{g} + \mathrm{h} \geq 0$; that is iff $-\mathrm{h} \leq \mathrm{g} \leq +\mathrm{h}$. The probability, P_c, of this occurring is

$$P_c = \int_{\alpha=0}^{+\infty} d\alpha \int_{\beta=-\alpha}^{+\alpha} d\beta \, f_{\mathrm{h}}(\alpha) f_{\mathrm{g}}(\beta|\mathrm{h} = \alpha); \tag{5.23}$$

where $f_{\mathrm{h}}(\alpha)$ is the pdf for h, and $f_{\mathrm{g}}(\beta|\mathrm{h}=\alpha)$ is the pdf for g conditioned on h. h and g are given by (5.4).

One approach to deriving the conditional pdf for g is to transform to polar coordinates, (5.10), as was done to obtain the square-law transformation (5.11). The random variable g conditioned on h becomes $g|_{h=\alpha} = |\vec{r}_1|\sqrt{\alpha + |\vec{r}_1|^2}\sin(\theta_0 - \theta_1)$. Derivation of a pdf for this function requires finding the inverse of the function $f(r) = r\sqrt{\alpha + r^2}$ — a problem we have not yet solved. However, we do have the measured BER vs. SNR curve from which to evaluate the noise performance.

The theoretical performance upper limit for quadrature transmission of two independent data streams, each of which carries orthogonal symbols from an alphabet of length 2, is given by coherent reception. In this case, the I-channel decision is independent of the Q-channel decision. The probability of a correct decision in one channel is $1 - P_\epsilon$; with P_ϵ given by (4.22). The probability of a correct decision for a symbol representing an (I,Q) dibit is $(1 - P_\epsilon)^2$. Therefore, the symbol error rate is

$$\mathrm{SER} = 1 - (1 - P_\epsilon)^2. \tag{5.24}$$

Figure 24 shows the experimental SER has function of the SNR for PIR of complex data flow in AWGN; and the theoretical SER for coherent reception from the same transmission. To achieve an SER of 10^{-3} requires 3.0 dB more signal power for experimental PIR than for theoretical coherent reception. At an SER of 10^{-6}, the loss is 3.4 dB. Provided that the implementation loss with respect to theoretical PIR is the same as the implementation loss for real data flow over the complex channel (Figure 22), we deduce that PIR is approximately 2.2 dB less power efficient than coherent reception for quadrature data transmission. Over the same SER range, for real data flow over the complex channel, PIR is approximately 0.9 dB less efficient than coherent reception (see Section 7.2). With coherent reception, going from single-channel to dual-channel transmission entails doubling the data rate and doubling the transmission power and achieving the same BER. With PIR, doubling the data rate by going to complex data flow at a constant BER requires approximately another 1.3 dB of transmitter power in addition to the 3 dB required when using coherent reception for quadrature data transmission.

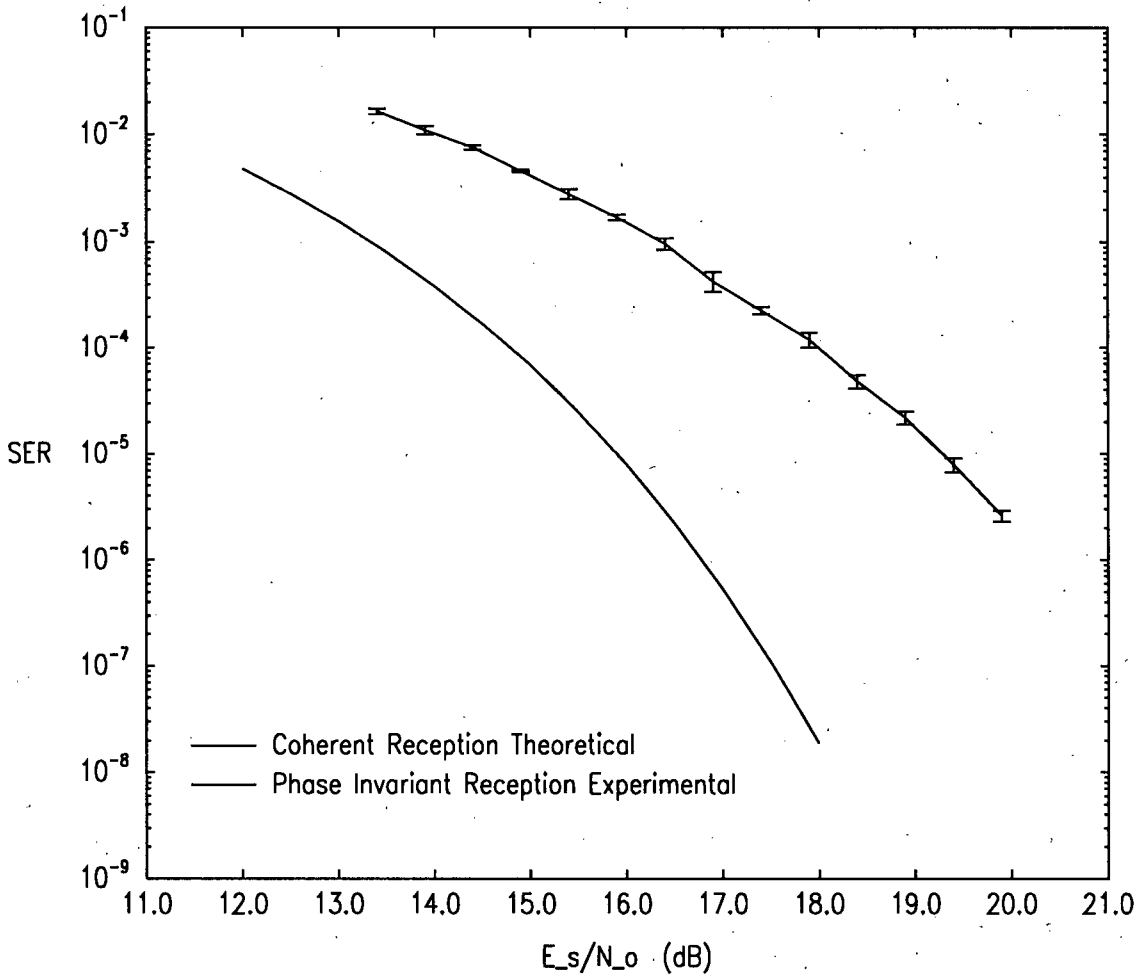

Figure 24 Symbol Error Rate vs Signal-to-Noise Ratio for Complex Word Length 1 Data in AWGN

Chapter 6 The BER Performance of CPSK in the Presence of Single-Tone Interference

6.1 Theoretical Analysis for Coherent Reception

In [2], a general expression for the probability of error in AWGN and single tone interference is derived. This provided the basis for simulations of the performance for specific choices of spreading sequence, word length and jammer parameters. Unfortunately, the PN sequence length there chosen for the simulations is longer than can be accommodated by the hardware of the receiver that we have implemented. To allow a quantitative comparison between theory and experiment, in this section we derive a numerical expression for the word length 1, PN sequence length 63 case.

Consider a single-tone jammer of angular frequency $\omega = \omega_o + \delta\omega$, with power P_J, at phase angle ϕ with respect to the information signal:

$$j(t) = \sqrt{2P_J}\cos(\omega t + \phi). \tag{6.1}$$

For convenience, the amount by which the angular frequency of the jammer differs from that of the signal carrier is written as a percentage of the angular chipping frequency:

$$\delta\omega = \varrho\omega_c = \frac{2\pi\varrho}{T_c}. \tag{6.2}$$

$\delta\omega$ will be chosen low enough so that the time integral of (2.12) for the interference terms of the receiver decision variables is well approximated by a summation over chipping times. Then (2.12) for the received interference, upon substitution of (8.1) for the interference, becomes:

$$J_m = \sqrt{E_J}u(\varrho, \phi, m); \tag{6.3}$$

where we have defined

$$u(\varrho, \phi, m) = \frac{1}{G}\sum_{i=0}^{G-1} p_m(iT_c)\cos(2\pi\varrho i + \phi), \tag{6.4}$$

and $E_J = P_J T_s$ to be the power of the jammer integrated over a symbol period.

In the word length 1 case, the probability of bit error in AWGN is:

$$P_e = \frac{1}{2}\mathrm{erfc}\left(\sqrt{\frac{E_b}{2N_o}}\right). \tag{6.5}$$

Because the interference terms are additive constants to the decision variables, it is clear that the probability of error conditioned on ϕ, conditioned on the reception of a 0 bit, is given by P_ϵ with $\sqrt{E_b}$ replaced by $\sqrt{E_b}+\sqrt{E_J}(u(\varrho,\phi,0)-u(\varrho,\phi,1))$; and therefore, with $\sqrt{E_b/N_o}$ replaced by $\sqrt{\mathrm{SNR}}(1+\sqrt{\mathrm{JSR}}(u(\varrho,\phi,0)-u(\varrho,\phi,1)))$; where $\mathrm{SNR}=E_b/N_o$ is the signal to noise ratio, and $\mathrm{JSR}=E_J/E_b$ is the jammer to signal ratio.

The function u can be written (next paragraph) as

$$u(\varrho,\phi,m)=\gamma(\varrho,m)\cos\phi, \tag{6.6}$$

so that the probability of error conditioned on ϕ, for equally likely transmission of 0 or 1, is

$$\begin{aligned}P_\epsilon[\phi]=\frac{1}{4}\Bigg(&\mathrm{erfc}\Bigg(\frac{1}{\sqrt{2}}\sqrt{\mathrm{SNR}}\Big(1+\sqrt{\mathrm{JSR}}(\gamma(\varrho,0)-\gamma(\varrho,1))\cos\phi\Big)\Bigg)\\&+\mathrm{erfc}\Bigg(\frac{1}{\sqrt{2}}\sqrt{\mathrm{SNR}}\Big(1+\sqrt{\mathrm{JSR}}(\gamma(\varrho,1)-\gamma(\varrho,0))\cos\phi\Big)\Bigg)\Bigg);\end{aligned} \tag{6.7}$$

where the first and second terms are due to the probability of bit error conditioned on the reception of a 0 or 1 respectively. The average bit error rate is

$$\mathrm{BER}=\frac{1}{2\pi}\int_{\phi=0}^{\phi=2\pi}P_\epsilon[\phi]\,d\phi. \tag{6.8}$$

To obtain the function γ, write (6.4) as

$$u(\varrho,\phi,m)=c(\varrho,m)\cos\phi-d(\varrho,m)\sin\phi; \tag{6.9}$$

with

$$\begin{aligned}c(\varrho,m)&=\frac{1}{G}\sum_{i=0}^{G-1}p_m(iT_c)\cos(2\pi\varrho i)\\ d(\varrho,m)&=\frac{1}{G}\sum_{i=0}^{G-1}p_m(iT_c)\sin(2\pi\varrho i);\end{aligned} \tag{6.10}$$

and redefine ϕ by an additive constant to obtain (6.6) with

$$\gamma(\varrho,m)=\sqrt{c^2(\varrho,m)+d^2(\varrho,m)}. \tag{6.11}$$

It is easily shown that

$$\gamma(\varrho,m)=\frac{1}{G}\left(\sum_{l=1-G}^{G-1}c_m(l)\cos(2\pi\varrho l)\right)^{1/2}; \tag{6.12}$$

where $c_m(l)$ is the aperiodic autocorrelation of the PN code sequence $p_m(i)=p(i-m)$.

We have written a C program which evaluates γ, and find that for a maximal length PN sequence with $G = 63$ and with $\varrho = 0.1$ that

$$\gamma(0.1, 0) = 0.143 \qquad \gamma(0.1, 1) = 0.048. \tag{6.13}$$

Figure 25 of the next section shows a plot of the BER versus JSR at an SNR equal to 12.0 dB, as given by (6.8). There we also give a set of measured points, for the same parameter values, for comparison.

6.2 Theoretical Analysis and Experimental Results for PIR

In this section we derive an analytic expression for the BER as a function of JSR and SNR with phase-invariant reception, for the same spreading and coding parameters ($G = 63$, $k = 1$) as the previous section. The intensity of computation prohibits a numerical evaluation of the resulting expression; so we rely on the measured curve to evaluate the performance.

With real word length 1 data flow over the complex channel in the presence of the single-tone interference as described by (6.1) and (6.2), the m'th correlator output, $\vec{r}_m$, has both I and Q channel interference components:

$$\begin{aligned} J_{m,I} &= \sqrt{E_J}u(\varrho, \phi, m), \\ J_{m,Q} &= -\sqrt{E_J}v(\varrho, \phi, m); \end{aligned} \tag{6.14}$$

where the functions u and v are defined by:

$$\begin{aligned} u(\varrho, \phi, m) &= \frac{1}{G}\sum_{i=0}^{G-1} p_m(iT_c)\cos(2\pi\varrho i + \phi), \\ v(\varrho, \phi, m) &= \frac{1}{G}\sum_{i=0}^{G-1} p_m(iT_c)\sin(2\pi\varrho i + \phi); \end{aligned} \tag{6.15}$$

with parameters ϱ and ϕ as defined in the previous section. We showed there that u can be written as per (6.6). v admits a similar representation:

$$\begin{aligned} u(\varrho, \phi, m) &= \gamma(\varrho, m)\cos\phi, \\ v(\varrho, \phi, m) &= \gamma(\varrho, m)\sin\phi; \end{aligned} \tag{6.16}$$

with γ given by (6.12).

The probability density functions for the decision variables $|\vec{r}_i|$ are obtained by the square-law transformation (5.11). Conditioned on jammer phase angle ϕ, and on the reception of the symbol representing a 0 bit, these are:

$$\begin{aligned} f_{|\vec{r}_0|}(\alpha) &= \frac{\alpha}{\pi N_o} \int_{\theta=0}^{2\pi} e^{\frac{-1}{N_o}((\alpha\cos\theta-(\sqrt{E_s}+\sqrt{E_J}\gamma_0\cos\phi))^2+(\alpha\sin\theta-(\sqrt{E_s}-\sqrt{E_J}\gamma_0\sin\phi))^2)}d\theta; \\ f_{|\vec{r}_1|}(\alpha) &= \frac{\alpha}{\pi N_o} \int_{0}^{2\pi} e^{\frac{-1}{N_o}((\alpha\cos\theta-\sqrt{E_J}\gamma_1\cos\phi)^2+(\alpha\sin\theta+\sqrt{E_J}\gamma_1\sin\phi)^2)}d\theta; \end{aligned} \tag{6.17}$$

where $\gamma_i = \gamma(\varrho, i)$. With these, the probability of error, conditioned on ϕ, and on the reception of a zero, is as given by (5.9) and (5.14):

$$P_\epsilon(\phi|0) = \int_0^\infty \int_0^\infty f_{|\vec{r}_1|}(\alpha+\beta) f_{|\vec{r}_0|}(\beta)\, d\alpha d\beta. \tag{6.18}$$

The probability of error conditioned on the reception of a one, $P_\epsilon(\phi|1)$, is given by (6.18) and (6.17) with γ_0 and γ_1 interchanged. Then, for equally likely reception of a zero or a one, the probability of error is

$$P_\epsilon = \frac{1}{4\pi} \int_0^{2\pi} (P_\epsilon(\phi|0) + P_\epsilon(\phi|1))\, d\phi. \tag{6.19}$$

This quintuple integral is too heavy to evaluate numerically to any useful degree of accuracy in any reasonable amount of time.

Figure 25 shows the measured curve corresponding to (6.19) at an SNR of 12 dB. The corresponding curve, derived in the previous section, for theoretical coherent reception is shown for comparison. The BER (theoretical optimal or experimental PIR) at JSR = -10 dB is only about 10 percent higher than in the absence of a jammer (JSR = $-\infty$ dB). The implementation loss at 12 dB SNR of PIR with respect to coherent reception, is approximately 1.8 dB; meaning that the experimental curve at a 1.8 dB higher SNR than that of the exhibited curve, roughly coincides with the theoretical curve shown. From the graph, it is evident that it is not until the jammer strength reaches the signal strength (JSR = 0 dB) that the BER starts to degrade significantly.

As for coherent reception, at $\delta\omega = 0$ the jammer does not contribute to the BER because the decision variables are all contributed to equally. At $\delta\omega = \infty$, the jammer does not contribute either because of the low-pass filtering performed to form the decision variables. The most pessimistic value of ϱ is dependent on the PN sequence used, but is near the value chosen ($\varrho = 0.1$) for any maximal length PN sequence.

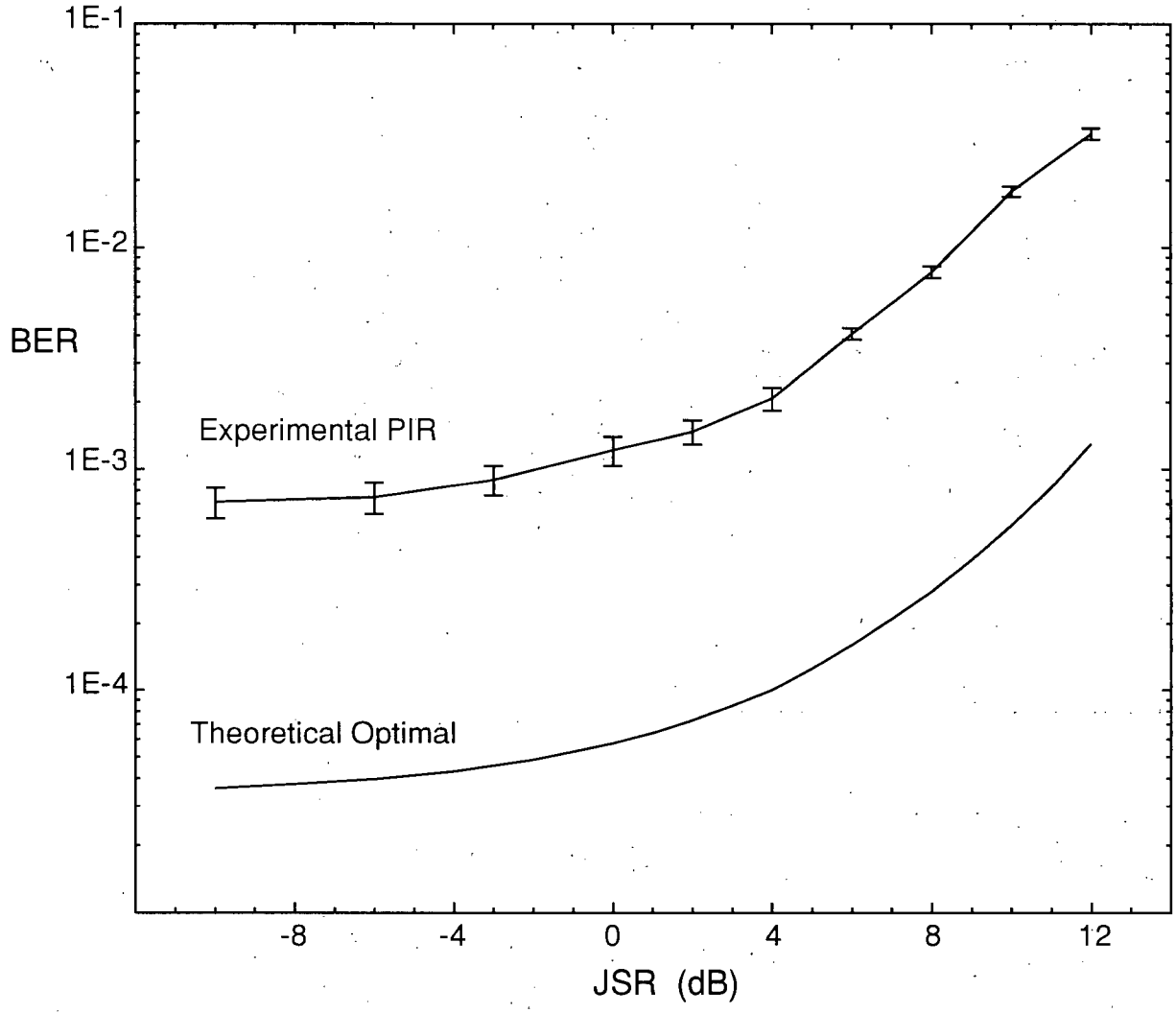

Figure 25 Theoretical and Experimental BER versus JSR at SNR=12.0 dB with $\delta\omega = 0.1\omega_c$

Chapter 7 Tracking and Acquisition in AWGN

In this chapter, we outline the analysis required to analytically predict the mean time-to-acquire and the mean time-to-lose-lock, in additive white gaussian noise, that the CPSK receiver achieves. The experimental curves of these quantities as functions of the signal-to-noise ratio are presented. They determine the minimum length of preamble, and the maximum length of body that should be used for packet transmission.

7.1 Mean Time to Acquire

Refer to Figure 9 of chapter 3 for the PN code acquisition algorithm. For M-ary CPSK the length, measured in number of chips, of an elementary code-phase shift, m_e, is

$$m_e = \frac{G+1}{M}. \tag{7.1}$$

The acquisition algorithm uses M data correlators, spaced by m_e chips, to search for lock by one chip increments.

At the end of a data symbol period, if the data correlator with the maximum output has an output which crosses $threshold1$, the maximum of it and the output of the corresponding early and late correlators is compared to $threshold2$. Let P_{s1} denote the probability of exceeding these two thresholds (or passing stage 1) given that the code is in fact aligned to within 1/2 chip. If stage 1 is not passed, the data correlators are precessed for the next attempt to pass stage 1. If stage 1 is passed, the output of the correlator which was largest in stage 1 is checked again on another data read, along with it's early and late counterparts. If the maximum of these three exceeds $threshold3$, the signal is declared to be acquired. Let P_{s2} denote the probability of passing stage 2, given rough (to within 1/2 chip) code alignment.

Let n_i denote the average number of attempts required to pass stage i when the code is aligned to within 1/2 chip. Then

$$n_i = \frac{1}{P_{si}} \qquad \text{for} \quad i = 1, 2. \tag{7.2}$$

Since the probability of passing stage 2 is independent of the probability of passing stage 1, the average number of attempts required to pass both stages is $n_1 n_2$. Provided that the thresholds are set such that the probability of false acquisition is negligible, attempts to pass stage 1, given that the code is aligned to within 1/2 chip, occur at m_e data symbol intervals.

Each failed attempt to pass stage 1, with code alignment to within 1/2 chip, costs m_e data symbol periods; while the subsequent attempt to pass stage 2 costs 1 data symbol period. Then since 1/2 chip code alignment first occurs, on average, after $(m_e + 1)/2$ data symbols, the mean time to acquire, measured in number of data symbols is given by

$$\overline{A} = (n_1 n_2 - 1)m_e + n_2 + \frac{m_e + 1}{2}. \tag{7.3}$$

In the absence of noise, this reduces to

$$\overline{A}_{ideal} = \frac{m_e + 1}{2} + 1. \tag{7.4}$$

The pdf for the output magnitude of the punctual complex correlator is given by the first of (5.13) for exact code alignment; and given by the same equation with $\sqrt{E_s}$ replaced by $\sqrt{E_s}/2$ for 1/2 chip-off code alignment. Obtaining P_{si} however, requires the joint pdf of the latter two — something which we have not yet been able to obtain. To derive the mean time-to-acquire when the probability of false acquisition is non-negligible, one must go to a Markov chain analysis, similar to that in [11].

Let the acquisition thresholds be written as fractions of the complex correlator magnitude for the noiseless reception of the corresponding data symbol:

$$thresholdi = c_i\sqrt{E_s}. \tag{7.5}$$

Then with $c_1 = 1/16, \quad c_2 = c_3 = 1/4$, and $G = 63$, we have found the mean time-to-acquire as a function of signal-to-noise ratio, as shown in Figure 26, for the experimental arrangement described in chapter 3. Each data point represents 20 measurements. The error bars represent one standard deviation due to the noise induced variance. A false acquisition was observed less than 1 out of 100 trials during these measurements. However, not enough measurements were made to collect meaningful false acquisition statistics.

Note that the ideal mean time-to-acquire reaches the value $\overline{A}_{ideal}$ given by (7.4) at an SNR of 13 dB. At SNRs greater than or equal to his value, a data packet with a preamble length of $l_{ideal} = ((G + 1)/M) + 1$ will be blocked with negligible probability. Assuming that the time-to-acquire is approximately gaussian, a preamble length, l_n, which will result in a blocking probability which is upper bounded by the probability of a normalized gaussian process falling n standard deviations outside of its mean is given by

$$l_n = \overline{A} + n\sigma - \overline{A}_{ideal} + l_{ideal}; \tag{7.6}$$

where $\overline{A}$ and σ are to be read from the graph of Figure 26.

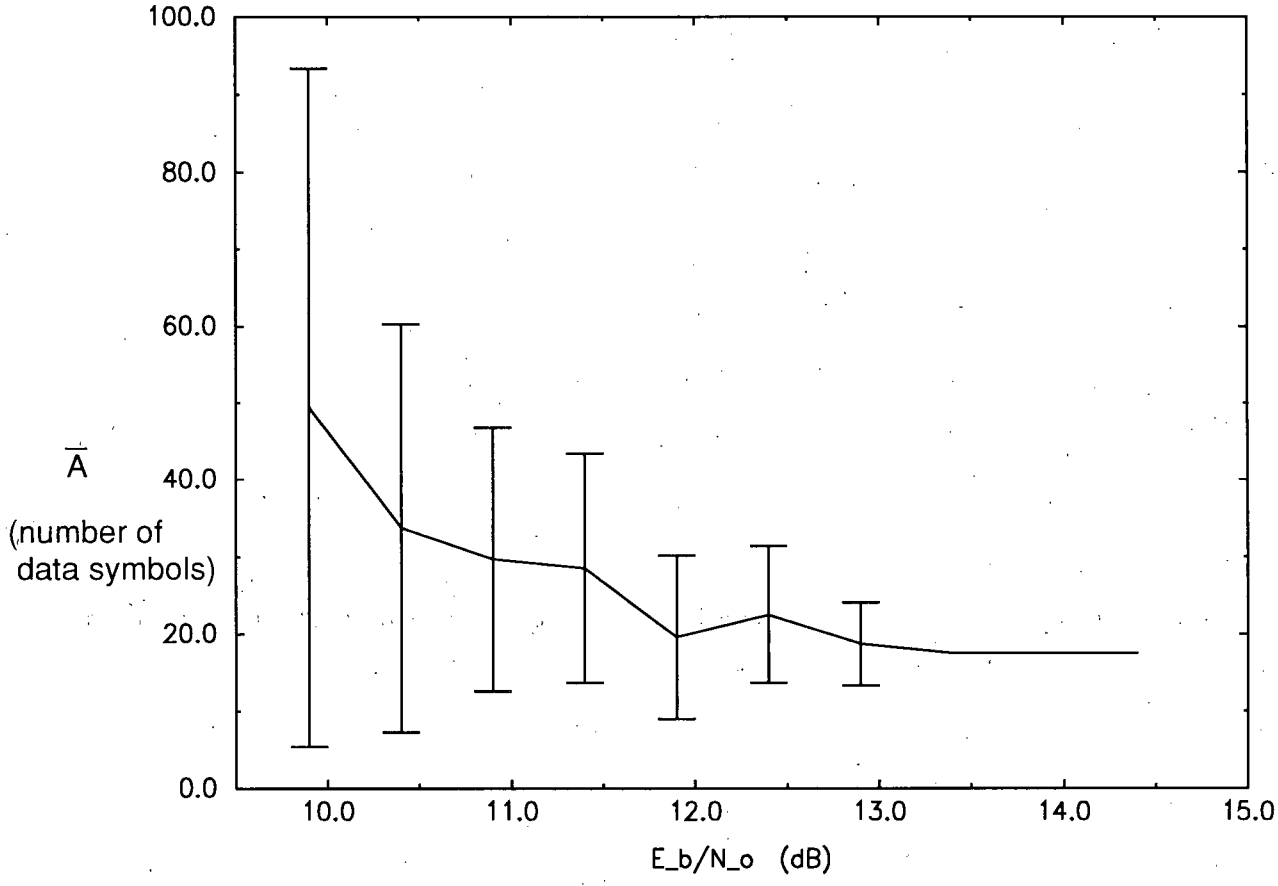

Figure 26 Mean Time-to-Acquire Measured in Data Symbol Durations vs SNR for G=63

7.2 Mean Time to Lose Lock

The mean time-to-lose-lock can be analysed by means of Markov chain theory. Let s_1 be the state for which the tracking loss flag is down; s_2 be the state for which the tracking loss flag is raised; and s_3 be the state representing a loss of lock. Let p denote the probability that the maximum of the outputs of the punctual, early, and late correlator corresponding to the received data symbol is below $threshold4$. Then the receiver makes one state transition per data symbol period in the Markov chain shown in Figure 27 with state transition probabilities

$$P_{1,1} = P_{2,1} = 1 - p, \quad P_{1,2} = P_{2,3} = p, \quad P_{3,3} = 1.$$

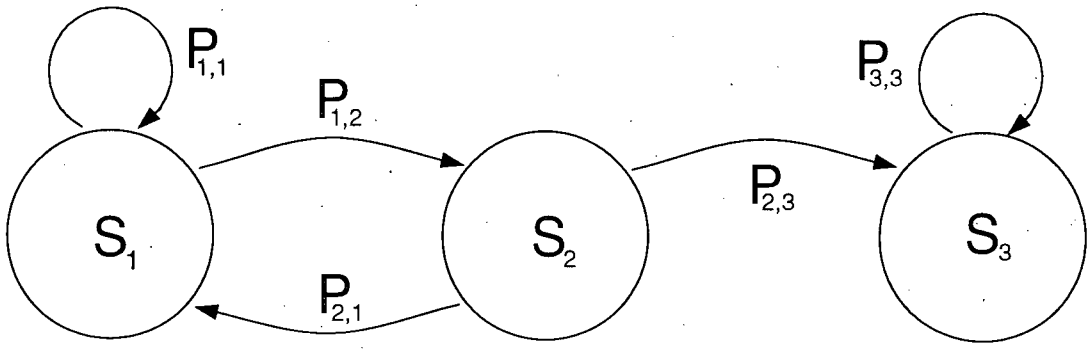

Figure 27 Markov Chain for Tracking Loss Calculation

The states s_1 and s_2 are the transient states, and state s_3 is an absorbing state. By the theory of finite Markov chains [12], the mean number of times, $n_i(s_j)$, that the process which started in transient state s_i is in a transient state s_j, is given by

$$\{n_i(s_j)\} = (I - Q)^{-1}; \tag{7.7}$$

where $Q = \{P_{i,j}\}$ is the transition probability matrix for the transient states, and I is the identity matrix. For the case at hand, since we start in state s_1, the mean time-to-lose-lock is given by

$$\overline{TL} = n_1(s_1) + n_1(s_2) = \frac{p+1}{p^2}. \tag{7.8}$$

As in the previous section, computation of p requires the joint pdf of the magnitudes of a complex correlator and its 1/2 chip displaced counterparts. Furthermore, it also depends on the symbol error probability, and on the degree to which the receiver is actually on track. To take into account the latter, the states need another set of quantum numbers corresponding to whether the receiver is on, 1 sample off, 2 samples off, or greater than 2 samples off track. Taking into account timing slippage, would also require specifying the direction of the off-track states.

With the tracking loss threshold $c_4 = 1/16$, as defined by (7.5), and a spreading factor of $G = 63$, we have found the mean time-to-lose-lock as a function of signal-to-noise ratio by experiment. It is only independent of the spreading factor for zero timing slippage. Each data point of the graph shown in Figure 28 represents 10 measurements. Since the error bars represent 1 standard deviation, provided that the process is approximately gaussian, the

maximum packet body length for nonblocked complete reception with $n\sigma$ confidence can be read from the graph.

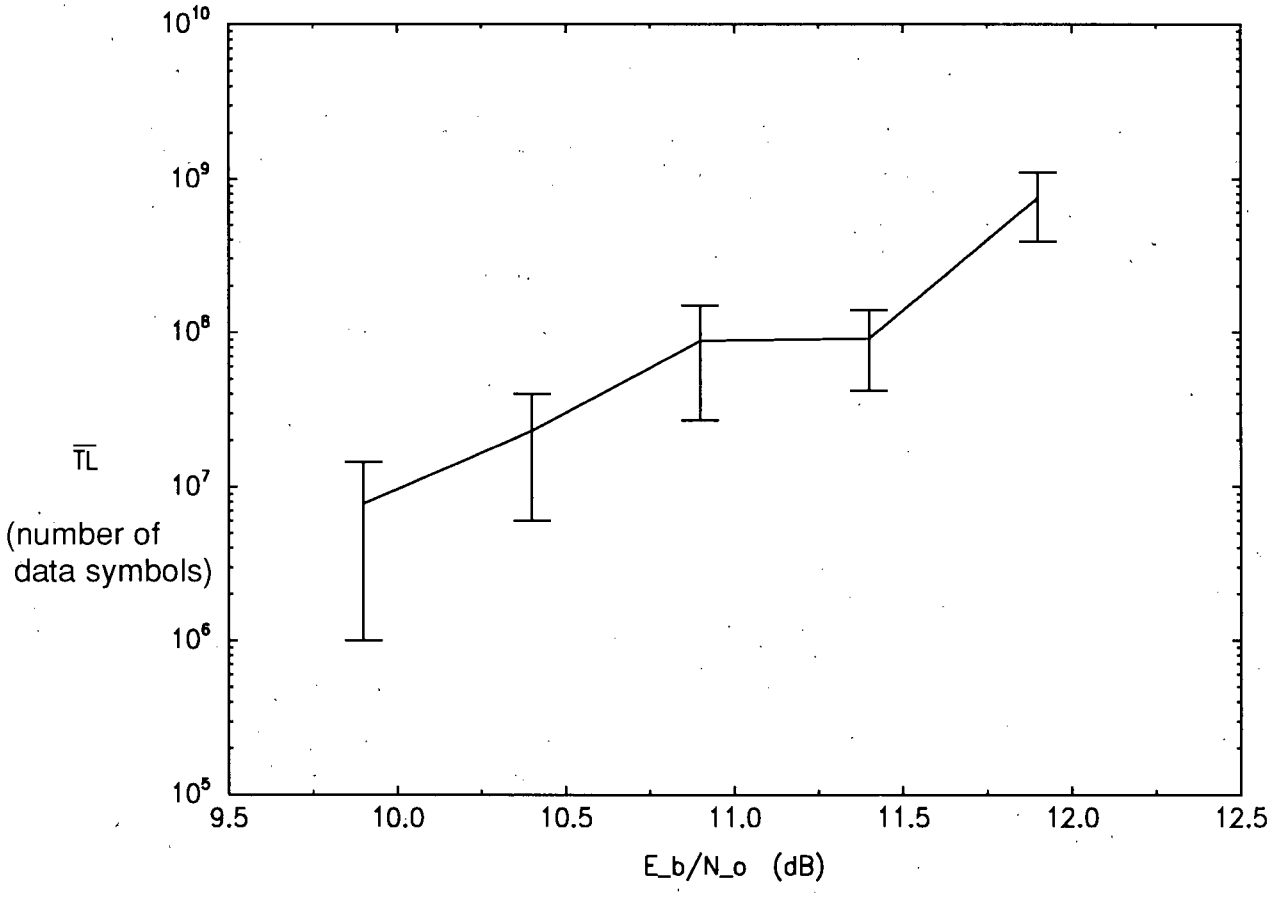

Figure 28 Mean Time-to-Lose-Lock Measured in Number of Data Symbols vs. SNR

Chapter 8 Conclusions

The design and implementation of a correlation receiver for CPSK has been presented. This involved solving the spreading code and carrier-wave synchronization problems. The steady state behaviour of the receiver has been investigated exhaustively in AWGN and single-tone interference; and the previous theoretical BER performance claims by the inventors of CPSK have been experimentally verfied. Beyond this, the present author has found the following.

When sampling at twice the chipping rate, applying a first order RC filter of 3–dB bandwidth equal to 1/2 the chipping rate, to the baseband signal before sampling, optimizes the SNR of the correlated signal to a value which is within 1 dB of the output SNR of an ideal matched filter. This result is not specific to CPSK — in the DS-SS literature, the LPF bandwidth is as here (for example, in [13]); but we have not been able to find a publication of the SNR optimization calculation.

The author's herein described tracking algorithm has been found to be very satisfactory. For coherent reception, we have found by theoretical analysis, that in AWGN with a timing slippage typical of a TTL oscillator, that the signal is kept on track well enough to incurr a BER performance degradation of less than 0.4 dB for SNRs less than about 14 dB.

The hereby invented Phase-Invariant-Reception method for CPSK for solving the carrier-wave synchronization problem eliminates the need for a phase-lock loop, or for a phase estimation DSP filter, by forming receiver decision variables that do not depend on the phase difference between the carrier-wave and the receiver's LO. This method is well suited for DSP because of the minimal amount of post-correlation processing that it requires. For single-channel data transmission and zero frequency difference between the carrier and the receiver LO, at a BER of 10^{-3}, PIR is 1.1 dB less power efficient than coherent reception; at a BER of 10^{-8} this loss of power efficiency goes down to 0.5 dB. This loss increases independently of the SNR as a function of the frequency difference according to the graph shown in Figure 23. For a 140 MHz carrier, commercial oscillators exist with frequency tolerances that would keep this additional degradation within 1 dB. The degradation due to non-zero frequency difference could be minimized by the addition of a frequency lock loop. The remaining implementation loss with respect to coherent reception can not be recovered.

This loss should serve as a bench mark for DSP filters which remove the phase rotation so as to allow coherent reception.

More work needs to be done on the dynamic aspects of CPSK reception. The acquisition algorithm presented is very simple and reasonably fast. The threshold values to use in AWGN were experimentally determined by lowering them from some high nominal value (thus shortening the acquisition time) until the probability of false acquisition started to become non-negligible. The resulting threshold values need to be fine tuned, either by more experimentation or by finding reasonably exact analytic expressions for the acquisition time and for the probability of false acquisition. This would allow tradeoff considerations between preamble length and acquisition failure rate under different SNR and JSR values to be analysed.

The algorithm for checking for loss of lock should be improved from the present 2 state algorithm to a 3 (or higher) state algorithm so as to improve the time to lose lock. Such an algorithm would make the loss of lock more immune to bit errors. Again either by more experimentation, or by solving the theoretical problems described in chapter 9, the best algorithm should be determined and its thresholds optimized. After this, a higher level performance evaluation which optimizes data throughput with respect to packet length needs to be done so that the modem can be used effectively in a computer network.

Bibliography

[1] Robert C. Dixon. *Spread Spectrum Systems with Commercial Applications*. Wiley-Interscience, 1994.

[2] Aries Y.C. Wong. Analysis of techniques to enhance the performance of direct sequence spread spectrum signaling for wireless data communications. Master's thesis, University of British Columbia, 1995.

[3] M.K. Simon. *Spread Spectrum Communications, Vols. 1,2,3*. Computer Science Press, 1985.

[4] J.M. Wozencraft and I.M. Jacobs. *Principles of Communication Engineering*. John Wiley & Sons, 1965.

[5] Spectrum Signal Processing Inc. *Quad C40 Processor Board User's Guide*, June 1993.

[6] Texas Instruments. *TMS320C4x User's Guide*, 1993.

[7] 3L Ltd. *Parallel C User Guide*, 1994.

[8] Chris Bowick. *RF Circuit Design*. Indianapolis, Ind. : H.W. Sams, 1982.

[9] Hansen Wang. *Spread Spectrum Modem IF Receiver, Technical Manual, Board Revision 1.0*, June 1995.

[10]K. Beeler and H. Kaufmann. Time integrating correlator for real-time processing of spread-spectrum signals. *Proc. Custom Integrat. Circ. Conf.*, May 1990.

[11]Unisys. *PA-100 Spread Spectrum Demodulator, Technical Data Sheet and User's Guide*, June 1993.

[12]Stanford Telecom. *STEL-2000A, Digital, Fast Acquisition, Spread Spectrum Burst Processor*, 1994.

[13]V.C.M. Leung and R.W. Donaldson. Confidence estimtes for acquisition times and hold-in times for pn-ssma synchronizer employing envelope correlation. *IEEE Trans. Comm.*, COM-26, Jan. 1982.

[14]J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. D. Van Nostrand Company, Inc., 1960.

[15]A.L. Welti and B. Bobrovsky. On optimal agc structure for direct sequence spread spectrum pn-code tracking. *IEEE Trans. Comm.*, 42(2/3/4), 1994.

Appendix A Maximal Length PN Sequence Generation

The Maximal Length Pseudo-Noise Sequences are the longest codes that can be generated by a shift register of a given number of stages l. Their length is

$$G = 2^l - 1. \tag{A1}$$

Figure 29 shows the 3–stage shift register with the appropriate linear feedback connections, for generating the $G = 7$ PN sequence; while Figure 30 shows the autocorrelation function, as given by (2.8) for the l-sequence.

In the implementation, the non-shifted PN sequence is chosen to be the one generated by initializing all the stages of the generating register with 1. The PN sequence is software generated and written to the DSP's memory during system initialization.

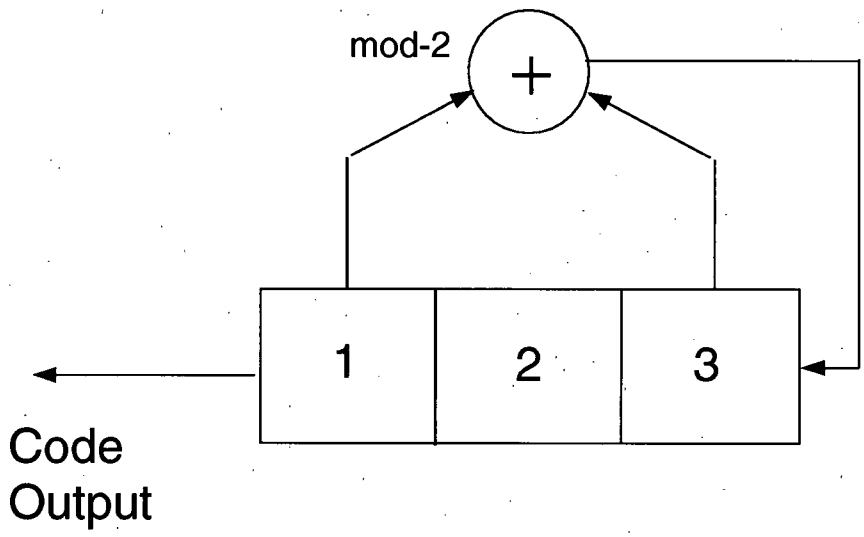

Figure 29 Three-Stage Maximal Generator

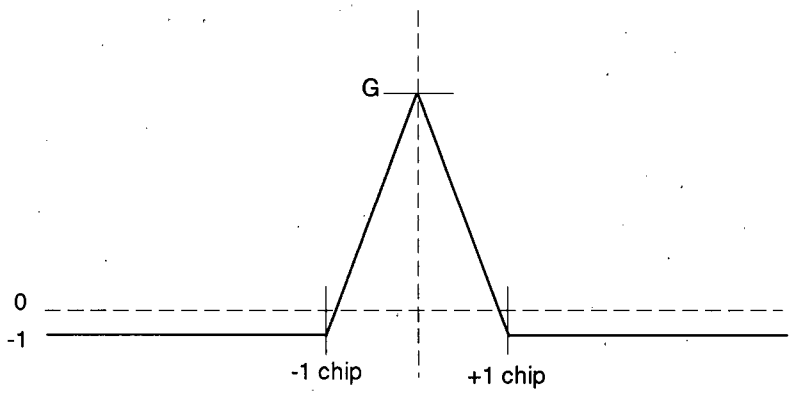

Figure 30 1-Sequence Autocorrelation Function

Appendix B IF Transmitter Modules

Each section of this appendix discusses and gives a schematic of a module of the IF transmitter shown in Figure 6.

B.1 Digital Communication Port Interface

The digital interface reads from the C40 communication port via four control signals: /CREQ, /CACK, /CSRTB, /CRDY. Of these, /CREQ and /CACK are for determining ownership of the data bus; and for unidirectional data flow, these can simply be tied high, provided that the C40 communication port used defaults to an output port at system reset. A sending C40 activates /CSTRB to indicate that it has placed a data byte on the data bus; while a receiving C40 activates /CRDY to indicate that it has received a data byte. The IF transmitter's digital interface (call it Tx) mimics the C40 communication port protocol in such a manner that the data flows according to the interfaces' 4 MHz clock. This is achieved with a D flip-flop which controls both the /CSTRB /CRDY handshaking and the clock signal for a following D flip-flop which reads the data line. See Figure 31 for the digital interface circuit schematic.

The logic elements are implemented by the FAST logic family; and the propagation delays and C40 response times are such that the circuit is guaranteed to function with a clock rate of up to 5.25 MHz. With a 4 MHz clock, the available, software controlled, chipping rates are divisions by 2 of 4 Mbps.

Because the C40 communication port is very high-speed, signal quality is very important. The value for the serial impedance matching resistor (39Ω) was determined by experimentally minimizing the ringing of the output /CRDY signal. The 10 kΩ pullup resistors are used to avoid unintended triggering after reset.

Figure 32 shows the circuit appropriate for a single channel system. In actuality, both I and Q channels are present. Because the first /CSTRB for the two channels is only asserted by the C40 at the same time to within some uncertainty, it is possible for the two channels to be skewed by one Tx clock period. To avoid this and ensure that the two channels are synchronized, the RESET signal of the I-channel control flip-flop is routed to RESET of the Q-channel control flip-flop, and vice-versa.

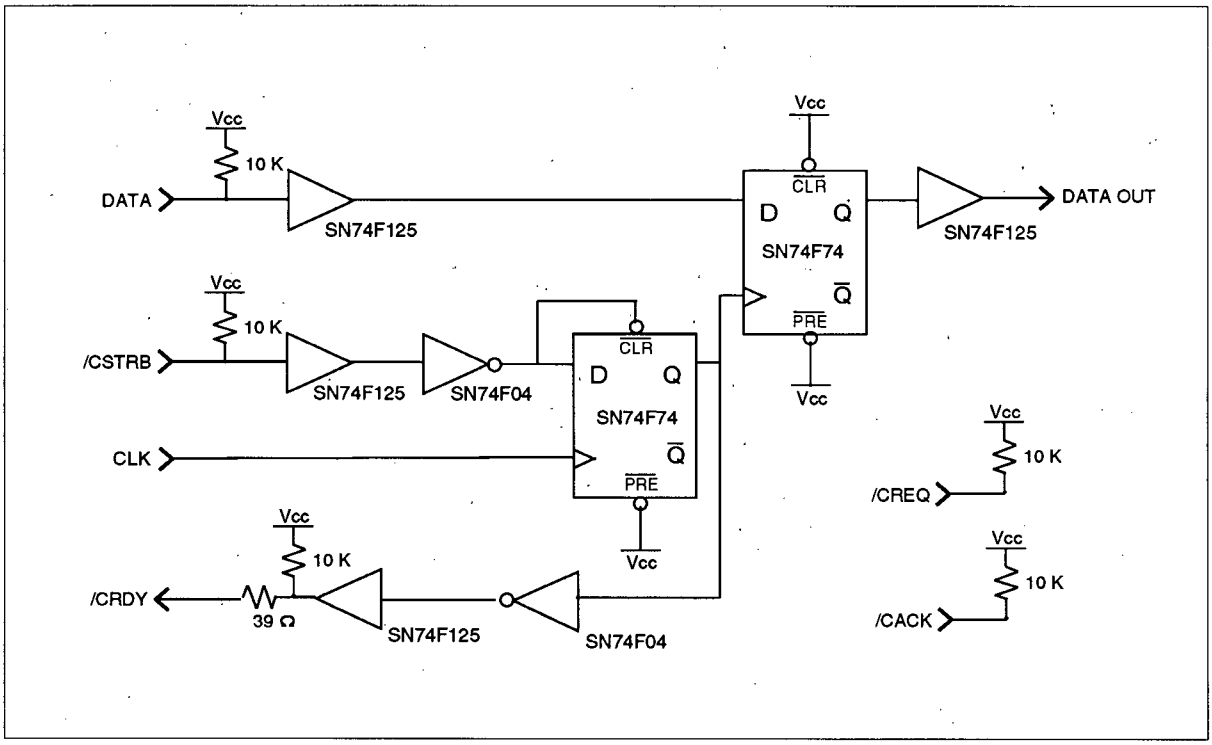

Figure 31 Communication Port Interface (one channel)

B.2 Level Shifter and Amplifier

The level shifter/amplifier converts the TTL logic valued data stream to a bipolar data stream appropriate for BPSK modulation. The circuit uses the high frequency operational amplifier AD9631 to implement a difference amplifier in which one input is the TTL signal, and the other input is a constant voltage signal held at a point midway between TTL high and TTL low. The constant voltage signal is obtained by a 20 kΩ voltage divider inserted between ground and the +5 volt supply, followed by a unity-gain buffer implemented by the operational amplifier LM741. The buffer holds the voltage divider output, V_{ref}, constant (up to a small jitter which is then taken out by a capacitor between ground and the buffer output) in spite of the large changes in the current drawn by the difference amplifier. Two Zener diodes are used to clip off the overshoot and the undershoot of the TTL input signal, V_{in}, to the difference amplifier. See Figure 32 for the circuit schematic.

The output voltage of the difference amplifier is given by

$$V_{out} = \frac{1 + R_2/R_1}{1 + R_3/R_4} V_{in} - \frac{R_2}{R_1} V_{ref}. \qquad (B1)$$

The resistors R_2 and R_4 are both chosen to be $150\Omega \pm 10\%$, while the variable resistors R_1 and R_3 are adjusted so as to make $R_2/R_1 = R_4/R_3 = A$. Then the output voltage is given by

$$V_{out} = A(V_{in} - V_{ref}). \qquad (B2)$$

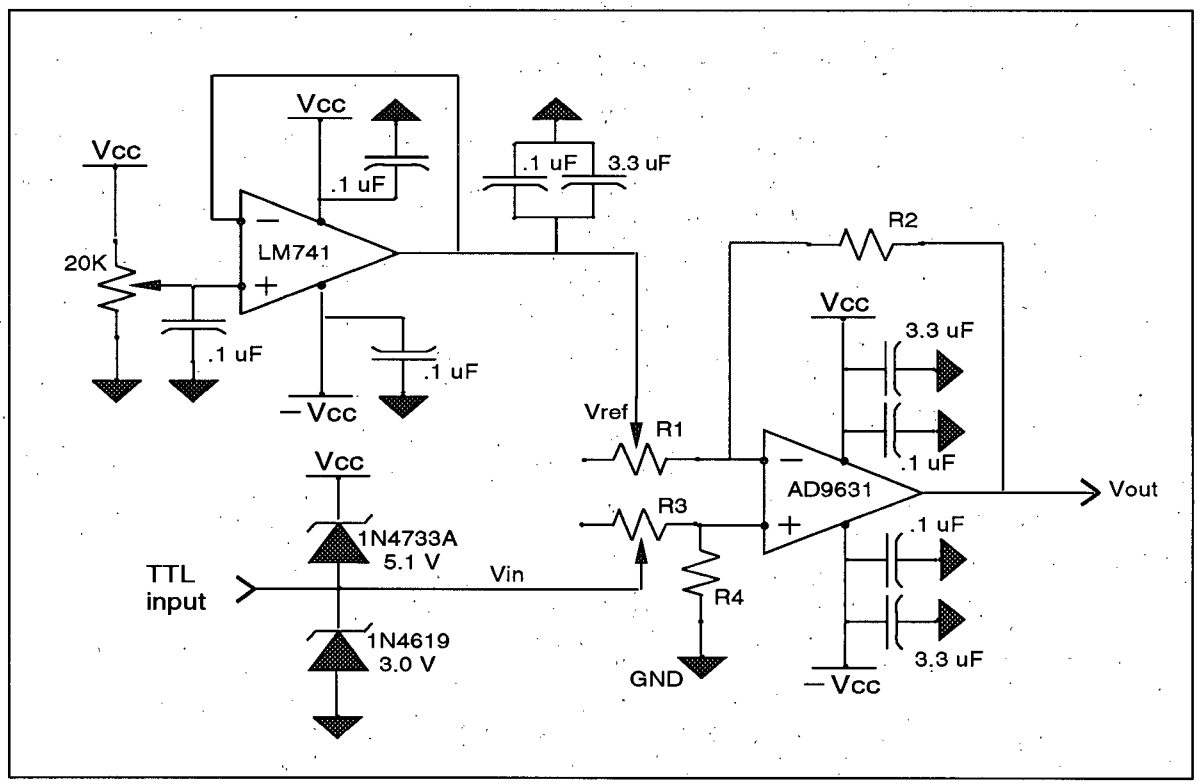

Figure 32 Level Shifter and Amplifier

B.3 BPSK Modulator and Bandpass Filter

The quadrature BPSK modulator performs double-sideband suppressed-carrier modulation with an RF carrier of frequency 140 MHz in both I and Q channels. It is implemented by Mini-Circuits MIQY-140M, whose characteristics are as follows:

— input and output impedance of 50 ohms
— operating LO frequency and power: 140 MHz at 10 dBm
— maximum I and Q current: 40 mA (2 V peak-to-peak)
— bandwidth: 10 MHz
— conversion loss: 6 dB
— carrier suppression: 30 dB

The RF modulator output is bandpass filtered to remove the image spectra at frequencies harmonic to 140 MHz. See Figure 33 for the modulator plus filter circuit schematic.

The filter consists of two identical cascaded stages whose circuit elements are determined by the formulae:

$$\begin{aligned} L_s &= \frac{\sqrt{2}R_o}{\omega_2 - \omega_1} & C_s &= \frac{1}{\omega_o^2 L_s} \\ C_p &= \frac{\sqrt{2}}{R_o(\omega_2 - \omega_1)} & L_p &= \frac{1}{\omega_o^2 C_p}; \end{aligned} \tag{B3}$$

where $R_o = 50\Omega$ is the characteristic impedance of the filter input and output, ω_o is the angular frequency band-center, and $\omega_2 - \omega_1$ is the angular frequency 3–dB bandwidth. With a band-center of 140 MHz and a bandwidth of 134 MHz, these are:

$$\begin{aligned} L_s &= 84nH & C_s &= 15pF \\ C_p &= 34pF & L_p &= 39nH. \end{aligned} \tag{B4}$$

The inductors where wound with 20 AWG tinned copper wire according to the following formula for an air-core, single-layer solenoid:

$$L = Fn^2 d; \tag{B5}$$

where n is the number of turns, d is the diameter of the coil, and F is a constant that depends on the ratio of the length to the diameter of the coil.

The extremely small values for the inductances make the circuit element parameter values hard to control — as the lumped parameter model is barely appropriate. The actual constructed filter ended up with a bandwidth of approximately 100 MHz about a band-center of 120 MHz. The 20 MHz band about the desired 140 MHz has a passband ripple of less than 1 dB about zero attenuation, so the filter is satisfactory.

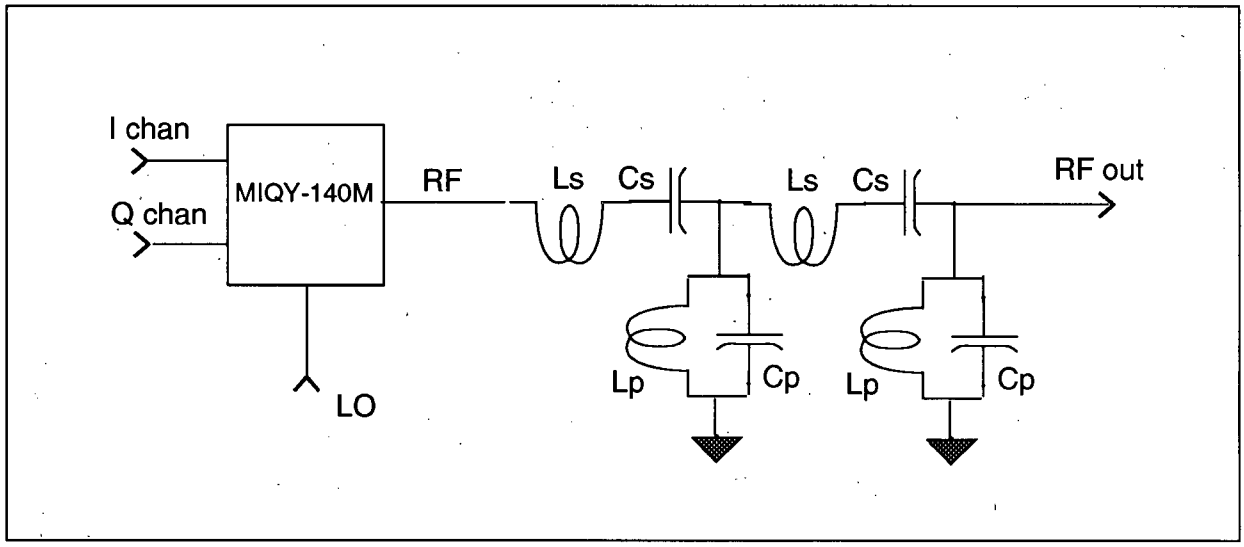

Figure 33 BPSK Modulator and Filter

Appendix C IF Receiver Modules

Each section of this appendix discusses and gives a schematic of a module of the IF receiver shown in Figure 7.

C.1 BPSK Demodulator and Low Pass Filter

A 3 dB pad improves the impedance matching between the BNC connector for the incoming IF signal and the QPSK demodulator, which is implemented by Mini-Circuits MIQY-140D, and is of the following specifications:

— input and output impedance of 50 ohms

— operating LO frequency and power: 140 MHz at 10 dBm

— maximum RF input power: 50 mW (1.6 V peak-to-peak)

— bandwidth: 10 MHz

— conversion loss: 5.6 dB

— maximum output current: 40 mA

The active low pass filter and amplifier has an overall gain of 28 dB equally distributed over 2 stages, each of which is implemented with Analog Device's high frequency AD843JN op amp in the basic inverting configuration. One of the stages implements the first order low-pass filter by a parallel RC feedback. The values of R and C were chosen so as to give the filter the 3–dB cut-off frequency of 0.25 MHz, as called for in Chapter 4.

The voltage offset adjustment, implemented with the LM741CN op amp, is necessary to shift the bipolar baseband signal to be centered about 1 volt, because that is the middle of the ADC range. The circuit includes a trim pot to precisely adjust the offset.

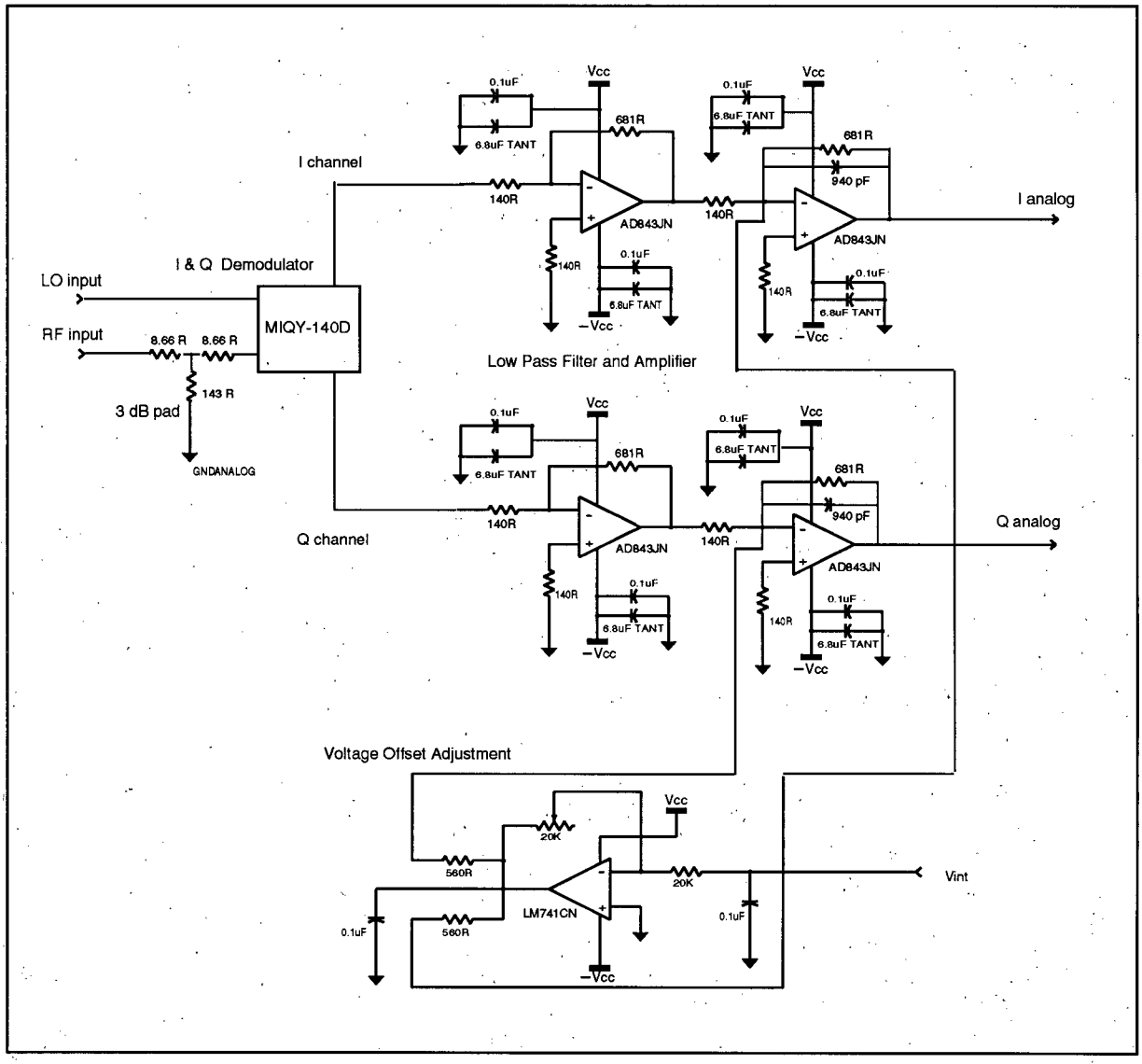

Figure 34 BPSK Demodulator and Low Pass Filter

C.2 Analog/Digital Converter

The I and Q signals are sampled on the rising edge of the ENCODE signal by Analog Devices AD9058JD Dual-Channel 8–Bit, 50 MSPS Flash ADC. It has an encode propagation time delay of 12 ns, and an input aperture time of 0.8 ns. It provides a +2 volt reference output, V_{int}, which is used by the voltage offset adjustment of the previous section to bring the LPF output signal into the center of the 0 to 2 volt analog input range of the AD9058JD.

Pull-down resistors are on the TTL output lines to approximately equalize the TTL high/low rise and fall times. The two Octal D Latches (SN74F373) latch the ADC output data.

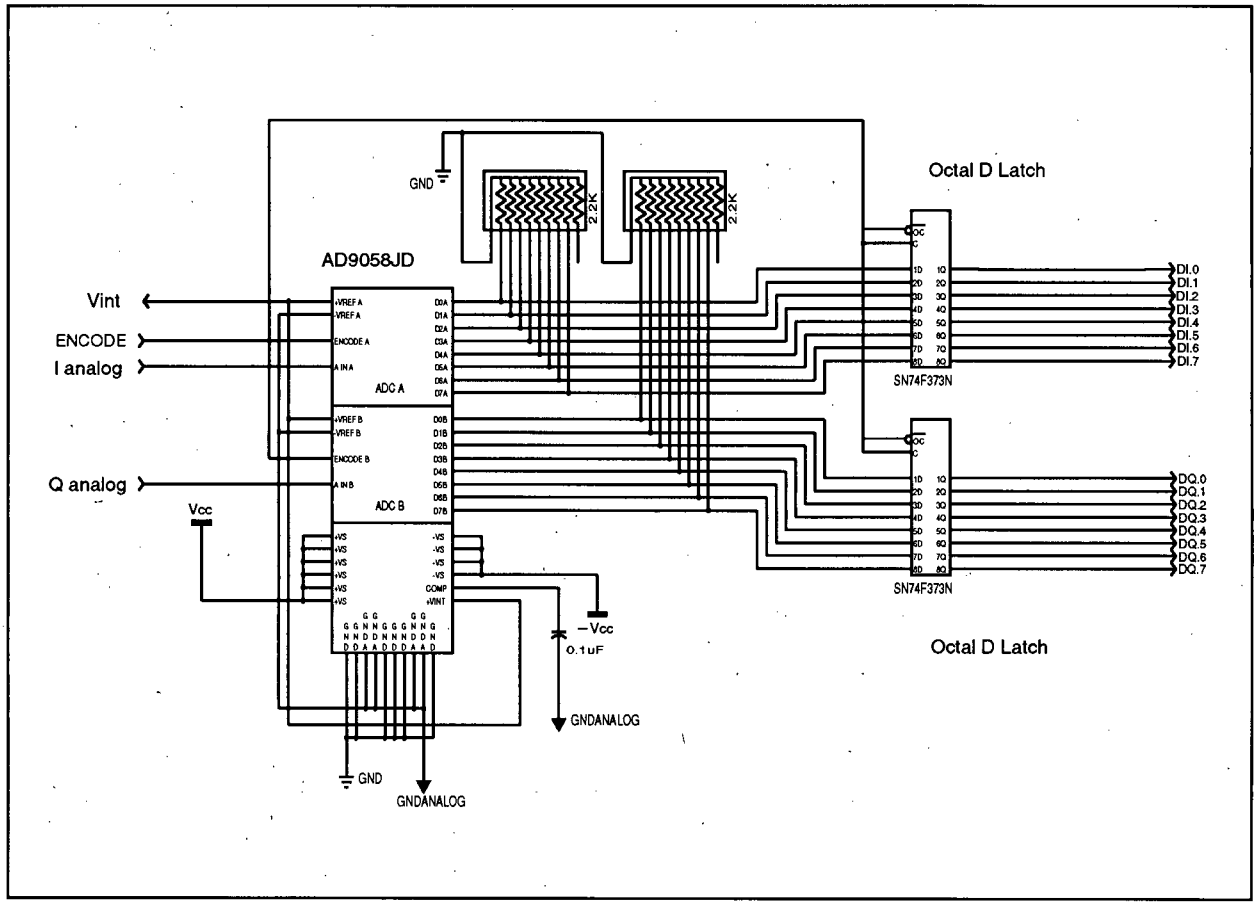

Figure 35 Dual Channel 8–Bit ADC

C.3 ADC Converter Control

The ADC Converter Control module takes as input the TTL clock oscillator, the QPC/C40 global reset, and one data line from a C40 communication port; and produces as output the ENCODE control signal for the ADC, and the write control signals, /WEN and WCLK, for writing ADC output to the bank of FIFO memory. The serial connection of two buffers (SN74F04N) provides a simple communication interface with the C40 providing the GO/STOP signal, clocked in by the D-flip-flop SN74F74N, for enabling/disabling the ADC conversion and FIFO writing. Each of the two JK flip-flops (SN74F109) perform a

division by two of their respective clock inputs. Thus, the FIFO write clock, WCLK, is driven at half the clock rate; while the FIFO write enable, /WEN, and the ADC encode control, ENCODE, signals are driven at one quarter of the receiver TTL clock oscillator rate.

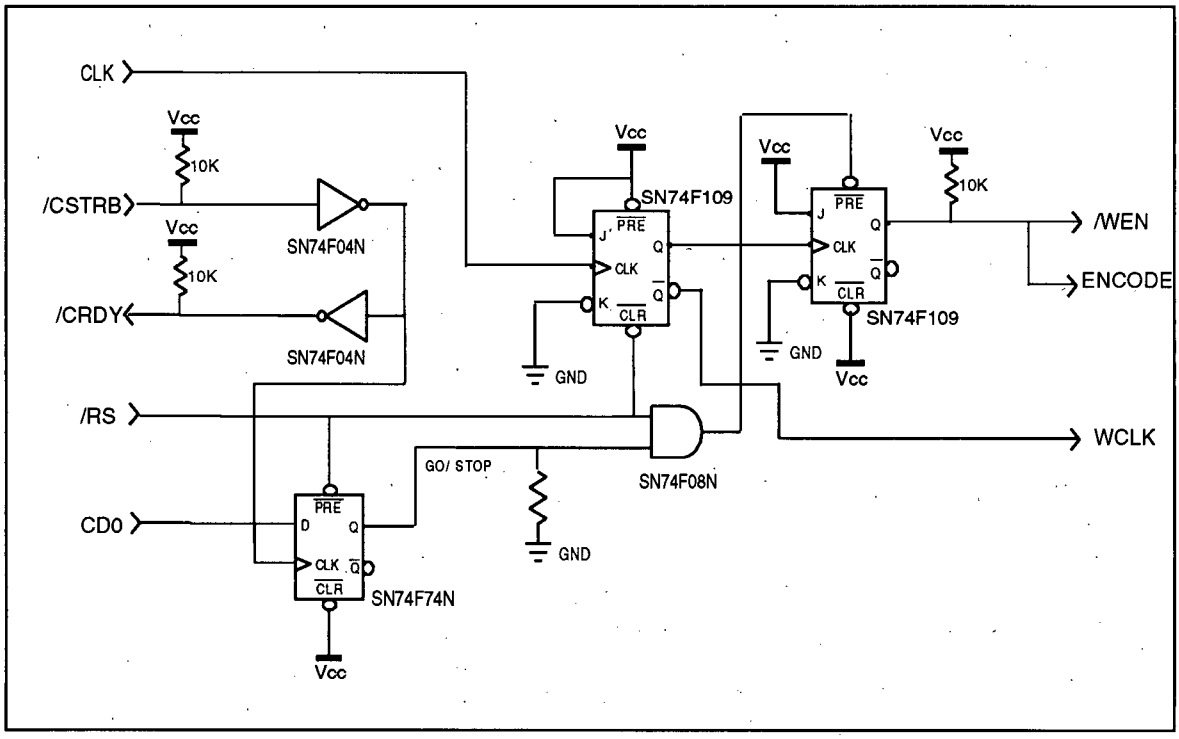

Figure 36 ADC Converter Control

C.4 FIFO Memory Banks

Each FIFO of the memory bank is TI's SN74ACT722X1L-25RJ. They are 2048 9–bit bytes of SRAM deep, with a 25 ns read or write access time. Either end of the FIFO can be operated in synchronous or in asynchronous mode. The write control signals, WCLK and /WEN, issued from the ADC conversion module, operate the input side of the FIFO in synchronous mode; while the read control signals, RCLK and /REN, issued from the FIFO/TMS320C40 communication port, operate the output side of the FIFO in asynchronous mode.

The FIFO activates the full flag, /FF, when the FIFO is full, indicating that the writes are blocked; and activates the empty flag, /EF, when the FIFO is empty, indicating that the reads are blocked. /EF is used by the FIFO/TMS320C40 communication as a control signal.

/FF is not used at present; but a future iteration of the receiver could use the full flag to indicate a system error — for normal operation, /FF should never be activated.

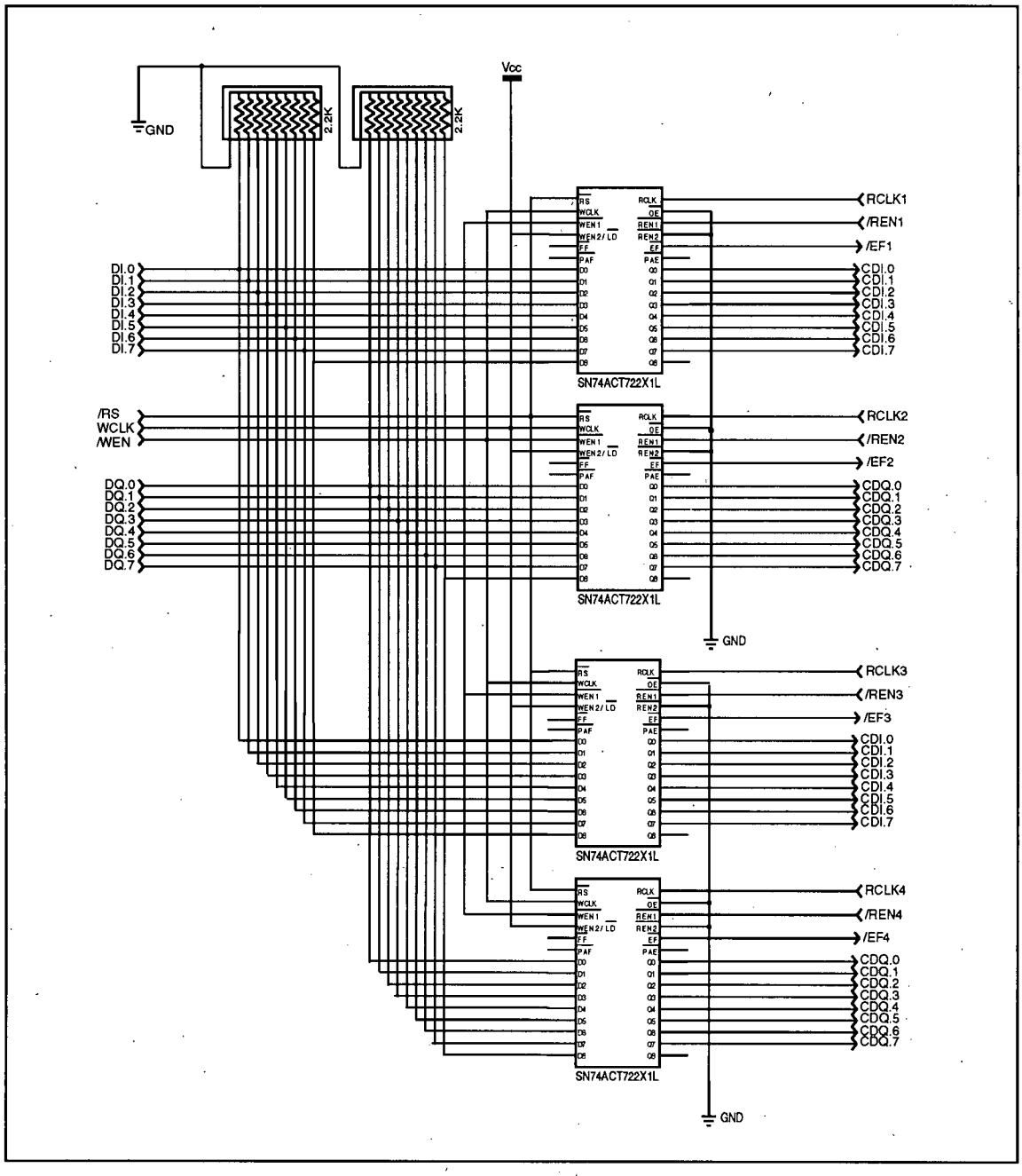

Figure 37 FIFO Memory Banks

C.5 FIFO/TMS320C40 Communication Port

Of all the receiver hardware, the FIFO/C40 communication port was the most challenging to design. This is because the C40 communication port is very high speed, and its communication protocol is rather particular; so that setup time requirements are stringent and signal quality is very important in order to avoid byte slippage. The C40 communication port transfers words in an asynchronous transmission of 4 bytes, followed by a word synchronizer delay. The port we have designed transfers bytes synchronously, and does not exercise an extra delay after every fourth byte. The resulting port speed is one half of that possible between two C40s.

The communication port has been extensively tested, and found to be reliable when driven by a clock of frequency up to 32 MHz. However, due to the C40 response time, and the propagation delays of the input and output buffers (SN74F125), we have found that the communication rate in bytes per second changes from clock frequency divided by 2 with a 16 MHz (or slower) clock, to clock frequency divided by 4 with a 20 MHz (or faster) clock. Therefore, we use a 16 MHz clock for a port speed of 8 MBPS.

The communication port schematic does not show the token forcer which is necessary to give the IF receiver ownership of the data bus. We have it implemented according to the schematic given in the TI TMS320C40 Data Book.

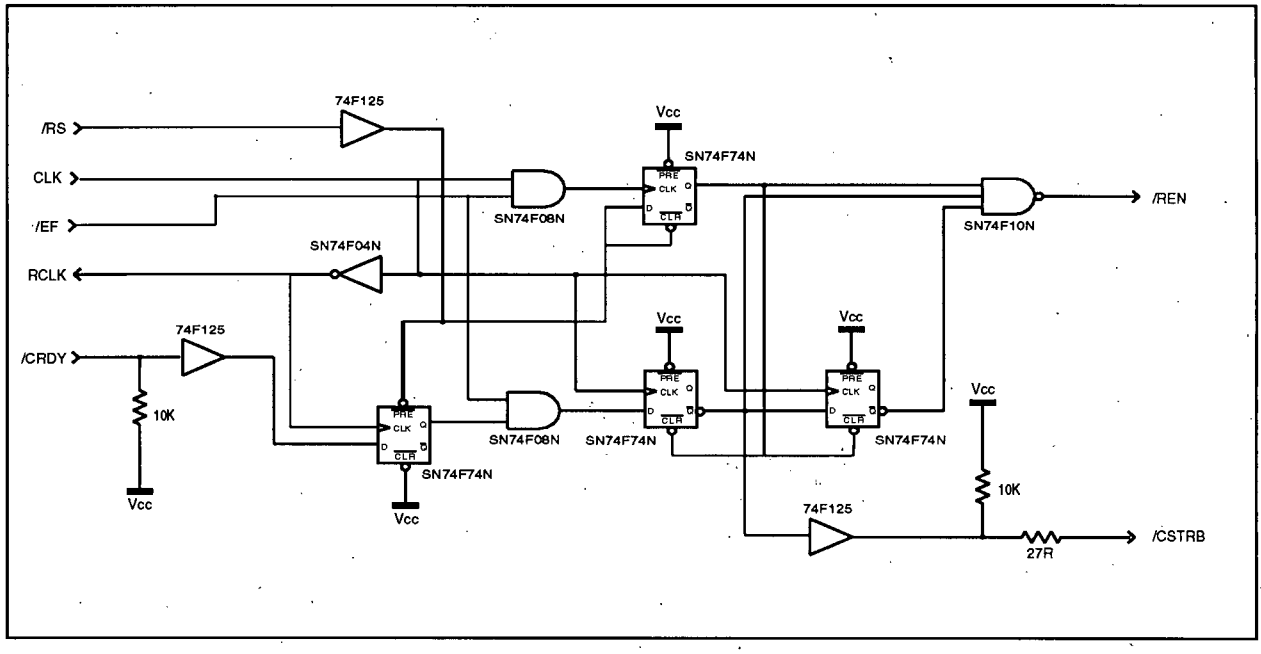

Figure 38 FIFO/TMS320C40 Communication Port

Appendix D DSP Code Listings

D.1 Master Receiver

```
/*
 * Master receiver for word length 1, quadrature data flow with Phase-
 * Invariant reception for a file sent over the IF link.
 * Hosted on processor A. Receives from a pair of complex correlators
 * on processor B, and an early and a late correlator on processor D.
 * The task reads from file datac (contains zeroes and ones) and passes
 * it to processor B which relays it to the transmitter on processor C.
 * The data which the receiver processes is written to file re_datac.
 * The amount of spreading is determined by a user supplied PN generator
 * shift register length.
 * The functionality includes acquisition which uses a two sample step
 * size. Once the code is acquired, the program tracks, decodes, and
 * writes to a buffer.
 */
#include <chan.h>
#include <stdio.h>
#include <math.h>                    /* for pow */
#include <stdlib.h>                  /* for calloc */
#include <compt40.h>                 /* for out_word */
extern reed();                       /* for reading from correlators */
float ci0=0;                         /* ci0 holds 0 I correlator output */
float cq0=0;                         /* cq0 holds 0 Q correlator output */
float ci1=0;                         /* ci1 holds 1 I correlator output */
float cq1=0;                         /* cq1 holds 1 Q correlator output */
float die=0;                         /* die holds e I correlator output */
float dqe=0;                         /* dqe holds e Q correlator output */
float dil=0;                         /* dil holds l I correlator output */
float dql=0;                         /* dql holds l Q correlator output */
main(int argc, char *argv[], char *envp[],
     CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
float c0=0;                          /* 0 correlator magnitude squared */
float c1=0;                          /* 1 correlator magnitude squared */
float de=0;                          /* early correlator mag squared */
float dl=0;                          /* late correlator mag squared */
float g;                             /* 0 corr. dot rotated 1 corr. */
float h;                             /* mag squared 0 minus mag squared 1 */
int M,N;                             /* PN shift reg length, PN length */
float threshold1;                    /* to detect within 1/2 chip */
float threshold2;                    /* to detect exact correlation */
float threshold3;                    /* for acq conferm */
float threshold4;                    /* to detect tracking loss */
int ESHIFT;                          /* length of elementary phase shift */
int acq=0;                           /* acq=0 means not acquired*/
int tflag=1;                         /* for tracker */
int tlflag=0;                        /* for tracker */
float max=0,maxd=0;                  /* max corr. output, max delayed */
int j=0;                             /* for tracker */
float temp;                          /* for tracking loss test */
int shift=0,shft=0;                  /* shifts for acq. and tracking */
int shiftd=0, shftd=0;               /* delayed shifts */
```

```
int slide;                          /* tracking adjustment */
int sld=0;                          /* slide adjustment */
int slided;                         /* delayed slide */
int i=0;                            /* sample counter */
int flag=-1;                        /* for flagging correlators */
int flagc;                          /* flagc=0 means malloc for corrs OK */
int flagg;                          /* flagg=0 means malloc for gen OK */
int *bufi,*bufq;                    /* pointers to buffered data */
int valuei,valueq;                  /* for reading data */
int datacount=0;                    /* input datacount */
FILE *outfile,*infile;              /* for data io */
infile = fopen("datac","r");        /* start by counting data */
while (fscanf(infile,"%d %d",&valuei,&valueq)==2) datacount=datacount+2;
fclose(infile);
printf("The number of bits to transmit is %d \n",datacount);
printf("Input PN shift register length (max=10): ");
scanf("%i",&M);                     /* read M */
chan_out_word(M,out_ports[0]);      /* send M to the correlators */
chan_out_word(M,out_ports[1]);      /* send M to delayed correlators */
N=pow(2,M)-1;
threshold1 = (2*N)/(16*16);
threshold2 = (2*N)/16;
threshold3 = threshold2;
threshold4 = threshold1;

ESHIFT=N-1;
slide=2*N-1; slided=2*N-1;
chan_in_word(&flagc,in_ports[0]);   /* confirmation from correlators */
if (flagc) {printf("Insufficient space for correlators\n");exit(1);}
chan_out_word(datacount,out_ports[0]);    /* send to corr */
bufi = (int *) calloc(datacount/2+1,1);   /* allocate and zero storage */
bufq = (int *) calloc(datacount/2+1,1);   /* allocate and zero storage */
if ( bufq == NULL )
{
   printf("The required rec storage space is not available. \n");
   exit(1);
}
chan_in_word(&flagg,in_ports[0]);   /* confirmation from generator */
if (flagg) {printf("Insufficient space for generator\n");exit(1);}
infile = fopen("datac","r");        /* pass data to  corr */
while (fscanf(infile,"%d %d",&valuei,&valueq)==2)
{
   chan_out_word(valuei,out_ports[0]);
   chan_out_word(valueq,out_ports[0]);
}
chan_out_word(flag,out_ports[0]);   /* flag end of file */
chan_out_word(flag,out_ports[0]);   /* with 2 -1's */
fclose(infile);
chan_in_word(&flag,in_ports[0]);    /* wait for correlators ready */
chan_in_word(&flag,in_ports[1]);    /* wait for delayed correlators */
chan_out_word(flag,out_ports[0]);   /* flag ready to correlator */
chan_out_word(flag,out_ports[1]);   /* flag delayed correlator */
/*  Send control signals to IF board */
  asm("        .data                          ");
  asm("o_addr: .word  00100092H               ");
  asm("fifosp: .word  0F4F4F4F4H              ");
  asm("frst:   .word  0F5F5F5F5H              ");
  asm("gostop: .word  0F7F7F7F7H              ");
  asm("rst:    .word  00000000H               ");
```

```
  asm("        .text                          ");
  asm("        push AR0                       ");
  asm("        push AR1                       ");
  asm("        push AR2                       ");
  asm("        push AR3                       ");
  asm("        ldpk @o_addr                   ");
  asm("        lda @o_addr, AR0               ");
  asm("        ldpk @fifosp                   ");
  asm("        lda @fifosp, AR1               ");
  asm("        ldpk @frst                     ");
  asm("        lda @frst, AR2                 ");
  asm("        ldpk @gostop                   ");
  asm("        lda @gostop, AR3               ");
  asm("        sti AR1,*AR0                   ");
  asm("        nop                            ");
  asm("        nop                            ");
  asm("        nop                            ");
  asm("        nop                            ");
  asm("        nop                            ");
  asm("        sti AR2,*AR0                   ");
  asm("        nop                            ");
  asm("        nop                            ");
  asm("        nop                            ");
  asm("        nop                            ");
  asm("        nop                            ");
  asm("        sti AR3, *AR0                  ");
  asm("        pop AR3                        ");
  asm("        pop AR2                        ");
  asm("        pop AR1                        ");
  asm("        pop AR0                        ");

for (i=0;i<=datacount;i++)
{
   if (acq==0)                          /* acquisition mode */
   {
      out_word(shift,0);                /* shift correlators */
      out_word(slide,0);
      out_word(shiftd,3);               /* shift delayed correlators */
      out_word(slided,3);
      slided = slide;
      shift = (shift+2)%(2*N);          /* increment shift by 2 samples */
      if (maxd>threshold1)              /* test for acquisition */
      {
         sld=0;
         (de>maxd)?(maxd=de,sld=-1):(maxd=maxd); /* adjust by 1 samp */
         (dl>maxd)?(maxd=dl,sld=+1):(maxd=maxd);
         if (maxd>threshold2)           /* test adjusted acquisition */
         {
            acq=1;shift=(2*N+shft-sld)%(2*N);j=2;
         }
      }
      maxd=max;
/*    printf("%e %e %e %e %d %d %d\n",c0,c1,de,dl,slide,shift,j); */
      shft=shiftd;
      reed();                           /* read correlators */
      c0 = ci0*ci0 + cq0*cq0;           /* form magnitudes squared */
      c1 = ci1*ci1 + cq1*cq1;
      de = die*die + dqe*dqe;
      dl = dil*dil + dql*dql;
```

```
      (c0>=c1)?(max=c0,shiftd=(2*N+shift-2)%(2*N)): /* find max and */
      (max=c1,shiftd=(ESHIFT+shift-2)%(2*N));       /* adjust shift */
   }
   else if (acq==1)                  /* acquisition confirmation */
   {
      out_word(shift,0);
      out_word(slide,0);
      out_word(shiftd,3);
      out_word(slided,3);

      if (j==0)
      {
         sld=0;
         (de>maxd)?(maxd=de,sld=-1):(maxd=maxd);
         (dl>maxd)?(maxd=dl,sld=+1):(maxd=maxd);
         if (maxd>threshold3)
         {                              /* acq confirmed */
            acq=2;slide=2*N-shift+sld-1;shftd=0;j=4;
         }
         else                           /* go back to acq mode */
         {
            acq=0;shift=(shift+2)%(2*N);
         }
      }
      j=j-1;
      maxd=c0;
/*    printf("%e %e %e %e %d %d %d\n",c0,c1,de,dl,slide,shift,j);   */
      shft=shiftd;
      reed();
      c0 = ci0*ci0 + cq0*cq0;
      c1 = ci1*ci1 + cq1*cq1;
      de = die*die + dqe*dqe;
      dl = dil*dil + dql*dql;

      (c0>=c1)?(max=c0,shiftd=(2*N+shift-2)%(2*N)):
      (max=c1,shiftd=(ESHIFT+shift-2)%(2*N));
   }
   else if (acq == 2)                   /* tracking/decoding mode */
   {
      out_word(0,0);                    /* write shift to (0,1) pair */
      out_word(slide,0);                /* write slide to (0,1) pair */
      out_word(shftd,3);                /* write shift to (e,1) pair */
      out_word(slide,3);                /* write slide to (e,1) pair */
      slide=2*N-1;
      if (j<=0)                         /* previous tracking adj done */
      {
         (de>dl)?(sld=-1,temp=de):(sld=+1,temp=dl);
         if (maxd<temp)                 /* test if tracking adj needed */
         {                              /* if tracker flag up, adjust */
            if (tflag) {j=4; slide=2*N-1+sld; tflag=0;}
            else tflag=1;               /* else raise flag */
         }
         else {temp=maxd; tflag=0;}     /* lower tracker flag */
         if (temp<threshold4)           /* test for tracking loss */
         {
            if (tlflag)                 /* if flag is up, exit */
            {
               printf("tracking loss \n");
               break;
```

```
            }
            else tlflag=1;              /* raise tracking loss flag */
         }
         else tlflag=0;                 /* lower tracking loss flag */
      }
      j=j-1;
      maxd=max;
      g = cq0*ci1-ci0*cq1;              /* form decision variables */
      h = (c0-c1)/2;
      if (g-h<=0)                       /* decode and buffer I and Q */
      {                                 /* channel data */
         if (g+h>=0) {*(bufi+i)=0; *(bufq+i)=0;}
         else        {*(bufi+i)=0; *(bufq+i)=1;}
      }
      else
      {
         if (g+h>=0) {*(bufi+i)=1; *(bufq+i)=0;}
         else        {*(bufi+i)=1; *(bufq+i)=1;}
      }

/*    printf("%e %e %e %e %e %e \n",c0,c1,de,dl,g+h,g-h);   */
      reed();                           /* read correlators */
      c0 = ci0*ci0 + cq0*cq0;           /* form magnitudes squared */
      c1 = ci1*ci1 + cq1*cq1;
      de = die*die + dqe*dqe;
      dl = dil*dil + dql*dql;
      (c0>=c1)?(max=c0,shftd=0):(max=c1,shftd=ESHIFT);
   }
}                                       /* end of while loop */
/* reset IF board */
  asm("        push AR0                    ");
  asm("        push AR1                    ");
  asm("        ldpk @o_addr                ");
  asm("        lda @o_addr, AR0            ");
  asm("        ldpk @rst                   ");
  asm("        lda @rst, AR1               ");
  asm("        sti AR1,*AR0                ");
  asm("        pop AR1                     ");
  asm("        pop AR0                     ");
outfile = fopen("re_datac","w");    /* decode shifts and write to file */
for (i=0;i<datacount/2+1;i++)
{
   valuei = *(bufi+i);
   valueq = *(bufq+i);
   fprintf(outfile,"%d %d  ", valuei,valueq);
}
fclose(outfile);
}                                       /* end of main */
```

```
;
; assembler routine for reading correlators
;
  .version 40
  .text
  .globl _reed
  .globl _ci0                    ; I chan 0 correlator input
  .globl _cq0                    ; Q chan 0 correlator input
  .globl _ci1                    ; I chan 1 correlator input
  .globl _cq1                    ; Q chan 1 correlator input
  .globl _die                    ; delayed I chan early correlator
  .globl _dqe                    ; delayed Q chan early correlator
  .globl _dil                    ; delayed I chan late correlator
  .globl _dql                    ; delayed Q chan late correlator
_reed:
  ldpk @i1_addr
  lda @i1_addr,AR0               ; correlator input port addr
  ldpk @i2_addr
  lda @i2_addr,AR1               ; delayed correlator input addr

  ldf *AR0,R1                    ; read from correlator
  ldpk @_ci0
  stf R1,@_ci0                   ; write to global variable
  ldf *AR0,R1
  ldpk @_cq0
  stf R1,@_cq0
  ldf *AR0,R1
  ldpk @_ci1
  stf R1,@_ci1
  ldf *AR0,R1
  ldpk @_cq1
  stf R1,@_cq1
  ldf *AR1,R1                    ; read from delayed correlator
  ldpk @_die
  stf R1,@_die                   ; write to global variable
  ldf *AR1,R1
  ldpk @_dqe
  stf R1,@_dqe
  ldf *AR1,R1
  ldpk @_dil
  stf R1,@_dil
  ldf *AR1,R1
  ldpk @_dql
  stf R1,@_dql
  rets

 .data
i1_addr: .word  00100041H        ; comm port 0 input fifo
i2_addr: .word  00100071H        ; comm port 3 input fifo
  .end
```

D.2 Pair of Complex Correlators

```
/*
 * We set up a pair of complex correlators to look for the 0 or the 1
 * represented by shifted or non-shifted PN sequence produced by gen.c
 * and correlate with corr() contained in file cor.asm. Corr() also
 * reads a shift and a slide from rec.c. There are 2 samples per chip.
 * The shift register length is read from rec.c and determines
 * the PN sequence. Before receiving, corr relays data file to be
 * transmitted from rec to gen.
 */
#include <chan.h>
#include <math.h>                    /* for pow */
#include <stdlib.h>                  /* for malloc */
extern void corr(void);              /* correlator written in assembler */
int N;                               /* PN sequence length */
int M;                               /* PN gen shift reg length */
int stage[] = {1,1,1,1,1,1,1,1,1,1};/* shift register for PN generator */
int connect[10][10]={{1,0,0,0,0,0,0,0,0,0},
       {1,1,0,0,0,0,0,0,0,0},
       {1,0,1,0,0,0,0,0,0,0},
       {1,0,0,1,0,0,0,0,0,0},
       {1,0,0,1,0,0,0,0,0,0},
       {1,0,0,0,0,1,0,0,0,0},
       {1,0,0,0,0,0,1,0,0,0},
       {1,0,0,0,1,1,1,0,0,0},
       {1,0,0,0,0,1,0,0,0,0},
       {1,0,0,0,0,0,0,1,0,0}};       /* connection vectors */
int *pn0, *pn1;                      /* pointers to the PN sequences */
void pn_gen(int *pn0, int *pn1);     /* the PN's generating func */
main(int argc, char *argv[], char *envp[],
   CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
   int flag;                         /* for processor synchronization */
   int datacount;                    /* for data relay */
   int valuei=0;                     /* for data relay */
   int valueq=0;
   int *p0;                          /* pointers to allocated space */
   int *p1;
   chan_in_word(&M,in_ports[0]);     /* read M from rec */
   chan_out_word(M,out_ports[0]);    /* pass M to generator */

   N = pow(2,M)-1;
   p0 = (int *) malloc(4*N+2);       /* allocate for aligned pointer */
   p1 = (int *) malloc(4*N+2);
   if (p1==NULL) chan_out_word(1,out_ports[1]);
   else chan_out_word(0,out_ports[1]); /* indicate to rec malloc result*/
   pn0 = (int *)((((unsigned int)p0>>(M+1))+1)<<(M+1));  /* aligned  */
   pn1 = (int *)((((unsigned int)p1>>(M+1))+1)<<(M+1));  /* pointers */
   chan_in_word(&datacount,in_ports[0]); /* relay data count */
   chan_out_word(datacount,out_ports[0]);
   chan_in_word(&flag,in_ports[1]);   /* read from gen */
   chan_out_word(flag,out_ports[1]);  /* indicate to receiver */
```

```
   while ( valuei != -1 )              /* relay data to generator */
   {
   chan_in_word(&valuei,in_ports[0]);
   chan_in_word(&valueq,in_ports[0]);
   chan_out_word(valuei,out_ports[0]);
   chan_out_word(valueq,out_ports[0]);
   }
   pn_gen(pn0,pn1);                    /* initialize PN's */
   chan_out_word(flag,out_ports[1]);   /* indicate readiness to rec */
   chan_in_word(&flag,in_ports[0]);    /* wait for receive to be ready */
   chan_out_word(flag,out_ports[0]);   /* flag gen to indicate ready */
   corr();                             /* call correlator function */
}
void pn_gen(int *PN0, int *PN1)        /* PN sequence generator */
{
   int i, j, temp;
   for (j=0; j<N ; j++)
   {
   temp=0;
   *(PN0+2*j) = 1 - 2*stage[0];        /* convert (0,1) to (1,-1) */
   *(PN0+2*j+1) = 1 - 2*stage[0];      /* and double up the entries */
   *(PN1+((2*j+((2*N)+2)/2)%(2*N))) = *(PN0+2*j); /* shifted PN seq */
   *(PN1+((2*j+1+((2*N)+2)/2)%(2*N))) =*(PN0+2*j);
   for (i=0; i < M; i++)
   {
  temp = temp^(stage[i]*connect[M-1][i]);
  if (i<M-1) stage[i]=stage[i+1];
   }
   stage[M-1]=temp;
   }
}

;
; assembler routine to implement a pair of real-time correlators
;
         .version 40
         .text
         .globl _corr
         .globl _pn0
         .globl _pn1
         .globl _N
_corr:                                 ; setup correlator
         ldpk @i_addr
         lda @i_addr, AR0              ; AR0 holds I data input port addr
         ldpk @q_addr
         lda @q_addr, AR1              ; AR1 holds Q data input port addr
         ldpk @is_addr
         lda @is_addr, AR2             ; shift/slide input port addr
         ldpk @o_addr
         lda @o_addr, AR3              ; AR3 holds output port addr
         ldpk @_pn0
         lda @_pn0, AR4                ; AR4 addresses 0 PN sequence
         ldpk @_pn1
         lda @_pn1, AR5                ; AR5 addresses 1 PN sequence
         ldpk @_N                      ; N is PN seq length
         lda @_N, BK                   ; for circular addressing mode
         mpyi 2,BK                     ; two samples per chip
         ldpk @center
```

```
        ldi @center,R6              ; ADC offset

        subi 1,BK,RC                ; read off 2N samples, create delay
        rptb delay
        ldi *AR0,R8
delay:  ldi *AR1,R8
top:
        lda AR4,AR6                 ; AR6 indexs 0 PN seq
        lda AR5,AR7                 ; AR7 indexs 1 PN seq
        ldi *AR2,IR1                ; read shift into IR1
        nop *AR6++(IR1)%            ; shift correlators
        nop *AR7++(IR1)%
        ldi *AR2,IR0                ; read slide
        ldi IR0, RC                 ; repeat slide+1 times
        ldi 0,R10                   ; R10 holds sum of I sample squares
        ldi 0,R2                    ; R2 holds 0 I sum
        rptbd fin
        ldi 0,R3                    ; R3 holds 0 Q sum
        ldi 0,R4                    ; R4 holds 1 I sum
        ldi 0,R5                    ; R5 holds 1 Q sum

        lbu0 *AR0,R0                ; load zeroth byte
        lbu0 *AR1,R1                ; load zeroth byte
        subi R6,R0                  ; subtract ADC offset
        subi R6,R1
        mpyi R0,*AR6   ,R8          ; I chan multiply for 0 correlator
        mpyi R1,*AR6++%,R9          ; Q chan multiply for 0 correlator
        addi R8,R2                  ; accumulate I chan 0 correlation
        addi R9,R3                  ; accumulate Q chan 0 correlation
        mpyi R0,*AR7   ,R8          ; mult and accumulate
        mpyi R1,*AR7++%,R9          ; for 1 correlator
        addi R8,R4
        addi R9,R5
        mpyi R0,R0                  ; I chan sample square
        mpyi R1,R1                  ; Q chan sample square
        addi R0,R10                 ; I chan sum of squares
fin:    addi R1,R10                 ; Q chan sum of squares

        float R2,R0
        float R3,R1
        float R4,R8
        float R5,R9
        float R10,R11               ; R11 holds sum of squares
        bz protect                  ; protect against zero division
        rsqrf R11,R10               ; R10 holds reciprocal sqrt
        mpyf R10,R0                 ; normalized 0 I correlator output
        mpyf R10,R1                 ; normalized 0 Q correlator output
        mpyf R10,R8                 ; normalized 1 I correlator output
        mpyf R10,R9                 ; normalized 1 Q correlator output
        stf R0,*AR3
        brd top
        stf R1,*AR3                 ; output correlator results
        stf R8,*AR3
        stf R9,*AR3
protect: ldf 0,R11                  ; output zeroes
        stf R11,*AR3
        brd top
        stf R11,*AR3
        stf R11,*AR3
```

```
        stf R11,*AR3
        .data
i_addr:  .word 00100061H          ; comm port 2 input fifo
q_addr:  .word 00100081H          ; comm port 4 input fifo
is_addr: .word 00100071H          ; comm port 3 input fifo
o_addr:  .word 00100072H          ; comm port 3 output fifo
center:  .word 0000007EH          ; ADC offset (127)
        .end
```

D.3 Tracking Correlators

```
/*
 * We set up a pair of correlators to look for the 0 or the 1
 * represented by shifted or non-shifted PN sequence produced by gen.c
 * and correlate with corr() contained in file cor.asm. Corr() also
 * reads a shift and a slide from rec.c. There are 2 samples per chip.
 * The shift register length is read from rec.c and determines
 * the PN sequence.
 */
#include <chan.h>
#include <math.h>                    /* for pow */
#include <stdlib.h>                  /* for malloc */
extern void corr(void);              /* correlator written in assembler*/
int N;                               /* PN sequence length */
int M;                               /* PN gen shift reg length */
int connect[10][10]={{1,0,0,0,0,0,0,0,0,0},
       {1,1,0,0,0,0,0,0,0,0},
       {1,0,1,0,0,0,0,0,0,0},
       {1,0,0,1,0,0,0,0,0,0},
       {1,0,0,1,0,0,0,0,0,0},
       {1,0,0,0,0,1,0,0,0,0},
       {1,0,0,0,0,0,1,0,0,0},
       {1,0,0,0,1,1,1,0,0,0},
       {1,0,0,0,0,1,0,0,0,0},
       {1,0,0,0,0,0,0,1,0,0}};       /* connection vectors */
int stage[] = {1,1,1,1,1,1,1,1,1,1}; /* shift reg for PN generator */
int *pn0, *pn1;                      /* pointers to the PN sequences */
void pn_gen(int *pn0, int *pn1);     /* the PN's generating func */
main(int argc, char *argv[], char *envp[],
   CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
   int flag;                         /* for processor synchronization */
   int *p0;                          /* pointer to allocated space */
   int *p1;
   chan_in_word(&M,in_ports[0]);     /* read M from rec */
   N = pow(2,M)-1;
   p0 = (int *) malloc(4*N+2);       /* allocate for aligned pointer */
   p1 = (int *) malloc(4*N+2);
   pn0 = (int *)((((unsigned int)p0>>(M+1))+1)<<(M+1)); /* aligned */
   pn1 = (int *)((((unsigned int)p1>>(M+1))+1)<<(M+1)); /* pointers */
   pn_gen(pn0,pn1);                  /* initialize PN's */
   chan_out_word(flag,out_ports[1]); /* indicate readiness to rec */
   chan_in_word(&flag,in_ports[0]);  /* wait for receive to be ready */
   chan_out_word(flag,out_ports[0]); /* flag gen to indicate ready */
   corr();                           /* call correlator function */
```

```
}
void pn_gen(int *PN0, int *PN1)       /* generate PN sequence */
{
   int i, j, temp;
   for (j=0; j<N ; j++)
   {
   temp=0;
   *(PN0+((2*N-1+2*j)%(2*N))) = 1 - 2*stage[0];  /* early correlator */
   *(PN0+((2*N+2*j)%(2*N)))   = 1 - 2*stage[0];
   *(PN1+((2*N+1+2*j)%(2*N))) = 1 - 2*stage[0];  /* late correlator  */
   *(PN1+((2*N+2+2*j)%(2*N))) = 1 - 2*stage[0];
   for (i=0; i < M; i++)
   {
  temp = temp^(stage[i]*connect[M-1][i]);
  if (i<M-1) stage[i]=stage[i+1];
   }
   stage[M-1]=temp;
   }
}

;
; assembler routine to implement a pair of real-time correlators
;
         .version 40
         .text
         .globl _corr
         .globl _pn0
         .globl _pn1
         .globl _N
_corr:                                ; setup correlator
         ldpk @i_addr
         lda @i_addr, AR0             ; AR0 holds I data input port addr
         ldpk @q_addr
         lda @q_addr, AR1             ; AR1 holds Q data input port addr
         ldpk @is_addr
         lda @is_addr, AR2            ; shift/slide input port addr
         ldpk @o_addr
         lda @o_addr, AR3             ; AR3 holds output port addr
         ldpk @_pn0
         lda @_pn0, AR4               ; AR4 addresses early PN sequence
         ldpk @_pn1
         lda @_pn1, AR5               ; AR5 addresses late PN sequence
         ldpk @_N                     ; N is PN seq length
         lda @_N, BK                  ; for circular addressing mode
         mpyi 2,BK                    ; two samples per chip
         ldpk @center
         ldi @center,R6               ; ADC offset
top:
         lda AR4,AR6                  ; AR6 indexes early PN seq
         lda AR5,AR7                  ; AR7 indexes late PN seq
         ldi *AR2,IR1                 ; read shift into IR1
         nop *AR6++(IR1)%             ; shift correlators
         nop *AR7++(IR1)%
         ldi *AR2,IR0                 ; read slide
         ldi IR0, RC                  ; repeat slide+1 times
         ldi 0,R10                    ; R10 holds sum of I sample squares
         ldi 0,R2                     ; R2 holds early I sum
         rptbd fin
```

```
        ldi 0,R3                ; R3 holds early Q sum
        ldi 0,R4                ; R4 holds late I sum
        ldi 0,R5                ; R5 holds late Q sum
        lbu0 *AR0,R0            ; load zeroth byte
        lbu0 *AR1,R1            ; load zeroth byte
        subi R6,R0              ; subtract ADC offset
        subi R6,R1
        mpyi R0,*AR6   ,R8      ; I chan mult for early correlator
        mpyi R1,*AR6++%,R9      ; Q chan mult for early correlator
        addi R8,R2              ; accumulate I chan early corr
        addi R9,R3              ; accumulate Q chan late corr
        mpyi R0,*AR7   ,R8      ; multiply and accumulate
        mpyi R1,*AR7++%,R9      ; for late correlator
        addi R8,R4
        addi R9,R5
        mpyi R0,R0              ; I chan sample square
        mpyi R1,R1              ; Q chan sample square
        addi R0,R10             ; I chan sum of squares
fin:    addi R1,R10             ; Q chan sum of squares

        float R2,R0
        float R3,R1
        float R4,R8
        float R5,R9
        float R10,R11           ; R11 holds sum of squares
        bz protect              ; protect against zero division
        rsqrf R11,R10           ; R10 holds reciprocal sqrt
        mpyf R10,R0             ; normalized 0 I correlator output
        mpyf R10,R1             ; normalized 0 Q correlator output
        mpyf R10,R8             ; normalized 1 I correlator output
        mpyf R10,R9             ; normalized 1 Q correlator output
        stf R0,*AR3
        brd top
        stf R1,*AR3             ; output correlator results
        stf R8,*AR3
        stf R9,*AR3
protect: ldf 0,R11              ; output zeroes
        stf R11,*AR3
        brd top
        stf R11,*AR3
        stf R11,*AR3
        stf R11,*AR3
        .data
i_addr:  .word 00100061H        ; comm port 2 input fifo
q_addr:  .word 00100091H        ; comm port 5 input fifo
is_addr: .word 00100041H        ; comm port 0 input fifo
o_addr:  .word 00100042H        ; comm port 0 output fifo
center:  .word 0000007EH        ; ADC offset (127)
        .end
```

D.4 Transmitter Code

```
/*
 * Transmitter code for word length 1, quadrature data transmission
 * over the IF link. We read from file data, copy the data to
 * the on-board memory, and then transmit a stream of
 * independent I and Q channel data, with the assembler function
 * tran(), which is the fastest possible transmitter.
 * This task goes on processor C.
 */
#include <chan.h>
#include <math.h>                       /* for pow */
#include <stdlib.h>                     /* for malloc */
int N;                                  /* length of the PN sequence */
int M;                                  /* number of PN generator stages */
int len;                                /* N, used by trans() */
int stage[] = {1,1,1,1,1,1,1,1,1,1}; /* shift reg for PN generator */
int connect[10][10]={{1,0,0,0,0,0,0,0,0,0},
       {1,1,0,0,0,0,0,0,0,0},
       {1,0,1,0,0,0,0,0,0,0},
       {1,0,0,1,0,0,0,0,0,0},
       {1,0,0,1,0,0,0,0,0,0},
       {1,0,0,0,0,1,0,0,0,0},
       {1,0,0,0,0,0,1,0,0,0},
       {1,0,0,0,1,1,1,0,0,0},
       {1,0,0,0,0,1,0,0,0,0},
       {1,0,0,0,0,0,0,1,0,0}};          /* connection vectors */
int *pni;                               /* pointer to I chan PN seq */
int *pnq;                               /* pointer to Q chan PN seq */
int *bufi;                              /* pointer to I chan input data */
int *bufq;                              /* pointer to Q chan input data */
int count;                              /* data count minus one */
void pn_gen(int *pni, int *pnq);
extern void trans();                    /* function written in assembler */
main(int argc, char *argv[], char *envp[],
  CHAN *in_ports[], int ins, CHAN *out_ports[], int outs )
{
   int flag;                            /* for processor synchronization */
   int valuei=0;                        /* for data input */
   int valueq=0;
   int i=0;                             /* a data index */
   int ESHIFT;
   int *pi;                             /* to allocate space */
   int *pq;
   int datacount;
   chan_in_word(&M,in_ports[0]);        /* read M from correlator */
```

```
    N = pow(2,M)-1;
    len = N;
    ESHIFT = (N-1)/2;                   /* length of elem code shift */
    pi = (int *)malloc(2*len);          /* allocate for aligned pointer */
    pq = (int *)malloc(2*len);
    pni = (int *)((((unsigned int)pi>>(M))+1)<<(M));   /* aligned */
    pnq = (int *)((((unsigned int)pq>>(M))+1)<<(M));   /* pointers */
    chan_in_word(&datacount,in_ports[0]);    /* read data count */
    count = datacount/2-1;
    bufi = calloc(datacount/2,1);       /* alloc storage for input data */
    bufq = calloc(datacount/2,1);
    if (bufq==NULL) chan_out_word(1,out_ports[0]);
    else chan_out_word(0,out_ports[0]);
    while ( (valuei!=-1)&&(valueq!=-1) )     /* read and store data     */
    {                                        /* data terminated by a -1 */
    chan_in_word(&valuei,in_ports[0]);
    *(bufi+i) = valuei*ESHIFT ;         /* store the shifts */
    chan_in_word(&valueq,in_ports[0]);
    *(bufq+i) = valueq*ESHIFT ;
    i++;
    }

    pn_gen(pni,pnq);                    /* initialize PN's */

    chan_in_word(&flag,in_ports[0]);    /* wait for corrs to be ready */
    chan_in_word(&flag,in_ports[1]);
    trans();                            /* transmit */
    while(1);
}
void pn_gen(int *PNI, int *PNQ)         /* PN sequence generator */
{
    int i, j, temp;
    int value;

    for (j=0; j<N ; j++)
    {
    value = -1*stage[0];                /* convert 1 to all ones word */
    *(PNI+j) = value;                   /* store I chan PN sequence */
    *(PNQ+j) = value;                   /* store Q chan PN sequence */

    temp=0;
    for (i=0; i < M; i++)
    {
   temp = temp^(stage[i]*connect[M-1][i]);
   if (i<M-1) stage[i]=stage[i+1];
    }
    stage[M-1]=temp;
    }
}


;
; assembler routine for real-time transmission
;
```

```
          .version 40
          .text
          .globl _trans
          .globl _bufi              ; points to I chan data buffer
          .globl _bufq              ; points to Q chan data buffer
          .globl _pni               ; points to I chan PN sequence
          .globl _pnq               ; points to Q chan PN sequence
          .globl _len               ; PN sequence length
          .globl _count             ; data count

_trans:
          push AR3
          push AR4
          push AR5
          push AR6
          push AR7
          push R4
          push R5
          push R6
          pushf R6
          push R7
          pushf R7
          push R8

          ldpk @_len
          ldi @_len,BK              ; for circular addressing
          ldpk @_pni
          ldi @_pni,R8              ; R8 addresses pni
          ldpk @_pnq
          ldi @_pnq,R9              ; R9 addresses pnq
          ldpk @i_addr
          lda @i_addr,AR2           ; I chan output port
          ldpk @q_addr
          lda @q_addr,AR3           ; Q chan output port
          ldpk @_bufi
          lda @_bufi,AR4            ; AR4 holds I channel data buffer
          ldpk @_bufq
          lda @_bufq,AR5            ; AR5 holds I channel data buffer
          ldpk @_count
          lda @_count,AR6           ; AR6 holds data count minus one
lp1:      ldi R8,AR0                ; set AR0 to pni start addr
          ldi R9,AR1                ; set AR1 to pnq start addr
          subi 1,BK,RC              ; for repeat mode
          ldi *AR4++,IR0            ; read I chan shift into IR0
          rptbd lp2
          ldi *AR5++,IR1            ; read Q chan shift into IR1
          nop *AR0++(IR0)%          ; shift pni index according to I data
          nop *AR1++(IR1)%          ; shift pnq index according to Q data
          ldi *AR0++%,R3            ; start of repeat block
       || ldi *AR1++%,R5            ; I/Q chan chips in R3/R5
          sti R3,*AR2               ; write chips to output ports
       || sti R5,*AR3
lp2:      sti R3,*AR2               ; repeat write for half speed
       || sti R5,*AR3               ; transmission
```

```
        db AR6,lp1
        pop R8
        popf R7
        pop R7
        popf R6
        pop R6
        pop R5
        pop R4
        pop AR7
        pop AR6
        pop AR5
        pop AR4
        pop AR3
        rets
        .data
i_addr: .word  00100052H          ; comm port 0 output fifo
q_addr: .word  00100062H          ; comm port 1 output fifo

        .end
```

D.5 Application Configuration

```
! Hardware configuration
processor root
processor second
processor third
processor fourth
wire ? root[0] second[3]
wire ? second[0] third[3]
wire ? third[0] fourth[3]
wire ? fourth[0] root[3]
! Software configuration
task rec4   ins=2 outs=2
task corr4  ins=2 outs=2
task corrd4 ins=1 outs=2
task gen4   ins=2 outs=1   OPT=CODE
place rec4 root
place corr4 second
place gen4 third
place corrd4 fourth
connect ? rec4[0] corr4[0]
connect ? rec4[1] corrd4[0]
connect ? corr4[0] gen4[0]
connect ? corrd4[0] gen4[1]
connect ? corr4[1] rec4[0]
connect ? gen4[0] corr4[1]
connect ? corrd4[1] rec4[1]
```