UNIVERSITY OF ALBERTA

DSP IMPLEMENTATION OF DECISION-AIDED ERROR CORRECTION ALGORITHM FOR DIGITAL CELLULAR RADIO

By

Seema Madan ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science.**

DEPARTMENT OF ELECTRICAL ENGINEERING

Edmonton, Alberta

Spring 1996

ISBN 0-612-10732-9

UNIVERSITY OF ALBERTA

LIBRARY RELEASE FORM

NAME OF AUTHOR: **SEEMA MADAN**

TITLE OF THESIS: **DSP Implementation of Decision-Aided Error Correction Algorithm for Digital Cellular Radio**

DEGREE: **MASTER OF SCIENCE**

YEAR THIS DEGREE GRANTED: **1996**

Seema Madan

Seema Madan
C4H / 22, Janakpuri
New Delhi, India 110058

Date: 25th January 1996.

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **DSP IMPLEMENTATION OF DECISION-AIDED ERROR CORRECTION ALGORITHM FOR DIGITAL CELLULAR RADIO** submitted by **Seema Madan** in partial fulfillment of the requirements for the degree of **Master of Science**.

Paul A. Goud

Dr. Paul A. Goud, Supervisor

Colin G. Englefield.

Dr. Colin G. Englefield, Co-Supervisor

Keith A. Stromsmoe

Dr. K. A. Stromsmoe, Internal Examiner

Dr. P. Gburzynski, External Examiner

Date: 22nd January 1996.

Dedicated to my dear parents, sisters, and husband
for their love, support and encouragement.

ABSTRACT

In mobile radio communication, the major cause of performance degradation for a traditional receiver based on differential detection is the multipath fading phenomenon associated with the channel. Amplitude and phase variations of the received signal due to fading result in severe and unpredictable random phase shifts superimposed onto the phase modulated transmitted signal. This leads to closure of the received signal eye-diagram, resulting in demodulation errors that exhibit an irreducible error floor. Decision-aided detection has been shown to be an effective error correcting method for BPSK modulation. Its performance for $\pi/4$DQPSK modulation as per the IS54 standard is evaluated in this research.

The decision-aided algorithm is used for tracking the phase variations due to the fading channel. The channel estimation is based on the fact that, in a typical case, only one out of one hundred first decisions is detected incorrectly. The reverse modulator recovers the phase change due to the channel, by subtracting the phase changes corresponding to the transmitted data from the total received phase change. The effect of any incorrect first decisions is reduced by a channel estimation filter, using samples before and after each decision sample. Once the phase change due to the fading channel is known, its effect on the differential detector output is removed so that the detected signal error rate is significantly reduced.

In this research, the transmitter and the flat fading channel model have been implemented using the Texas Instruments DSP TMS320C50, using equivalent complex base-band signals for the $\pi/4$DQPSK modulation. The modulated signal, distorted by the channel, is fed digitally to a TMS320C30 DSP where the decision-aided error correction algorithm is implemented. The test results show that the algorithm lowers the error floor, obtained at the output of a conventional differential detector, by a factor of more than 200. The performance is also evaluated for various values of signal to noise ratio (SNR). Under adequate SNR conditions, this method of error correction is quite attractive as compared to other techniques such as coding, insertion of pilot symbols, or the diversity method, because it requires no overhead bits. The implementation of the algorithm requires $\cong$167 instruction cycles/symbol; thus, for the TMS320C30 DSP @60ns instruction cycle, only 10μs/symbol are needed. This would allow the error correction method to be implemented for symbol rates up to 100ks/s. Also, since the algorithm does not require any modification to the frame structure of the signal, it may be preferred if the transmitting signal format has already been standardized. A second method of detection, which attempts to obtain the true channel characteristic from the incoming signal is also proposed, so as to obtain better results for the lower range of SNR values. This method has been shown to be useful for locating the errors, to remove the errors however, a complex correction algorithm may be required.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisors, Dr. Paul A. Goud and Dr. Colin G. Englefield, for their interest, advice and support throughout my graduate program. I would also like to thank Dr. Ray E. Rink and Dr. Wayne D. Grover along with my supervisors for introducing me to the field of Digital Signal Processing, Telecommunications and Wireless Communications.

My sincere thanks to Telecommunications Research Laboratories and its sponsors, for the award of a Telecommunication Research Scholarship, and the unique and stimulating research environment. Also, I extend my thanks to the research staff and colleagues at TR Labs for many helpful discussions, particularly David Clegg and Jason Lamont for their help in PCB fabrication. Thanks are also due to Albert Lau for his help in setting up the initialization routine for the DSP platform.

Finally, I thank my family and friends for their encouragement and support, throughout the completion of this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

$\pi/4$DQPSK	$\pi/4$ Shifted Differential Quadrature Phase Shift Keying
AFD	Average Fade Duration
AMPS	Advanced Mobile Phone Service
ARQ	Automatic Request for Retransmission
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
b/s/Hz	bits per second per Hertz
CDMA	Code Division Multiple Access
CDF	Cumulative Distribution Function
CT2	Cordless Telephone 2
DCS 1800	Digital Cellular System 1800
DECT	Digital European Cordless Telecommunications
DSP	Digital Signal Processor
EIA/TIA	Electronic Industries Association/Telecommunication Industry Association
FDMA	Frequency Division Multiple Access
FEC	Forward Error Control
FIR	Finite Impulse Response
FM	Frequency Modulation
GSM	Global System for Mobile Communications
IIR	Infinite Impulse Response
IS54	Interim Standard 54
IS95	Interim Standard 95
IS136	Interim Standard 136
ISI	Inter-Symbol Interference
kb/s	kilo-bit per second

kbaud	kilo-baud
LCR	Level Crossing Rate
LO	Local Oscillator
MAHO	Mobile Station Assisted Handover
MSC	Mobile Service Switching Center
NADMCS	North American Dual Mode Cellular System
NMT	Nordic Mobile Telephone
NTT	Nippon Telephone and Telegraph Company
OK-QPSK	Offset Keying - Quadrature Phase Shift Keying
PBPF	Pilot Band-Pass Filter
PCB	Printed Circuit Board
PDC	Personal Digital Cellular
PDF	Probability Density Function
QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency
SER	Symbol Error Rate
SNR	Signal to Noise Ratio
SRRC	Square Root Raised Cosine
TACS	Total Access Communication System
TCT	Tone Calibrated Technique
TDMA	Time Division Multiple Access
VSELP	Vector Sum Excited Linear Predictive Coding
WLAN	Wireless Local Area Network

LIST OF SYMBOLS

C, D	Vector measurement for phase difference between estimated and extracted channel
CHI, CHQ	Vector measurement for fading channel
D0	Transmitted data symbol
D1	First decision: decison made about the transmitted symbol at the output of differential detector
D2	Second decision: decision made about the transmitted symbol at the output of decision-aided detector
f_d	Doppler frequency
I_k, Q_k	Received signal vector at the output of the sampler
IF	Improvement factor
SI, SQ	Vector measurement for received signal
U, V	Vector measurement for received differential phase angle
Up, Vp	Vector measurement for extracted channel
Upf, Vpf	Vector measurement for filtered channel estimate
Up_{new}, Vp_{new}	Vector measurement for extracted channel after error correction

1. INTRODUCTION

General Background

The introduction of radio telephony in the world of conventional public telephone services was motivated by the need to communicate from remote locations or while on the move, as from a car or other vehicle. To decouple the telephone from its wires to the local exchange, thereby allowing mobility for the user, radio communication with the fixed wireline network needs to be used. This presents several challenges, such as: the efficient use of the limited frequency spectrum, the lossy and dispersive nature of the transmission medium, extreme random fading of the received signals, and the variable location of the mobile terminal. Over the past decade, wireless communications has advanced to the point where a different scale of mobility is provided to users for different applications. However, the quality and capacity of communication over radio channels is yet to meet the growing demand.

1.1 Mobile Radio Communications

The driving forces behind the evolution and development of mobile radio communication from the first generation onward are: system capacity, extent of coverage or mobility (cellular, cordless, paging, satellite-based wireless, etc.), various transmission services for different applications (voice, data, messaging, video) and cost.

Experiments on radio communication were first conducted by Hertz in 1880. In 1897 Marconi demonstrated successful transmission of a radio signal over an 18 mile path. In 1921, the first land mobile radio telephone was installed by the Detroit Police Department, in the USA. As technology and demands increased, the trend in 1938 was to use higher frequencies (30MHz range).

Further developments led to the growth from simplex (push to talk) to full duplex operation at 150MHz in 1964. Subsequently (1970), frequency channel splitting was used, but which was soon found to be of limited in use in meeting the demand for service. In 1974, the Bell telephone system introduced the system *Advanced Mobile Phone Service (AMPS)* which was designed for use in a cellular planned network. The number of customers that could be served increased tremendously due to the division of the coverage area into cells, and the reuse of frequencies in non-adjacent cells; these features are shown in Fig. 1.1. To provide for seamless transfer of a call from one cell to another, it was proposed to execute a *handoff* operation as the mobile unit crosses cell boundaries.

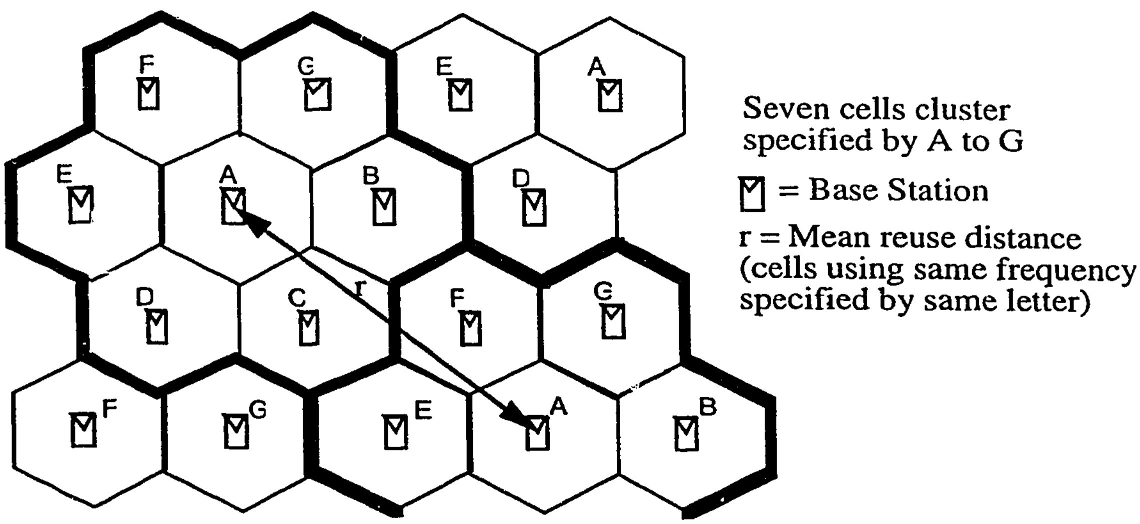


Fig. 1.1. Cellular mobile radio system.

Other first generation cellular systems, such as TACS, NMT, NTT and many others similar to AMPS were introduced in different parts of the world [1]. All these system use *frequency division multiple access (FDMA)*, and *analog frequency modulation* for speech transmission and *frequency shift keying* for signaling. Systems based on this technology are currently being used by more than 30 million subscribers worldwide. Additionally, first generation cordless telephones using analog radio technologies were developed, but they provide low mobility (both in range and user's speed) and have low power compared to cellular.

A potential for further increase in capacity was made possible in early 1990, with the deployment of the second generation cellular systems using digital technology. These systems use *time division multiple access (TDMA)* or *code division multiple access (CDMA)*, low rate digital speech coding and bandwidth efficient modulation for more efficient use of the limited frequency spectrum. In addition to voice transmission, they offer such capabilities as point-to-point short messaging, broadcast messaging, and group addressing, etc.. The major standards are [2] the *Pan-European Global System for Mobile communications (GSM)* and *DCS1800*; *IS54* based on TDMA and *IS95* based on CDMA in North America; and *Personal Digital Cellular (PDC)* in Japan. Also, to provide increased user capacity in a quasi-static environment, digital cordless technology standards *CT-2 (second generation cordless telephone)* and *DECT (Digital European Cordless Telephone)* in Europe, and several different Industrial Scientific Medical band Technologies in the United States have evolved. These systems are, however, not compatible.

At the present time, work is in progress to evolve a third generation of cellular systems which will be capable of offering better voice quality (comparable to the fixed network), a suitable range of video and data services, comparable performance in vehicular, portable, rural, urban and indoor environments, and provision for wider mobility. Several wireless technologies currently provide wireless personal communications, but they need to overcome significant existing limitations. For example, digital cordless has a limited range;

digital cellular needs better voice encoding, has marginal circuit quality, limited data capabilities, holes in coverage and poor in-building coverage; wide area data needs to overcome a limited data rate, no voice capability and high cost; Wireless Local Area Networks (WLANs) have limited coverage and have no standards; paging and messaging systems are leading towards two way messaging and/or voice [3]. This implies that a lot of work is still required.

1.2 Digital Cellular System (IS54)

Amidst the continuous evolution of new wireless communication technologies, efforts are also being made to improve the performance of existing cellular systems. The proposed second generation digital cellular communication system for North America (IS54) is currently being introduced into the market. The performance of this system is degraded significantly by the radio propagation channel [4]. This makes it essential to investigate and incorporate a suitable technique to combat the adverse effects of the propagation channel. This section presents the IS54 standard. The effects of the radio propagation channel on transmitted information signal are discussed in the next section. This is followed by a proposed technique to counteract the channel impairment. Finally, a thesis overview is presented.

The introduction of IS54 in the market is being driven by the need to serve the increasing number of subscribers. Instead of increasing system capacity by reducing the cell size and erecting more base-stations, which is a difficult and expensive solution, the use of digital technology was proposed by the EIA/TIA (Electronic Industry Association/Telecommunication Industry Association) subcommittee TR45.3 in the form of Interim Standard IS54. This is also known as the *North-American Dual Mode Cellular System (NADMCS)* because of the dual nature of the mobile stations that ensures compatibility with the previously adopted AMPS analog system specifications. In less densely populated areas, where conversion from analog to digital can be slower (because the increased capacity offered by the digital system may not be required immediately), there exist a mix of analog and digital terminals as well as different types of base station equipment. Systems using the IS54 standard thus allow a smooth transition from the analog to the digitally implemented services to roaming subscribers.

The access scheme for IS54 is TDMA, and not FDMA as used in AMPS, because it allows easy transition to the new system enabling no change in the frequency plan of the old system. Also, with TDMA, the *mobile station assisted handover* (MAHO) procedure becomes faster and more accurate. In this process, the mobile unit measures the signal strength on channels from near-by base stations and reports them to its current base station. This reduces the signaling load between the base stations and the mobile services switching center (MSC). The time division multiple access scheme also facilitates the use of a flexible data rate by assigning the amount of time required (i.e. half the number of time slots for half rate channels) by each user.

The bit rate of a speech codec determines the capacity of a cellular system. The lower the bit rate, the lower the amount of spectrum consumed for one connection, permitting more simultaneous connections within the system bandwidth. The current implementation of the speech coding algorithm is *vector sum excited linear prediction coding* (VSELP) which uses 7.95kb/s for speech coding alone and 13kb/s with error protection [5].

The IS-54 system retains the 30kHz channel bandwidth of AMPS with a transmitter frequency in the 824-894MHz band. Each frequency channel provides a raw RF bit rate of 48.6kb/s. The modulation scheme has a spectral efficiency of 1.62b/s/Hz. This is achieved using a linear modulation method, called *π /4 shifted differential quadrature phase shift keying* (π/4 DQPSK), at a 24.3 kbaud channel rate. This has various advantages over other linear modulation schemes such as QPSK, OK-QPSK [6]. Since the encoding is differential, loss of data due to phase slips is reduced, enabling the use of simple non-coherent receivers. A *square root raised cosine* pulse shaping filter with *roll-off factor* equal to 0.35 is used to limit the bandwidth of the transmitted signal at the transmitter.

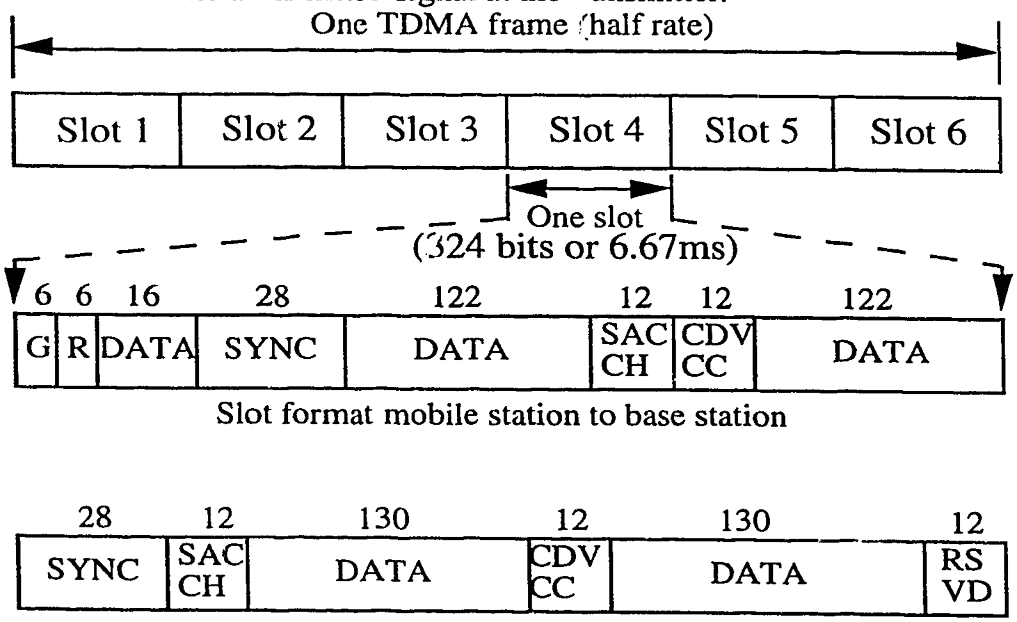


Fig. 1.2. Slot and frame structure for IS54.

The frame and time slot structure for each traffic channel in IS54 is shown in Fig. 1.2. The TDMA frame has a duration of 40ms for half rate channels, and the time slot duration is 40/6 ms for both full rate and half rate channels. The time slot formats for traffic to and from the base station are different. Each time frame has a length of 1944 bits (972 symbols) and is divided into 6 slots of exactly 324 bits (162 symbols) in length. Each full rate traffic channel uses two equally spaced time slots of the frame, while each half rate channel uses one time slot of the frame. At a bit rate of 48.6 kbits/sec, 25 frames per second are transmitted.

For the purpose of this research, most of the parameters are taken from this standard. Because of the limited capacity of the available hardware and ease of implementation, some changes are made. The details of the relevant parameters that are selected are covered in later chapters.

1.3 Propagation Channel

The radio signal transmitted from a mobile radio base station reaches a mobile terminal receiver after passing through a very harsh propagation channel. Because of terrain features and man-made structures, a line-of sight between the transmitting and receiving antennas seldom exists. Multipath propagation caused by reflection, diffraction and scattering is the principal mode of propagation for the information signal. The time-varying multiple paths formed by irregular terrain cause the signal energy to reach the receiver with different time delays, attenuations and phases. This gives rise to severe distortion of the information carried by the channel and results in many problems and limitations for mobile radio systems. To design an optimum method of mitigating the impairments caused by the complex time-varying multipath transmission channel, it is necessary to understand the impairments and to determine under what conditions their effects are most prominent.

1.3.1 Path Loss

The spatial separation between the transmitters and the receivers causes path-loss effects. This is primarily due to terrain effects and the presence of *radio* *wave scatterers* along the path within the mobile radio environment. Antenna height, topography of the coverage area (hills, valleys), reflective properties of the surrounding structures (walls, buildings), and path obstructing features of intervening structures (building penetration), are some of many factors that affect path loss. Based on path loss measurement models, a tradeoff is made between coverage area and link power for a particular antenna design and radio frequency. Fig. 1.3 shows the path loss region.

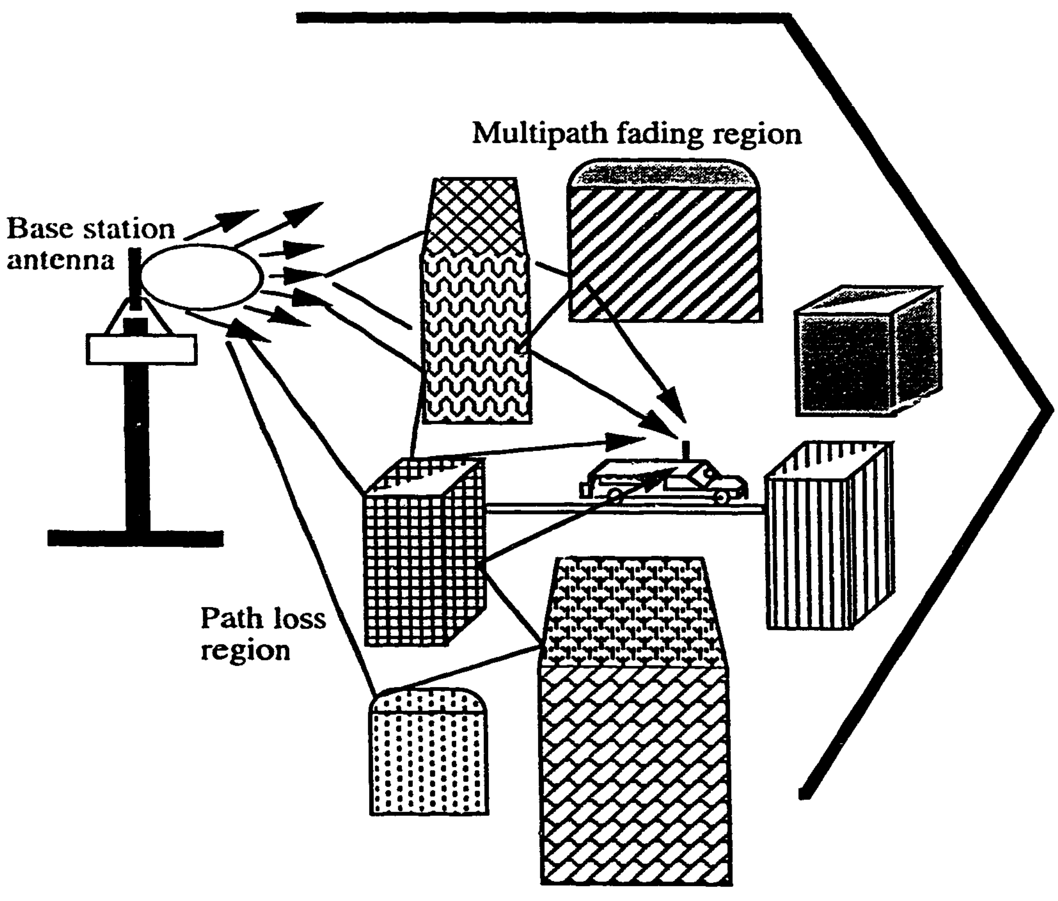


Fig. 1.3. Mobile radio environment.

1.3.2 Fading

The signal received by the mobile station consists of large number of plane waves whose amplitudes, phases and angles of arrival, relative to the direction of vehicle motion, are random. Their interference can be constructive or destructive. Fig. 1.4 illustrates this phenomenon.

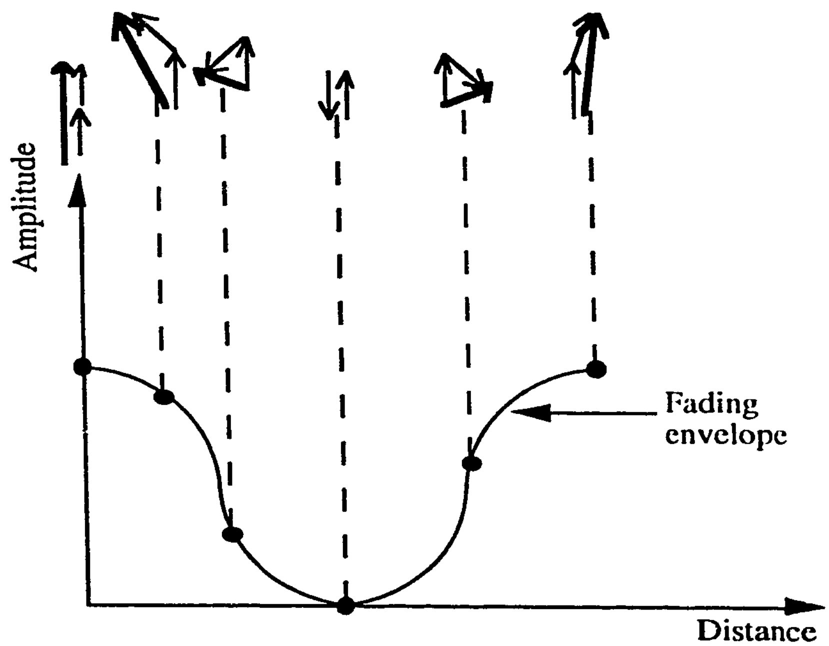


Fig. 1.4. Fading phenomenon.

Plane waves interfere and produce a varying field strength pattern with maxima and minima spaced about a quarter wavelength apart. With the short wavelengths of mobile radio frequencies, the received signal fades rapidly and deeply as the mobile moves around. Some fades are as low as 40dB below the mean signal strength. By reciprocity, the base station receiver experiences the same rapid fading. Signal fluctuation due to multipath effects caused by man-made structures in a local area is called *short term fading*. It is rapid and short lived as the mobile moves around. *Long term fading*, on the other hand, is a slow variation mainly due to topology. *Shadowing*, where the mobile unit is located in the shadow region of mountains, hills etc. is also a cause of long term fading. The severity of multipath fading is shown in Fig. 1.5, which presents a typical segment of a fading signal received at a mobile unit. This is a cause of signal degradation and poor quality at receiver. The statistical characteristics of fading must be studied in order to find suitable methods for counteracting fading effects.

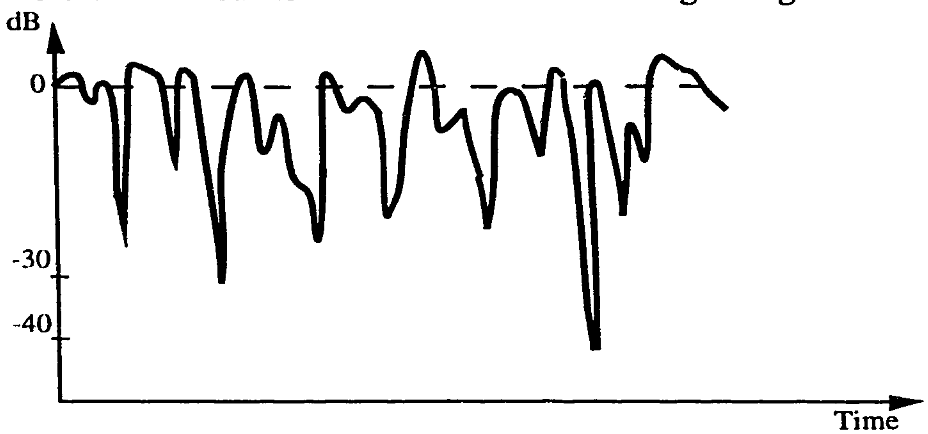


Fig. 1.5. Typical received signal strength variation.

1.3.3 Doppler effect

Vehicle motion relative to the base station transmitter causes a Doppler shift in the carrier frequency of the received signal. The amount of Doppler shift depends on the speed and direction of travel of the mobile unit with respect to the received signal and carrier frequency. Its value is very much less than the carrier frequency and determines the rate of short term fading, i.e., fades/sec. This creates a random frequency modulation of the transmitted carrier.

1.3.4 Delay spread

Random reflections and scattering that result in multipath cause replicas of the transmitted signal to reach the receiver with different propagation delays. As shown in Fig. 1.6, a single sharp transmitted pulse would be received as a smeared and spread out form of the transmitted signal. This time dispersion, together with the symbol duration, determines if the time spreading is large enough for each symbol to overlap with its adjacent symbol to produce intersymbol interference (ISI).

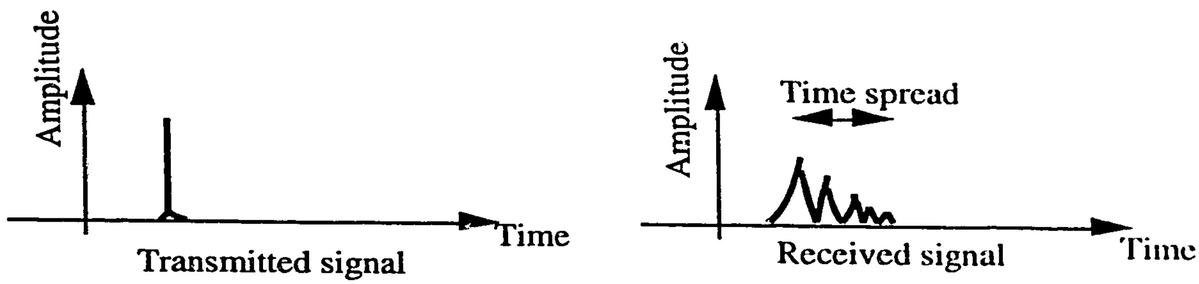


Fig. 1.6. Delay spread phenomenon.

The average rms or worst case value of the delay spread is dependent upon the geometric relationships between the transmitter, the receiver and the surrounding physical environment. Typically, the rms delay for an indoor environment is between 20ns to 300ns [7]. For an outdoor environment it is less then 8μs for urban areas and between 10μs to 30μs for hilly and mountainous areas. For a delay spread δ, symbol time T, and signal bandwidth W, the channel is dispersive in time or selective in frequency when $\delta >> T$ and $\delta >> 1/W$. The channel is frequency selective because, at the same time, a signal fades different amounts for different frequencies within the bandwidth.

1.3.5 Noise

On a radio link electrical noise generated by external sources enters the receiver through the antenna. Internal noise, such as thermal noise, in the receiver, also contributes to the total noise in the system. Sources of man-made noise at the base station are automotive noise, power generating facilities, industrial equipment, consumer products like lighting systems, medical equipment, electrical trains and buses. When selecting the base station site and antenna height, its surroundings are considered for noise level. All the noise sources combined together are generally characterized as additive white Gaussian noise (AWGN) [8].

1.4 Decision-Aided Detection

A receiver based on the conventional differential detector exhibits significant performance degradation due to the detrimental effects of the radio propagation channel. Considerable efforts have been made in the past to reduce the undesired effects of the multipath phenomenon. Techniques such as pilot symbol insertion, error correction coding and diversity can be used for performance improvement [9]. Some other signal processing schemes have also been proposed in the literature that are based on channel prediction, using the transmitted symbols alone [10, 11].

In this work, a receiver based on a decision-aided channel estimation and error correction algorithm has been investigated for $\pi/4$DQPSK modulation. Decision-aided feedback is used to estimate the phase variations due to a flat fading channel. Since, channel phase variations directly contribute to demodulation errors, estimation and removal of the channel phase variations can greatly improve the quality of the received signal. The channel estimation relies on the fact that typically, 99 out of 100 received symbols are detected correctly by the differential detector. Thus, the differential detector decisions (first decisions) may be used by a *reverse modulator* to remove the modulation from the received signal and recover the channel phase information. This extracted phase information is also typically correct 99 out of 100 times. To reduce the effect of incorrect first decisions on the extracted channel, a low pass *channel estimation filter* is used. The smoothed channel characteristic, along with the differential detector decisions are then used for making a new decision for the data, thereby correcting many of the errors made in the first decision. The reverse modulation is done by complex conjugate multiplication of the baseband received signal vector and the first decision signal vector, and the channel estimation filter is a simple FIR filter. This technique can be implemented in real time using a *digital signal processor (DSP)*. Decision aided detection has been shown to be an effective error correcting method for binary phase shift keying (BPSK) [12, 13]. Its performance for $\pi/4$DQPSK modulation as per the IS54 standard is studied in this project, in order to achieve improved error performance for the North American digital cellular standard.

1.5 Thesis Overview

This chapter has presented the various deleterious effects of the radio propagation channel on the signal transmitted as per digital cellular standards. The objective of this research is to improve the performance of IS54 based systems through the use of a decision-aided channel estimation and error correction algorithm.

Chapter 2 describes the basic structure of a digital transceiver and details of the channel model. The effect of a flat fading single path channel on the bit error rate, causing an irreducible error floor, is described next. An overview of channel compensation techniques which can be employed to improve system performance are presented.

Chapter 3 explains the structure of the decision-aided detector which is investigated in this thesis. The functional characteristics of each component in the channel estimator and error corrector module are described.

Chapter 4 presents the experimental implementation of the digital transceiver with the decision-aided detection, using two *Texas Instruments* DSPs. The test results obtained from the implemented detector are presented and discussed in chapter 5.

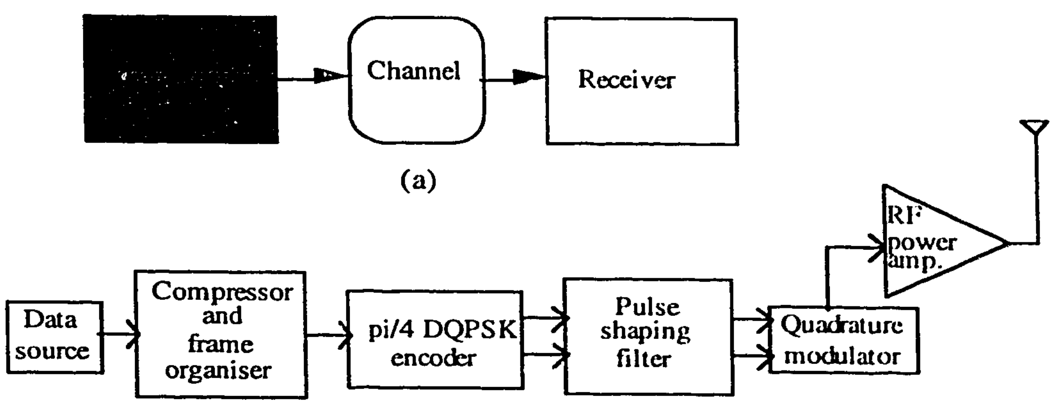
Chapter 6 summarizes and discusses the research results, and outlines areas for possible future research work.

2. TRANSCEIVER STRUCTURE AND ERROR FLOOR REDUCTION TECHNIQUES

In this chapter, the basic structure of a π/4DQPSK transmitter and model of the fading channel used, are described. The structure of the demodulator and differential detector used to recover the transmitted data are presented next. The effect of the channel on the bit error rate and the occurrence of an irreducible error floor is then discussed. This is followed by background information on various methods for lowering the irreducible error floor, including a brief discussion on their performance and limitations in combating fading effects.

2.1 Structure of π/4DQPSK Transmitter

The basic structure of the communication system and a simplified model of the transmitter based on the IS-54 standard are shown in Fig. 2.1a and b. Each module of the transmitter is described briefly in the following sections:



(b)

Fig. 2.1. (a) Basic communication system model. (b) Transmitter structure.

2.1.1 Data Source

The data source is a binary digital information source such as digitized speech (64kb/s) or digital data, which is to be transmitted.

2.1.2 Data Compressor and Frame Organizer

The data compressor is used to eliminate redundancy in the speech signal, so as to reduce its bit rate. According to the IS54 standard, the *vector sum excited linear predictive (VSELP)* coding technique is used to compress the digitized voice signal from 64kb/s to 7.95kb/s [5]. This information payload, along with overhead bits and signaling bits, is then mapped onto a standard frame format, to produce a 48.6kb/s serial bit stream.

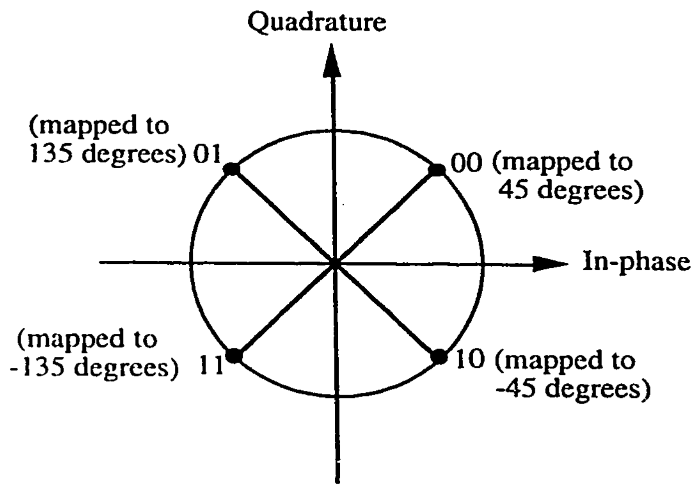


Fig. 2.2. Gray coding for π/4DQPSK.

2.1.3 π/4DQPSK Encoder

The serial data stream is encoded into 2-level symbols using a Gray code format, as shown in Fig. 2.2, to reduce the probability of bit error. Each symbol is encoded differentially onto the in-phase and quadrature components (SI and SQ) of the equivalent complex baseband signal. The differentially encoded signal is obtained in accordance with the π/4DQPSK modulation format of Table 2.1.

TABLE 2.1
RELATION BETWEEN SERIAL INPUT DATA AND DIFFERENTIAL PHASE ANGLE.

Input Data (b_1, b_0)	Differential Phase Angle ($\Delta\phi$)
0,0	$+\pi/4=+45^o$
0,1	$+3\pi/4=+135^o$
1,0	$-\pi/4=-45^o$
1,1	$-3\pi/4=-135^o$

Fig. 2.3 shows the signal constellation of $\pi/4$DQPSK modulation. The eight phase points can be considered to be formed by superimposing two quadrature phase shift keying (QPSK) signal constellations offset by 45^o relative to each other. During each symbol period, a phase angle from one of the two QPSK constellations is transmitted. The two constellations are used alternately, so that each successive symbol is mapped into the phase changes given in Table 2.1.

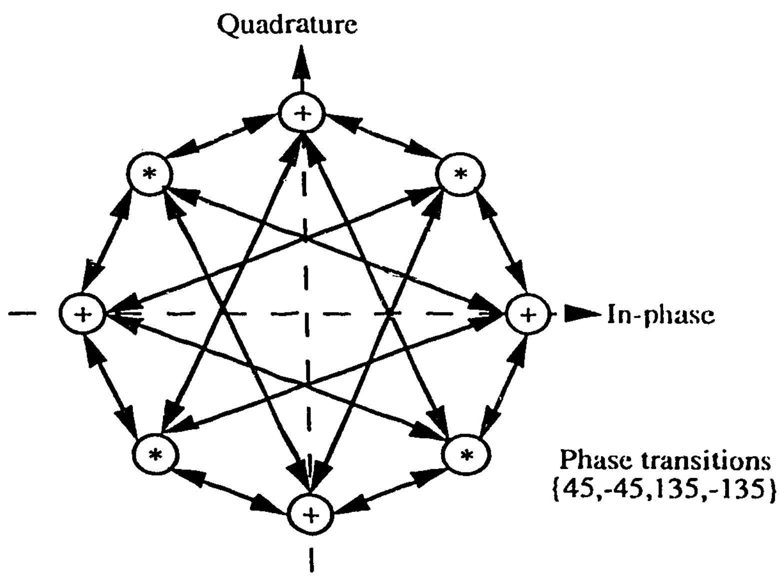


Fig. 2.3. Constellation for π/4DQPSK modulation.

2.1.4 Pulse Shaping Filter

The differentially encoded signals SI and SQ are passed through a baseband filter to limit the bandwidth of the transmitted signal. According to the IS54 standard, a *square root raised cosine (SRRC) filter* with a roll-off factor (α) of 0.35 should be used. The frequency response of this type of filter is defined as [14]

$$|H(f)| \equiv \begin{cases} 1 & 0 \le |f| \le \dfrac{1-\alpha}{2T} \\ \sqrt{\dfrac{1}{2}\left\{1-\sin\left[\dfrac{\pi(2fT-1)}{2\alpha}\right]\right\}} & \dfrac{1-\alpha}{2T} \le |f| \le \dfrac{1+\alpha}{2T} \\ 0 & |f| \le \dfrac{1+\alpha}{2T} \end{cases} \tag{2-1}$$

where α is the roll-off factor and T is the symbol duration. Fig. 2.4 shows the frequency response for a symbol rate of 1.2ks/s with $\alpha = 0.35$.

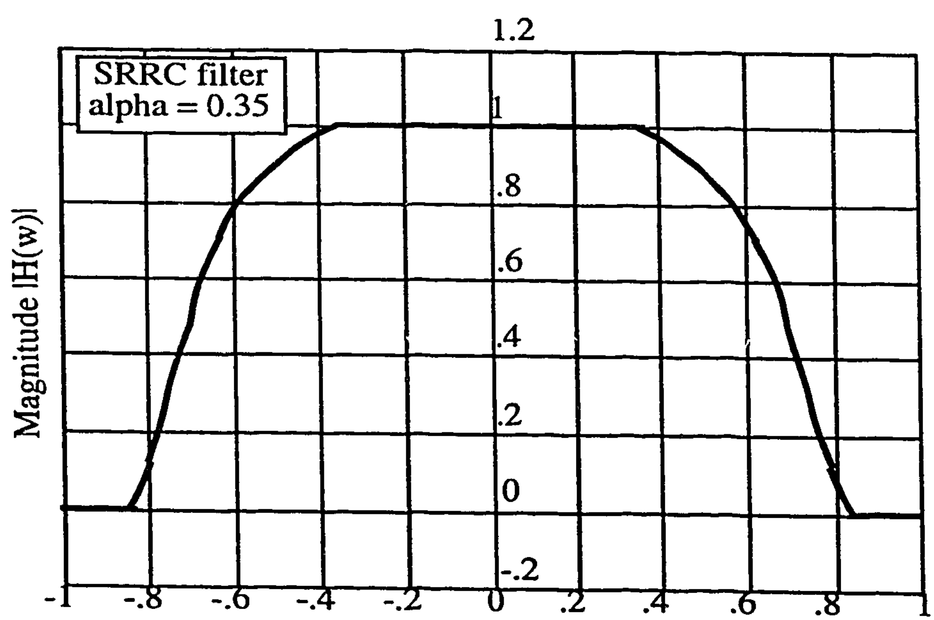


Fig. 2.4. Frequency response of pulse shaping filter.

2.1.5 Radio Frequency (RF) Section

The RF section in the transmitter has a quadrature modulator, a power amplifier and an antenna. The main components of the quadrature modulator are the oscillator which generates the carrier frequency, and the phase splitter, mixers and in-phase combiner which constitute the modulator. The carrier frequency is specified in the IS54 standards [5], as shown in Table 2.2.

TABLE 2.2
IS54 SPECIFICATIONS FOR CARRIER FREQUENCY.

Transmitter	Channel Number	Center Frequency (MHz)
Mobile	$1 \le N \le 799$	$0.030N + 825.000$
	$990 \le N \le 1023$	$0.030(N - 1023) + 825.000$
Base	$1 \le N \le 799$	$0.030N + 870.000$
	$990 \le N \le 1023$	$0.030(N - 1023) + 870.000$

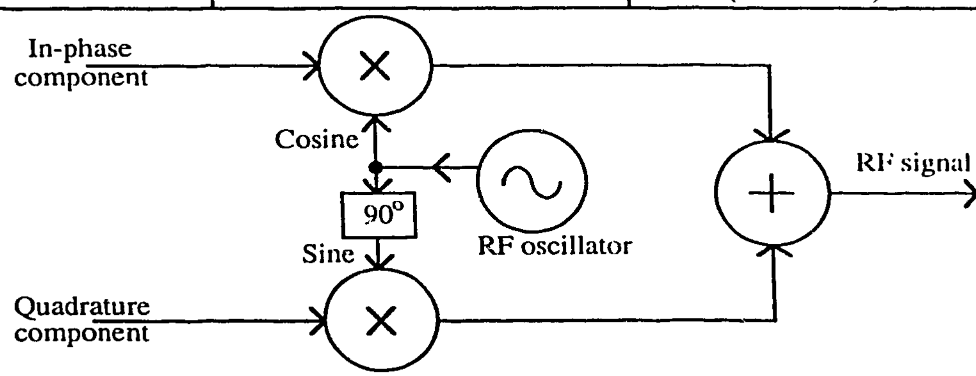


Fig. 2.5. Quadrature modulator.

The output of the oscillator is split into two components with a 90° phase shift, producing carriers that may be represented by $\sin(w_C t)$ and $\cos(w_C t)$. The mixers multiply these phase-shifted signals by the in-phase and quadrature filtered baseband signals, as shown in Fig. 2.5. The output of the mixers are then added together by the combiner, resulting in the modulated RF signal. Finally, the signal is amplified and fed to the antenna. If h(t) is the time domain response of the square root raised cosine filter, then the resultant phase modulated transmitted signal is given by

$$s(t) = \sum_{n} h(t - nT)\cos(w_c t + \phi_n) \tag{2-2}$$

where w_c is the carrier frequency of transmission and ϕ_n is the absolute phase angle for the n^{th} symbol period (i.e. $[n+1]T \geq t \geq nT$).

2.2 Radio Propagation Channel

Multipath propagation has already been mentioned as a principal feature of mobile radio channels. The characterization of multipath phenomena is presented in this section. Since the transmitted signal in the case of IS54 is narrow-band, and the delay spread in urban areas is less than 8 μsec, all the signal frequency components are affected by the channel in a similar way. The fading is therefore called *flat fading*, implying that there is no frequency-selective behavior. Several multipath models have been proposed to explain the observed statistical characteristics of the radiated electromagnetic fields, and the associated signal envelope and phase [15]. However, they all lead to comparable statistical properties of the field for large numbers of constituent waves. A model based on scattered waves, describing the characteristics of the signal received by an antenna mounted on a mobile unit, is considered here.

2.2.1 Scattering Model

If the transmitted signal is vertically polarized, the field components seen at the receiver would normally be horizontally traveling waves. In general, due to scattering and diffraction the waves travel in other directions as well [16]. Since the wavelength of the signal is relatively small, each reflection may be considered as a ray. The signal at the receiving point is thus the resultant of N plane waves caused by multipath reflections of the transmitted wave. Each plane wave has a random amplitude and angle of arrival at different locations. A typical component wave is shown in Fig. 2.6 with a spatial frame of reference.

The n^{th} incoming wave has an amplitude C_n and phase ϕ_n with respect to an arbitrary reference, and spatial angle of arrival α_n and β_n. The parameters C_n, ϕ_n, α_n and β_n are all random and statistically independent. The phase angles are assumed to be uniformly distributed from 0 to 2π. The amplitude C has a mean square value of

$$E\{C_n^2\} = \frac{E_o}{N} \tag{2-3}$$

where E_o is a positive constant.

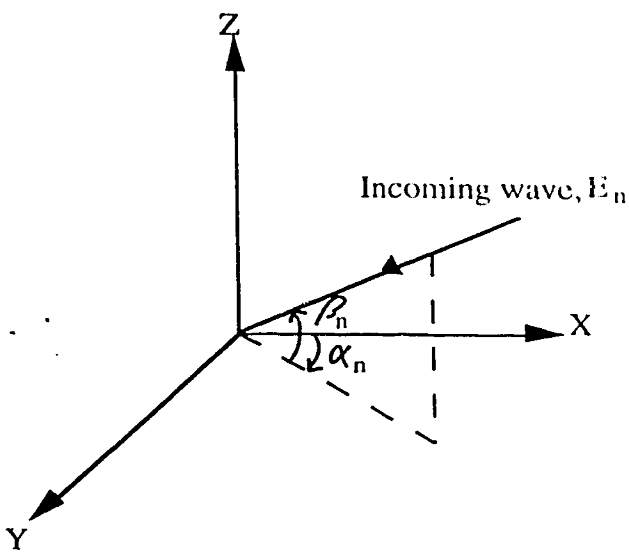


Fig. 2.6. Scattering model.

For N such waves, the resultant field at any receiving point (x_o, y_o, z_o) is

$$E(t) = \sum_{n=1}^{N} E_n(t). \tag{2-4}$$

If the transmitted signal is an unmodulated carrier, then

$$E_n(t) = C_n \cos(w_o t - \frac{2\pi}{\lambda}[x_o \cos\alpha_n \cos\beta_n + y_o \sin\alpha_n \cos\beta_n + z_o \sin\beta_n] + \phi_n) \, . \tag{2-5}$$

The mobile receiving point is assumed to be moving with a velocity υ in the x-y plane in a direction making an angle γ to the x-axis. The co-ordinates of the receiving point after a unit time are ($\upsilon \cos \gamma$, $\upsilon \sin \gamma$, z_o) and the received field is now given by

$$E(t) = CI(t)\cos w_c t - CQ(t)\sin w_c t \tag{2-6}$$

where CI(t) and CQ(t) are the in-phase and quadrature component that would be detected by receiver, w_c is the carrier frequency, and λ is the wavelength of the transmitted carrier frequency. Here,

$$CI(t) = \sum_{n=1}^{N} C_n \cos(w_n t + \theta_n) \tag{2-7}$$

$$CQ(t) = \sum_{n=1}^{N} C_n \sin(w_n t + \theta_n) \tag{2-8}$$

$$w_n = \frac{2\pi\upsilon}{\lambda}\cos(\gamma - \alpha_n)\cos\beta_n \tag{2-9}$$

$$\theta_n = \frac{2 z_o \pi}{\lambda}\sin\beta_n + \phi_n\ . \tag{2-10}$$

Due to the motion of the receiver, each wave experiences a Doppler frequency shift. The Doppler frequency will have a maximum value of $f_m = \pm\frac{\upsilon}{\lambda}$, a value which is very much less than the carrier frequency (typically f_m is less than 100Hz). If the value of N is sufficiently large (> 6), then, by the central limit theorem, the quadrature components CI(t) and CQ(t) will be independent Gaussian processes which are completely characterized by their mean value and autocorrelation function. The mean of E(t) is zero, since the mean values of both CI(t) and CQ(t) are zero. Also, CI(t) and CQ(t) have an equal variance, σ^2, equal to the mean square value or the mean power.

The probability distribution function of CI and CQ is thus [17]

$$p_x(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp(-\frac{x^2}{2\sigma^2}) , \tag{2-11}$$

where x=CI(t) or CQ(t) and $\sigma^2 = E\{C_n^2\} = \frac{E_o}{N}$.

2.2.2 Channel Characteristics

The Gaussian model described above may be used to predict the field-measured statistics of the signal. These statistical properties can be used to duplicate multipath characteristics in an experimental laboratory study or for the purpose of simulation. These characteristics of the channel are important parameters for the design and evaluation of different radio systems.

2.2.2.1 Power Spectra

The spectrum of the received signal is strictly bandlimited to the range $\pm f_m$ around the carrier frequency, and is given by [18]

$$A_0(f) = \begin{cases} \dfrac{E_0}{4\pi f_m\sqrt{1-\left(\dfrac{f}{f_m}\right)^2}} & for \ |f| \le f_m \\ 0 & elsewhere \end{cases} \tag{2-12}$$

where f_m is the maximum Doppler shift. It may be noted that, the power spectral density becomes infinite at f_c+f_m and f_c-f_m but is immediately zero outside these limits. For a typical mobile radio system the value of the maximum Doppler shift f_m will be in the range 10 to 100Hz; although this frequency shift is small, it may cause deterioration of the message information.

2.2.2.2 Statistics of Received Signal Envelope and Phase

The envelope r(t) of the complex baseband signal E(t), received at a radio receiver is given by

$$E(t) = CI(t) + jCQ(t) \tag{2-13}$$

$$r(t)=\sqrt{CI^2(t)+CQ^2(t)}\ . \tag{2-14}$$

Since CI and CQ are both Gaussian distributed with equal variances σ^2, then r(t) has a Rayleigh probability density function (pdf) [19]

$$p_r(r)=\frac{r}{\sigma^2}\exp(-\frac{r^2}{2\sigma^2}) \tag{2-15}$$

where the mean power for the Rayleigh pdf is $2\sigma^2$. The probability that the envelope does not exceed a specified value R is given by the cumulative distribution function

$$prob(r \leq R) = P_r(R) = \int_0^R p_r(r)dr = 1-\exp(-\frac{R^2}{2\sigma^2})\ . \tag{2-16}$$

The received signal phase is a uniformly distributed random process, given in terms of CI(t) and CQ(t) by

$$\theta(t)=tan^{-1}(\frac{CQ(t)}{CI(t)}) \tag{2-17}$$

with $$p_\theta(\theta)=\frac{1}{2\pi}\quad , \qquad 0\leq\theta\leq 2\pi \quad . \tag{2-18}$$

Since the signal is composed of a number of components of random phase, the resultant phase is random and takes on all values in the range (0, 2π) with equal probability. The probability distribution function gives information such as the overall percentage of time, or the overall percentage of locations, for which the envelope lies below a specified value. However, it does not provide information about how this time is distributed. The level crossing rate (LCR) and average fade duration (AFD) are used to provide a quantitative measure of this property.

2.2.2.3 Level Crossing Rate and Average Fade Duration

The level crossing rate at any specified signal level is the expected rate at which the envelope crosses a specified level in the positive-going (or negative-going) direction. Mathematically, the level crossing rate is

$$n(r=A)=\frac{N}{T} \tag{2-19}$$

where N=total number of crossings over T second length of data. It can be shown that the theoretical LCR is given by [15]

$$n(r=A)=\sqrt{\frac{\pi}{\sigma^2}}Af_m\exp(-\frac{r^2}{2\sigma^2}) \tag{2-20}$$

where A is a specified level, f_m is the maximum Doppler shift and σ^2 is the mean power of the envelope. The average fade duration ($\bar{\tau}$), is used to quantitatively measure the average period of a fade below a specified level A. Mathematically, AFD is given by the total duration of N fades below level A, divided by N,

$$\bar{\tau}(r=A)=\frac{\sum_{i=1}^{N}t_i}{N} \tag{2-21}$$

where t_i is the duration of the i^{th} individual fade. Theoretically, [15]

$$\bar{\tau}=\sqrt{\frac{\sigma^2}{\pi}}\frac{\exp(\frac{A^2}{2\sigma^2})-1}{Af_m} \quad . \tag{2-22}$$

2.2.2.4 Envelope Correlation

The envelope correlation characteristic, based on time and space separation, has its importance in *diversity* and other applications. The correlation coefficient based on time separation is given by [20]

$$\varsigma_r(\tau)=J_o^2(\beta v\tau) \tag{2-23}$$

where J_o is the Bessel function of the first kind of order zero and τ is the time separation between envelopes. The correlation coefficient will be zero whenever $J_o(\beta v\tau)$ is equal to zero, implying that for certain values of τ, the probability that two envelopes of the same signal, separated by τ, fade simultaneously is zero. The first zero occurs when $v\tau/\lambda = 0.38$, which happens when the time separation is equal to $\tau = 0.38\lambda/v$. Spatial correlation is related to time correlation by $d = v\tau$, which therefore implies that two antennas separated by $d = 0.38\lambda$ will receive uncorrelated envelopes.

2.2.3 Channel Model

The simulation model for fast flat fading that characterizes the statistical properties described above is shown in Fig. 2.7. Its implementation and performance is described in Chapter 4 and 5. Two Gaussian random generators are used to create an equivalent complex baseband signal with a Rayleigh distributed envelope. The Gaussian processes are fed into two lowpass filters with cutoff frequency equal to the maximum Doppler frequency in order to produce the correct amplitude versus frequency characteristic for the fading channel. The outputs of these shaping filters are the in-phase component, CI(t), and the quadrature phase component, CQ(t), of the channel. Their combined effect on the transmitted signal is to produce a uniformly distributed phase, a Rayleigh envelope and the required Doppler power spectrum.

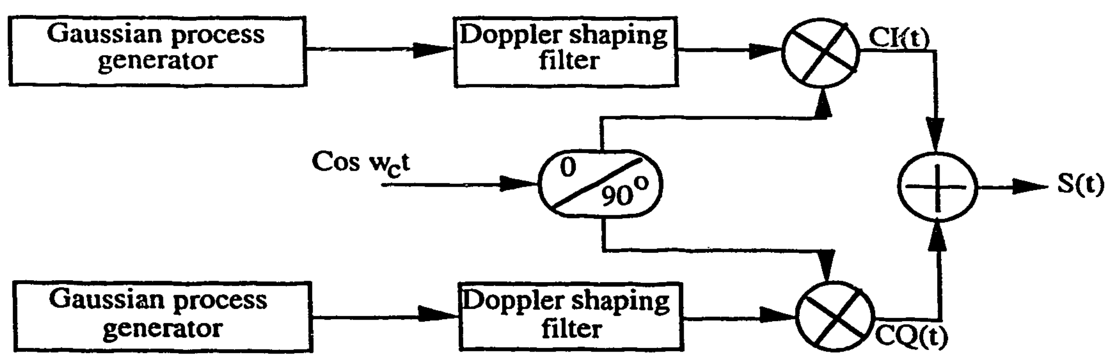


Fig. 2.7. Radio channel model.

2.3 Structure of $\pi/4$DQPSK Receiver

2.3.1 Demodulator Section

This is very similar to the transmitter RF section in terms of components, but the sequence of operation and the signal flow are in the opposite direction. The demodulator recovers the base-band signal and clock from the modulated carrier. Base-band recovery is accomplished by multiplying the incoming signal with the local oscillator signal, and then low-pass filtering.

The receiver filter used is exactly the same as that used in the transmitter. The cascade of the two square root raised cosine filters gives an over all raised cosine filter which, as per Nyquist's second theorem, is a zero ISI filter. That is, the impulse response of this filter passes through zero at all sampling intervals except the desired symbol instant. The zero ISI condition holds true only for the ideal condition of no fading and distortion. The other purpose of the receiver lowpass filter is to limit the noise and remove unwanted frequency products resulting from the frequency down-conversion process.

The next stage is the sampler, which converts the signal to a discrete time signal at a sample rate equal to the symbol rate; the sampling frequency and timing are determined by the timing recovery circuitry. The clock recovery circuit is used to maintain the optimum decision point for the data, so that the ISI effect from neighboring symbols is minimized.

2.3.2 Detector

$\pi/4$ shifted, differentially encoded quadrature phase shift keying is amenable to a number of different detection techniques. According to IS54, any detection scheme can be chosen so long as it meets the specified minimum performance. In general, the various proposed detection methods are classified as coherent, differential and discrimination. A coherent detector is more complex to implement then the other methods due to the required additional circuitry for the carrier recovery. However, the performance of a coherent detector is 2-3dB better than that of a non-coherent detector in Gaussian and slowly fading channels. In a fast fading channel however, coherent detection results in a higher irreducible error floor than either differential or discriminator detection. The three non-coherent detection realizations, namely baseband differential detection, IF band differential detection and FM discriminator detection, have an equivalent BER performance [21]. Each method has its own design challenges. In the first method, the design of the local oscillator is quite intricate. If the local oscillator has a frequency difference of Δf relative to the unmodulated carrier, the phase drifts by $2\pi\Delta fT$ during one symbol duration (T). This phase drift causes a performance degradation. In the last two methods, the challenge lies in the design of the bandpass filter, which requires a specified amplitude and phase response [21]. Also, this filter needs to minimize the effect of noise and ISI. The baseband differential detection scheme has been selected for this research project from an implementation simplicity view-point.

2.3.2.1 Baseband Differential Detector

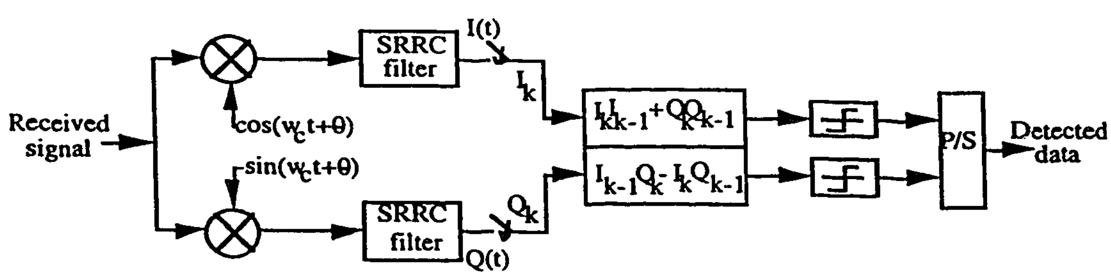


Fig. 2.8. Block diagram of the baseband differential detector.

The block diagram of the baseband differential detector is shown in Fig. 2.8. The local oscillator (LO) is assumed to have the same frequency as the transmitter, and with no phase difference. The effect of a different receiver LO frequency from the transmitter LO is described in [22]. Here, it is assumed that the receiver sampling clock is jitter free and synchronous with the transmitter frequency, and with the full eye opening occuring at t=nT, for the ideal channel. The two branches of the receiver recover the in-phase I(t) and quadrature Q(t) components of the received signal. The detected phase angle of the received signal is given by $\phi = \tan^{-1}[Q(t)/I(t)]$.

At the sampling instant t=kT, assuming that the channel is an ideal Nyquist channel and that the sampler output is I_k and Q_k, then

$$I_k = cos(\phi_k) \tag{2-24}$$

$$Q_k = sin(\phi_k) \tag{2-25}$$

where ϕ_k is the signal phase at t=kT, which is preserved through the Nyquist channel. The change of phase angle over one symbol period contains the transmitted information. The data is detected by determining

$$U_k = \cos(\phi_k - \phi_{k-1}) = I_k I_{k-1} + Q_k Q_{k-1} \tag{2-26}$$

$$V_k = \sin(\phi_k - \phi_{k-1}) = I_{k-1} Q_k - I_k Q_{k-1} \ . \tag{2-27}$$

The U_k and V_k values determine the quadrant of the transmitted signal. The decision for the data is then made based on the Gray code encoder used in the transmitter.

The detected data is determined according to the following conditions:

$$\begin{aligned} &D1_i = 1, \text{ if } U_k > 0; \\ &D1_q = 1, \text{ if } V_k > 0; \\ &D1_i = 0, \text{ if } U_k < 0; \\ &D1_q = 0, \text{ if } V_k < 0; \end{aligned} \qquad (2\text{-}28)$$

where $D1_i$ and $D1_q$ are the two consecutive bits encoded by each symbol.

2.3.3 Performance Measurement

Whenever a signal is detected based on a noisy observation, there is a possibility of making an error. The error probability is defined by,

$$P_e = Pr(D1 \neq D) \qquad (2\text{-}29)$$

where D1 is the detected data and D is the transmitted data.

The performance of a receiver is measured by sending a known sequence of data from the transmitter. The detected sequence at the receiver is then compared to the transmitted data sequence and the number of symbols or bits in error are measured. The signal and noise powers are measured at the detector input in order to calculate the signal to noise ratio (SNR). A good receiver design is one that minimizes the probability of error for a given signal to noise ratio.

2.4 Fading Effect: Irreducible Error Floor

When the received signal is corrupted by the multiplicative fading process that gives rise to the Rayleigh fading phenomenon, the bit error rate is higher than it would be in a Gaussian noise environment. The bit error rate (BER) is a function of the average signal level and the coherence bandwidth based on the delay spread. The BER is, in general, high for a low SNR, but decreases as the average signal level increases. At a certain signal level, however, the BER curve flattens out and the BER becomes constant even though the signal level continues to increase. This bit error rate, called an *irreducible error* or *error floor*, that persists even for high SNR values, is due to *random FM*, as explained below.

Random FM is the direct result of the phase noise introduced by the fading channel. Because the received signal phase, θ, varies with location, movement of the mobile produces a random change of θ with time, equivalent to random phase modulation. This is normally called random FM, because the time derivative of θ causes frequency modulation which will be detected by any phase sensitive detector; e.g. a differential detector or FM discriminator. The detected phase shifts due to the channel appear as noise to the receiver; in terms of the in-phase and quadrature signal components, the FM noise is given by

$$\dot{\theta}=\frac{d\theta}{dt}=\frac{d}{dt}[\tan^{-1}\frac{CQ(t)}{CI(t)}] \tag{2-30}$$

The characteristics[18] of the pdf and power spectrum, of $\dot{\theta}$ are functions of the vehicle speed υ or, alternatively, of the Doppler frequency υ/λ. Although, the highest probabilities occur for small values of $\dot{\theta}$, large excursions due to deep fades can also occur. The random FM spectrum [15, 18] shows that the energy is mainly confined to the frequency range $0 \leq f \leq 2f_m$, from where it falls off as $1/f$; it is insignificant beyond $5f_m$.

Various methods, described in the literature, to improve the performance of the receiver in a fading environment are discussed in the following section.

2.5 Error Floor Reduction Techniques

2.5.1 Pilot Symbol Insertion

The pilot symbol insertion method has the potential to estimate the random phase in a Rayleigh fading system. The block diagram of such a transmitter is depicted in Fig. 2.9. In the transmitter, a known symbol, called a pilot symbol, is inserted in each frame containing N symbols (including the pilot symbol). This is achieved using N-1:1 multiplexers that sample one pilot symbol followed by N-1 data symbols. The block diagram of the corresponding coherent demodulator is depicted in Fig. 2.10a. Assuming perfect frame synchronization, the receiver identifies the pilot symbols through a duplicate generator at its end. The fading distortion is measured using these known pilot symbol as

$$W(kNT) = cos(\theta_{kn})\, U(kNT) + sin(\theta_{kn})\, V(kNT) \tag{2-31}$$

$$Z(kNT) = cos(\theta_{kn})\, V(kNT) - sin(\theta_{kn})\, U(kNT) \tag{2-32}$$

where, U(t) and V(t) are the in-phase and quadrature components of the received signals and W(t) and Z(t) are the estimation of fading channel. θ_{kn} is the phase of carrier at t=kNT. It is assumed here that a jitter free synchronous clock is generated in the receiver, and that the data is ISI free at t=nT.

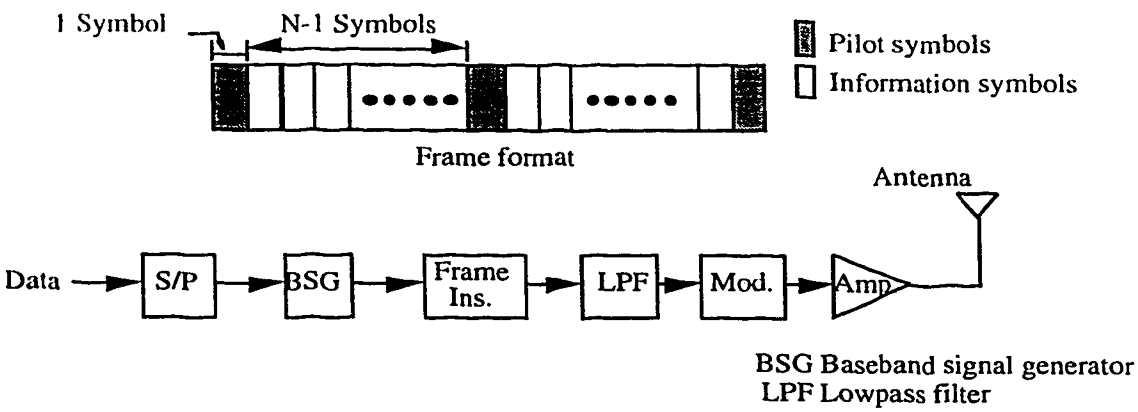


Fig. 2.9. Transmitter for pilot symbol insertion method.

The distortion in the other unknown symbols is compensated by interpolating the sequence of the measured information. Fig. 2.10b shows a more detailed configuration of the fading estimation and compensation section. The fade characteristics are used to recover the phase of the transmitted carrier which contains the information to be detected. A detailed mathematical model, and the analytical and numerical performance of this method have been given by [23,24].

This approach significantly reduces the error floor caused by the Doppler spread in a fast fading channel. However, because the pilot signals do not carry any message information, they represent a waste of power and frequency spectrum.

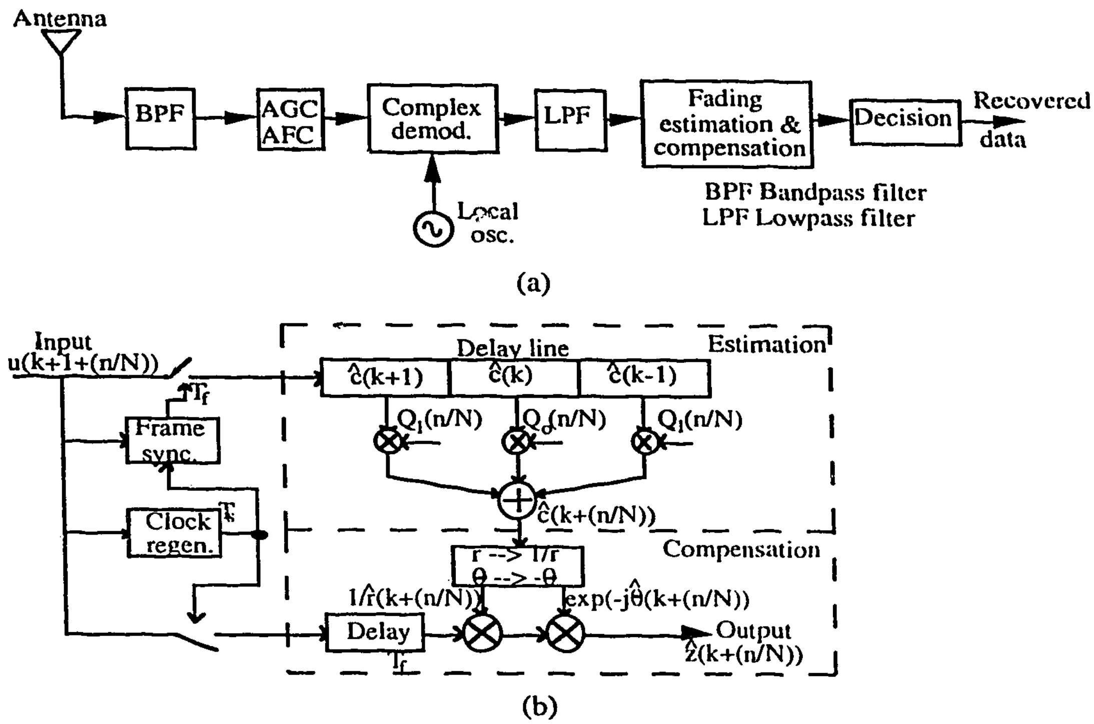


Fig. 2.10. (a) Receiver for pilot symbol insertion method.
(b) Fading estimation and compensation for pilot symbol insertion method.

2.5.2 Reference Tone Approach

Pilot-tone aided coherent demodulation is another efficient method to improve the performance of coherent demodulation in a fast Rayleigh fading environment. Fig. 2.11a is a block diagram of the complex baseband representation of the tone calibrated technique (TCT) system [25]. The transmitted power is split between the data bearing signal and low level pilot tone in such a manner that they do not interfere with each other, for example by separating them in frequency, Fig. 2.11b. Transparent tone in band (TTIB) [26], creates an artificial null for the pilot tone by splitting and separating positive and negative frequency components. The total transmitted power is [25]

$$P_T = \frac{A^2 R_s (1+r)}{2} \tag{2-33}$$

$$r = \frac{a_p^2}{A^2 R_s} \tag{2-34}$$

where R_S is the symbol rate, r is the ratio of pilot tone power to data signal power, A is data signal level, and a_p is the pilot dc level.

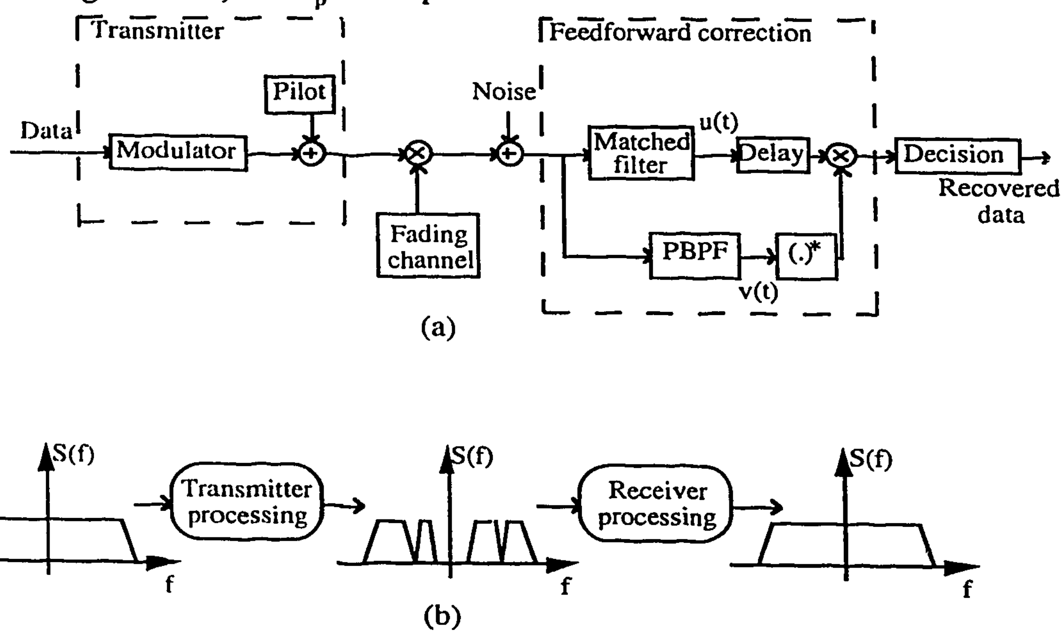


Fig. 2.11. (a) System based on pilot tone insertion.
(b) Pilot tone insertion using spectral manipulation.

The receiver extracts the pilot with a *pilot bandpass filter (PBPF)* and uses the result as a phase reference to demodulate the data coherently. The two principal design tradeoffs are the fraction of the power devoted to the pilot tone, and the bandwidth of the PBPF. If the filter is too narrow-band, the frequency offset and Doppler spread will prevent the filter output from following the channel fluctuations, and if it is too wide it will admit too much noise. The matched filter output for a fading rate $f_d \ll R_S$ is

$$u(kT) = c(kT)Am_k + n_u(kT) \tag{2-35}$$

where c(kT) represents the time varying complex gain of channel, m_k is the information signal and n_u is the noise component. Because the pilot tone is located in a spectral notch, there is no significant pilot tone energy in the matched filter output. The PBPF output v(t) and matched filter output u(t) are used to calculate the decision variable, as shown in Fig. 2.11a, on which the binary decision is made. Further details, analysis and results are given in the literature [25, 26].

The principal virtue of tone calibrated transmission is that it suppresses the irreducible error rate. However, it requires complex signal processing at the receiver to create a spectral null at the carrier frequency or DC in baseband. Also, tone insertion results in inefficient bandwidth utilization since the pilot tone increases the required bandwidth, for the same data rate.

2.5.3 Coding and Interleaving

Coding and interleaving are quite successful approaches to combating channel fading effects. These methods are based on *Shannon's theorem* that a stationary channel can be made arbitrarily reliable, given that a fixed fraction of the channel is used for redundancy. The two major types of error control are *Automatic Request for Retransmission (ARQ)* and *Forward Error Control (FEC)* [27]. Both techniques add redundancy to the data prior to transmission in order to reduce the effect of errors that occur during transmission. ARQ utilizes redundancy to detect errors and, upon detection, it requests a repeat transmission. Two well-known techniques for retransmission are selective repeat and "Go-back N". FEC utilizes the redundancy so that a decoder can extract the correct data from the corrupted received signal. FEC codes have various forms and properties and are broadly classified as *block codes* and *convolutional codes*.

A convolutional encoder operates on the input bit stream such that each information bit can affect a finite number of consecutive symbols; i.e., coded message bits in the encoder output. The functional architecture of the encoder has multistage shift registers, modulo-2 adders, and interconnection between the two as per the constraint length, as shown in Fig. 2.12a. The decoder of these codes has to know the history of the decoded stream, that is, the values held in the shift registers, before it is able to decode a particular bit. Therefore, the decoder is more complex and an error may propagate if the decoder makes a mistake in the history of the stream.

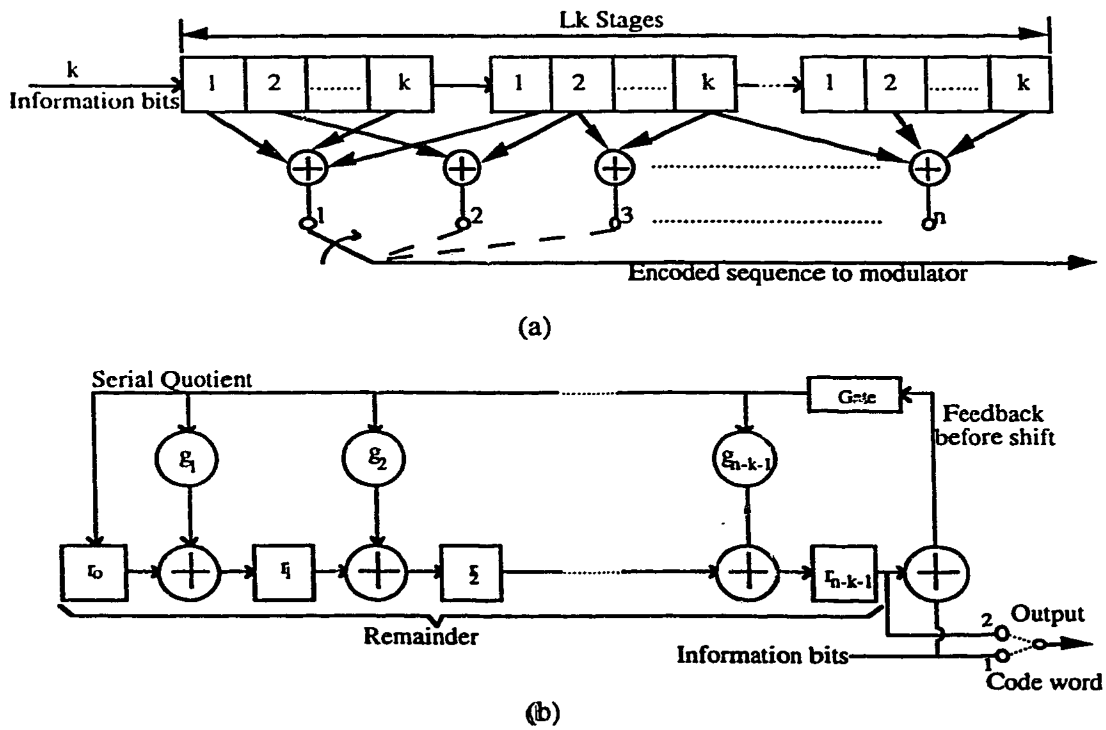


Fig. 2.12. (a) Convolutional encoder. (b) Block encoder.

In block coding, the input data is split into discrete blocks and each block is independently processed by an encoding algorithm to add redundancy and produce a longer block, as shown in Fig. 2.12b. The decoder works on a similar basis; each block is individually processed. For block coding, synchronization is of major concern in the decoder because the block boundaries need to be known.

Efficient use of codes capable of correcting random errors requires the conversion of burst errors into random errors. This is achieved by spacing the individual bits of a each data block far apart in time so that they encounter independent fading. This technique is called interleaving. The simple *block interleaving* uses memory containing M rows and K columns, with a coded stream written into memory row by row, and read out column by column. Successive bits of the coded stream then appear as every K^{th} bit of the transmission stream. In reception, the coded stream is restored by using a corresponding M*K memory as a deinterleaver.

Coding reduces the BER considerably; however the redundant bits used in a bandwidth constrained channel reduces the maximum throughput. Another consideration is that, to improve the performance at the receiver end, the data framing needs to be modified at the transmitter end. This is sometimes not practicable, because frame format is generally set by standards.

2.5.4 Diversity

The effect of fading can be combated by the use of diversity techniques either at the base station or the mobile station. Diversity reception works on the principle that if two or more independent samples of a random process are taken, then these samples will fade in an uncorrelated manner. These two uncorrelated fading signals, received via independently fading paths, are then combined to reduce the effect of the fades. A *macroscopic diversity* scheme is used to reduce the effect of long term fading on the signals received from two or more different antennas at different base station sites. A *microscopic diversity* scheme, on the other hand, is used to reduce the effect of short term or multipath fading on signals received from two or more different antennas, but at only one receiving site.

There are six different types of microscopic diversity; *space, frequency, polarization, field component, angle and time* [28]. In space diversity, two antennas separated physically can provide two signals with low correlation among their fades. The separation required varies with antenna height. A separation of 0.5λ can be used to obtain an almost uncorrelated signal at the mobile unit. As long as the correlation coefficient is less than 0.2, the two signals are considered to be uncorrelated. Frequency diversity is obtained by using two mobile-radio signals using two slightly different carrier frequencies. The frequency separation is chosen to be greater than the coherence bandwidth, so that the two signals fade independently. However, because of the congested frequency spectrum, this is not really a practicable solution. In polarization diversity, the transmitter power is fed to two differently polarized antennas, to transmit orthogonal electric field components. Two polarized antennas receive the two orthogonal signals with different fading effects. However, the splitting of the power at the transmitter results in a 3dB power reduction. Field component diversity is based on the electromagnetic theory concept that, when an E-field is propagating, an H-field is always associated with it and carries the same message information. Energy density antennas are used for field component diversity. Angle diversity makes use of two or more directional antennas pointed in different directions at the receiving site, so as to receive uncorrelated fading signals. Time diversity means transmitting identical messages in different time slots to yield two uncorrelated fading signals at the receiver. The time separation needs to be greater than the coherence time of the channel.

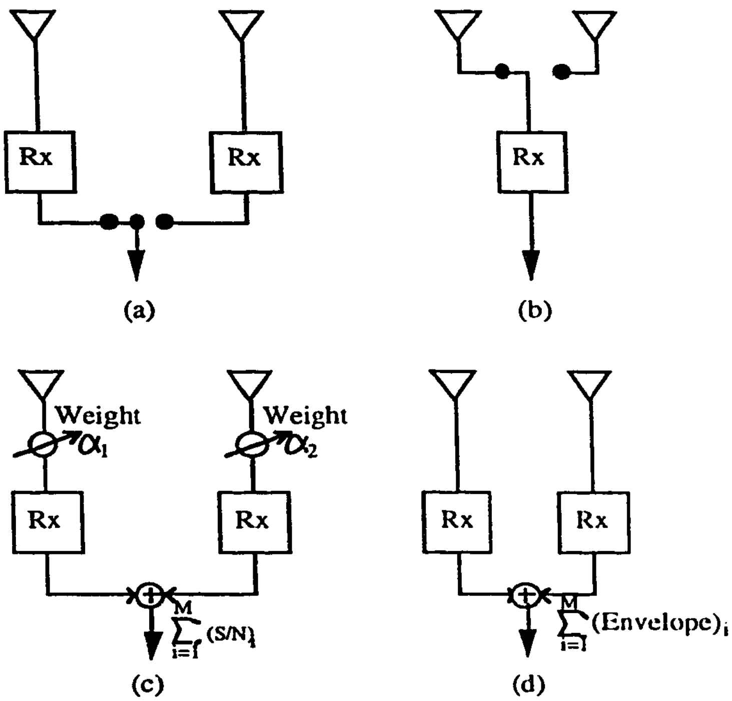


Fig. 2.13. Diversity combiners. (a) Selective combining. (b) Switched combining. (c) Maximal-ratio combining. (d) Equal-gain combining.

After the creation of diversity branches there are four major ways to combine them. These are the *selective, switched, maximal ratio and equal gain combining techniques*, as shown in Fig. 2.13. In selective combining, the strongest signal among all the branches is selected by monitoring each branch with a separate receiving front end. Switched diversity, uses the branch that has a signal level above the threshold value and switches the branch whenever the signal falls below the threshold. It needs only one receiver, but the performance is greatly affected by threshold level and switching noise. Maximal ratio is the best combining technique, as each branch signal is combined with proper weights that depend on the individual signal strengths. When combining signals from all the different branches, the phase and time delay for each branch needs to be taken into account. Equal gain combines the signal of individual branches to get a sum of the instantaneous fading envelopes.

Diversity improves the transmission quality significantly against multipath fading and interference. However it is complex and expensive from an implementation view-point. Also, for a small-size portable unit, it may not be an attractive option, because of limitations due to size and hardware power consumption.

2.6 Summary

This chapter has presented the basic structure of the transmitter, fading channel and receiver. The effect of a flat fading channel on the transmitted signal is discussed, followed by the various available schemes to improve the performance of the receiver. The limitations of each scheme are also presented in terms of bandwidth, power or complexity. This leads us to explore the decision-aided algorithm for detection and error correction, which has a minimum penalty in terms of the above parameters. This algorithm is discussed in the next chapter.

3. DECISION-AIDED CHANNEL ESTIMATION AND ERROR CORRECTION

This chapter proposes a method for reducing the error floor which is based on the data modulation alone, i.e. no additional information bits apart from data bits need to be transmitted. The aim is to achieve a practicable error reduction method, with no overhead and improved performance, that can be implemented using a digital signal processor (DSP) with a small amount of memory requirement and moderate computational complexity. A block diagram and its components are described in the following sections, giving the functional details. At the end of this chapter, a second approach for the detection and correction of errors is discussed.

3.1 Decision-Aided Detection

The basic structure of a receiver with decision-aided detection is shown in Fig. 3.1. A traditional differential detector, as described in Chapter 2, is used to make the first decision. This is then followed by the decision-aided detector which extracts the channel information and makes a second decision on the transmitted data, using knowledge gained from the first decision and the received signal. It is for this reason that the scheme is called *decision-aided detection*. In a test situation, the second decision is compared with the transmitted data to determine the error performance.

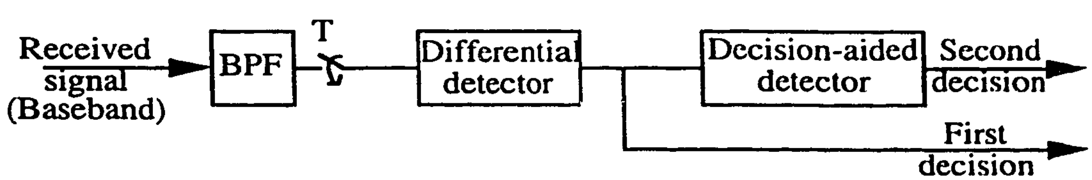


Fig. 3.1. Block diagram of receiver.

3.1.1 Block Diagram

The block diagram of a differential detector with decision-aided detection is shown in Fig. 3.2.

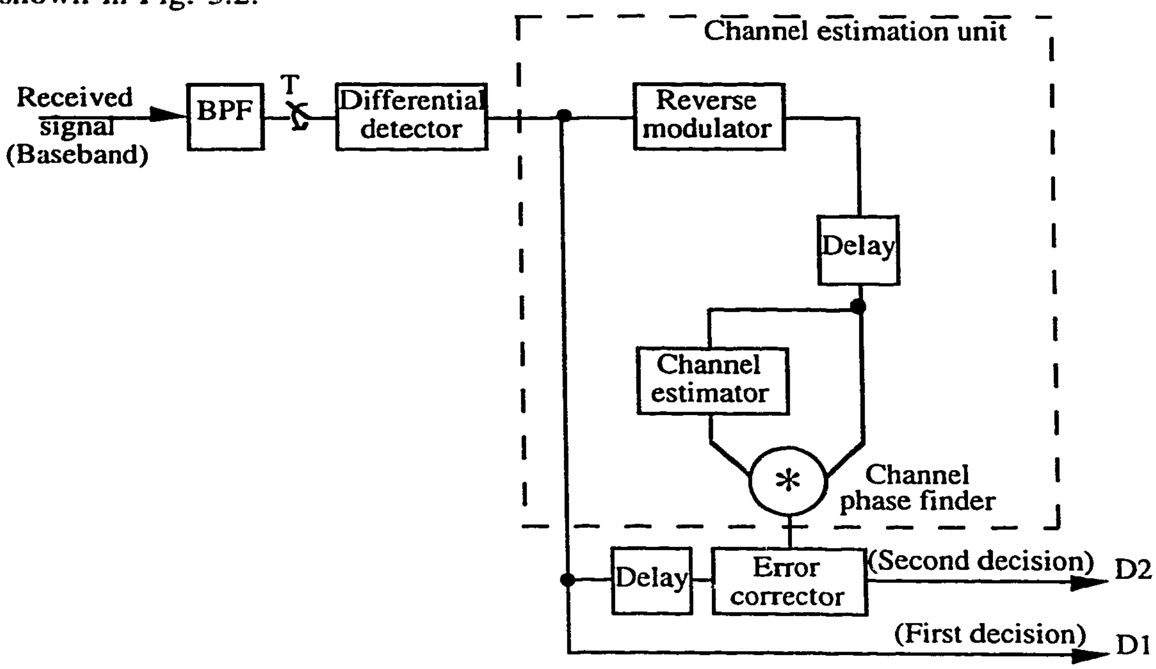


Fig. 3.2. Block diagram for decision-aided detector.

At the receiver input, the transmitted π/4DQPSK signal as affected by Rayleigh fading is down-converted from RF to complex baseband and sampled at $f_S = 1/T_S$. This complex baseband input r, is given by

$$r(nT_s) = s(nT_s)\varepsilon(nT_s) + \eta(nT_s) \qquad (3\text{-}1)$$

where s is the transmitted signal, ε is the fading process and η is the additive noise. The signal passes through the receive filter in order to limit additive noise while minimizing the distortion of the desired signal. The output of the receive filter, r_z, which has been decimated to a rate of 1 sample per symbol, is given by

$$r_z(kT) = s(kT)\varepsilon(kT) + \eta_z(kT) \qquad (3\text{-}2)$$

The receive filter output is fed to a conventional differential detector, in order to make decisions about the transmitted information. The output signal of this correlator is given by

$$r_t([k+1]T)r_t(kT)^* = \{s([k+1]T)\varepsilon([k+1]T)+\eta_1([k+1]T)\}\{s(kT)\varepsilon(kT)+\eta_1(kT)\}^*$$
$$= U + jV \quad \text{where U is the real part and V is the imaginary part.}$$

(3-3)

The ability to decode the transmitted information, s, is impaired by the additive noise, η_1, and the multiplicative fading, ε. The receiver could eliminate the effects of η_1 and ε if it were able to estimate them. However, for practical purposes, the additive noise is unpredictable and its effect can only be reduced by increasing the transmitted signal power. In contrast, the fading process can be estimated because it is relatively narrow-band compared to the data signal; thus the receiver's performance can be significantly enhanced by estimating the effect of the channel.

The function of the decision-aided detector is to compute an estimate of ε. It consists of two basic units: a channel estimation unit and an error correction unit. The core of the decision-aided detector is the channel estimation unit, which comprises three parts: a reverse modulator, an estimation filter and a channel phase finder. The reverse modulator removes the modulation from the received signal based on the signal data as determined by the first decision **D1**. For the case of no ISI and no detection errors in **D1**, the phase of the resulting signal is a noisy estimate of the fading channel distortion.

$$Up + jVp = r(nT_s - D)\,\hat{s}^*(nT_s - D) \tag{3-4}$$

$$Up + jVp = \varepsilon(nT_s - D) + \eta_2(nT_s) \tag{3-5}$$

where D is the delay due to the receive filter.

In the practical situation, for a mobile radio channel, typically one out of hundred first decision errors are incorrect; therefore, the carrier recovery at the output of reverse modulator (**Up** and **Vp**) is distorted. However, since the recovered carrier samples are 99% correct, a good estimate of the channel phase may be obtained by filtering the resulting channel samples. This is the function of the channel estimation filter. Once the fading estimate has been made, then the first decision is changed to accommodate the effect of the channel phase angle, resulting in a second decision **D2**. The following section will describe the functions and characteristics of the channel estimator and the error correction unit.

3.2 Channel Estimation

3.2.1 Reverse Modulation

The signal at the output of the differential detector (**U** and **V**) represents the in-phase and quadrature components of the phase change, as determined by the differential detector. Using this phase change, a first decision, **D1**, is made on the received symbol; if there is no error caused by the fading channel, then the detected symbol will be the same as the transmitted symbol. As described earlier in Chapter 2, the information symbols in $\pi/4$DQPSK modulation are transmitted as the difference between successive carrier phase angles. The possible values of these phase differences are {±45° and ±135°}. This means that, under ideal channel conditions (i.e. no noise or fading distortion), the received phase differences are limited to these four values. Thus, the **U** and **V** components of the detector output, if plotted on a vector diagram, must lie along axes that are at ±45° or ±135°, as shown in Fig. 3.3a. The labels '00', '01' ... on each of these axes indicate the corresponding demodulated symbols. However, due to the random phase modulation caused by the fading, the actual received **U** and **V** vectors do not lie exactly in these ideal directions. The deviation of the actual vector from the ideal vector is represented by **Up** and **Vp**, as shown in Fig. 3.3b. **Up** and **Vp** are a measure of the phase deviation caused by the non-ideal channel, i.e. intersymbol interference, additive white Gaussian noise, and Rayleigh fading. It should be noted here that **Up** and **Vp** are measured with respect to one of the four output ideal directions. The differential detector provides an output **U** and **V** and the decoder chooses the closest ideal vector direction in order to make a decision (Fig. 3.3a).

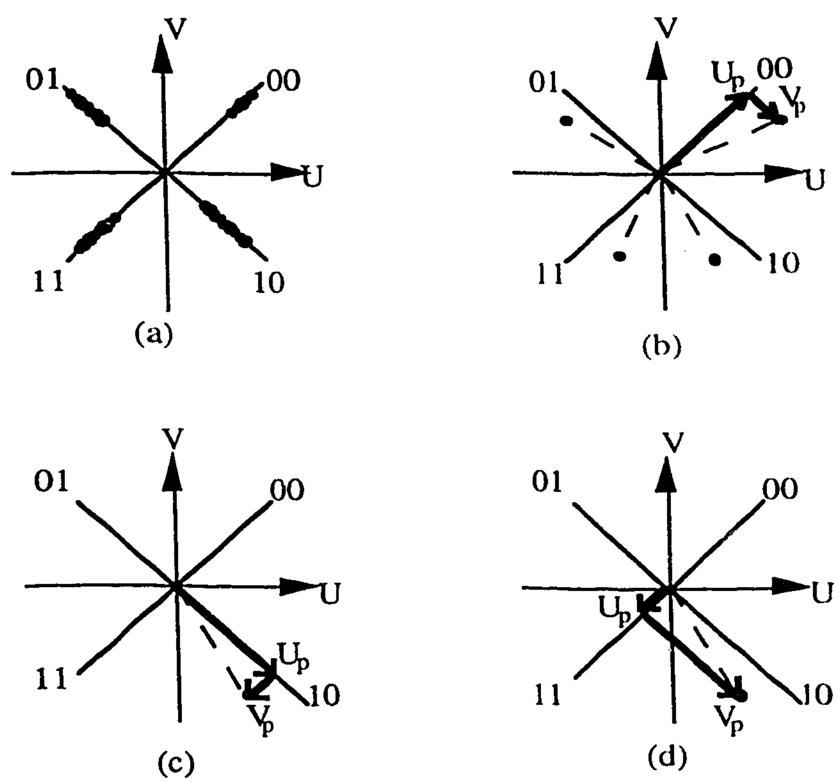


Fig. 3.3. Vector diagram for reverse modulation process.

When the effects of fading and noise are small, then **U** and **V** will lie very close to one of the ideal directions. Fig. 3.3b shows this situation. The received signal is relatively high, and **U, V** lie close to +45°. **Up** is large and **Vp** is slightly negative. The decoded symbol corresponding to the received signal will be "00". During a fade the received signal will be smaller and the detector output will be closer to the origin. This condition is shown in Fig. 3.3c. The vector **U**, **V** lies closest to the -45° axis and the received symbol will be decoded as '10'; **Up** and **Vp** will be as shown. For this case however, noise and distortion may easily have caused a significant amplitude and phase error, moving **U, V** away from its correct direction and causing an error in the detected data. For example, the true data corresponding to Fig. 3.3c could be '11', but the channel has caused a phase error of almost 90°, placing **U**, **V** in the wrong quadrant. It is important to notice that any data errors will also cause the values of **Up** and **Vp** to be incorrect. With reference to the direction '11', **Up** and **Vp** should actually be taken as in Fig. 3.3d. It is this error in **Up** and **Vp** that is used by the error detection and correction algorithm. Because the fading is a narrow-band process, the effect of the fading changes only slowly from one symbol to the next. Further, because typically 99 out of 100 symbol decisions are made correctly, only 1% of the **Up**, **Vp** values are incorrect. By using a low-pass filter to smooth out the **Up** and **Vp** values, the effect of the incorrect decisions can be made very small, and a good estimate of the fading channel characteristic can be made.

3.2.2 Estimation Filter

A good estimate of the phase change due to the fading channel (**Upf** and **Vpf**), for each decision point, is determined from the **Up** and **Vp** values, using a filtering operation to remove, as far as possible, the effect of any incorrect **Up** and **Vp** values. Here, to achieve the best possible performance both past and "future" values of **Up** and **Vp** are used. Thus an interpolation method is used, rather than a simple prediction based on past values only. With respect to each first decision (D1), this requires storage of past and future values of the channel characteristics, in order to estimate the channel phase angle and then make a second decision. This is realized by storing successive values of **Up** and **Vp** as determined by the first decision D1, and making the corresponding second decisions several (20 to 30) symbol periods later.

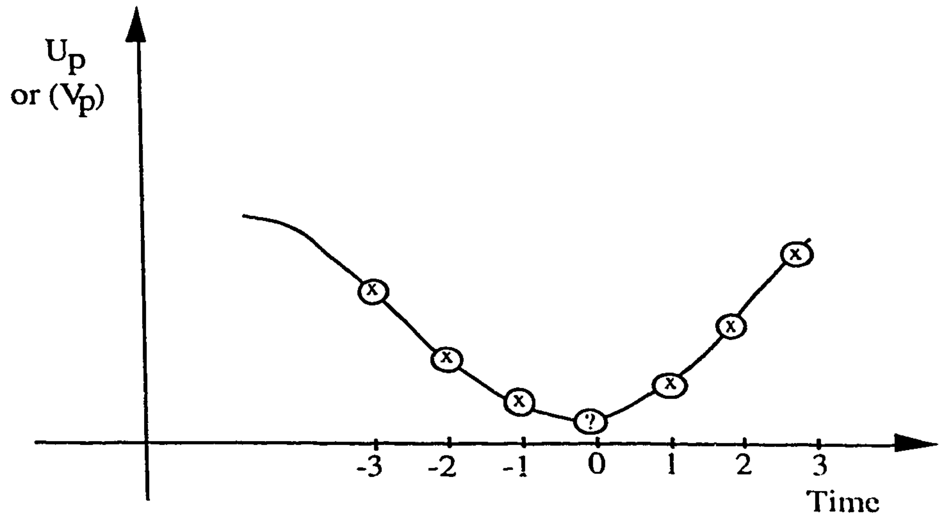


Fig. 3.4. Channel estimation.

The estimator is designed as a lowpass filter, since the fading channel characteristic is a narrow-band process having a two-sided bandwidth equal to 4 times the maximum Doppler frequency [15]. The fading bandwidth is doubled due to the multiplicative correlation process in the differential detector. A lowpass filter smoothes out the channel estimates, thereby removing the errors caused by any wrong decisions, and also removing most of the noise. However the passband of the estimating filter has to be a tradeoff between a narrow and wide passband. A narrow passband is desired so as to reduce the effect of noise at low SNR, but at the same time, it has to be wide enough to pass the fading channel information without distortion.

The Rayleigh fading estimation filter determines the values of **Upf** and **Vpf.** The **Upf** values corresponding to each decision point are obtained from the **Up** values using an FIR transversal filter. As shown in Fig. 3.4, **Up** values both before and after the decision point are used, with the filter output corresponding to the tap at k=0. The **Vpf** values are obtained similarly. Mathematically

$$Upf(n) \equiv \sum_{k=-N}^{N} C_k Up(n-k) \tag{3-6}$$

where the C_k are the filter coefficients. The total length of the filter is 2N+1, using N values on each side of the decision point. The channel estimation using this filter is not always perfect since, due to the data errors, erroneous samples of **Up** or **Vp** will enter the filter at k=-N and will move through to k=0 before they can be removed.

Various techniques can be used to determine a set of appropriate C_k values. These values must be chosen so that, when the channel samples are correct, the estimated values are exactly equal to the input values. This requires a flat spectral response over the bandwidth of the fading channel. The sampling theorem provides the basic approach to choosing a set of coefficients. This theorem states [29]:

A signal bandlimited to B Hz (i.e., a signal whose Fourier transform is zero for all $|w| > 2\pi B$*) is uniquely determined by its values at uniform intervals less than 1/2B seconds apart,*

or, alternatively,

If a continuous signal is multiplied by an ideal sampling function, then the signal can be recovered from the sampled version only if the original signal spectrum is entirely contained within a bandwidth of less than half the sampling rate.

The original signal can be recovered from the samples by using an ideal lowpass filter that passes the signal spectrum centered on zero frequency and extending to within half the sampling frequency on either side. This filter also masks out all the undesired aliasing images. In the time domain, the inverse Fourier transform of an ideal lowpass filter gives the sync function. Thus, to recover the original time domain waveform, the sampled signal must be convolved with a unit sync function. In other words, a lowpass waveform can be re-constructed from its samples by summing weighted sync functions, where the weights are the discrete sample values. Mathematically,

$$g(t) = sam_{T_s}[g(t)] * sync\left(\frac{t}{T_s}\right) \tag{3-7}$$

or, for discrete signals,

$$g(t) = \sum_{n=-\infty}^{\infty} g(nT_s)sync(2B\tau n) \tag{3-8}$$

where $g(nT_s)$ are the signal samples given, in our application by **Up** and **Vp**. The filter coefficients are given by the values of sync($2B\tau n$). In principle, a filter designed in this way will give perfect reconstruction of the desired waveform. However, one important limitation for any practical implementation of this result is the infinite duration of the sync function. Variation from the sync function can be made by choosing different lowpass filter shapes, keeping a flat lowpass characteristic, but allowing variation in the roll-off and stopband characteristics.

Another approach for determining the best possible estimate is to use a least square error fit or a polynomial curve fit method for the coefficient calculation [30]. For either of these methods, a polynomial can be found from which **Upf** and **Vpf** can be calculated. Sample values of **Up** (or **Vp**) are considered on both sides of the decision point for which the estimate of **Upf** (or **Vpf**) is required. A polynomial of degree m can be represented as,

$$p(x) = a_o + a_1 x + a_2 x^2 + \dots\dots + a_m x^m = \sum_{k=0}^{m} a_k x^k. \tag{3-9}$$

With equally spaced sample times, however, the required estimated value reduces to a sum of weighted sample values, as given by Eq. (3-6). The least square method aims at minimizing the discrepancy between the data points and the approximating polynomial, given by

$$Q(f,p) = \sum_{i=-n}^{n} \left[f_i - p(x_i) \right]^2 \tag{3-10}$$

where $\{x_{-n}.....x_{-2}\ x_{-1}\ x_0\ x_1\ x_2......x_n\}$ represent the domain point at the sampling instant and $\{f_{-n}\ .....f_{-2}\ f_{-1}\ 0\ f_1\ f_2\ .....f_n\}$ are the corresponding sample values. The degree of the polynomial and the number of points used in the curve fitting procedure need to be varied in order to test the accuracy of estimation for the best possible estimate of the channel.

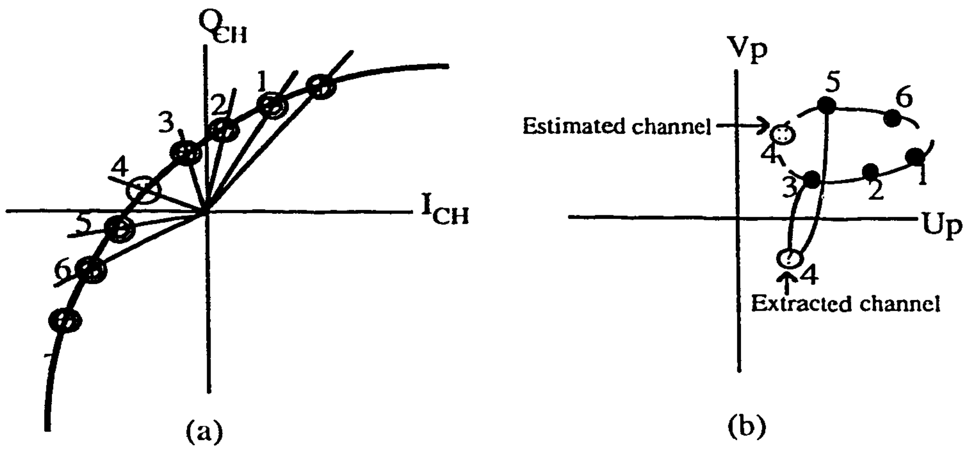
The two approaches are not totally independent. According to the first approach any function can be represented as a sum of sines and cosines, i.e. a Fourier series. For the second approach, polynomials can be used to represent any function. However, expanding sine and cosine functions in polynomial form shows the relation between the two approaches. Another technique available for designing the estimating filter is to use a signal processing based algorithm [31], such as the window method, Remez exchange algorithm, or the frequency sampling based design, etc. Since all these approaches can be represented in terms of Eq. (3-6), they all lead to an estimation filter design based on a lowpass FIR filter.

The estimate of the channel leads to two cases. In the first case, the estimated channel is almost the same as the original sample, indicating that no error has been made. If the two values are very different, then an error may have occurred. When the error is corrected, then, in order to improve future estimates, it may be required to replace the original values of **Up** and **Vp** with the corrected values **Upnew** and **Vpnew**. The decision criteria and values for replacement are further described in Section 3.3, the error correction section.

3.2.3 Channel Phase Finder

Once the fading process estimate is complete, the **Upf** and **Vpf** values so obtained are compared with the initially extracted channel information **Up** and **Vp**. This gives information about the phase change due to the channel, information which was not available while making the first decision about the transmitted data. When the first data decisions are all correct, then ideally **Upf** and **Vpf** will closely follow the **Up** and **Vp** values. However, when $\Delta Ø_{channel}$ (channel phase change) plus the effect of noise, is >45° then a data error will be made and the extracted **Up** and **Vp** will be placed in the wrong quadrant (as illustrated earlier in Fig. 3.3(c) and (d)) as a result of which **Up** and **Vp** may be interchanged, or the signs may be reversed. However, since typically only 1 out of 100 channel samples is incorrect, and the estimation filter will smooth out the values that are in error, and **Upf** and **Vpf** will still give a good estimate of $\Delta Ø_{channel}$. By comparing these values with the original **Up** and **Vp**, a second decision **D2** may be made.

The error correction procedure is illustrated in Fig. 3.5. Fig. 3.5a shows a typical sequence of channel values I_{CH}, Q_{CH} during a fade. This diagram also shows the **Up** and **Vp** values that should be extracted in order to correctly quantify the phase change due to the channel. However, this phase change will only be determined correctly if this incremental angle is less than ±45°. Fig. 3.5b shows the values of **Up** and **Vp** that will actually be obtained. The values are all correct except for sample point #4 which corresponds to the channel phase of ≈ 60° that occurs from sample point 3 to 4 (in Fig. 3.5a). For this symbol, the data is decoded in the wrong quadrant (+90° compared to the true data) due to the excessive channel phase change, and consequently the channel is extracted with an error of -90°. This point #4 is extracted incorrectly. The dashed line in Fig. 3.5b shows the trajectory of the filtered **Upf**, **Vpf** values. Because most of the **Up**, **Vp** samples are correct, the one point in error affects the **Upf, Vpf** values only slightly. By comparing **Up, Vp** with **Upf, Vpf** at each sample, a check is made on the first data decision and a correction made if required.



(a) (b)

Fig. 3.5. (a) Typical channel. (b) Error correction.

The various angles are related by,

$$\Delta\varnothing_{\text{extracted\_channel}} = \Delta\varnothing_{\text{received}} - \Delta\varnothing_{\text{data,first\_decision}} \qquad (3\text{-}11)$$

$$\Delta\varnothing_{\text{data,second\_decision}} = \Delta\varnothing_{\text{received}} - \Delta\varnothing_{\text{estimated\_channel}} \,. \qquad (3\text{-}12)$$

Using phasor notation, the channel error may be defined in terms of quantities **C** and **D**, where

$$\mathbf{C} + \mathbf{j}\,\mathbf{D} \cong \Delta\varnothing_{\text{estimated\_channel}} - \Delta\varnothing_{\text{extracted\_channel}} \,. \qquad (3\text{-}13)$$

The values of **C** and **D** are easily calculated from

$$\mathbf{C = Up\ Upf + Vp\ Vpf} \qquad (3\text{-}14a)$$

$$\mathbf{D = Up\ Vpf - Upf\ Vp}. \qquad (3\text{-}14b)$$

3.3 Error Correction

Finally, the receiver removes the effect of the additional channel phase (**C** and **D**) from the first decision (**D1**) thereby giving the second decision (**D2**). There are four possible cases for the second decision. These are: no change in D1, change **D1** by -90°, change **D1** by +90°, and lastly, change **D1** by 180°. The shift of angle refers to the Gray code quadrant used for transmission, and this corresponds to the detected data, as shown in Fig. 3.6.

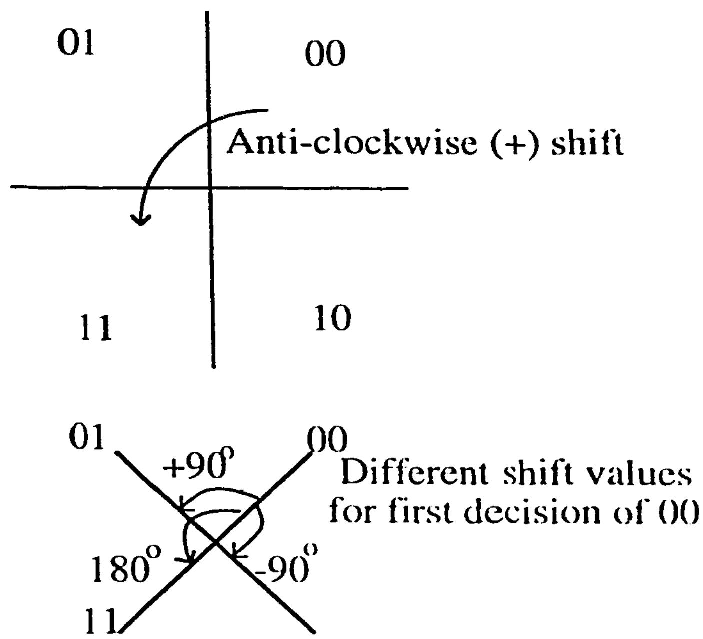


Fig. 3.6. Gray code to shift first decision.

C+jD is a measure of how closely (**Up**, **Vp**) agrees with (**Upf**, **Vpf**). Since the data decisions are all based upon phase, then a correct decision should align (**Up**, **Vp**) and (**Upf**, **Vpf**) as closely as possible, indicating that **C+jD** should represent a zero phase angle. From the value of **C+jD**, the phase is determined and a choice of **D2** as compared to **D1** is made as shown in Table 3.1.

TABLE 3.1
RELATION BETWEEN ang(C+jD) AND DATA SHIFT.

Case number	Condition Ø=ang(C+jD)	Data shift
I	-45<Ø<45	No Shift
II	45<Ø<135	-90 Shift
III	-135<Ø<-45	+90 Shift
IV	135<Ø<-135	180 Shift

The same conditions are more clear from the vector diagram shown in Fig. 3.7. Since the channel estimate makes use of subsequent symbols, there is a delay between the second decision instant and the first decision instant. The amount of delay depends on the length of the estimation filter used. This delay has to be taken into account while shifting data **D1**, as per the Table 3.1, to obtain **D2**.

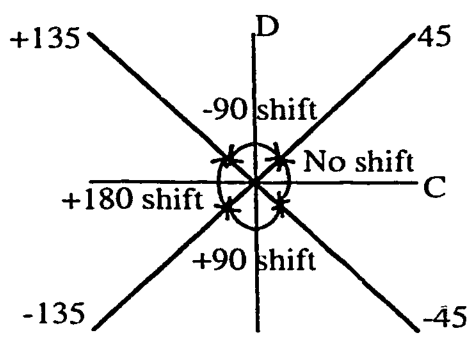


Fig. 3.7. Decision range for different shift values.

Referring to the example given in the previous subsection, shown in Fig. 3.5, the correction of the data is made so as to align the (**Up**, **Vp**) values with the (**Upf**, **Vpf**) values. In the diagram, modulation point #4 gives better agreement if it is rotated through +90°, indicating the original data decision **D1** should be corrected by -90°.

As discussed in the previous section, errors in **D1** causes errors in (**Up**, **Vp**) which leads to errors in the filtered channel estimates (**Upf**, **Vpf**). When erroneous values enter the domain of the estimation filter, then the accuracy of the channel estimates (**Upf**, **Vpf**) will be impaired. However, the inaccuracy will be small because errors usually occur during fades, during which time the signal level is small. Once an erroneous value reaches the center of the filter, then, when the data is corrected, the (**Up**, **Vp**) value can also be changed. The experimental results (given in Chapter 4) show that, when this is done, the corrected error rate is somewhat improved. The corrected or new (**Up**, **Vp**) values are found from the initial values, as given in Table 3.2.

TABLE 3.2
RELATION OF (Up, Vp) AND DATA SHIFT WITH (Upnew, Vpnew).

Upnew	Vpnew	DATA SHIFT
Up	Up	No Shift
-Vp	Up	-90 Shift
Vp	-Up	+90 Shift
-Up	-Vp	180 Shift

3.4 Scheme II for Decision-Aided Detection

This method is an extension of the method described in Sections 3.2 and 3.3. The only difference is that the reverse modulation is fed back to extract the channel phase angle from the received signal, unlike scheme I, where the output of the differential detector is used to obtain the channel information. In scheme I the channel phase change (for each symbol) is obtained, whereas in scheme II an attempt is made to track the true magnitude and phase angle of the channel characteristic (versus time). The motivation for trying to recover the channel, rather than the "differential channel" is that during a fade the differential detector output becomes very small and can easily become lost in the noise. The incoming signal is less affected by noise, so the channel recovered from the signal should have a higher "channel-to-noise" ratio.

The principal disadvantage of trying to obtain the true channel angle is that, for a differentially encoded signal, the phase changes due to the detected data must be accumulated, and the overall total phase angle must be subtracted from the incoming signal (the complex baseband signal SI+jSQ) in order to retrieve the channel. Whenever a data detection error occurs, the recovered channel suffers a phase discontinuity rather than having just a single point in error. While it is not difficult to detect that a phase discontinuity has occurred, it has been found difficult to locate the exact symbol point that is in error. For this method then, the error detection and correction algorithm must be considerably more mathematically sophisticated and computationally intensive than for the simple method of scheme I.

3.4.1 Block Diagram

The block diagram of the error detection and correction process is shown in Fig. 3.8. It consists of a differential detector, a re-modulation unit, a channel recovery unit, an error detection unit and an error correction algorithm. As in scheme I, the signal is first detected by a conventional differential detector. The past decisions $D1_i$ are used for re-modulation, but in order to determine the data contribution to the received signal SI+jSQ, all the decision information must be accumulated. The cumulative phase angle for the data is subtracted from the incoming signal in order to recover the channel. If there is no decision error, then the channel magnitude and phase will be correctly recovered from the received signal.

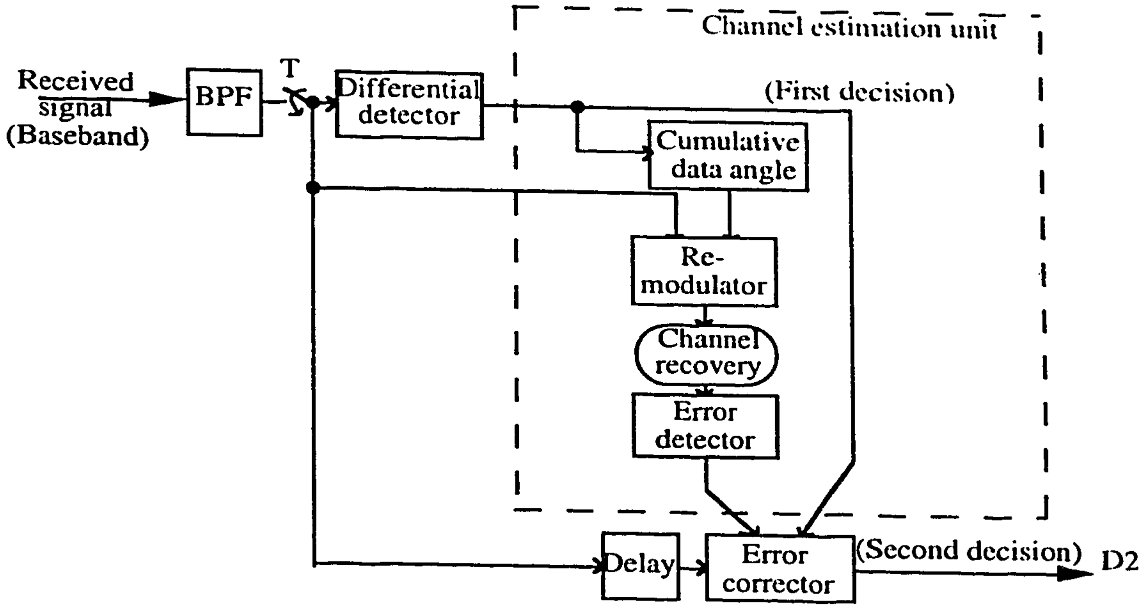


Fig. 3.8. Block diagram for scheme II (Decision-aided error detection).

For every error in the demodulated data, however, the accumulated channel angle will be incorrect by either 90°, 180°, or 270°. This will affect not only the point in error, but will shift the phase of all later points as well. Because of this, the channel is recovered as a sequence of smooth segments with discontinuities at the points where the data (D1) errors occur, as shown in Fig. 3.9.

It is the function of the error detector to locate the discontinuities, and then determine how best to remove them. That is, a choice of data (or a change of phase angle) must be made that will give the smoothest possible channel characteristic. Simple filtering is not sufficient in this case. One method that can be used to locate the errors is to calculate the derivatives of the recovered channel characteristic. Since errors cause fairly sharp discontinuities, giving sudden changes of slope, etc., then the magnitude of derivative values is much larger in the vicinity of an error than elsewhere. This locates the error to within a few symbol locations, but it does not however give the exact position. Using the derivatives thus enables various error zones to be identified.

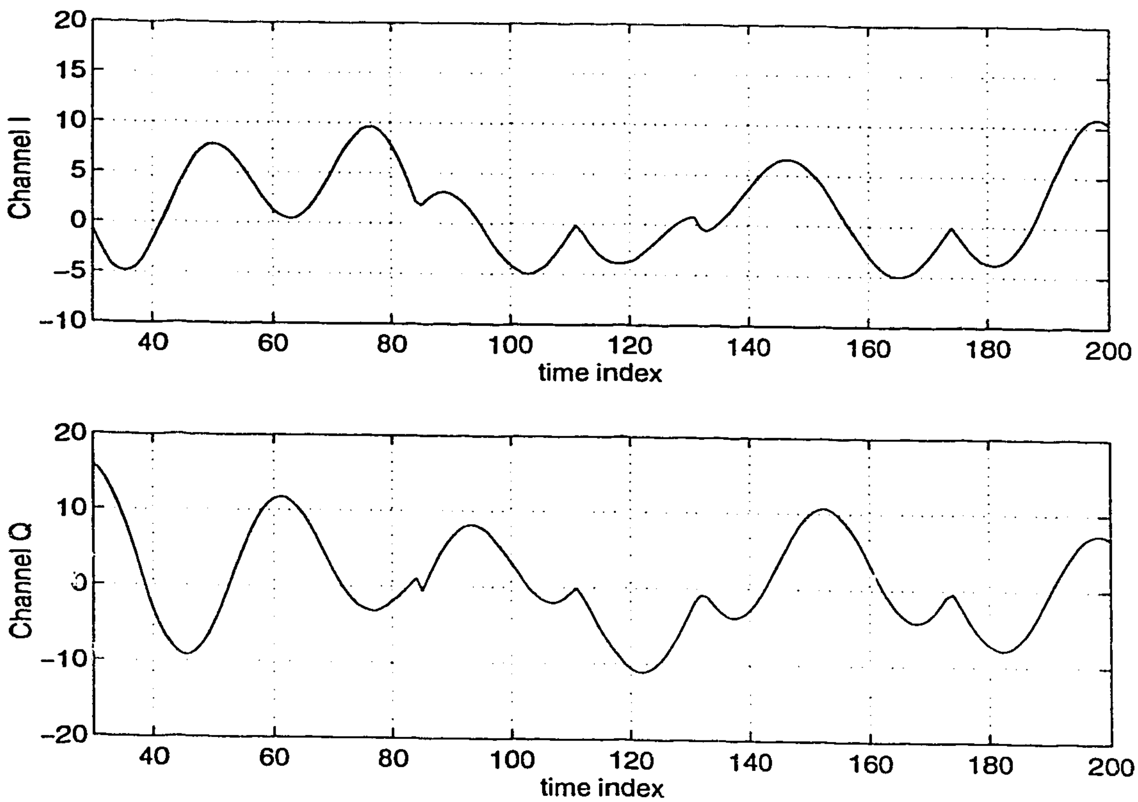


Fig. 3.9. Recovered channel.

3.4.2 Channel Estimation

Although the error zones appear quite distinct, it is difficult to determine exactly which symbol (or symbols) is in error. A number of techniques using the channel derivatives have been attempted with varying degree of success, but a completely satisfactory method has not yet been determined. This is an area where further work is required. Scheme II is further discussed in Chapter 5, together with some initial results.

3.5 Summary

In this chapter, decision-aided detection for $\pi/4$DQPSK signal is discussed and an algorithm capable for reducing the irreducible error floor is presented. The output of the differential detector is used to estimate the channel characteristic, and this information is then used to make a second decision about the transmitted data. The scheme is based on the transmitted symbol information only; i.e., no coding or overhead information is used. A second method which attempts to obtain the true channel characteristic from the incoming signal is also discussed. The DSP implementation and experimental results are presented in the following chapters.

4. DSP IMPLEMENTATION OF DECISION-AIDED DETECTOR

Digital signal processors are presently very popular in digital communication system design as they provide potentially improved capability, performance, repeatability, flexibility, size and cost. The aim of this chapter is to present the real time digital signal processor implementation of the transceiver model described in the previous chapters. The hardware and software structure of the transmitter, channel and receiver, and the performance measurement module for scheme I are discussed in detail.

4.1 Transmitter

The basic structure of the transmitter is implemented using the fixed point Texas Instruments DSP TMS320C50. The key features of this DSP are given in Table 4.1 [32, 33].

TABLE 4.1
FEATURES OF DSP TMS320C50.

50ns instruction cycle (20MIPS)
9K × 16 bit on-chip program/data RAM
1056 × 16 bit dual-access on-chip data RAM
32 bit ALU, accumulator and accumulator buffer
8 auxiliary registers with dedicated arithmetic unit
2 indirectly addressed circular buffers
TDM serial port; Software controlled interval timer

The same DSP is programmed to simulate the channel as well, details of which are given in the next section. Because of the need to program the slowly varying channel as well as the transmitted signal in a single DSP, then in order to evaluate the performance of the scheme, the transmitted data is limited to a lower rate (2.4kb/s). The various components of the transmitter are discussed in the following sub-sections.

4.1.1 Data Source

The data source in an actual system is either speech converted to digital form, or digital data from a fax machine, or a computer, etc.. This signal has no memory and can be considered to be completely random. The data stream is then compressed and placed into a specified standard frame structure [5]. However, to generate the data for testing, a maximal length pseudo random binary sequence generator is used.

The two commonly used methods to generate such random sequences are the multiplicative congruential method or mixed congruential method, and the M-sequence method [34]. The sequence generated by the mixed congruential method is based on a recurrence relation given by

$$Y_i = AY_{i-1} + C(\text{mod}\, R) \qquad i = 1,2,3\ldots\ldots\ldots \qquad (4\text{-}1)$$

where i is a discrete time interval, Y_i is the output pseudorandom sequence, and A, C and R are constants; also, $Y_0 > 0$, $A > 0$, $C \geq 0$, $R > Y_0$, $R > A$, $R > C$, and mod is the modulo operator. The mathematical definition of the M-sequence method is given by recurrence relation

$$X_i = \text{mod}\, 2\sum_{k=1}^{m} \alpha_k X_{i-k} \qquad i = 1,2,3\ldots\ldots\ldots \qquad (4\text{-}2)$$

where i is a discrete time interval, $X_i \in (0,1)$ is the output pseudorandom sequence, m is the tap length, $\alpha_k \in (0,1)$ are the tap coefficients, and the summation sign with mod2 represents modulo-2 addition. The multiplicative congruential method requires mathematical computations like multiplication, addition and the modulo-R operation as shown in Eq. (4-1). Thus the implementation based on this method requires a significant number of instruction cycles. By contrast, the M-sequence method requires only a modulo-2 summation or exclusive-OR operation, as shown in Eq. (4-2). The implementation simplicity of the M-sequence method, makes it a more suitable choice for realizing the pseudorandom binary sequence source.

The M-sequence generator that functions in accordance with expression (4-2) comprises an m-bit shift register and the set of modulo-2 adders for feedback. Since X_k is repeated every 2^m-1 outputs, it is required to set the value of m large enough so as to obtain independent and random X_k over the testing period. In this experiment, the length of shift register, m, is selected to be 31 so as to obtain a sequence length equal to $2^{31}-1$. Thus, for the data source of 1.2ks/sec, the output data sequence has a period of 1.8×10^6 sec and gives 2×10^9 symbols before repeating. The tap coefficients used in the feedback circuit, for a selected value of shift register length, is obtained from a standard M-sequence characteristic polynomial table [34]. In order to initialize the shift register for sequence generation, a non-zero value must be stored in at least one register.

The mathematical definition of the implemented data source is given by

$$\alpha_k = \begin{cases} 1 & k = 3,31 \\ 0 & (1 \le k \le m) \cap (k \ne 3,31) \end{cases}$$

$$X_i = X_{i-3} \oplus X_{i-31} \qquad i = 0,1,2\ldots\ldots \tag{4-3}$$

where $\oplus$ is the Exclusive-OR operator. Fig. 4.1 shows that the source consists of 31 clocked registers or memory locations and one modulo-2 addition operation.

These clocked memory locations form a 31 tap-delay line, and the contents of locations 3 and 31 are used as the inputs to the modulo-2 adder.

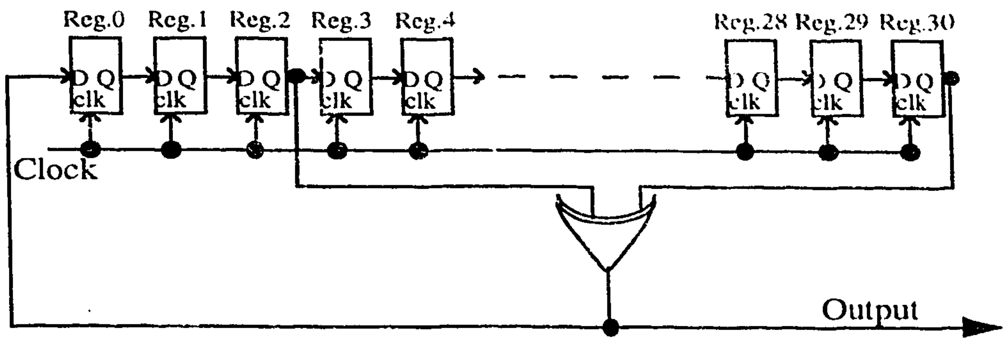


Fig. 4.1. M-sequence generator.

The software implementation of the same function is best done using a circular buffer. Since the circular buffer provided in the C50 has the limitation that it may be incremented or decremented by one step at a time, this could not be used directly. However, an address table was used to achieve the same action. Since the sequential bits of data are used by the encoder in pairs (refer Section 2.1.3), two independent generators are used to generate the data symbol, to avoid serial-to-parallel conversion. Also, since the channel implementation needs another two random sequence generators, a combined procedure that generates 16 uncorrelated random sequences is used. Further details of the implementation structure is given in Section 4.2.1, together with the channel implementation.

4.1.2 Encoder and Modulator

Each pair of bits from the data source is encoded using a Gray code format, where each symbol corresponds to a particular differential phase angle ($\Delta\phi$), as required by the π/4DQPSK modulation format. See Table 2.1 for the relation between the input pair of binary data and the phase change $\Delta\phi$. The output of the phase encoder is fed to the modulator in order to obtain the in-phase (I) and quadrature phase (Q) signals. The modulation scheme is defined as

$$S_k = S_{k-1} e^{j\Delta\phi(b1_k, bo_k)} \tag{4-4}$$

where k is the output symbol index, S_k is the k^{th} complex output symbol from the modulator, and $\Delta\phi(b1_k,bo_k)$ is the phase encoder function corresponding to bo_k and $b1_k$ (the first and second data bits at the k^{th} index). This shows that, S_k is generated by rotating the phase of the previous symbol S_{k-1} by an angle $\Delta\phi$ $(b1_k,bo_k)$, where $\Delta\phi$ takes on one of $\{\pm\pi/4, \pm3\pi/4\}$. This produces a constellation diagram for the modulator signal output with eight possible states (see Fig. 2.3).

For software implementation simplicity, the above modulator equation is converted to a state equation given by

$$i_k = \left(i_{k-1} + \Delta\phi_s\left(b1_k, bo_k\right)\right) \bmod 8 \tag{4-5}$$

where $k \in [0,1,2,3,4,5,6,7]$ is the state number. Each state number represented by k corresponds to one of the 8 output symbols (i.e. State 0,1 etc.) and $\Delta\phi_S(b1_k,bo_k)$ is obtained by mapping $\Delta\phi$ in $\Delta\phi(b1_k,bo_k)$ to these states. Since state 0 is taken as the reference while mapping, the four possible differential angles provide the four state values {1, 3, 5, 7} to $\Delta\phi_S(b1_k,bo_k)$. Fig. 4.2a and b shows the constellation state diagram and the $\Delta\phi_S(b1_k,bo_k)$ state diagram.

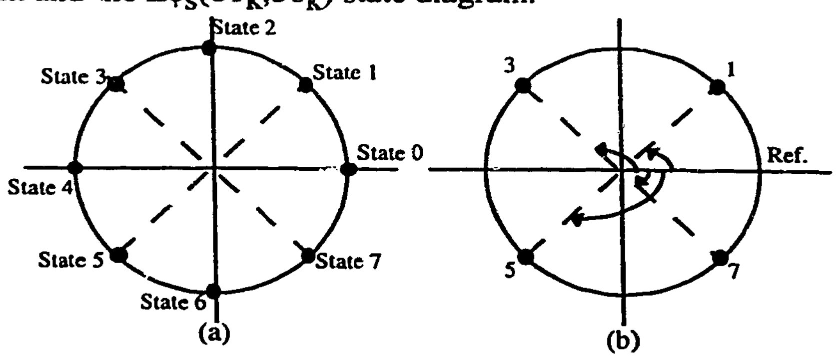


Fig. 4.2. (a) Constellation state diagram. (b) Phase change state diagram.

For the purpose of DSP implementation, the state equation is further simplified by replacing the modulo eight operation with an AND operation, using the binary constant "111". Eq. (4-5) may then be re-written as

$$i_k = \left(i_{k-1} + \Delta\phi_s\left(b1_k, bo_k\right)\right) AND \text{ "00000111"} \qquad (4\text{-}6)$$

The complete software module to realize the encoder and modulator functions requires the phase encoder state generator function $\Delta\phi_S(b1_k,bo_k)$ and the state equation of Eq. (4-6). Instead of using a table to generate $\Delta\phi_S(b1_k,bo_k)$ from the data and then obtain i_k from $\Delta\phi_S$, digital logic is used to further simplify the implementation. The relationship between the input data bits and the corresponding state values of $\Delta\phi_S(b1_k,bo_k)$ in binary form are as shown in Table 4.2.

TABLE 4.2
RELATION BETWEEN DATA BITS AND STATE GENERATOR FUNCTION.

Input data (b_1,b_0)	$\Delta\phi_S(b1_k,bo_k)$ in decimal	$\Delta\phi_S(b1_k,bo_k)$in binary (s_2,s_1,s_0)
0,0	1	0,0,1
0,1	3	0,1,1
1,1	5	1,0,1
1,0	7	1,1,1

Representing i_k as $s_2s_1s_0$, then i_k can be easily obtained from each p[illegible]ta bits as

$$\begin{aligned} s_o &= 1 \\ s_1 &= b_0 \oplus b_1 \\ s_2 &= b_1 \end{aligned} \qquad (4\text{-}7)$$

Thus, the generation of $\Delta\phi_S(b1_k,bo_k)$ utilizes one Exclusive-OR logic operation and some simple memory manipulation. The software implementation of the modulator, i.e. Eq. (4-6), consists of a look-up table, an addition, and an AND operation. The look-up table stores the complex values of the eight output symbol and is indexed by i. The index i is obtained by adding the new state value of the encoder to the previous state value and performing the AND "111" operation on the resulting value. The resulting index i is used to obtain the in-phase and quadrature components of the data modulator output from a look-up table, as listed in Table 4.3. At the same time this index value overwrites the previous state value for use in the next calculation.

TABLE 4.3
LOOKUP TABLE FOR IN-PHASE AND QUADRATURE PART OF MODULATOR OUTPUT.

Index i	In-phase component	Quadrature component	Index i	In-phase component	Quadrature component
0	1	0	4	-1	0
1	0.707	0.707	5	-0.707	-0.707
2	0	1	6	0	-1
3	-0.707	0.707	7	0.707	-0.707

4.1.3 Pulse Shaping Filter

The output of the modulator is passed through a pulse shaping filter. This filter serves two basic purposes: first, to modify the spectrum of the modulated signal to meet the regulatory constraints on the shape of the transmitted spectrum or on out-of-band power, etc.; second, to control the time domain characteristics of the impulse response, such as the pulse shape and zero crossings, so as to limit intersymbol interference (ISI). The two main design parameters for designing this filter are the spectral bandwidth and the zero crossings of the corresponding time response. In order to have no ISI, the zero crossings in the time domain must be spaced at the symbol period. Nyquist has shown that a time domain pulse p(t) will have zero crossings once every T seconds (apart from t=0) if its Fourier transform P(f) satisfies the following constraint [35]

$$\sum_{k=-\infty}^{\infty} P(f+kR_s)=T_s \qquad \text{for } |f|<R_s/2 \tag{4-8}$$

One type of filter that gives this property is the raised cosine filter which has P(f) defined by

$$P(f)=\begin{cases} T_s & 0\le |f|\le \dfrac{1-\alpha}{2T_s} \\ T_s\cos^2\left\{\dfrac{\pi}{4\alpha}\left[2|f|T_s-1+\alpha\right]\right\} & \dfrac{1-\alpha}{2T_s}\le |f|\le \dfrac{1+\alpha}{2T_s} \\ 0 & otherwise \end{cases} \tag{4-9}$$

The factor α can be chosen to optimize the frequency response while maintaining the zero ISI condition. Plots of P(f) for several values of α are shown in Fig. 4.3.

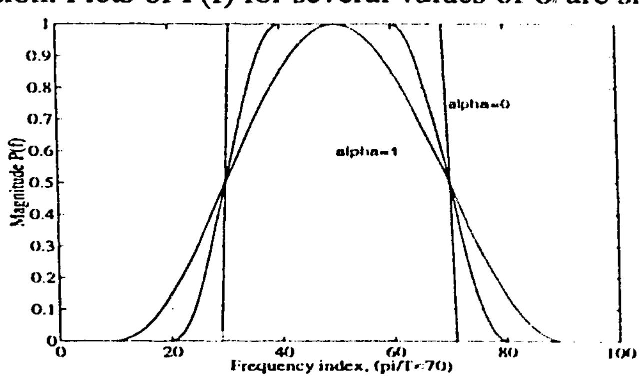


Fig. 4.3. Frequency domain response of the raised cosine filter.

Note that α=0 gives an ideal "brick wall" low pass filter response. The impulse response of a raised cosine filter p(t) has following properties:

$$p(0) = 1$$

$$p(kT) = 0 \qquad k = \ldots\ldots -2,-1,+1,+2,\ldots\ldots \quad . \qquad (4\text{-}10)$$

To provide the same filtering at both the transmitter and the receiver, the required P(f) can be split between two filters, chosen such that

$$P(f) = H_T(f)H_R(f) \qquad (4\text{-}11)$$

where $H_T(f)$ and $H_R(f)$ are the filter transfer functions at transmitter and receiver respectively.

Based on the above conditions and criteria, the IS54 standard specifies the transmit filter to be a square root raised cosine (SRRC) filter with a roll off factor (α) of 0.35. The frequency domain definition of this filter is given in Eq. (2-1). The impulse response of this filter is given by Eq. (4.12) [14]; and the frequency characteristic and impulse response of this filter are given in Fig. 4.4.

$$h(t) = \begin{cases} 1 - \alpha + 4\dfrac{\alpha}{\pi} & t = 0 \\[2ex] \dfrac{\alpha}{\sqrt{2}}\left[\left(1 + \dfrac{2}{\pi}\right)sin\left(\dfrac{\pi}{4\alpha}\right) + \left(1 - \dfrac{2}{\pi}\right)cos\left(\dfrac{\pi}{4\alpha}\right)\right] & t = \pm\dfrac{T}{4\alpha} \\[2ex] \dfrac{sin\left[\pi(1-\alpha)\dfrac{t}{T}\right] + 4\alpha\dfrac{t}{T}cos\left[\pi(1+\alpha)\dfrac{t}{T}\right]}{\alpha\dfrac{t}{T}\left[1 - \left(4\alpha\dfrac{t}{T}\right)^2\right]} & \text{for all other } t \end{cases} \qquad (4\text{-}12)$$

where T is the symbol period.

The pulse shaping filter is implemented in digital form in the C50 DSP. The two basic classes of filters are the *infinite impulse response* (IIR) and the *finite impulse response* (FIR). The FIR filter is selected for the pulse shaping filter because of its various advantages. A FIR filter has a linear phase response and is always stable, with a finite and constant delay [36]. Fig. 4.5 shows the structure of a FIR filter using an N-tap discrete delay line and N tap coefficients. The most recent N data samples are stored in the delay line, and are multiplied by the corresponding tap coefficients. The products are then summed together to obtain the filter output at each sampling instant. The characteristics of the FIR filter depend on the tap coefficients (h_i), and the length of the delay line (N). The tap coefficients are calculated by digitizing the impulse response h(t) of the pulse shaping filter at the sampling rate. For accurate implementation, the length N should be chosen large enough that all the significant values of the h(t) samples are included in the tap coefficients. For DSP implementation, N cannot be too large. For this filter, therefore, Matlab© (computer software) has been used to obtain an optimal design.

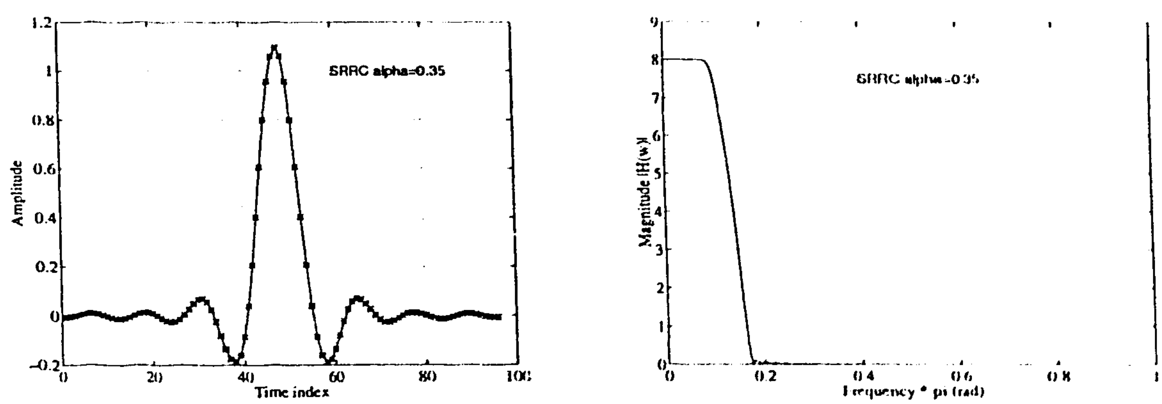


Fig. 4.4. Time domain and frequency domain response of SRRC filter.

Again, for accuracy of representation and signal processing, oversampling of the signal is required. In this work, an oversampling rate of 8 has been chosen, giving a sampling rate of 9600samples/second for a symbol rate of 1200symbols/sec. Since the symbols are generated at the lower rate, interpolation is required to bring the sampling rate up to 9600/second.

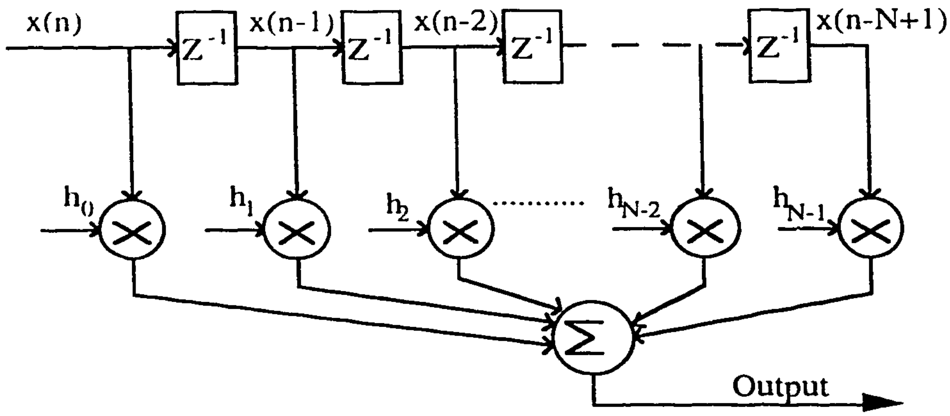


Fig. 4.5. FIR filter structure.

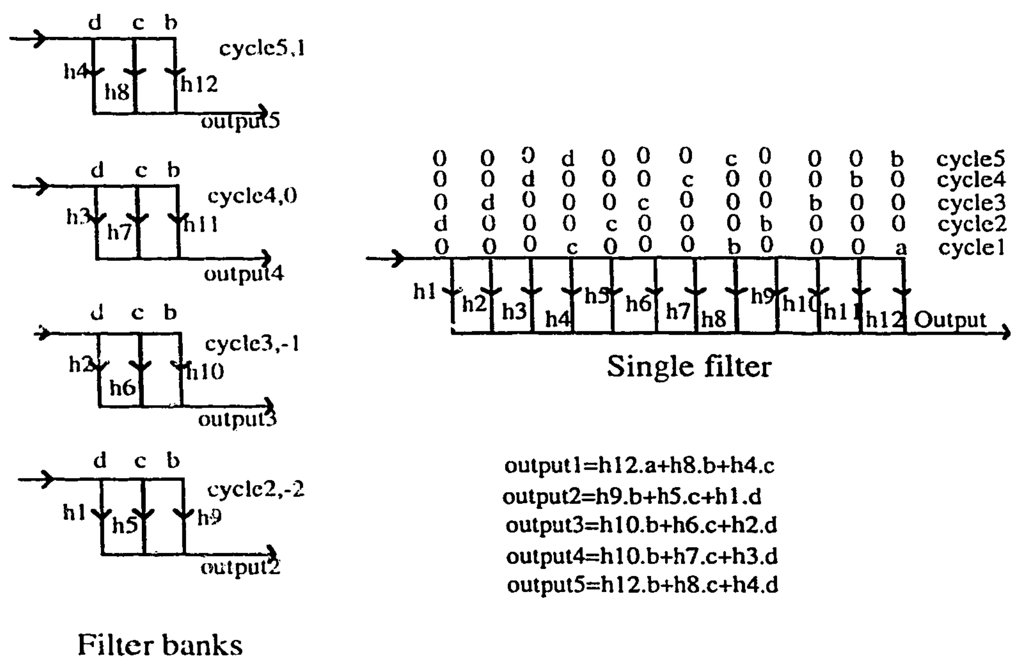


Fig. 4.6. Multirate interpolation filter structure (M=4).

As shown in Fig. 4.6 the interpolation can be performed by inserting zeroes in between the data symbol values, and then filtering this expanded data stream. However, if implemented as a simple N-tap FIR filter, each filtering operation would take N DSP clock cycles, and, as can be seen from Fig. 4.6, the majority of the multiplication results in zero. It is more time efficient to use a multirate structure by dividing the filter into M sub-filters, where M is the oversampling ratio, and each sub-filter includes only those h(t) values that are multiplied by the actual data values [37]. Each sub-filter thus has only N/M coefficients; the filtering operation for each sample is also reduced to N/M. The grouping and operation of the sub-filters, for M=4, is shown in Fig. 4.6. The input counter directs the data input values to each sub-filter in turn. The h(t) coefficient values shown in each block correspond to the appropriate values in Fig 4.6. In the DSP the sequential use of each sub-filter is obtained using incremental addressing to access the coefficient blocks.

Matlab© is used to design the SRRC filter. A value of N=96 was found to be sufficient to accurately obtain the desired response. With an oversampling factor (M) of 8, this requires 8 filter blocks with 12 coefficients in each block. Since the output of the $\pi/4$DQPSK modulator is a complex signal (I and Q components), two of these filters are implemented in the TMS320C50 DSP. To facilitate the software development and to reduce the required number of instruction cycles, an object-oriented programming [38] technique is used in the filter implementation. The DSP program consists of 8 filter objects, with each object containing two 12-tap FIR filters, one for the I signal and the other for the Q signal. The timing of each object is controlled in a synchronous mode rather than burst mode, i.e. each filter object operates synchronously with the sampling clock and not 1/8th of the sampling clock. The output of each object is the transmitter output, as shown in Fig. 4.7. For transmission to the receiver this output must be sequentially multiplied by the in-phase and quadrature components of the fading channel. The implementation of the fading channel is explained in the next section.

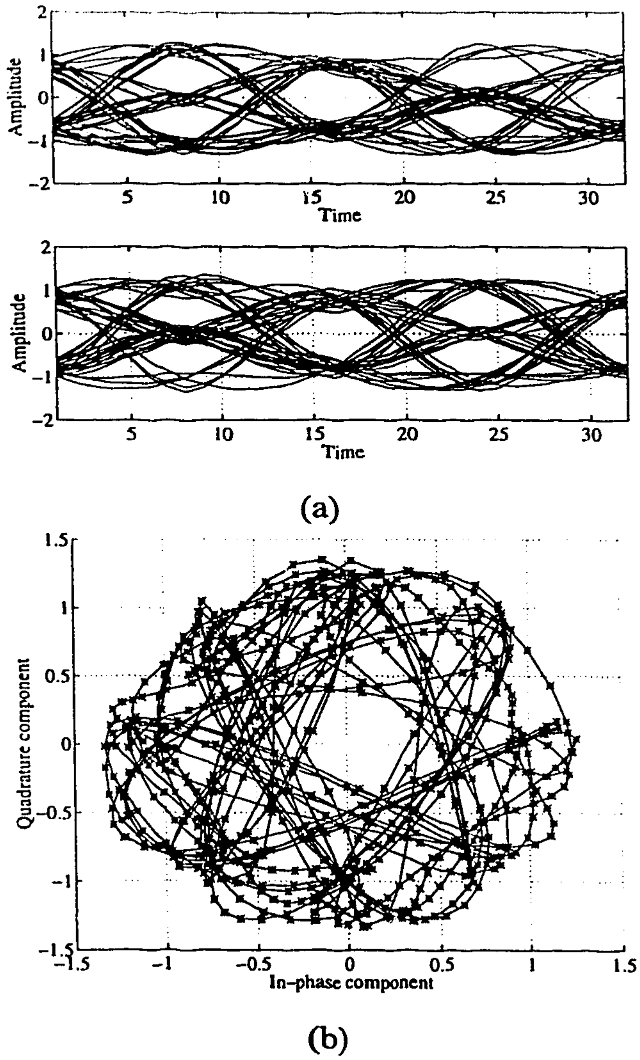


Fig. 4.7. Transmitted signal. (a) Eye-diagram. (b) Constellation diagram.

4.2 Flat Fading Channel

As described in Chapter 2, the transmitted signal propagates through the fading channel before reaching the receiver antenna mounted at the mobile unit. Different methods of simulating the baseband equivalent of the Rayleigh fading characteristics are described in Jakes [18]. The filter shaping method is preferred because of its simplicity of implementation in the DSP [39]. The block diagram of the fading simulator is shown in Fig. 4.8; its implementation is explained in the following sub-sections.

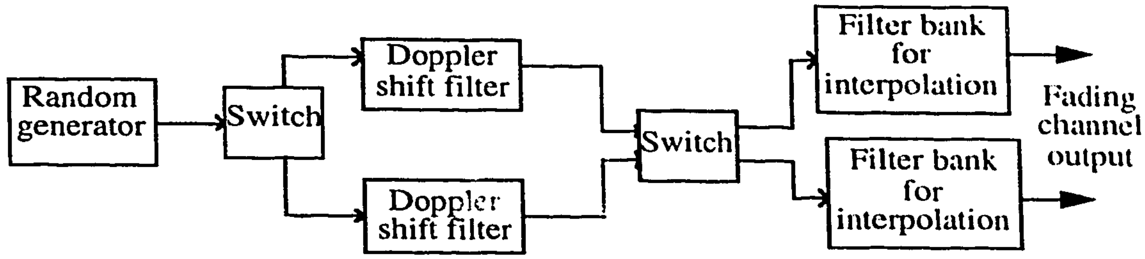


Fig. 4.8. Block diagram of fading channel model.

4.2.1 Gaussian Sources

A complex base-band equivalent channel characteristic with a Rayleigh distributed envelope may be easily obtained using two Gaussian noise generators, one for the in-phase and the second for the quadrature component of the channel amplitude. These Gaussian signal (processes) are CI(t) and CQ(t) respectively of Eq. (2-6). In order to generate two independent Gaussian sources, the central limit theorem is invoked, which states that [19]

The distribution of the sum of N statistically independent and identically distributed random variables with finite mean and variance approaches a Gaussian distribution as N approaches infinity.

Since the channel characteristic must be spectrally shaped, the I and Q components are obtained as the output of two FIR filters. Uniformly distributed random numbers are used as input to these filters. Since each filter output consists of the sum of the input random numbers as modified by the filter coefficients, then, for a sufficiently long filter the central limit theorem will apply and the output will approach the Gaussian pdf. The uniformly distributed random sources are generated as a pseudo-random binary sequence (PRBS) using the M-sequence method, whose principle of operation is explained in Section 4.1.1. In order to ease the problem of implementing the four random generators (two used for data and two used for the fading channel), 16 parallel shift registers (PRBS sources) are used, instead of 4 independent shift registers. Each generator has a different seed so that the output of each generator is independent of the others. The number of parallel sources, 16 in this case, is limited by the word length of the DSP memory. The shift register length and tap values used in each of these generators must be identical for circular buffer implementation and parallel operation. As mentioned already, the sequence length is selected to be (2^{31}-1) since this gives a very long period of random generation, e.g. at a symbol rate of 1.2 ksymbols/sec, the sequence period is equal to 20.7 days. In order to further make sure that all the sources are independent, 16 equally spaced seed values are chosen from the 31 bit maximal length sequence; these seed values are then entered in the random generator block as 31 16-bit words, as shown in Fig. 4.9. Every cycle of the random generator produces a new random word, giving 16 independent binary bit sources. For each random 16 bit binary word, 7 bits are used for each of the **chI** (in-phase) and **chQ** (quadrature) random number inputs, and the remaining two bits are used to generate the data symbol as shown in Fig. 4.9. Care must be taken while extracting and storing each 7 bit segment from the 16 bit word, so as to maintain the appropriate sign bit in each source. After obtaining the uniformly distributed words, the filtering is done using a 121 point FIR filter. This is a large enough number of points that the output will approach a Gaussian noise source, as per the central limit theorem. Separate averaging must be done for **chI** and **chQ**, therefore two filters are required. The required tap weightings for shaping the power spectrum of these fading generators are obtained as described in the following sub-section, (4.2.2).

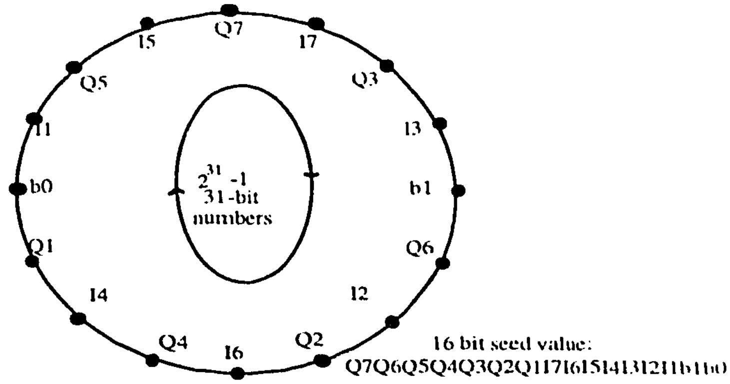


Fig. 4.9. Seed value generator for random generator.

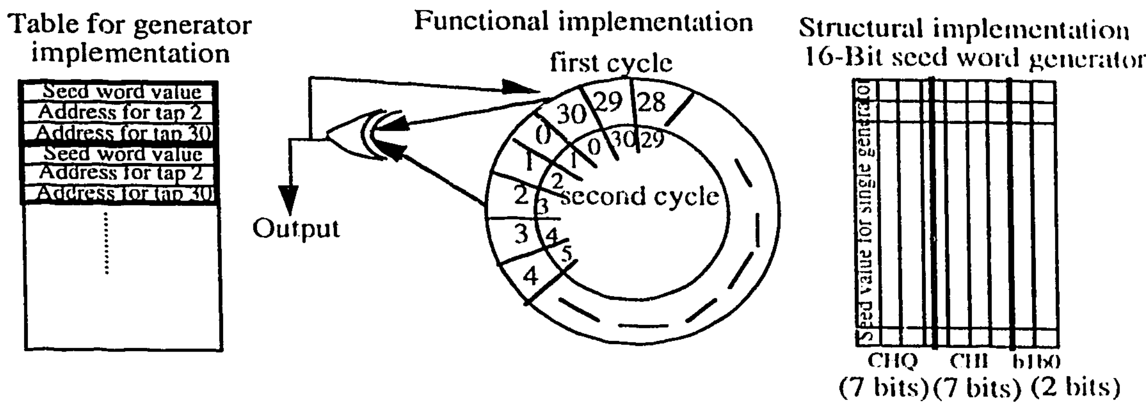


Fig. 4.10. Circular buffer implementation of random 16-bit word generator.

The actual software module for the 16-bit word generator is implemented using a "memory table" that simulates the action of a circular buffer. The circular buffer available in the C50 could not be used because of its limitation in increment count to ±1. The "memory table" uses 31 groups of 3 memory locations to implement a 31 length shift register. The first location in each group is a seed value (16 bit word) for the 16 bit generator. The second location in the group is the memory address containing the seed word (register) corresponding to tap 2, and the third location of the group has the address of the seed word (register) corresponding to tap 29. These taps are selected for maximal random sequence length as described in Section 4.1.1. Unlike an independent shift register, the circular buffer implementation does not have fixed tap coefficient locations or registers. This is due to the dynamic change in register tap number; i.e., instead of shifting up the seed bits in the registers (locations), the register number is changed such that the overall effect is the same, as shown in Fig. 4.10. This implementation is not straight forward, but it requires less computation than other methods and is therefore preferred.

4.2.2 Doppler Shift Filter

The Doppler shift filter serves two purposes: one is to produce a Gaussian distributed output and the second is to shape the output in order to obtain the required Doppler power spectrum. Two Doppler shift filters are used, one each for the in-phase (chI) and quadrature components (chQ) of the channel. These filters are designed to simulate the measured spectrum as it would be received in a practical mobile radio environment. It has been found experimentally that the received signal strength is very close to the analytical result, provided that an omnidirectional antenna is mounted at the receiver. It has been proved, as discussed in Section 2.2.2.1, that the spectrum can be formulated as [18]

$$
\begin{aligned}
S_s(f) &= \frac{C}{\sqrt{f_m^2 - f^2}} && -f_m \le f \le f_m \\
&= 0 && \text{otherwise} \qquad (4\text{-}13)
\end{aligned}
$$

where $S_S(f)$ is the power spectrum of the fading signal, C is a constant depending on the antenna gain pattern and path attenuation factor, and f_m is the maximum Doppler frequency. The range of the Doppler frequency is between 0 and 83.3Hz, for a vehicle speed of up to 100km/hr when a carrier frequency of 900MHz is used. For a maximum Doppler shift frequency of 80Hz, the received signal spectrum is as shown in Fig. 4.11. The Doppler shift filters are designed so that their transfer functions match this spectral shape as closely as possible.

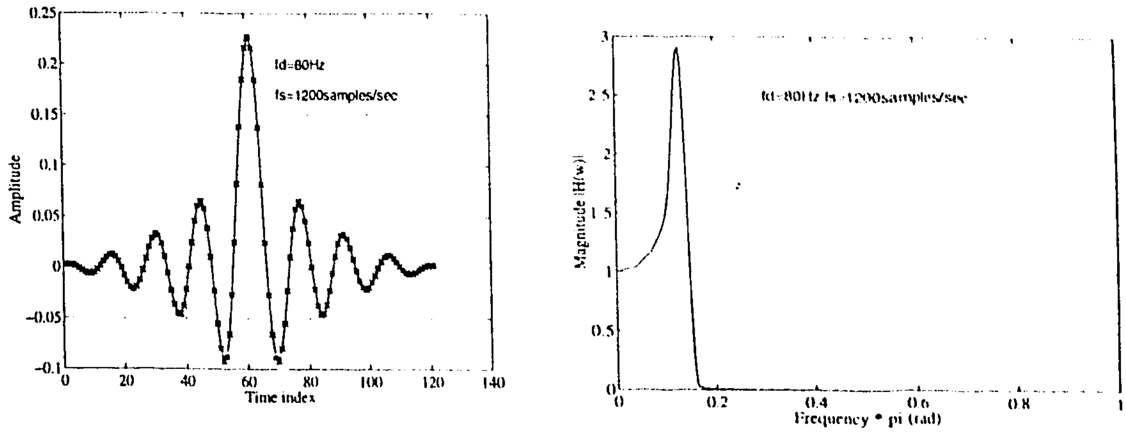


Fig. 4.11. Time domain and frequency response of Doppler shift filter.

The sampling rate of the channel needs to be the same as that of the data, namely 9600 samples/sec. Because it is difficult to design a filter with a high sampling rate to cut-off frequency ratio, the Doppler shift filter is initially designed with a sampling frequency of 1200samples/sec and a cutoff frequency of 80Hz. Interpolation is then used to increase the sampling rate, and if necessary, change the cut-off frequency at the same time; e.g., 16 times interpolation is used to produce an effective 9600 samples/sec with a 40Hz Doppler shift.

The Doppler shift filter is designed as an FIR (linear phase) filter, with a tap length of 121 coefficients. The *least square method* of filter design, from the Matlab© signal toolbox, is used to obtain the specified frequency and magnitude characteristics. An IIR filter with fewer taps could have been used but FIR is preferred over IIR because of its inherent stability and more pronounced peak at the maximum Doppler frequency. The power spectral density at the frequency f_m is about 10dB higher than that at $f=0$. The requirement of a variable bandwidth filter (variable Doppler shift) is implemented using a variable sampling rate for both the filter and the Gaussian Noise generator. A programmable counter is used to provide the required sampling rate for each simulation case. The hardware and software control of the sampling rate and the Doppler shift frequency is further explained in the following sub-section.

4.2.3 Interpolating Filter

Two interpolation filters, one each for the in-phase and quadrature components of the channel are required. Interpolation increases the sampling rate of the Doppler shift filter output to a value equal to that of pulse shaping filter output for the data, thereby facilitating the digital superposition of the channel upon the transmitted signal. The multirate implementation, as discussed for the pulse shaping filter, is used so as to save instruction cycle time. The design of the filter depends on the interpolation factor and the required suppression of the undesired images.

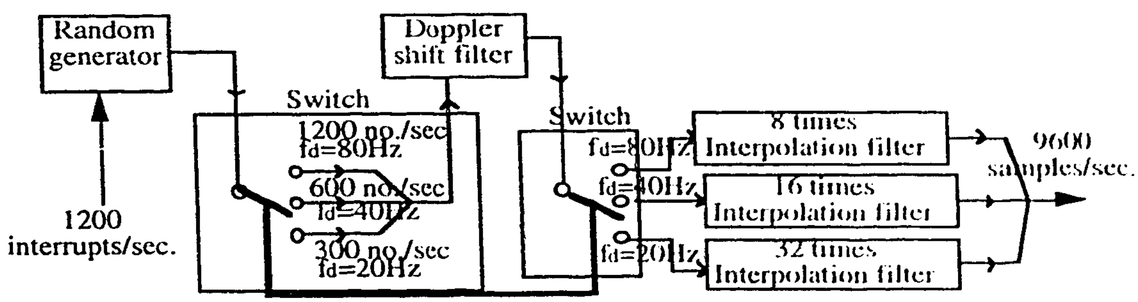


Fig. 4.12. Software architecture for fading channel.

The interpolation filter is designed as a FIR filter using the Matlab© signal processing toolbox. The tap length of the FIR filter depends on the sampling rate and the interpolation factor. The architecture of the algorithm for fading sample generation, for different Doppler frequencies, using a software switch, counter and interrupts is shown in Fig. 4.12. For f_d = 40 Hz, and a sampling rate of 600 samples/sec, the Doppler shift filter uses the same coefficients as were used for the Doppler shift filter with f_d=80Hz, at the sampling rate of 1200samples/sec. However a software counter is first used to lower the input rate to 600 samples/sec, and f_d=40Hz, then an interpolation filter with oversampling factor of 16 is used to interpolate between these samples, suppressing undesired images between (600-f_d) to (4800-f_d). This effectively produces the same waveform as for the 80Hz Doppler shift filter, but slowed down in time by a factor of 2. For f_d = 20Hz, the software counter adjusts the Doppler shift filter input to 300 samples/sec; the interpolation filter is then designed to interpolate between these samples with oversampling factor of 32 suppressing undesired images between (300-f_d) to (4800-f_d). The tap length of the designed interpolation filter is 96 for f_d equal to 40Hz, and 352 for f_d equal to 20Hz. Multirate implementation as discussed in Section 4.1.3, is used for these interpolation filters. The output of the fading simulator is shown in Fig. 4.13.

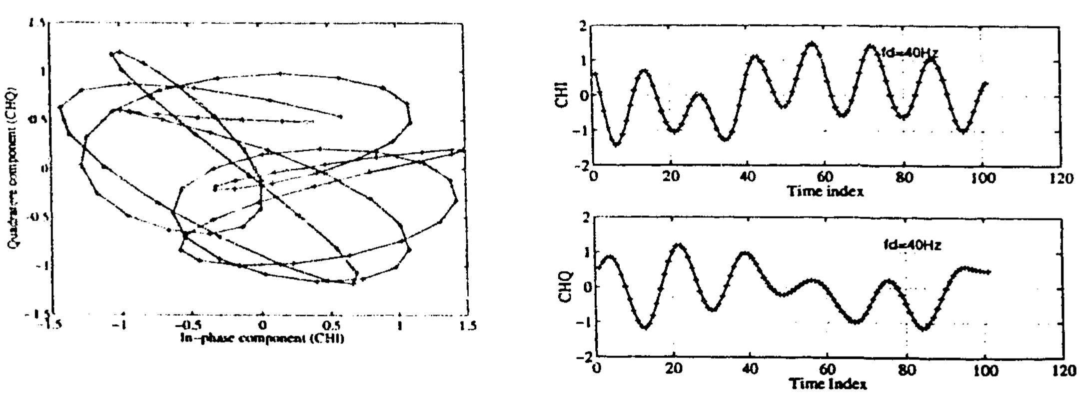


Fig. 4.13. Fading channel simulator output.

Although the filter structure is simple, it takes a large number of instruction cycles to generate the complete fading process, especially for the lowest Doppler frequency value, because of the increase in tap length of the filters. It is for this reason that the data rate was limited to 1200symbols/second. Since both the transmitter and fading channel are sequentially implemented in one DSP, each generated data symbol has to wait for the fading sample generation process to be completed.

4.3 Channel Superposition on Transmitted Signal

4.3.1 Complex Multiplication

Once the fading channel and transmitter output samples are calculated they are digitally combined using complex multiplication. The software module for this performs the following simple calculations,

$$rxI + jrxQ = (SI + jSQ)(chI + jchQ), \tag{4-14}$$

so that

$$rxI = SI * chI - SQ * chQ \tag{4-15}$$

and

$$rxQ = SI * chQ + SQ * chI \tag{4-16}$$

The output of the combiner is then fed sequentially to the DSP TMS320C30 using an external interface circuit, which is discussed in the following sub-section. This signal, representing the baseband equivalent of the signal received by the mobile unit receiver antenna, is shown in Fig. 4.14. By comparing Figs. 4.7, 4.13 and 4.14, the distortion of the transmitted signal due to the fading channel is clear.

4.3.2 Interface

The interface circuit is required to buffer the output of the TMS320C50 and transfer it to TMS320C30. The design of the interfacing circuit depends on the operating mode used by the DSP program. Since the synchronous mode of operation is used, the circuit operates at the sampling frequency to buffer the data between two DSPs.

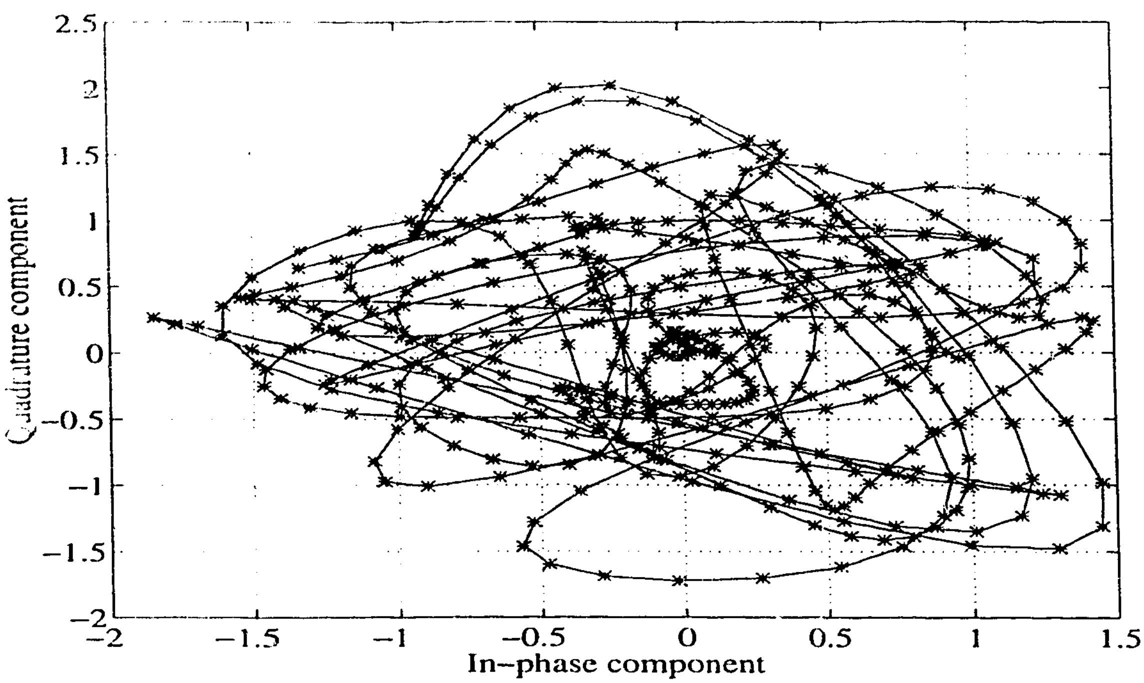


Fig. 4.14. Received signal waveform.

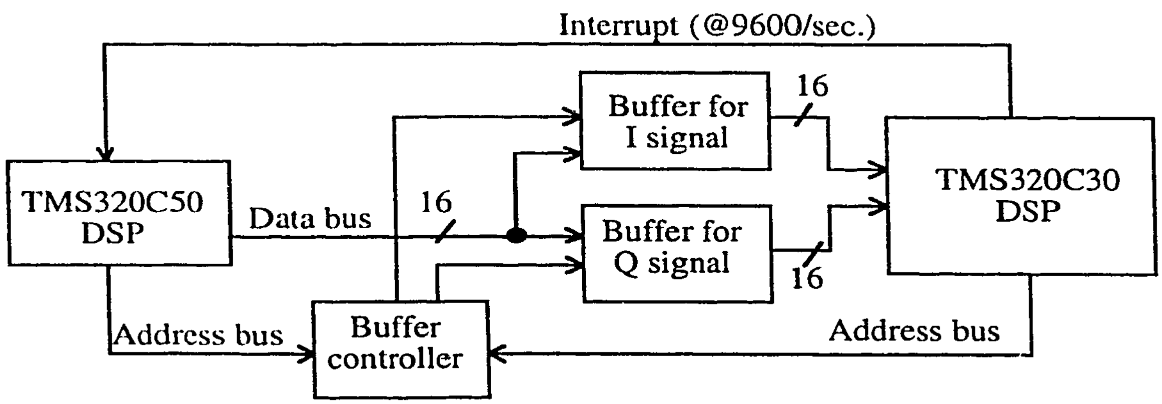


Fig. 4.15. Block diagram of interface circuit between DSP C50 and C30.

The block diagram of the interface circuit is shown in Fig. 4.15. It consists of two components: two 16-bit buffers for the in-phase and quadrature components of the received signal, and a buffer controller. The two 16-bit buffers provide a data access interface between the TMS320C50 DSP and the

TMS320C30 DSP. The buffer controller circuit monitors the TMS320C30 DSP address bus and the TMS320C50 address bus, and generates an appropriate control signal for the 16-bit buffers. These control circuits are used by the buffer for two functions: to latch a new set of samples from TMS320C50 (two operations, to write two 16 bit words into the buffer), or to enable the buffers for access by the TMS320C30 (a single read operation of all 32 bits of data). Initially the interface circuit was built using an SK-10 bread-board for prototype testing. Since the bread-board capacitance makes it unreliable at the clock rate of 20MHz, a printed circuit board (PCB) was built. The details of the PCB design process are shown in Appendix A.

4.4 Receiver

The receiver structure, as described in Chapters 2 and 3, is implemented using the Texas Instruments TMS320C30 DSP development platform [40, 41, 42] and an IBM PC486 compatible computer. The key features of this DSP are given in Table 4.4. A synchronization circuit is not designed in the receiver; instead, the sampling rate is controlled via a hardware interrupt on the C30 board. The basic building blocks of the receiver are described in the following sub-sections.

TABLE 4.4
FEATURES OF DSP TMS320C30.

60ns instruction cycle (16.7MIPS and 33.3MFLOPS)
Two 1K $\times$ 32 bit dual-access on-chip RAM
32-bit instruction and data words, 24-bit addresses
40/32 bit floating point/integer ALU and multiplier
8 extended-precision registers
On-chip Direct Memory Access (DMA) controller
Two serial ports, two 32 bit timers

4.4.1 Additive White Gaussian Noise Source

In nature, the ideal transmission of information is degraded by two types of phenomena: imperfect equipment (distortion), and the presence of "generalized noise", i.e., any signal other than the desired signal itself. The modeling of this noise as a Gaussian random process is well known. For simulation purposes we therefore need to be able to pseudorandomly generate such a process. Random samples generated independently at intervals of T_S correspond to a noise source with uniform power spectral density over a bandwidth equal to $1/T_S$. To simulate spectrally shaped Gaussian noise, "white" noise needs to be passed through a filter whose transfer function produces the desired shape. In the software implementation, the shift register method of PRBS generation is used to generate white noise. This is filtered by the SRRC filter at the receiver front end.

The noise generation program comprises modules to generate, add and measure the noise signal. A circular buffer of block length 28 is used as a shift register, with feedback taps 3 and 28, to obtain a maximum sequence of length $2^{28}-1$. This length is chosen so as to make sure that the sequence is independent of all the other sequences previously obtained, which used a 31 length shift register. The complete 32-bit word length of the TMS320C30 is used to generate two components (16 bits each for the real and imaginary parts) of the complex noise process. 28 initial seed values are randomly selected so as to initiate the noise generation. Each 16-bit component is moved to the least significant part of the two noise output words using an arithmetic shift so as to retain the sign bit value. The integer values so stored are changed to floating point values so as to obtain good dynamic range for the noise power, and also so that they may be multiplied by a variable scaling factor. This scaling factor can be changed to obtain different levels of noise power, and thus different SNR values for the signal. The generated complex noise is added to complex received signal, i.e. the real and imaginary part of signal and noise are added separately, and the resulting output is fed to the receiver bandpass filter (a SRRC filter, identical to the transmitter filter). Power calculations use the sum of the square of the measured signal amplitude at the output of the bandpass filter, divided by the number of samples. To calculate the noise power and signal power so as to measure the SNR, an extra sub-module is added into the program. In this power measurement module, either the noise signal or the received signal is set to zero prior to the input bandpass filter, so that the other quantity may be measured, i.e. two runs of the program are used. It is essential to make both noise and signal power measurements because the transmitted signal power is not normalized to one. The ratio of the signal and noise powers is calculated manually, off-line, to find the SNR in dB.

4.4.2 Receiver Bandpass Filter

The bandpass filter used at the front end of the receiver is exactly the same as the pulse shaping filter at the transmitter. As specified by the IS54 standard, the frequency response is the square root raised cosine shape with α=0.35, as discussed previously in Section 4.1.3. This filter is implemented digitally in the C30, using an FIR structure. The only difference between the transmitter and receiver filters is in their implementation. Unlike the pulse shaping filter at the transmitter, the bandpass filter at the receiver is not used for interpolation or oversampling. Thus, the multirate form of implementation can not be used at the receiver end. Also, since the C30 is a floating point DSP, all the calculations for the receiver filtering operation are done in floating point format.

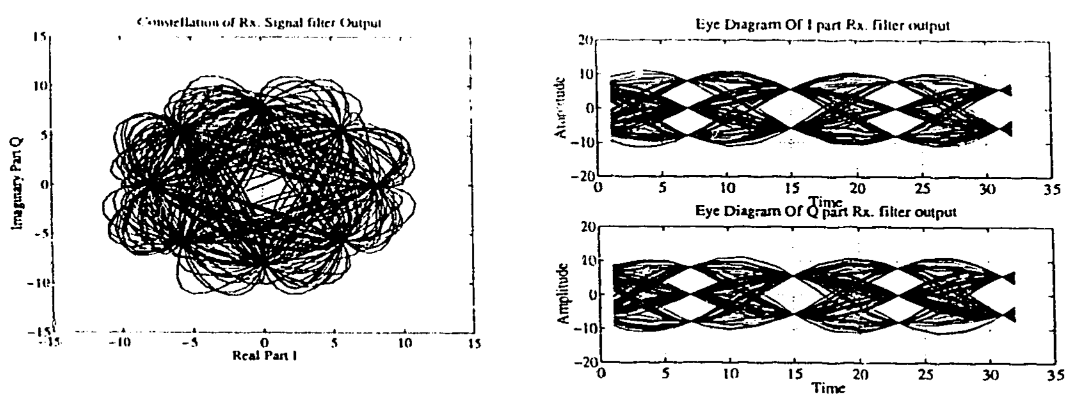


Fig. 4.16. Zero ISI waveform.

In order to observe the implementation correctness, the transmitter and receiver are connected back to back, with the noise and channel turned off (noise=0, channel=1). The constellation diagram and the eye diagram for this test are as shown in Fig. 4.16. The diagrams display a sharp pinch point at the sampling instants, indicating zero ISI, and the eye is wide open; this may be compared with the transmitter filter output shown earlier in Fig. 4.7. The overall zero ISI is obtained because the combination of two square root raised cosine filters give a raised cosine filter, which meets the Nyquist criterion.

An important point to note here is that the zero intersymbol interference criterion is no longer valid once the effect of the fading channel and noise has been impressed on the transmitted signal. This is because the overall transfer function is the product of the transfer function of the two filters and the channel, which does not meet the Nyquist criterion. The received signal at the output of the bandpass filter (Fig. 4.17) does not exhibit sharp pinch points in either the eye-diagram or the constellation diagram, and the eye is no longer wide open at the sampling instant.

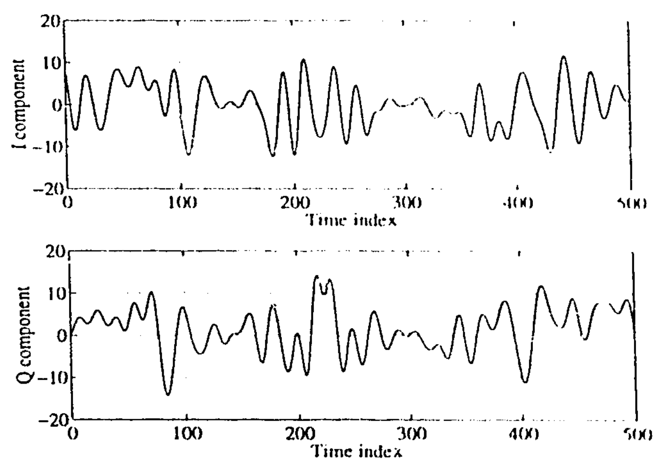


Fig. 4.17. Receiver bandpass filter output.

4.4.3 Decision-Aided Detector

The bandpass filter output is sampled once per symbol period, in order to extract the sample corresponding to the instant when the eye is wide open. This is required by the detector so as to make its decision at the optimum time. In order to simplify the implementation, the sampling time is controlled by a C30 interrupt, rather than using a timing recovery algorithm. At the appropriate symbol instant, one sample is recovered from the received 8 samples per symbol period. The appropriate symbol instant is the instant at which the pinch point is observed for the case of the back-to-back transmitter and receiver test. This is equivalent to the assumption that the transmitter and receiver are synchronized to the same clock. However, this assumption is not critical to the evaluation of the decision-aided error correction scheme because timing errors primarily affect D1, and the error improvement is measured as the ratio of D2 to D1.

4.4.3.1 Differential Detector

Differential detection, as discussed in Chapters 2 and 3, is used to make the first decision about the transmitted data, based on the information at the output of the receiver bandpass filter. This method of detection is preferred over coherent detection because it does not require complex processes for carrier synchronization, e.g. tracking, acquisition, lock detection, false lock prevention etc. Differential detection is accomplished by comparing the received signal phase in a given symbol interval (of duration T_S seconds) with that in the previous symbol interval and making a decision on the difference between these two phases. In the absence of channel distortion and noise, the differential phase detected by the detector will be exactly equal to one of the 4 possible transmitted phase shifts $\{\pm 45°, \pm 135°\}$; these are the differential phase shifts imposed by the $\pi/4$DQPSK modulation.

The DSP software implementation for the differential detector uses the rectangular co-ordinate system instead of the polar co-ordinate system, thereby avoiding any $\tan^{-1}$ operations. The in-phase and quadrature phase components at the output of the bandpass filter are stored for both the k^{th} and $(k-1)^{th}$ symbol instants, where k is an integer. The phase differential is obtained by simple multiplication and addition/subtraction operations derived from the vector division of the signals at the times k and k-1. The signals are

$$r_k\angle\phi_k = I_k + jQ_k \quad \text{and} \quad r_{k-1}\angle\phi_{k-1} = I_{k-1} + jQ_{k-1} \quad , \tag{4-17}$$

and the phase difference can be found from,

$$\frac{r_k\angle\phi_k}{r_{k-1}\angle\phi_{k-1}} = \frac{I_k + jQ_k}{I_{k-1} + jQ_{k-1}} = U + jV \tag{4-18}$$

Thus, $$U = b(I_k I_{k-1} + Q_k Q_{k-1}) \tag{4-19}$$

and $$V = b(Q_k I_{k-1} - I_k Q_{k-1}) \tag{4-20}$$

where b is scaling factor. For a typical signal, Fig. 4.18 shows the set of samples corresponding to the vector U+jV.

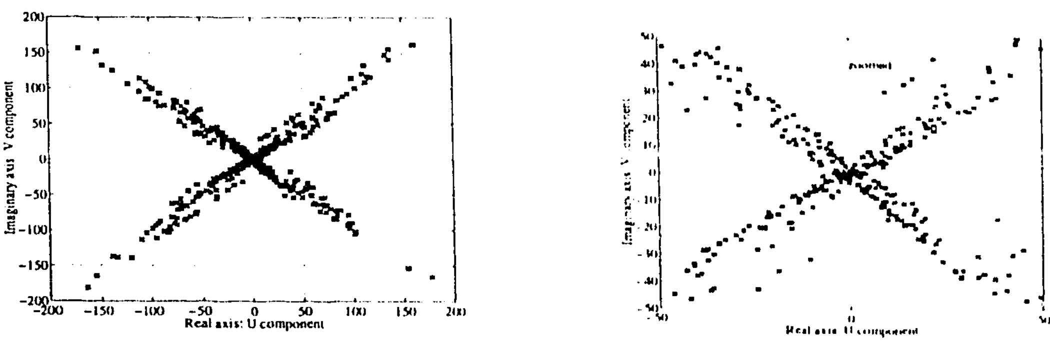


Fig. 4.18. Differential detector output (U+jV).

The differential phase angle is $\tan^{-1}(V/U)$, but the decision about the transmitted data can be made based on the quadrant in which the vector corresponding to U+jV lies. For DSP software implementation, the decision is made using a small look-up table, the indices (i, j) of which are determined using the sign bit (positive or negative) of U and V. The indices calculation and look-up table entries for the first decision (D1) are shown in Table 4.5.

TABLE 4.5
LOOK-UP TABLE FOR FIRST DECISION.

$U \geq 0$	i=0	for i+j=0	D1=00
$U < 0$	i=2	for i+j=1	D1=10
$V \geq 0$	j=0	for i+j=2	D1=01
$V < 0$	j=1	for i+j=3	D1=11

The first decision values are stored in circular buffers of length 31, i.e. the 31 most recent first decisions are stored and the older values are over-written by the new values in circular fashion, one by one. These stored values will be used while making the second decision using the decision-aided error correction algorithm. The first decision, in conjunction with the transmitted data, is used to calculate the primary symbol error rate, as explained in the error measurement section (Section 4.5).

4.4.3.2 Reverse Modulator

Since the channel is a multipath fading channel with AWGN noise, the detected differential phase, as indicated by **U** and **V**, deviates from the ideal data phase change values of {±45°, ±135°}. The deviation from the ideal phase angle, caused by the effect of the fading channel and noise, and called here the "extracted channel" is extracted by the reverse modulator. Note that the first decision (D1) is used by the reverse modulator to determine the transmitted phase change. A plot of the extracted channel phase, using a Matlab© simulation, is shown in Fig. 4.19. Since the fading is a narrowband process, the extracted channel should be band-limited to the $2f_d$ (f_d is the maximum Doppler frequency). However, Fig. 4.19 shows some high frequency components; i.e., some points are out of place, which occurs mainly at the times when the transmitted symbols are not detected correctly by the differential detector. This is expected because during deep fades the phase changes caused by the channel can be very large, leading to incorrect first decisions, and thus incorrect data phase information fed back to the reverse modulator. This directly results in errors in the extracted channel information.

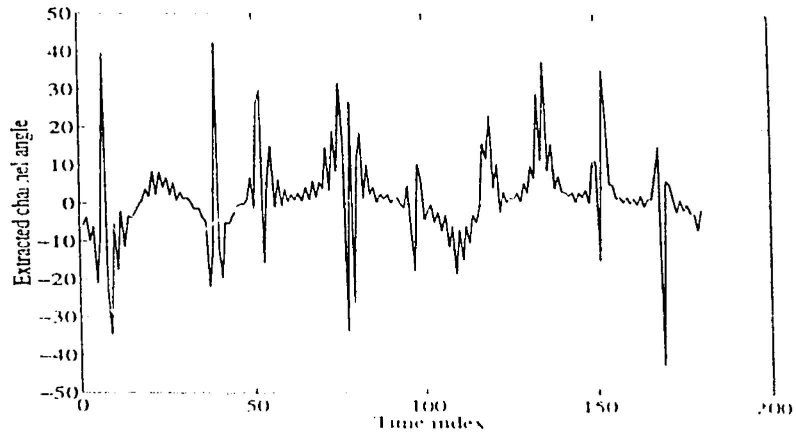


Fig. 4.19. Extracted channel using angles.

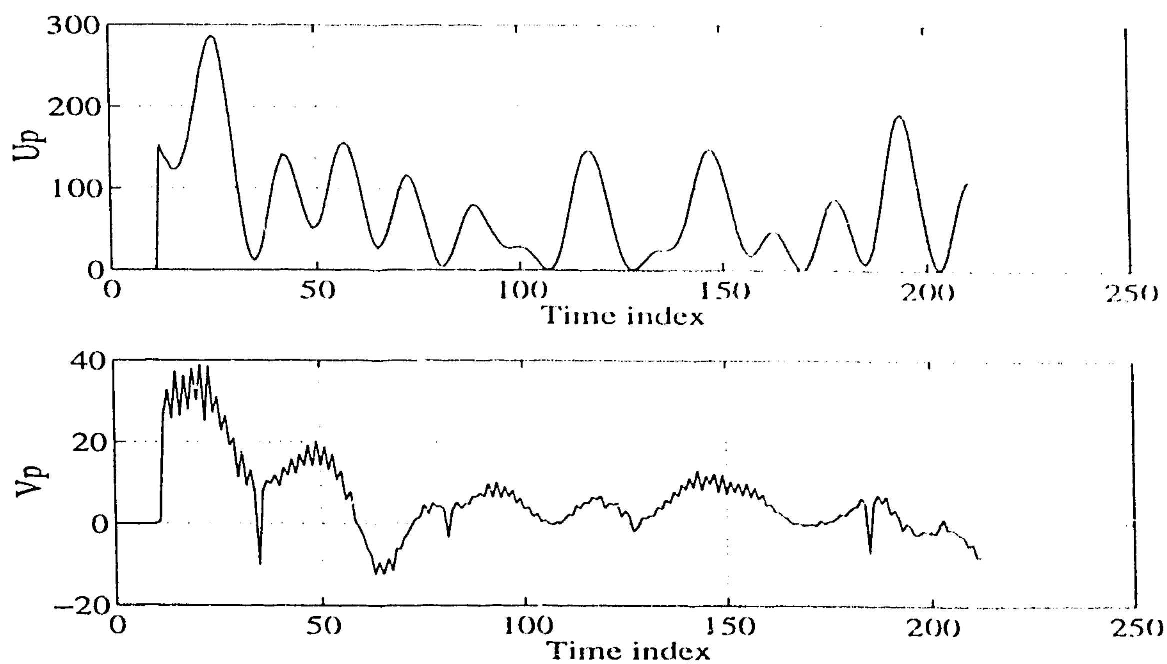


Fig. 4.20. Extracted channel using Up and Vp.

For DSP implementation, the use of $\tan^{-1}$ is undesirable and therefore the reverse modulator finds the extracted channel phase information in terms of **Up** and **Vp**, as discussed in Chapter 3. Fig. 4.20 shows a typical **Up** and **Vp** graph. In the implementation, the phase deviation is always calculated with reference to the corresponding detected data phase change. This means that for each symbol, the reference corresponds to one of the four possible transmitted phase changes {±45°, ±135°}, depending on the quadrant in which the received signal phase change corresponding to **U+jV** lies. Determination of **Up** and **Vp** for each of the four possible data phases is illustrated in Fig. 3.3b.

Mathematically,

if $$U + jV = re^{j\Delta\phi_{rx}},$$

then
$$U_p + jV_p = re^{j\left(\Delta\phi_{rx} - 45°\right)} \text{ for D1=45°, (first quadrant)}$$
$$U_p + jV_p = re^{j\left(\Delta\phi_{rx} - 135°\right)} \text{ for D1=135°, (second quadrant)}$$
$$U_p + jV_p = re^{j\left(\Delta\phi_{rx} + 135°\right)} \text{ for D1=-135°, (third quadrant)}$$
$$U_p + jV_p = re^{j\left(\Delta\phi_{rx} + 45°\right)} \text{ for D1=-45°, (fourth quadrant).} \quad (4\text{-}21)$$

Calculating Up and Vp in terms of U and V, and using rectangular co-ordinates, then

$$U_p + jV_p = (U + jV)e^{-j45°} \qquad \text{for D1=45°,}$$
$$U_p + jV_p = (U + jV)e^{-j135°} \qquad \text{for D1=135°,}$$
$$U_p + jV_p = (U + jV)e^{+j135°} \qquad \text{for D1=-135°,}$$
$$U_p + jV_p = (U + jV)e^{+j45°} \qquad \text{for D1=-45°.} \quad (4\text{-}22)$$

The resulting values of **Up** and **Vp** are given in Table 4.6.

TABLE 4.6
REVERSE MODULATOR OUTPUT.

D1 values	Up (in terms of U and V)	Vp (in terms of U and V)
+45°	(U+V)/sqrt(2)	(V-U)/sqrt(2)
+135°	(V-U)/sqrt(2)	(-U-V)/sqrt(2)
-135°	(-U-V)/sqrt(2)	(U-V)/sqrt(2)
-45°	(U-V)/sqrt(2)	(U+V)/sqrt(2)

Thus the reverse modulation process is simplified to a conditional addition and/or subtraction operation. The conditions for the **Up** and **Vp** calculations are based on the first decision value, which depends on **U** and **V** (**U** and **V** in turn correspond to the differential received phase angle). Therefore, to implement this conditional logic, it is possible to use simple logic instead of a conditional branch instruction, which helps to reduce the instruction cycle time. Thus, the sign extraction of **U** and **V** is used by the implemented software module, as given in Table 4.7, for the reverse modulation.

TABLE 4.7
LOGIC FOR REVERSE MODULATOR OUTPUT.

Condition	Intermediate variables for Up; where Up = j+l	Intermediate variables for Vp; where Vp = k+m
$U \geq 0$	j = U	k = V
$U < 0$	j = -U	k = -V
$V \geq 0$	l = V	m = -U
$V < 0$	l = -V	m = U

These values of **Up** and **Vp** corresponds to the fading estimate as affected by the noise. Since the first detection is not always correct, this estimate includes the effect of any incorrect first decisions as well. Therefore, the next step in the decision-aided detector is to minimize the effect of the incorrect first decisions on the fading estimate using the channel estimator.

4.4.3.3 Channel Estimator and Error Corrector

The channel estimator is the heart of the decision-aided detector and the performance of the algorithm depends largely on this part. The goal of the estimator, as discussed in detail in Chapter 3, is to obtain the best possible estimate of the phase changes due to the fading channel, using the extracted channel information at the output of the reverse modulator. Since, in a typical case, one out of one hundred first decisions are incorrect, the channel recovery at the output of the reverse modulator contains 1% incorrect estimates. The estimation filter is designed to remove this distortion as far as is possible, based on both past and future values of the recovered channel phase; i.e., an interpolation based method of estimation is used rather than a prediction based method. The design of the estimator is based on the fact that the fading is a narrow-band process having a bandwidth of twice the maximum Doppler shift.

As discussed in Chapter 3, in designing the channel estimation or reconstruction filter, a compromise must be made between the filter length and the accuracy of the spectral characteristics. Thus, the number of points (N) that can be used for estimation, on each side of the decision point (k=-N to 0 to N), is taken to be in the range of 2 to 30. Due to data errors, erroneous samples of **Up** or **Vp** enter the estimation filter at k=-N. Any channel sample in error has its maximum effect as it reaches the middle of the filter, since the h(0) coefficient has the largest value. One improvement that may be made in the filter design is to modify the filter by setting the middle filter coefficient value to zero. Since the gain of the filter must be kept at unity, the impulse response of the modified filter is then required to be

$$h(i) = \begin{cases} 0 & for\ i = 0 \\ \dfrac{h(i)}{1 - h(0)} & elsewhere \end{cases} \quad . \tag{4-23}$$

It should be noted however that any data errors will still have an effect on the channel estimate for k=-N to -1. If the error is corrected at k=0, then the channel data Up and Vp will have no errors in the right hand part of the filter (k>0). Some other practical considerations for designing the filter are: it is necessary that some of the filter coefficients be negative; and, also, the filter shape should be such that the coefficients away from the estimation point dies away towards zero, i.e. at the ends of the filter.

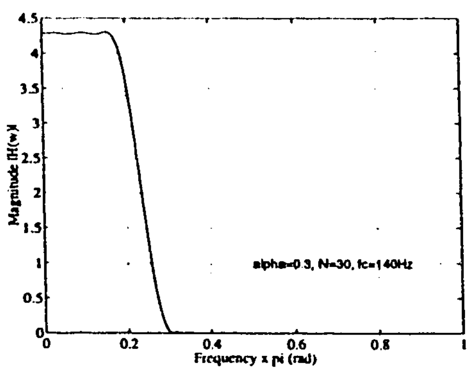
A number of "trial runs" with both the DSP and Matlab© were used to evaluate various possible filter designs. The coefficients of each filter tested were obtained using either the signal processing tool-box of Matlab, or a numerical analysis technique. In the various tests, the cut-off frequency, roll-off factor, and/or the degree of polynomial etc., and the length of the filter were changed, and tests were conducted in order to measure the error rate of the second decision as compared to the first decision. The Improvement Factor (IF) is thus defined as the ratio of the number of errors in the first decision to the number of errors in the second decision. Whatever filter design method is used, the filter should have a flat passband and a cutoff frequency of at least 2 times the maximum Doppler shift. Note that for these preliminary tests no noise is added at the receiver end.

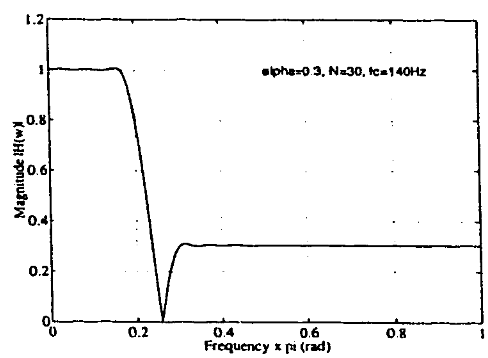
Initially, filter coefficients were calculated by sampling (1.2ks/s) the time domain response of a raised cosine filter, and changing the design parameters such as roll-off factor, cut-off frequency, and the length of the filter. The results of these tests, together with the relevant parameters are given in Table 4.8.

TABLE 4.8
TEST RESULTS BASED ON RAISED COSINE ESTIMATION DESIGN.

Roll-off (α)	Cut-off (f_c)	Points on each side (N)	Improvement factor (IF)	Doppler frequency (f_d)
0.30	140 Hz	30	7.2	20 Hz
0.50	140 Hz	20	14.03	20 Hz
0.30	80 Hz	20	1.7	20 Hz
0.30	190 Hz	20	11.72	20 Hz

For comparison purposes, Fig. 4.21a shows the frequency response of the first filter with both h(0)≠0 and h(0)=0. Fig. 4.21b shows the response for the other three filters with only h(0)=0.





h(0)≠0 h(0)=0

Fig. 4.21. Estimation filter design based on raised cosine filter.
(a) Filter response with h(0)≠0 and h=0.

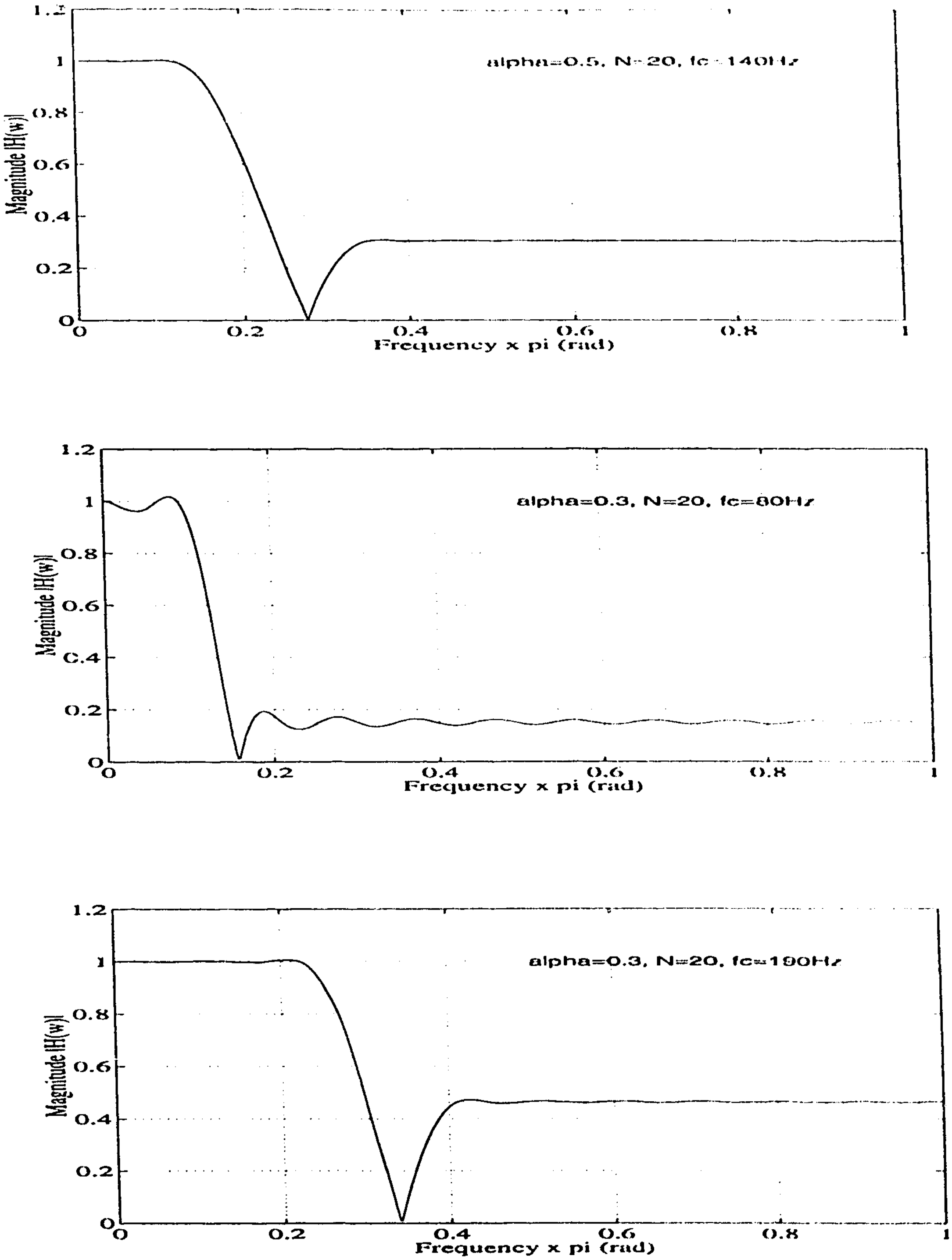


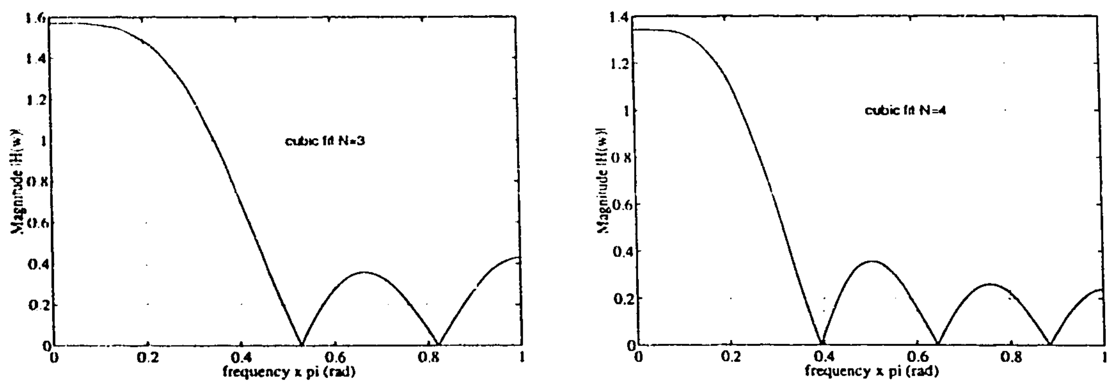
Fig. 4.21. (b) Filter response with h(0)=0.

Since the improvement factor was not significantly large, further tests were tried using least square error and curve fitting. Two sets of filter coefficients based on a least square fit were then used for estimation. The first test uses a cubic fit through three points on each side of the estimation point, while the second test uses four points on each side. The calculated values of the coefficients are:

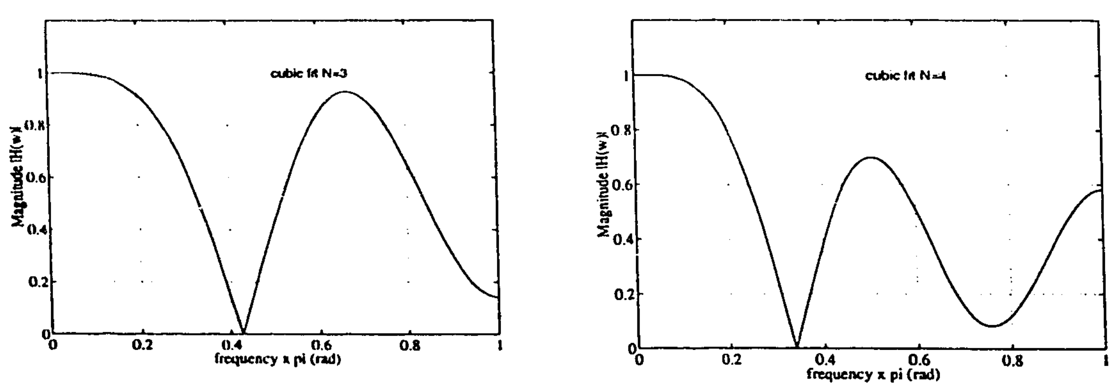
Test I :: [-1/7 3/14 6/14 0 6/14 3/14 -1/7]

Test II :: [-21/172 14/172 39/172 54/172 0 54/172 39/172 14/172 -21/172]

The frequency responses of these filters are shown in Fig. 4.22a and b.



(a)



(b)

Fig. 4.22. Estimation filter design based on least square fit. (a) Filter response with h(0)≠0. (b) Filter response with h(0)=0.

The improvement factor, IF, defined already is found to be about 11.8 for $f_d = 20Hz$, as shown in Table 4.9.

TABLE 4.9
TEST RESULTS BASED ON LEAST SQUARE ERROR FIT.

Test	D1 errors	D2 errors	Run length	Improvement factor (IF)
I	320	27	60477	11.85
II	398	34	73441	11.705

The next experiment used coefficient values calculated using the polynomial fit approach. Again two different sets of coefficients were tested: the first set uses a cubic polynomial fit through two points on each side; the second, uses a fifth order polynomial through three points on each side. The values of the calculated coefficients are:

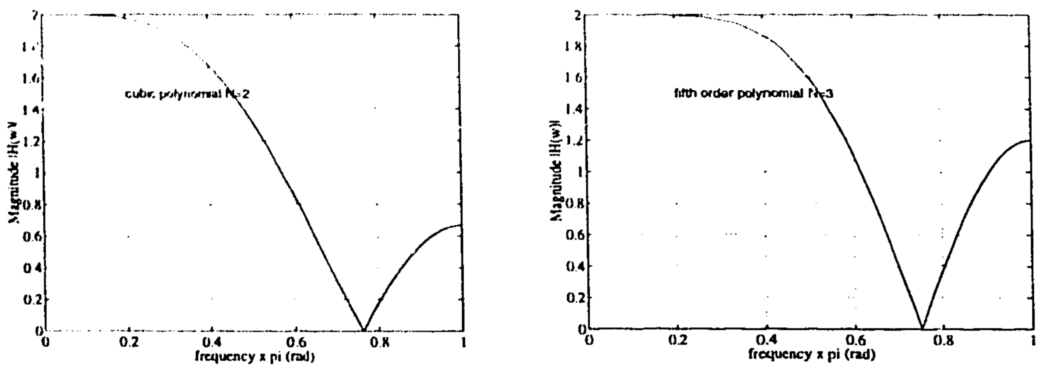
Test I :: [-1/6 2/3 0 2/3 -1/6]

Test II :: [0.05 -0.3 0.75 0 0.75 -0.3 0.05]

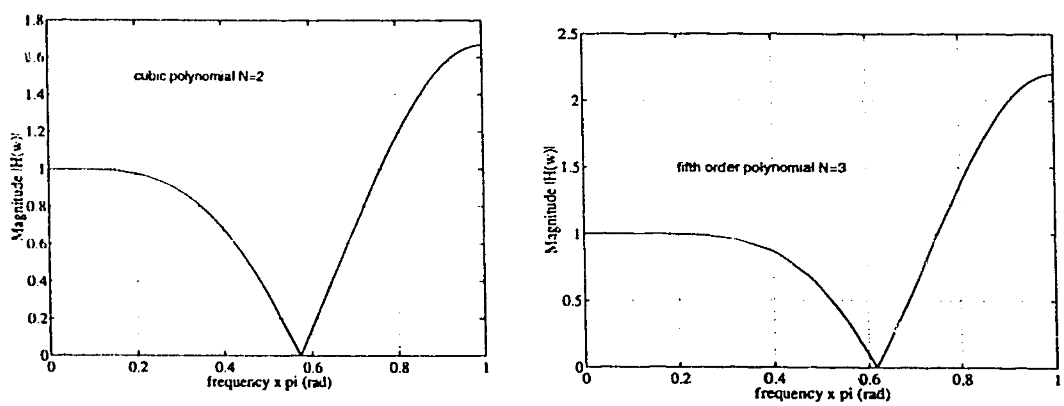
The frequency responses of these filters are shown in Fig. 4.23a and b. The improvement factor for this case is found to be only about 2.4 for f_d=20Hz, as shown in Table 4.10.

TABLE 4.10
TEST RESULTS BASED ON POLYNOMIAL FIT.

Test	D1 errors	D2 errors	Run length	Improvement factor (IF)
I	374	155	69478	2.419
II	321	145	61292	2.213



(a)



(b)

Fig. 4.23. Estimation filter design based on polynomial fit. (a) Filter response with h(0)≠0. (b) Filter response with h(0)=0.

The spectrum of the various filters used in the above tests, together with the results can be used as a guide to improving the performance. The polynomial fit and least square fit have relatively high (>220Hz) cut-off frequencies and high stop-band ripples. However the passband is very flat. For the least square fit both the cut-off frequency and the stopband ripple are reduced when the number of points (N) is increased from 3 to 4. For the raised cosine filter (with h(0) = 0), the cut-off frequency depends on the design, but the passband has some noticeable ripple. Based on these tests, it is concluded that a filter with a very flat passband and low ripple in the stop band, with a tap length range of 20 to 30, and with an approximate cut-off of 4 times f_d may be an appropriate design.

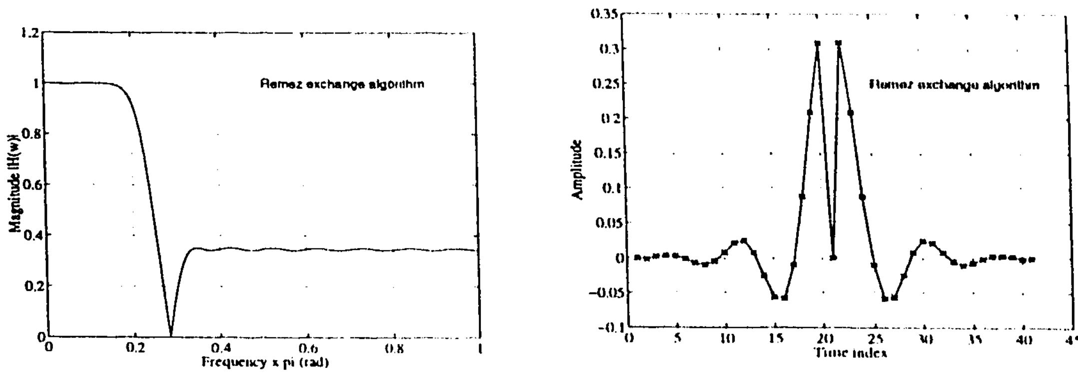


Fig. 4.24. Estimation filter design based on Remez exchange algorithm.

Therefore, a filter design based on the *Remez Exchange Algorithm* [31] is tested as an FIR filter structure with the flexibility that the passband can be made very flat by weighting the passband ripple by a large factor as compared to the stopband. The parameters chosen are:

fr=[0 0.1333 0.3333 1],

mr=[1 1 0 0],

wr=[500 1],

br=remez(40,fr,mr,wr),

where fr defines the vector of desired frequency response with 1 corresponding to $f_s/2$; mr is the vector containing the desired magnitude response at the frequencies specified in fr; wr is the weighting factor on passband and stopband; and br is the vector containing the tap coefficients returned by the Remez algorithm. The number "40" is the total number of taps, excluding the estimation point, (i.e. N=20 in this case). The calculated coefficients were modified in order to set the middle point to zero, and to normalize the gain to unity. The magnitude response of this filter is shown in Fig. 4.24. The improvement factor is now found to be >250, as shown in Table 4.11.

TABLE 4.11
TEST RESULTS BASED ON REMEZ EXCHANGE ALGORITHM.

f_d	D1 errors	D2 errors	Run length	Improvement factor (IF)
40 Hz	1508	6	61373	251.3
20 Hz	3767	10	602342	376.7

After making the error correction, **Up** and **Vp** in the estimation filter was changed so as to remove the incorrect **Up** and **Vp** from the right hand part of the filter, as discussed in Chapter 3. This software module when tested, shows the improvement factor increases from >250 to >275, as shown in Table 4.12.

TABLE 4.12
TEST RESULTS BASED ON REMEZ EXCHANGE ALGORITHM
(WITH Up_{new} AND Vp_{new}).

f_d	D1 errors	D2 errors	Run length	Improvement factor (IF)
40 Hz	40904	127	6523048	322.078
20 Hz	60169	215	9652384	279.856

The estimated channel values, **Upf and Vpf,** are shown in Fig. 4.25.

When the noise is added at the receiver, the estimation is expected to deteriorate because a smooth curve is difficult to fit through the sample points, thereby degrading the improvement factor. The results for different values of SNR are presented in Chapter 5.

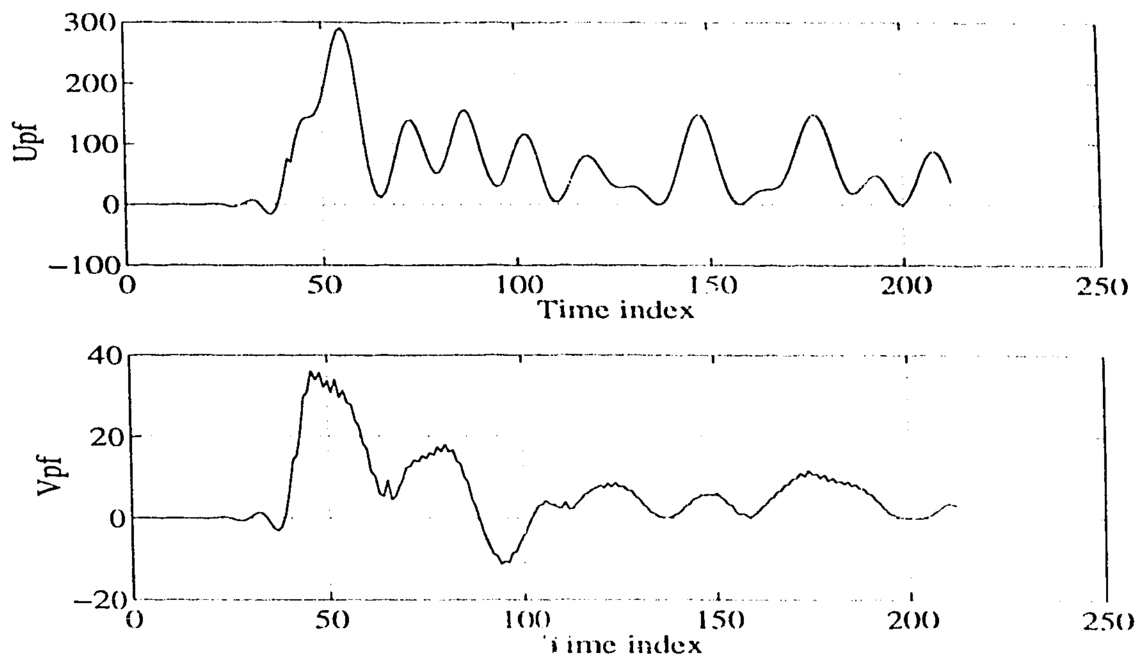


Fig. 4.25. Estimated channel (Upf and Vpf)

4.5 Error Measurement

The measurements of the symbol error rate (SER) and the signal to noise ratio (SNR) are used to find the performance of the error correction algorithm. The SNR measurement is discussed in Section 4.4.1. Two software SER testing modules are used, one for the differential detector output (first decision) and the other for the decision-aided detector output (second decision). The simulation is run long enough to obtain an accurate estimation of the error rate. Typically, at least 100 errors are counted to produce a 95% confidence interval, i.e. the error rate is measured within 25% of the actual BER 95% of the time [35]. The implementation of the SER tester requires two modules, as discussed in the following subsections.

4.5.1 Duplicate Data Source with Delay and Storage

To determine the SER, storage of the transmitted data information is required in order to compare it with the first and second decisions, because of the delay that exists between these decisions. The amount of delay between the transmitted data and the first decision depends on the lengths of the transmitting and receiving filters. Further, to make the second decision, additional delay exists due to the estimation filter. To store the data information, a circular buffer implementation is used. The length of this buffer depends on the delay; new data entering this buffer overwrites old data that has already been used by the comparator for the SER measurement. For implementation, since the transmitter is implemented in the C50 and the SER tester is in the C30, an additional duplicate data source was used in the C30.

4.5.2 Comparators

To measure the error rate, a simple comparison operation for equality is used between the detected data and the transmitted data. A software counter is incremented for each inequality. The total number of transmitted symbols is also counted. The overall SER for both D1 and D2 are determined using these two counter outputs.

4.6 Summary

This chapter has presented the DSP implementation of the transmitter using $\pi/4$DQPSK modulation, the flat-fading channel, and the receiver with differential and decision-aided detection. The design and characterization of the filter responses, using Matlab© simulation, have also been presented. The performance of the implemented error correction algorithm will be discussed in the next chapter.

5. RESULTS

This chapter presents the test results for the performance of the decision-aided detector. First, the sampling rate, Doppler shift parameters, and fading channel spectrum etc., are described in Section 5.1. This is followed by the performance results for the decision-aided error correction algorithm for both an AWGN channel and a Rayleigh flat fading channel. The last section presents some preliminary results for the implementation and performance of Scheme II.

5.1 Sampling Rate, Doppler Shift and Related Parameters

The basic structure of the test setup for the detector, error correction algorithm, and BER measurement is shown in Fig. 5.1. Since the transmitter and the fading channel are both implemented in the same DSP, the rate at which data can be generated is affected by the instruction cycles required for the generation of the fading sample. The 150 instruction cycles for data and 350 instruction cycles for the fading sample requires 25μsec., limiting the maximum sample rate to 40 ksamples/sec for the C50, which has an instruction cycle time of 50ns. Since the exact length of the various filters and the software implementation structure were not known initially, and to enable a flexible software architecture, the sample generation rate for the testing was limited to 9.6ksamples/sec; with an oversampling factor of 8 the data symbol rate is thus chosen to be 1.2ks/sec.

The error rate due to fading depends on the ratio of the Doppler shift frequency and to the symbol rate. For a mobile unit moving at 100km/hr and for a 900 MHz carrier frequency, the Doppler frequency will be 83.3 Hz. Therefore, for simulation of the IS54 system, which has a symbol rate of 24.3ks/sec and carrier frequency of 900 MHz, the ratio of Doppler shift to symbol rate needs to be $\cong 0.003$.

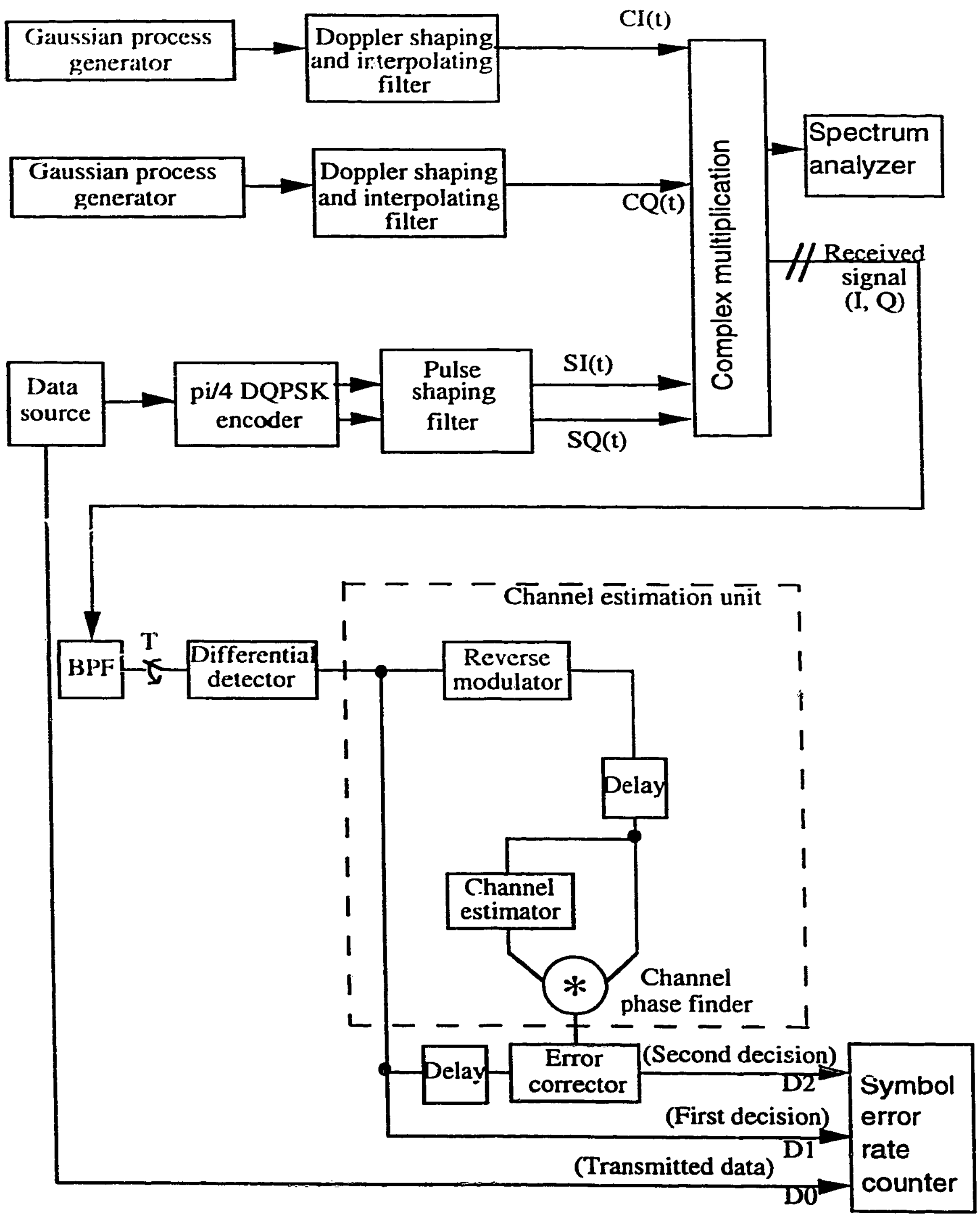


Fig. 5.1. Test setup for decision-aided detector.

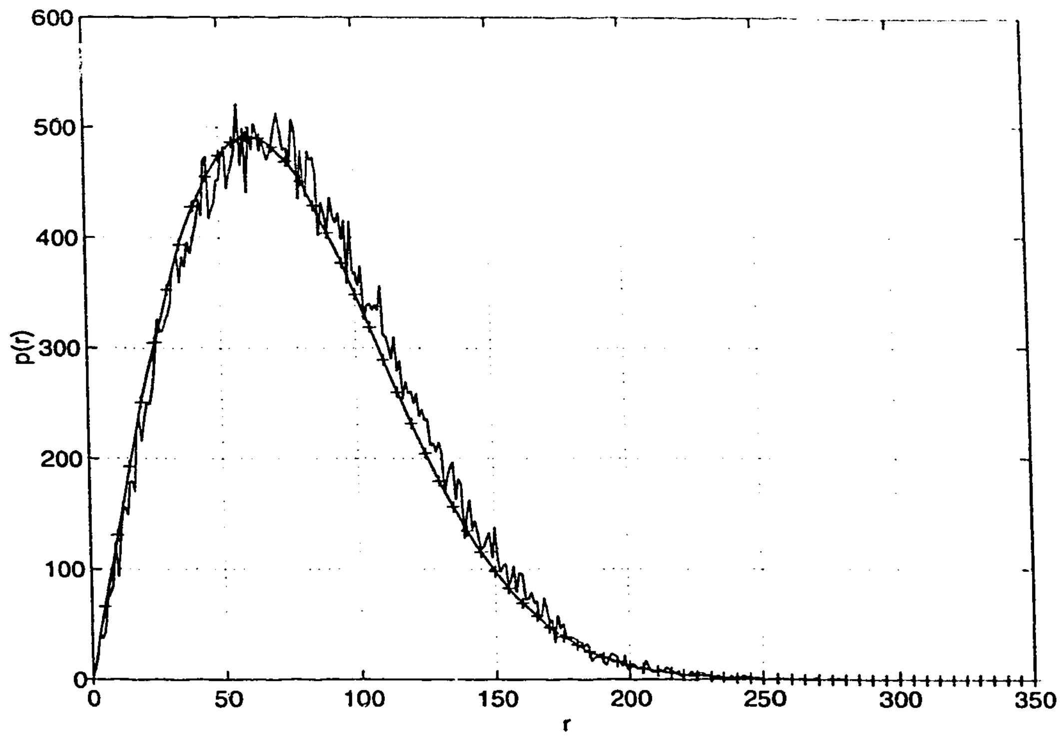


Fig. 5.2. PDF of envelope of the fading channel (from theory and simulation).

For a symbol rate of 1.2ks/s, this would require a Doppler frequency $f_d \cong 4$Hz. Generating a slowly fading sample (as compared to the data rate) is limited by the implemented test-setup, since simulation of slow fading requires very long FIR filters. Therefore, the algorithm is tested with relatively faster fading, $f_dT=0.016$ and 0.033, corresponding to 20Hz and 40Hz Doppler shift values respectively; the filters for these Doppler shifts are not too long (up to 350 tap coefficients). These f_d rates are used to test the performance of the error correction algorithm; apart from their practicability they have the added advantage that the primary D1 error rate is higher, so that the test run-times do not have to be as long. The error correction process, in principle, does not depend on the fading rate and symbol period. The implementation of the error correction algorithm requires $\cong$ 167 instructions/symbol; thus for the TMS320C30 DSP at

60ns/instruction, only 10μs/symbol are needed. This would allow the error correction method to be implemented for symbol rates up to 100ks/sec.

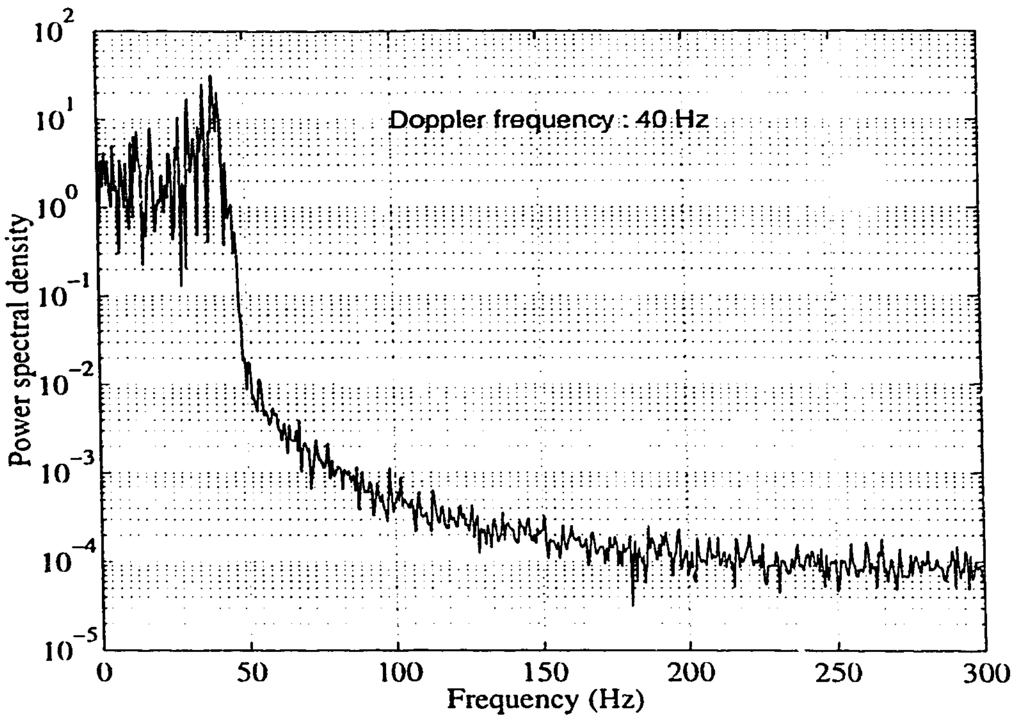


Fig. 5.3. Relative power spectrum of fading signal (Doppler frequency =40Hz).

Two important parameters for the simulation of the fading channel are the probability density function (pdf) and the frequency content. If the simulation results are to be valid, these two parameters should closely follow the theoretical and measured quantities. For the simulated channel in this work, the pdf is shown in Fig. 5.2 and the power spectrum for Doppler shift frequencies (f_d) equal to 40Hz and 20Hz is shown in Fig. 5.3 and Fig. 5.4 respectively. The total time period of observation in the DSP is approximately 10min. (i.e. @600 symbols/sec number of symbols =360,000). The spectrum observed using the spectrum analyzer closely follows the expected Doppler spectrum obtained with an isotropic antenna, with a peak at f_d and with very little power at higher frequencies (Refer Eq. (2-12)). The waveforms shown in the figures are obtained using Matlab©, because of the difficulty of obtaining a printout from the spectrum analyzer.

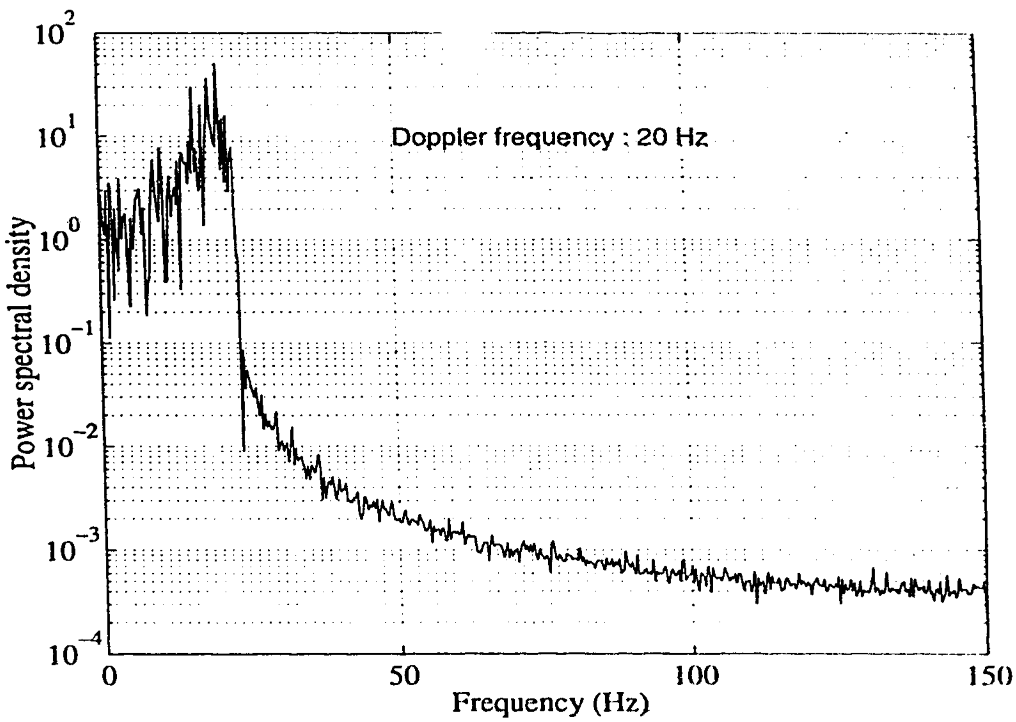


Fig. 5.4. Relative power spectrum of fading signal (Doppler frequency =20Hz).

5.2 Performance of Decision-aided Detector

The symbol error rate for both differential detection and decision-aided detection are obtained, and then compared to demonstrate the performance improvement quantitatively. As described earlier, the symbol error rate is measured for different signal to noise ratios.

5.2.1 AWGN Channel

Fig. 5.5 shows the system performance for both D1 and D2 in an AWGN channel. The performance for D1 (the differential detector) is good (BER<10^{-5}) for high values of SNR, but degrades at low SNR as the effect of the noise becomes significant; for this case there is no observable error floor. For a target

BER of 10^{-3}, the SNR must be about 12dB or greater. As can be seen, the error rate for the decision-aided detector (D2) is no better than the differential detector output (D1); in fact, there is a 2dB degradation in performance. This is because, for an AWGN channel, when the channel estimation filter in the decision-aided detector tries to estimate the noisy channel, it fails because the noise is inherently random. In fact, the channel estimation filter smoothes out the noise thus giving a very poor estimate for each individual point. Thus, for an AWGN channel, the first decision at the output of differential detector, is the better choice. However, the primary cause of error in cellular communication is the effect of the fading channel. Hence, the performance of the decision-aided error correction algorithm under Rayleigh fading conditions is more significant. This is presented next.

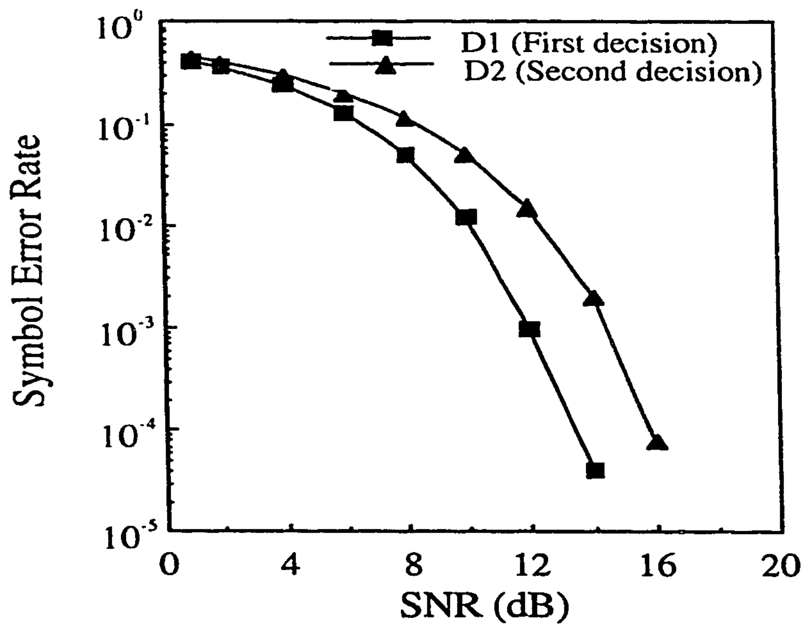


Fig. 5.5. SER performance for π/4DQPSK in an AWGN channel.

5.2.2 Fading Channel and AWGN, f_d=40Hz

Fig. 5.6 shows the error rates D1 and D2 for a channel with fast Rayleigh flat-fading and AWGN. The result at the output of the differential detector (D1) shows the significant degradation in system performance that occurs due to the phase shifts caused by fast fading. For all values of SNR, the system performance is considerably worse than the result for AWGN only. More importantly, the error performance shows an irreducible error floor; that is, the SER falls as the SNR is increased but then, above a certain SNR value, the SER becomes constant. The signal power can be increased, but there is no improvement in performance. This is due to the random FM effect of the fading channel. The actual error floor value depends on the value of the Doppler frequency. For f_dT=0.0333, (f_d=40Hz) the error floor occurs at a SNR of 30dB, with a symbol error rate of 2.4×10^{-2}. Thus, due to random FM, the system can not meet a target BER of 10^{-3}, no matter how high the signal power. These results show clearly that fading severely affects the performance of the π/4DQPSK modulation system. A similar result would be obtained for other phase shift modulation schemes.

The error rate (D2) at the output of the decision-aided detector, for the Rayleigh flat fading channel with f_dT=0.033, shows that the error rate falls to below 10^{-4} before the onset of the error floor. Fig. 5.6 clearly shows the improvement in error rate that has been achieved by the error correction scheme, provided that the SNR is sufficiently high. The error floor is observed to be reduced from 2.4×10^{-2} to 7.6×10^{-5}, a reduction in error rate of about 300.

5.2.3 Fading Channel and AWGN, f_d=20Hz

Fig. 5.7 shows the system performance in a fast Rayleigh flat-fading channel, for f_dT=0.0166 (f_d=20Hz). The results are similar to those for f_d=40Hz. The error rate floor is reduced from 5.4×10^{-3} to 1.5×10^{-5}, which is again an improvement of about 300 in the SER, after execution of the error correction algorithm. As for the previous case (f_d=40Hz), a SNR of 60 dB is required to obtain the greatest error improvement.

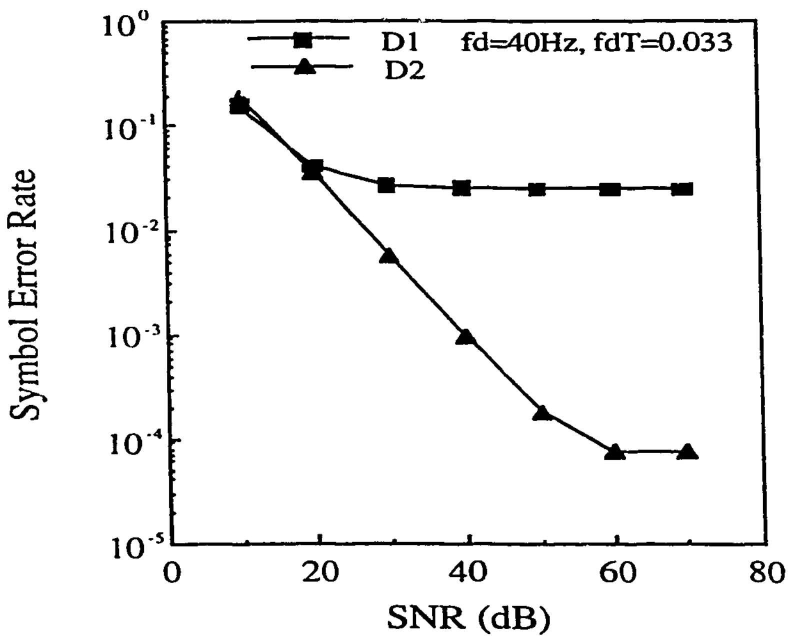


Fig. 5.6. SER performance of decision-aided detector for flat fading channel (f_d=40Hz).

These performance curves (Fig. 5.6 and Fig. 5.7) also show that, at the lower SNR values, when the noise power dominates, the error correction algorithm fails to show any improvement. The main reason for this poor performance is the random characteristic of the noise, which cannot be estimated by the channel estimation filter (same behavior as for the case of a pure AWGN channel). An added factor is the noise multiplication in the differential detector, which causes a fairly rapid deterioration in the quality of the recovered channel at low SNR values. Thus, the error correction works only when the D1 errors are due to the random slow channel phase modulation, not when the errors are due to Gaussian noise.

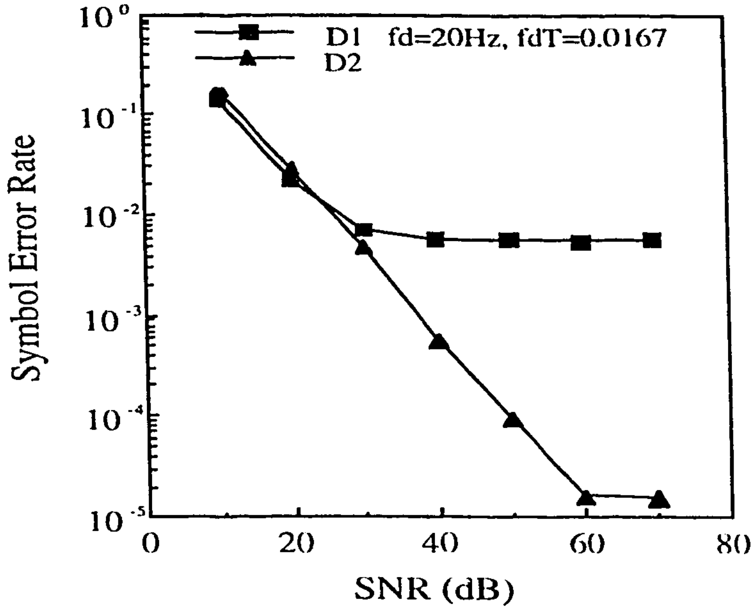


Fig. 5.7. SER performance of decision-aided detector for flat fading channel (f_d=20Hz).

5.3 Scheme II

As discussed in Chapter 3, the extracted channel waveform shows clear discontinuities at the points where the first decisions are incorrect. One method of locating and removing these discontinuities, and hence also correcting the errors is to calculate the derivatives of the extracted channel waveforms (Ich and Qch). Fig. 5.8a shows a typical recovered channel waveform in the vicinity of an error. The channel is smooth on either side of the error and hence the derivatives do not change very quickly in these regions. There is an abrupt discontinuity however at the point of error. In Fig. 5.8a, the most obvious discontinuity is in the slope, but the higher derivatives are also affected. Calculating the derivatives of the waveform (both I and Q) therefore provides one means of locating the error.

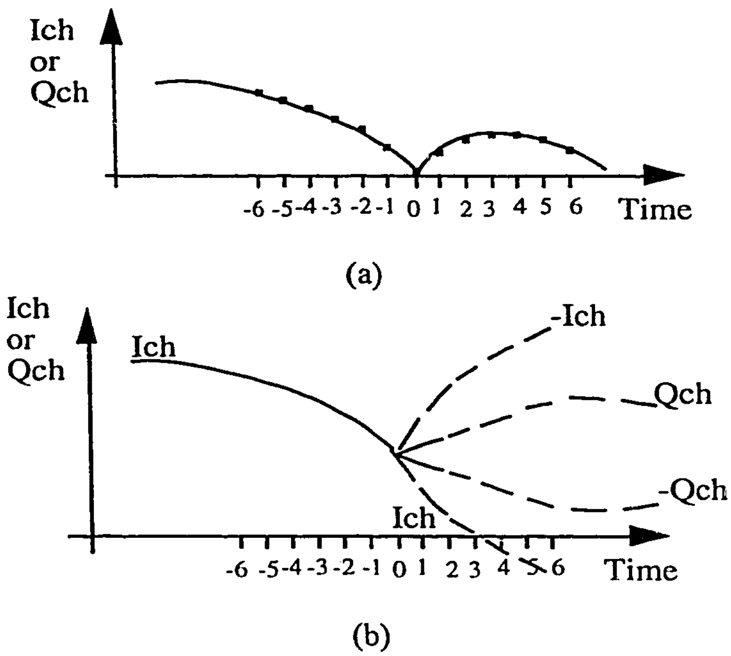


Fig. 5.8. Scheme II channel smoothing. (a) Recovered channel. (b) Actual channel possibilities.

It has been shown earlier that, whenever an error occurs, the channel waveform (Ich, Qch) suffers a phase shift that is a multiple of 90°. Thus, as the channel is reconstructed from the D1 decisions, there are, at each point, four possibilities; the channel may be recovered correctly, but there are three possibilities for making an error that involve the interchange of I and Q, and/or a change of sign. These four possibilities are illustrated in Fig. 5.8b. For correction of the errors the derivatives for the past channel values (the left side points in Fig. 5.8b) must be compared with the four possibilities for the derivatives based on the future (right side) values.

In this work, the derivatives have been calculated based on a least square cubic polynomial fit through 7 points. The derivatives at point 0 in Fig. 5.8 have been calculated using a cubic polynomial fitted to the points -6 to 0, giving a "forward prediction" of the derivatives, and also the point 0 to +6 giving a "backward prediction". The numerical values of the derivatives can be estimated using FIR filters with the coefficient values:

af_coeff = [1/36 -1/36 -1/36 0 1/36 1/36 -1/36]
bf_coeff = [-23/84 1/3 25/84 -4/84 -31/84 -1/3 33/84]
cf_coeff = [97/126 -269/252 -103/126 8/21 25/18 269/252 -31/18]
df_coeff = [-4/7 6/7 4/7 -3/7 -8/7 -4/7 16/7]
ar_coeff = [0 0 0 0 0 0 0 0 -1/36 1/36 1/36 0 -1/36 -1/36 1/36]
br_coeff = [0 0 0 0 0 0 0 0 33/84 -1/3 -31/84 -4/84 25/84 1/3 -23/84]
cr_coeff = [0 0 0 0 0 0 0 0 -31/18 269/252 25/18 8/21 -103/126 -269/252 97/126]
dr_coeff = [0 0 0 0 0 0 0 0 16/7 -4/7 -8/7 -3/7 4/7 6/7 -4/7]

where "af" gives the forward third derivative, "bf" gives the forward second derivative, etc., and "ar, br, cr, dr" are the reverse estimates.

For the recovered channel waveform of Fig. 3.9, the values of the forward and reverse third derivatives, "af" and "ar", are shown in Fig. 5.9. To combine the derivative information into one parameter, the difference of these two values can be taken and squared. Thus for Ich

$$a_I=(af-ar)^2 \tag{5-1}$$

and for Qch,

$$a_Q=(af-ar)^2 \quad . \tag{5-2}$$

Using a_I and a_Q, the error zones show up very distinctly as shown in Fig. 5.10. To combine the I and Q channel results, these values may be added and squared. Thus, the final test parameter for location of the errors is

$$Err\_ch= (a_I+a_Q)^2 \quad . \tag{5-3}$$

This is plotted in Fig. 5.11. As can be seen, the value of "Err_ch" is normally very small but has distinct peaks whenever the channel is in error.

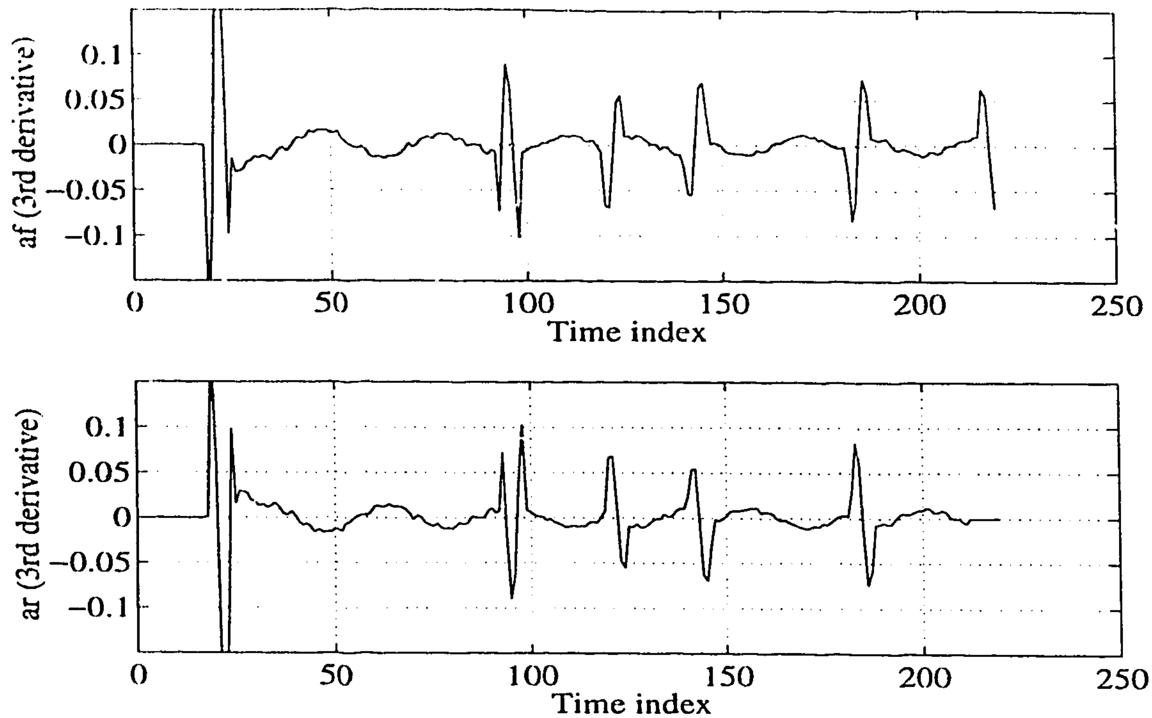


Fig. 5.9. Error zone detection using af, ar.

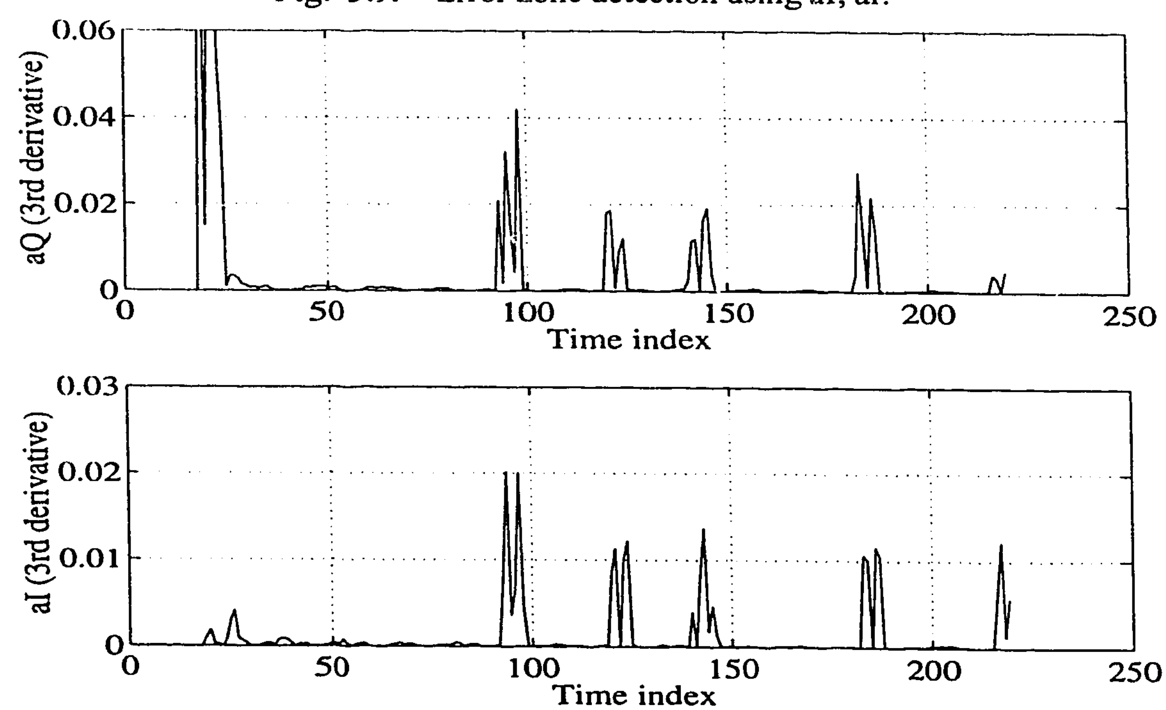


Fig. 5.10. Error zone detection using a_I, a_Q

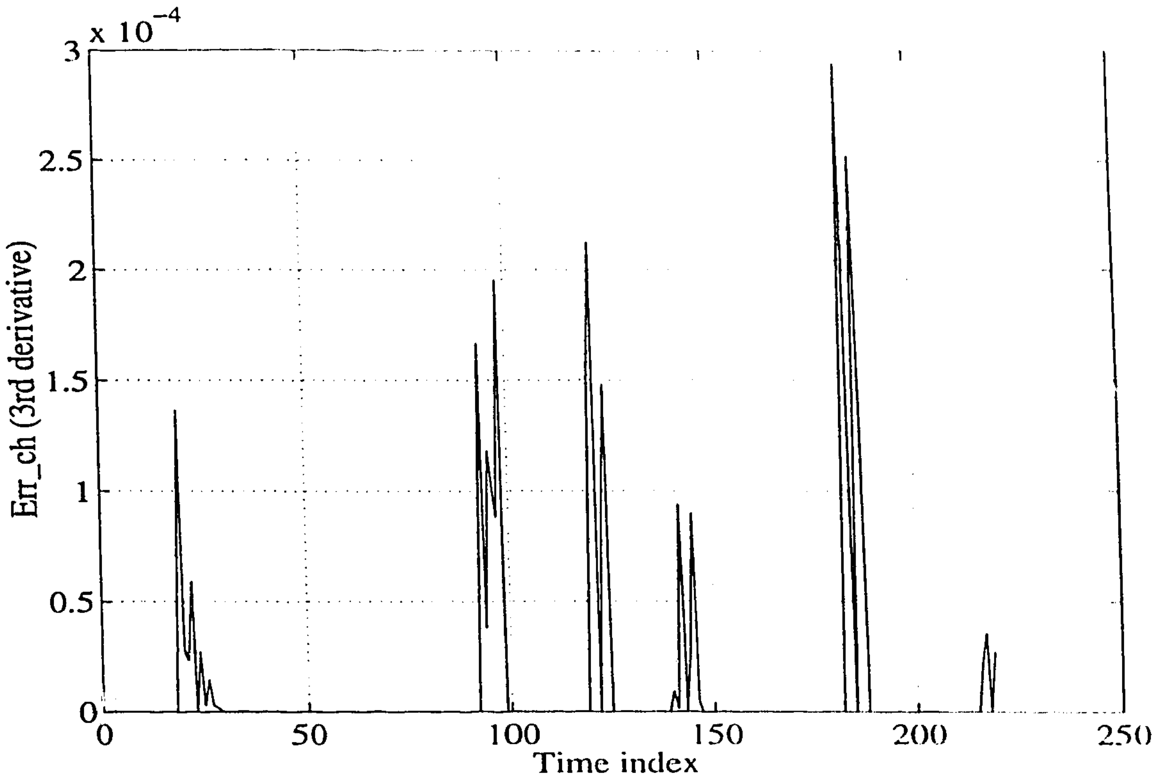


Fig. 5.11. Error zone detection using Err_ch.

The errors for the test shown in Fig. 5.9 are located at symbol points [87, 113, 114, 134, 135, 176]. The detected error zones, using the third derivatives "af and ar", appears as pronounced peaks. Comparing the value of "Err_ch" with a fixed threshold value, the error zones are determined as [85 to 90, 112 to 116, 134 to 137, and 175 to 179]. Various tests show that although the error point is in the error zone, its exact position cannot be determined. Because of this every point within the error zone must be considered as a possible error location.

The simplest approach to smoothing the channel, and hence correcting the errors, is to consider the points within the error zone, sequentially one point at a time. The four possible channel (Ich, Qch) interchanges corresponding to no shift, 90°, 180° and 270° shift are tested at each candidate point. A choice is made, based upon the derivatives, corresponding to the minimum discontinuity in the new channel. Based on this new channel, a corresponding new decision is made for the data. Fig. 5.12 and 5.13 shows the straightening of the channel and the consequent elimination of peaks from the smoothed channel.

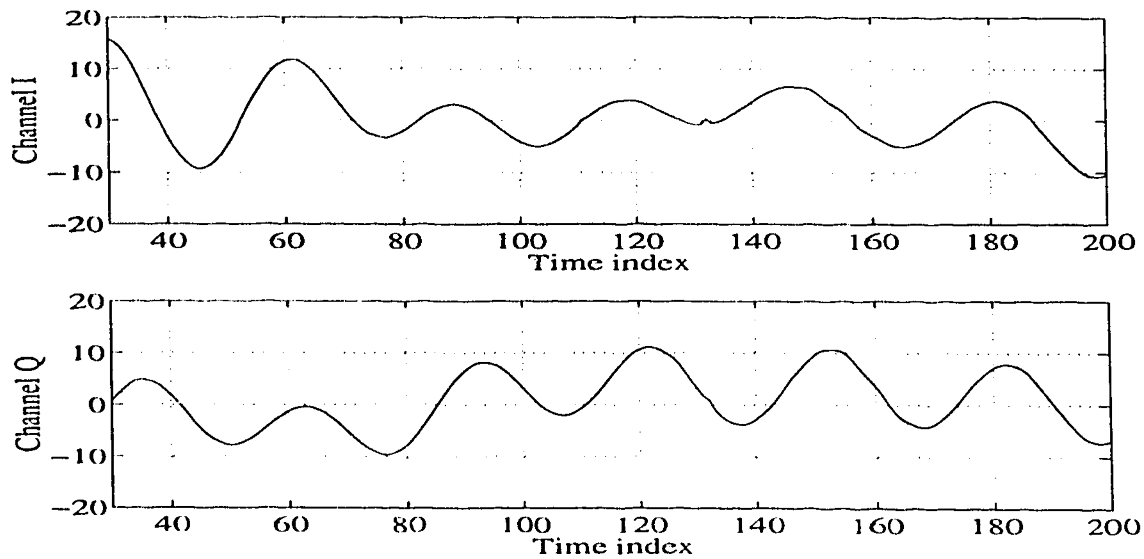


Fig. 5.12. Smooth channel using derivatives.

All the D1 errors have been removed except the double errors at 134, 135. Since some errors can be corrected, it appears that smoothing of the recovered channel may be a viable approach to the error correction problem. However, some problems encountered by this approach are:

- In testing the derivatives and choosing the minimum possible channel discontinuity, a change to the recovered channel may be indicated even though the actual discontinuity is at a neighbouring point. This results in the introduction of an additional error, rather than the removal of the actual error.
- Detection of the error peaks is sensitive to the range of points and to the chosen threshold value.

- Due to noise and ISI, the recovered channel is not always smooth, resulting in further deterioration as the choice of the new I and Q channel values becomes more prone to error.

Thus, it appears that a point-by-point approach may not be satisfactory. Other algorithms, more complex than that used in scheme I, may need to be investigated.

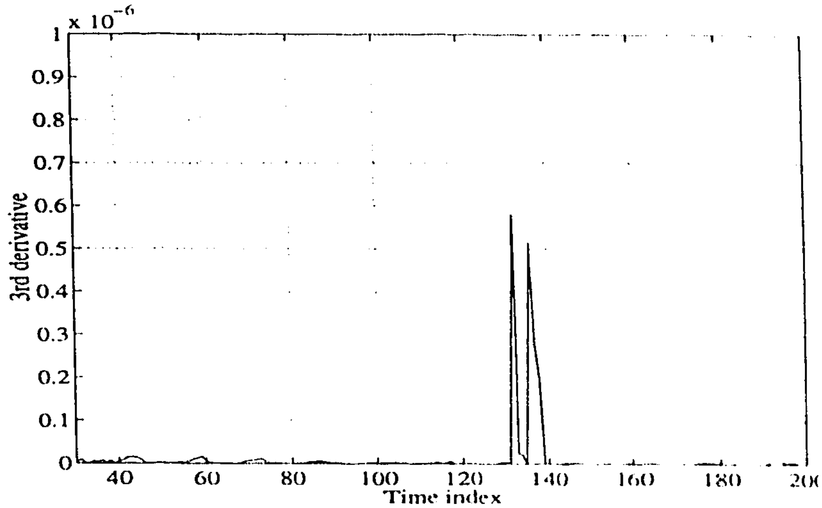


Fig. 5.13. Scheme II smooth channel peak detection.

5.4 Summary

The potential of a decision-aided error correction algorithm to improve the performance of cellular systems based on π/4DQPSK modulation is demonstrated by tests carried out in real time using digital signal processors. A simple method of error correction (Scheme I) works well when the SNR value is high. It lowers the error floor, obtained at the output of conventional differential detector, by a factor of more than 200. The proposed scheme II, which is intended to give better results for the lower range of SNR values, has been shown to be useful for detecting the presence of errors, but more investigation is needed to determine a suitable error correction algorithm.

6. SUMMARY AND CONCLUSIONS

A phase modulated signal, such as π/4DQPSK, is severely distorted by the short term fading arising from multipath propagation in the radio channel. The fading causes large variations in the amplitude of the received signal, but more importantly causes random phase modulation. These random phase variations appear as noise to the detector, resulting in a performance degradation for the conventional differential detector. The detection error can be reduced to a certain extent by increasing the signal power; however, beyond a certain level any further increase in the signal power fails to reduce the error rate, so that the BER versus SNR characteristic exhibits an irreducible error floor. This necessitates a method to mitigate the effect of the fading and random FM so as to reduce the error rate at the output of the detector.

Diversity reception, signal coding and pilot symbol insertion are known to be effective techniques for combating the undesired effect of fast random fades and the resulting random phase modulation. These methods, however, suffer from one or more drawbacks such as a bandwidth penalty, implementation complexity, and/or a power overhead. In this project, a decision-aided error correction algorithm for a π/4DQPSK modulated signal, based on the information data alone, has been investigated and implemented. This method achieves simple and practicable detection using a DSP, which requires only a small amount of memory, little computational power, and does not require any change in the signal format at the transmitting end. The latter is an attractive feature for systems with an already standardized transmitting format. The decision-aided detector consists of a reverse modulation unit, a channel estimation unit and an error correction unit, in addition to the normal differential detector. The complete error correction algorithm has been implemented in a TMS320C30 DSP, using an IBM-compatible PC-based assembler and debugger.

In order to evaluate the performance of the decision-aided detector, a simplified structure for the transmitter and fading channel has been implemented using a TMS320C50 DSP. The complex baseband equivalent of the received signal is obtained digitally by multiplying the transmitted signal and the fading channel characteristic. A custom designed printed circuit board (PCB) with buffer and control circuitry is used to interface the two DSPs for parallel transfer of this signal. An additive white Gaussian noise (AWGN) source at the receiver input has been simulated in the C30, whose power output is software controlled .

The implemented transmitter has three building blocks: the data source, the $\pi/4$DQPSK encoder and the pulse shaping filter. The flat fading channel is also implemented with three components: the Gaussian noise source, the Doppler shift filter and the interpolation filter. A multirate filtering technique is used to have the flexibility to change the Doppler frequency. At the receiver, the bandpass filter and the decision-aided detector are the two major modules. A second data generator and the symbol error counter are also implemented in the C30, to obtain the algorithm performance results. The test results for the transmitter output show that the spectral content is as specified by the IS54 standard. Test results for the envelope of the fading channel signal show that the pdf closely follows the Rayleigh probability density function. The power spectrum characteristics of the fading channel for the Doppler frequencies of 40Hz and 20Hz were tested for accuracy using a spectrum analyzer.

The decision-aided error correction algorithm is tested for symbol error rate for different signal to noise ratios (SNR) and its performance is compared to that of the conventional differential detector. The test results shows that the irreducible error floor at the output of decision-aided detector is reduced by a factor of more than 200 compared to that of the differential detector. The actual irreducible error rate depends on the product of Doppler frequency and symbol period (f_dT). For f_dT=0.033, the error floor is reduced from 2.4×10^{-3} to 7.6×10^{-5}. A second test using f_dT=0.0166, gives a differential detector output symbol rate of 5.4×10^{-3}, which reduces to 1.5×10^{-5} after execution of the decision-aided algorithm.

The test results show that the decision-aided algorithm has good error correction performance; this is achieved without any significant overhead or complexity. The error correction performance depends significantly on the transfer function of the channel estimation filter. Conceptually the design of the filter is based on the fact that the fading is a narrowband process with a bandwidth of two times the maximum Doppler shift frequency. However, various tests using different estimation filters show that the performance is sensitive not only to the nominal cut-off frequency, but also to the filter length (number of taps) and the pass-band and stop-band ripple. The tests found that the major requirements of the filter for improved performance are a very flat passband, a narrow transition band, and an impulse response with tap coefficients that decrease smoothly to zero at the filter ends. The channel estimation process deteriorates, and consequently the error correction performance also deteriorates, at low signal-to-noise ratio values. The test results for the performance evaluation are limited to two f_dT values because both the fading channel and the transmitter model are implemented in a single DSP. Since the fading process is slow compared to transmitted signal rate, a large number of instruction cycles are required to generate one fading sample.

The slower the fading compared to the sampling rate, the longer is the filter length (or number of instruction cycles) required by the Doppler shift and interpolating filters. The implementation of the error correction algorithm requires ≅167 instruction cycles/symbol; i.e., for the TMS320C30 DSP with an instruction cycle time of @60ns, it takes only 10.02μs/symbol. Thus, the algorithm is easy to use, with a maximum possible symbol rate of ≅100ks/s.

The final error tests also show that the error correction algorithm has good performance at high SNR values, but degrades as the noise becomes more dominant. This is a limitation of the algorithm. To improve the performance at low SNR values, Scheme II was investigated. The motivation here was to apply the decision-aided channel recovery directly to the received signal components rather than to the differential detector output, which suffers from the adverse effect of noise multiplication. Implementation of Scheme II has been only partially successful. It has been shown that the error zone can be detected within 4-6 symbols, but the exact position of an error is difficult to determine. For this reason the errors are hard to correct without a computationally intensive and sophisticated algorithm.

A number of ideas for further research to explore decision-aided detection are:

- Since the performance of the decision-aided error correction algorithm could be tested for only a few f_dT values in the present hardware setup, it would be interesting to change the hardware setup so as to use one separate DSP for the fading channel, in order to test the scheme for lower f_dT values. Although lowering the f_dT value from f_d=40Hz to f_d=20Hz produced a lowering of the error floor, smaller f_dT values may not give even better results because, with very slow fading, burst errors may occur during the fades, which may be hard to correct.

- In this thesis, the decision-aided detector has been tested only for a flat fading channel. The performance of the detector should be investigated further for the case of a frequency selective fading channel.
- The second method (as discussed in Chapter 3 and 5) tested in this thesis locates the approximate position of an error, but an efficient error correction algorithm still needs to be determined. This needs to be explored as it may lead to improved performance under low SNR conditions.
- Decision-aided detection could also be tested in combination with other schemes such as diversity and equalization, since this may lead to a further improvement in the BER performance.
- In this work, the channel estimation filter is designed for $f_d = 40$Hz, and then the coefficients are kept constant. An adaptive algorithm should be investigated that will estimate a suitable set of tap coefficients in real time. This may be particularly suitable for systems where only slow fading is encountered.

REFERENCES

[1] D. M. Balston and R. C. V. Macario, *Cellular radio systems*. London: Artech House, 1993.

[2] J. E. Padgett, C. G. Gunther, and T. Hattori, "Overview of wireless personal communications," *IEEE Commun. Mag.*, vol. 33, no. 1, pp. 28-41, Jan. 1995.

[3] D. C. Cox, "Wireless personal communications: What is it?," *IEEE Personal Commun.*, vol. 2, no. 2, pp. 20-35, Apr. 1995.

[4] V. Fung and T. S. Rappaport, "Bit-error simulation of $\pi/4$DQPSK in flat and frequency selective fading mobile radio channels with real time applications," *Proc. IEEE Int. Conf. on Commun.*, Denver, Co, pp. 553-557, June 1991.

[5] EIA/TIA Interim Standard, "Cellular System Dual-Mode Station - Mobile Station-Base Station Compatibility Standard," IS-54, *Electronic Industries Association*, May 1990.

[6] P. S. Mundra, T. L. Singal and R. Kapur, "The choice of a digital modulation scheme in a mobile radio system," *Proc. IEEE Veh. Technol. Conf.*, Secaucus, NJ, pp. 1-4, May 1993.

[7] J. B. Andersen, T. S. Rappaport, and S. Yoshida, "Propagation measurements and models for wireless communications channels," *IEEE Commun. Mag.*, vol. COM-33, no. 1, pp. 42-49, Jan. 1995.

[8] L. W. Couch II, *Digital and analog communication systems*. New York: Macmillan, Second edition, 1987.

[9] S. Stein, "Fading channel issues in system engineering," *IEEE J. Select. Areas Commun.*, vol. SAC-5, no. 2, pp. 68-88, Feb. 1987.

[10] H. Leib, "Data-aided noncoherent demodulation of DPSK," *IEEE Trans. Commun.*, vol. COM-43, no. 2/3/4, pp. 722-725, Feb./Mar./Apr. 1995.

[11] A. Svensson, "Coherent detector based on linear prediction and decision feedback for DQPSK," *Electron. Lett.*, vol. 30, no. 20, pp. 1641-1642, Sep. 1994.

[12] M. Li, A. Bateman, and J. P. McGeehan, "Decision feedback channel estimation - A precursor for adaptive data transmission management," *Proc. IEEE 41st Veh. Technol. Conf.*, pp. 730-734, St. Louis, Mo, May 1991.

[13] Unpublished work by C G. Englefield at *Bristol University*, U.K., 1990

[14] S. Chennakeshu and G. J. Saulnier, "Differential detection of $\pi/4$-shifted DQPSK for digital cellular radio," *Proc. IEEE 41st Veh. Technol. Conf.*, pp. 186-191, St. Louis, Mo, May 1991.

[15] J. D. Parsons, *The mobile radio propagation channel*. New York: John Wiley, 1992.

[16] T. Aulin, "A modified model for the fading signal at a mobile radio channel," *IEEE Trans.*, VT-28, no. 3, pp. 182-203, 1979.

[17] E. A. Lee and D. G. Messerschmitt, *Digital communication*. London: Kluwer Academic, Second edition, 1994.

[18] W. C. Jakes Jr., *Microwave mobile communications*. New York: John Wiley, 1974.

[19] John G. Proakis, *Digital communications*. New York: McGraw-Hill, Third edition, 1995.

[20] William C. Y. Lee, *Mobile communication engineering*. New York: McGraw-Hill, 1982.

[21] K. Feher, "Modems for emerging digital cellular mobile radio system," *IEEE Trans. Veh. Technol.*, vol. VT-40, no. 2, pp. 355-365, May 1991.

[22] C. L. Lui and K. Feher, "Noncoherent detection of $\pi/4$-QPSK systems in a CCI-AWGN combined interference environment," *IEEE Veh. Technol. Conf.*, San Francisco, CA, vol. VT-39, pp. 83-93, 1989.

[23] S. Sampei and T. Sunaga, "Rayleigh fading compensation method for 16QAM in digital land mobile radio channels," *Proc. IEEE Veh. Technol. Conf.*, San Francisco, CA, vol. VT-39, pp. 640-646, May 1989.

[24] J. K. Cavers, "An analysis of pilot symbol assisted modulation for Rayleigh fading channels," *IEEE Trans. Veh. Technol.*, vol. VT-40, no. 4, pp. 686-693, Nov. 1991.

[25] J. K. Cavers, "Performance of tone calibration with frequency offset and imperfect filter," *IEEE Trans. Veh. Technol.*, vol. VT-40, no. 2, pp. 426-434, May 1991.

[26] A. Bateman, "Feedforward transparent tone-in-band: its implementations and applications," *IEEE Trans. Veh. Technol.*, vol. VT-39, no. 3, pp. 235-243, Aug. 1990.

[27] E. R. Berlekamp, R. E. Peile, and S. P. Pope, "The application of error control to communications," *IEEE Commun. Mag.*, vol. 25, no. 4, pp. 44-56, Apr., 1987.

[28] J. D. Parsons, J. H. Henze, P. A. Ratliff, and M. J. Withers, "*Diversity techniques for mobile radio reception,*" *IEEE Trans. Veh. Technol.*, vol. VT-25, no. 3, pp. 75-84, Aug. 1976.

[29] A. Bateman and W. Yates, *Digital Signal Processing Design*. New York: Computer Science Press, 1989.

[30] C. F. Gerald and P. O. Wheatley, *Applied numerical analysis*. New York: Addison-Wesley, Fourth edition, 1989.

[31] A. V. Oppenheim and R. W. Schafer, *Digital signal processing*. New Jersey: Prentice-Hall, 1975.

[32] *TMS320C5x user's guide*, Texas Instruments, 1994.

[33] *TMS320C5x DSP starter kit user's guide*, Texas Instruments, 1994.

[34] V. N. Yarmolik and S. N. Demidenko, *Generation and application of pseudorandom sequences for random testing*. New York: John Wiley, 1988.

[35] M. C. Jeruchim, P. Balaban and K. S. Shanmugan, *Simulation of communication systems*. New York: Plenum Press, 1992.

[36] L. B. Jackson, *Digital filters and signal processing*. London: Kluwer Academic, 1989.

[37] J. A. Mitchell, "Multirate filters alter sampling rates even after you've captured the data," EDN, Aug. 1992.

[38] B. Stroustrup, *The C++ programming language*. New York: Addison Wesley, Second edition, 1993.

[39] R. A. Goubran, H. M. Hafez, and A. U. H. Sheikh, "Implementation of a real-time mobile channel simulator using a DSP chip," *IEEE Trans. Ins. Measure.*, vol. IM-40, no. 4, pp. 709-714, Aug. 1991.

[40] *TMS320 floating point DSP assembly language tools user's guide*, Texas Instruments, 1993.

[41] *TMS320C3x user's guide*, Texas Instruments, 1990.

[42] *Theory, algorithms, and implementation*, vol. 2, Texas Instruments, 1990.

APPENDIX A.

PCB FOR INTERFACE BETWEEN TEXAS INSTRUMENTS TMS320C50 AND TMS320C30 DSPs.

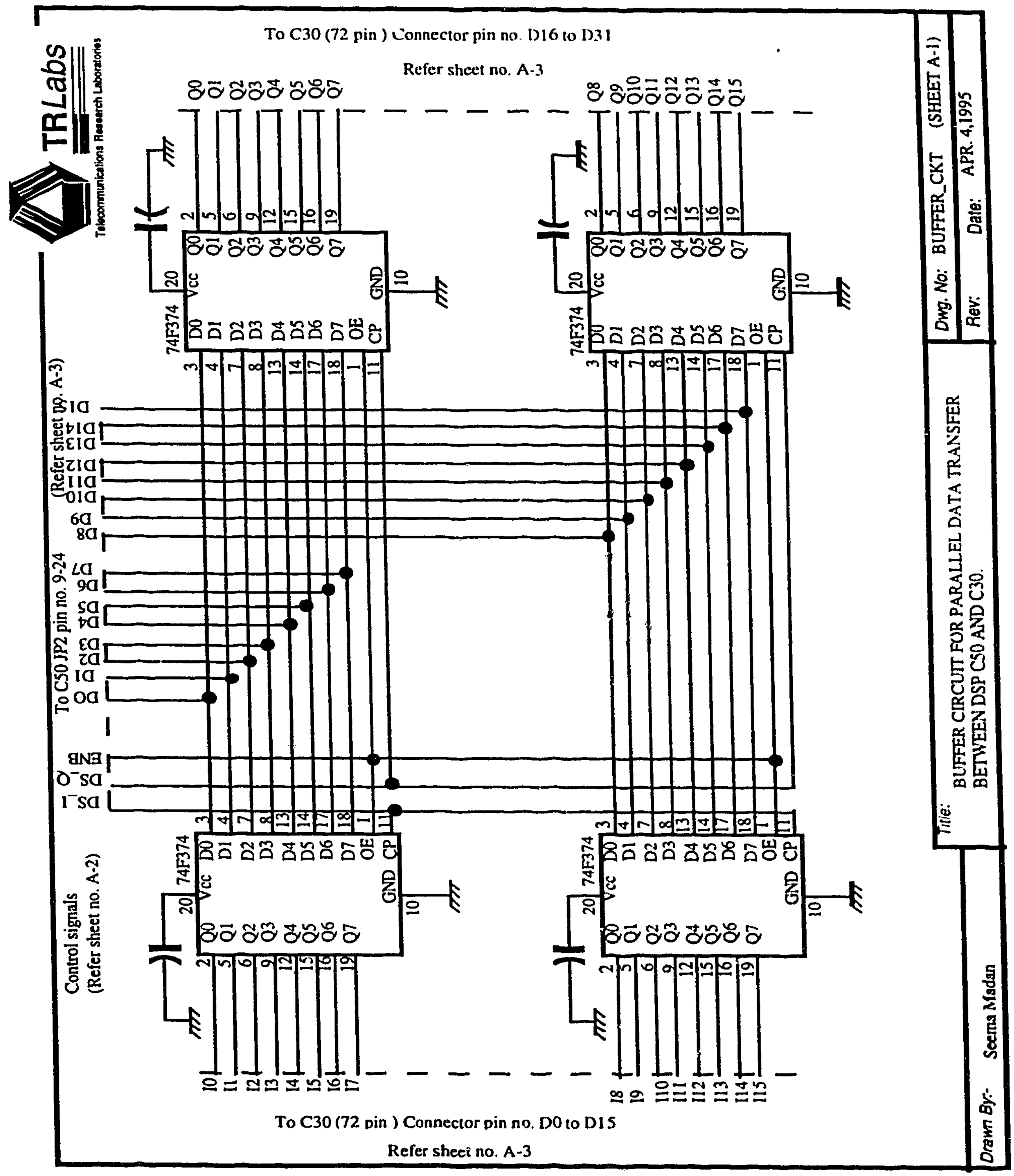


Fig. A.1. Circuit diagram of buffer circuit.

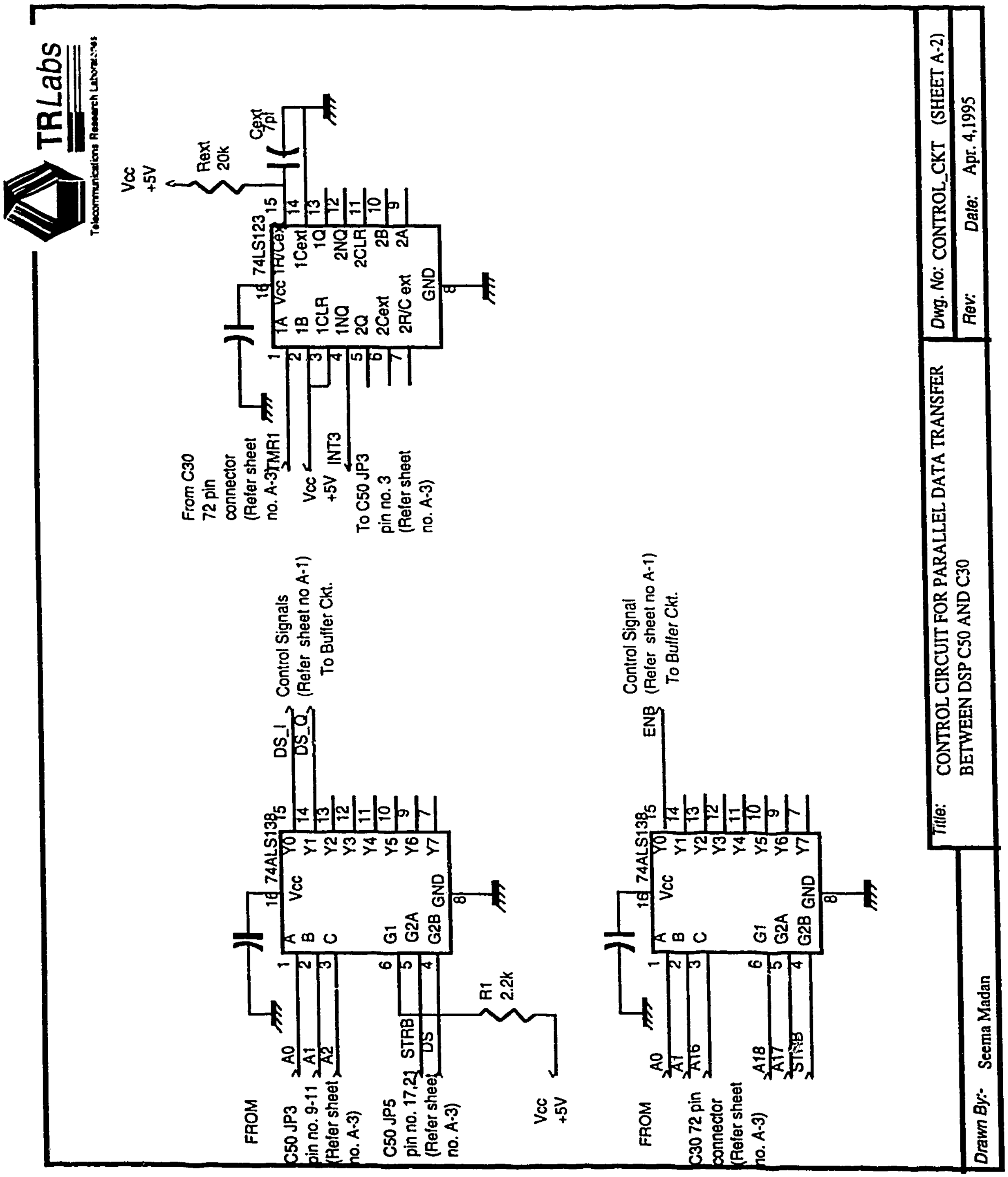


Fig. A.2. Circuit diagram of control circuit.

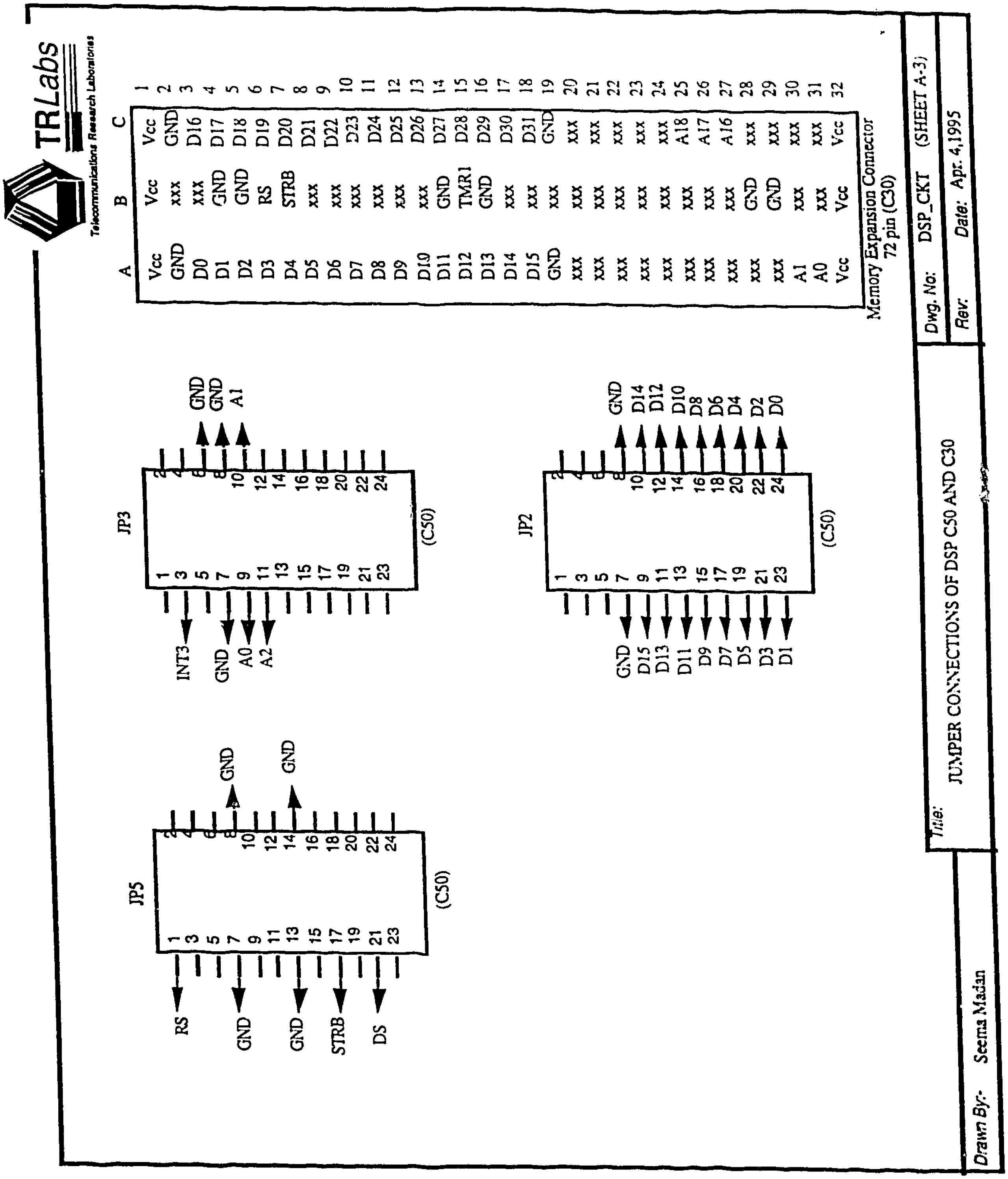


Fig. A.3. Jumper pin connections for C50 and C30 DSP.

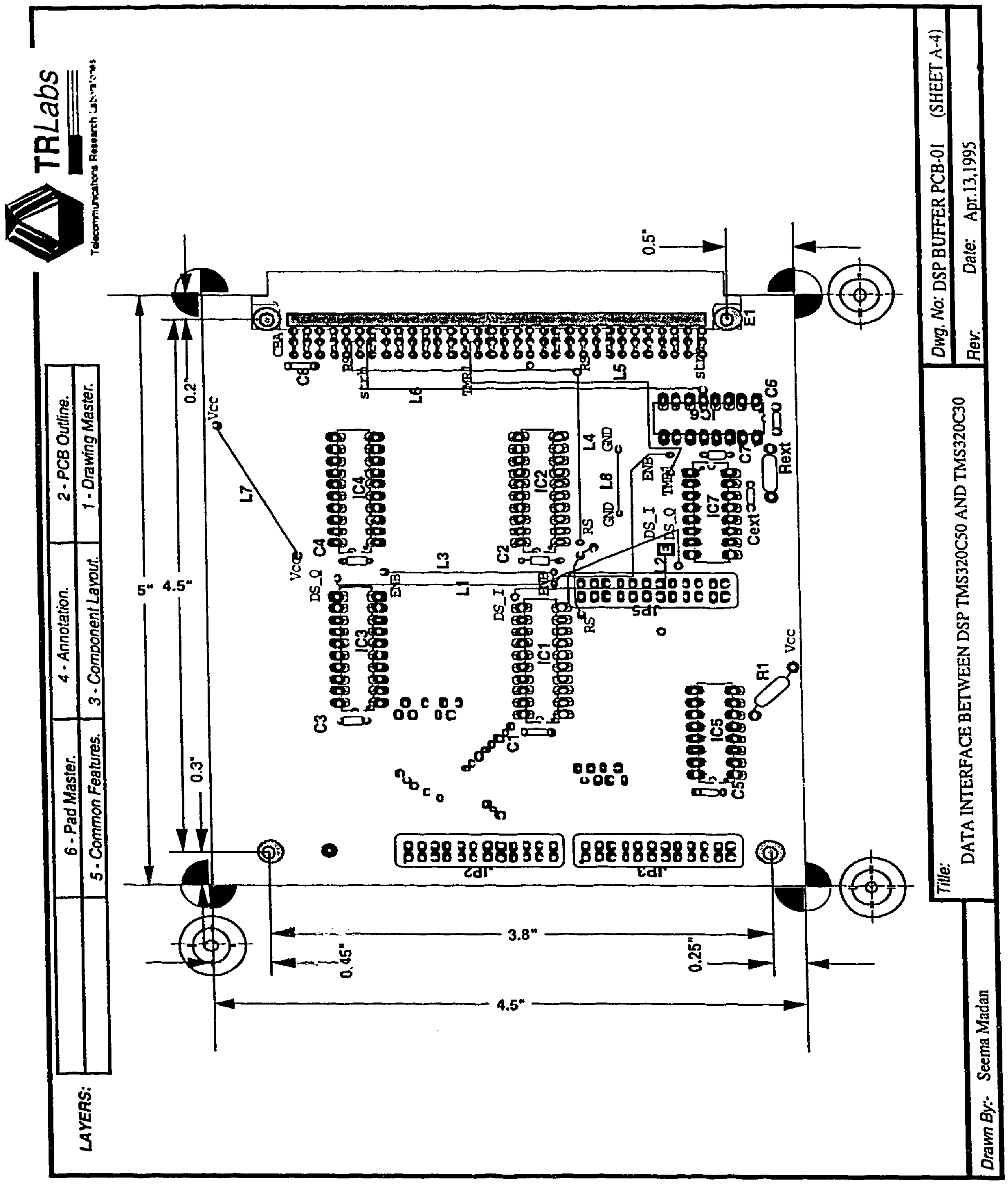


Fig. A.4. Component layout and common features for PCB.

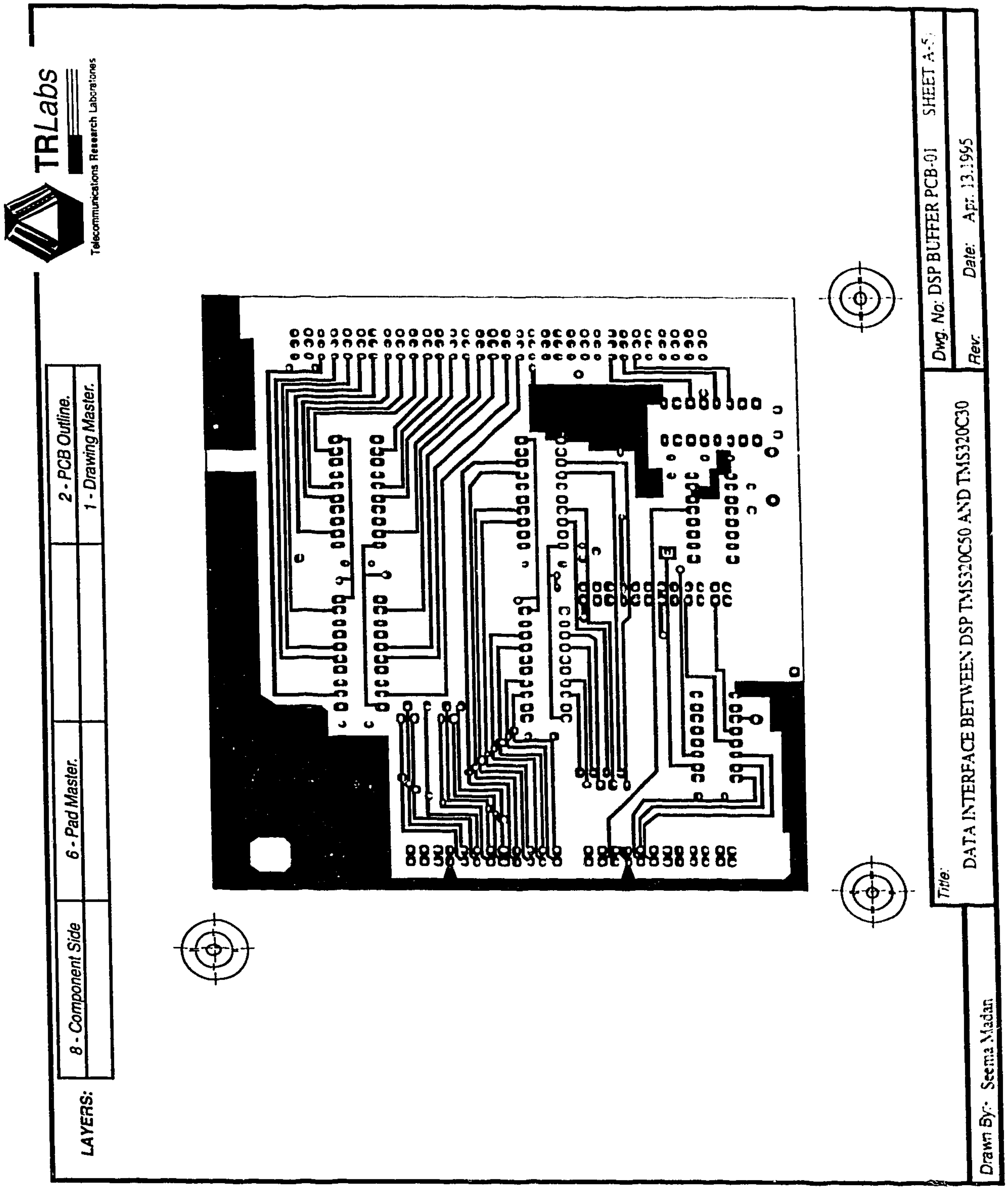


Fig. A.5. Component side for PCB.

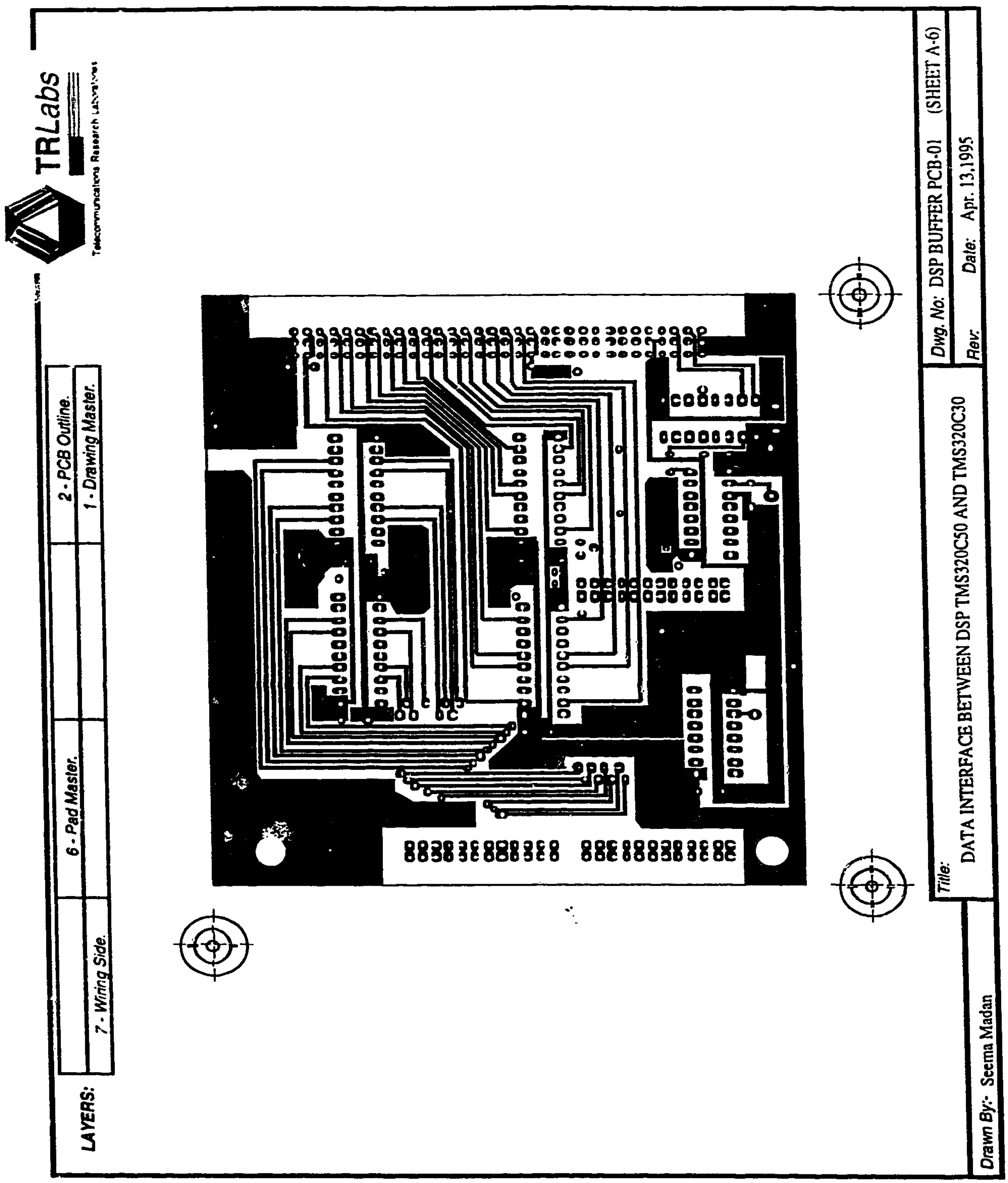


Fig. A.6. Wiring side for PCB.

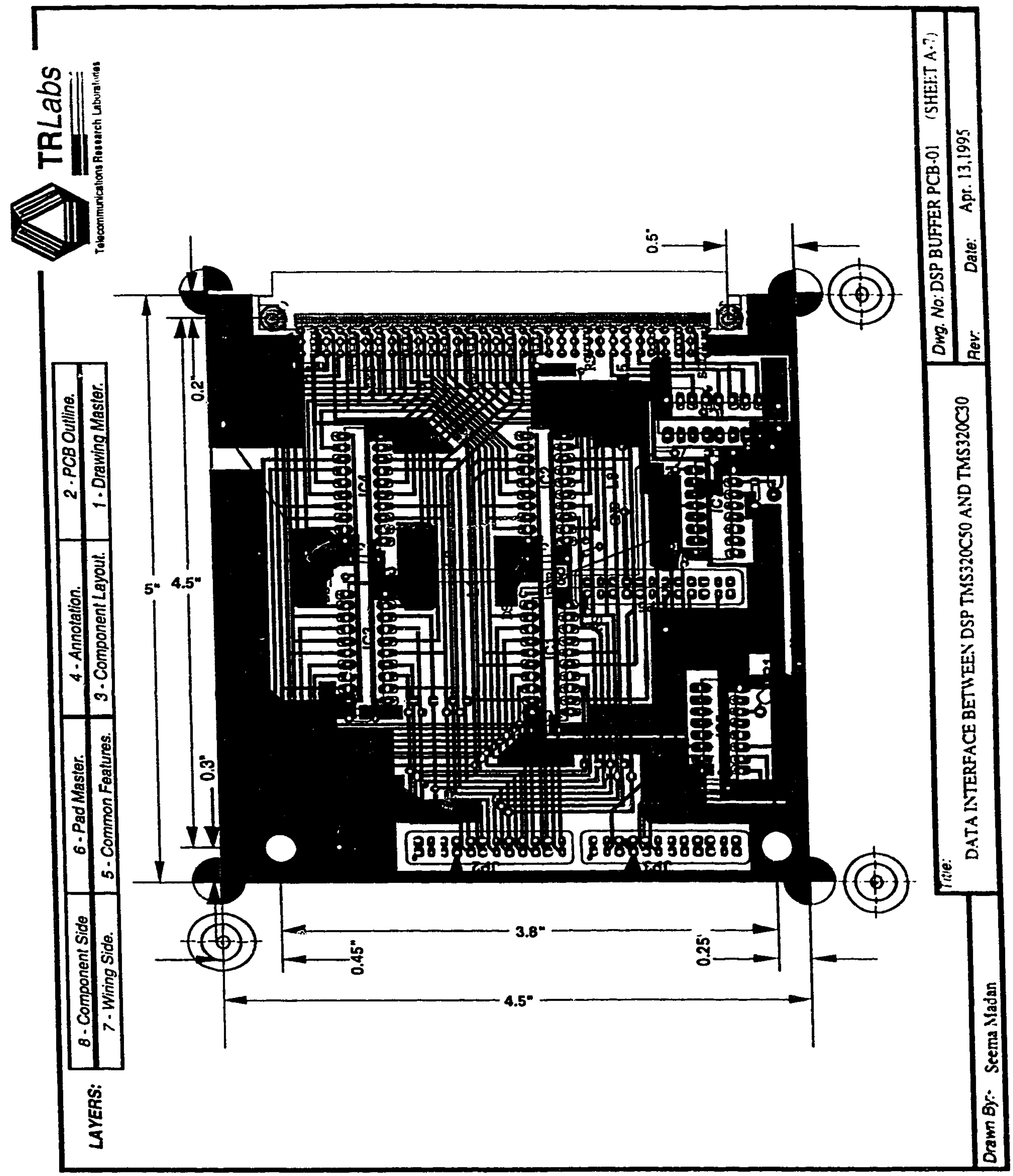


Fig. A.7. Double layer PCB.

APPENDIX B.

DSP PROGRAM LISTS FOR REAL TIME TRANSCEIVER IMPLEMENTATION

```
*PROGRAM : SHIFT7.ASM
*TABLE FOR SHIFT REGISTER SIMULATION
*USED FOR RANDOM GENERATOR FOR PRWS
*Total group = 31 = tap length
*Each group has three words
*First word of group is seed value/random no.
*Second word is address for 30th seed value
*Third word is updated address for 2nd seed value
*Groupings as per:
* bit 0 >> DATA I
* bit 1 >> DATA Q
* bit 2 to bit 8 >> CHANNEL I
* bit 9 to bit 15 >> CHANNEL Q

300h    .word    58129    ;sec
301h    .word    354h
302h    .word    306h
303h    .word    49340    ;seed2
304h    .word    357h
305h    .word    309h
306h    .word    2807     ;seed3
307h    .word    35ah
308h    .word    30ch
309h    .word    44744    ;seed4
310h    .word    300h
311h    .word    30fh
312h    .word    12312    ;seed5
313h    .word    303h
314h    .word    312h
315h    .word    49261    ;seed6
316h    .word    306h
317h    .word    315h
318h    .word    27436    ;seed7
319h    .word    309h
320h    .word    318h
321h    .word    61712    ;seed8
322h    .word    30ch
323h    .word    31bh
324h    .word    53[illegible]    ;seed9
325h    .word    30fh
326h    .word    31ch
327h    .word    45869    ;seed10
328h    .word    312h
329h    .word    321h
330h    .word    29397    ;seed11
331h    .word    315h
332h    .word    324h
333h    .word    29938    ;seed12
334h    .word    318h
335h    .word    327h
336h    .word    43613    ;seed13
337h    .word    31bh
338h    .word    32ah
339h    .word    35111    ;seed14
340h    .word    31ch
341h    .word    32dh
342h    .word    12935    ;seed15
343h    .word    321h
344h    .word    330h
345h    .word    17692    ;seed16
346h    .word    324h
```

```
347h    .word   333h
348h    .word   43955   ;seed17
349h    .word   327h
350h    .word   336h
351h    .word   12509   ;seed18
352h    .word   32ah
353h    .word   339h
354h    .word   4038    ;seed19
355h    .word   32dh
356h    .word   33ch
357h    .word   53693   ;seed20
358h    .word   330h
359h    .word   33fh
360h    .word   23906   ;seed21
361h    .word   333h
362h    .word   342h
363h    .word   14497   ;seed22
364h    .word   336h
365h    .word   345h
366h    .word   38290   ;seed23
367h    .word   339h
368h    .word   348h
369h    .word   26323   ;seed24
370h    .word   33ch
371h    .word   34bh
372h    .word   30551   ;seed25
373h    .word   33fh
374h    .word   34ch
375h    .word   41830   ;seed26
376h    .word   342h
377h    .word   351h
378h    .word   51741   ;seed27
379h    .word   345h
380h    .word   354h
381h    .word   23023   ;seed28
382h    .word   348h
383h    .word   357h
384h    .word   42080   ;seed29
385h    .word   34bh
386h    .word   35ah
387h    .word   56061   ;seed30
388h    .word   34ch
389h    .word   300h
390h    .word   29151   ;seed31
391h    .word   351h
392h    .word   303h
```

```
*PROGRAM   DSC2.ASM
*FILTER COEFF FOR DOPPLER SHIFT FIR 121 LENGTH FILTER
* This is calculated using freq. response graph in matlab
*Q15 notation is used here.

        .ps     0990h
coeffp  .int    75,82,62,23,-33,-102,-164,-203,-210,-167
        .int    -82,39,174,298,383,400,334,190,-20,-259
        .int    -485,-646,-705,-632,-426,-108,269,639,927,1072
        .int    1022,773,347,-187,-741,-1206,-1484,-1504,-1239,-708
        .int    7,786,1491,1976,2127,1884,1262,334,-750,-1806
        .int    -2621,-3024,-2884,-2156,-881,800,2684,4519,6052,7071
        .int    7429
        .int    7071,6052,4519,2684,800,-881,-2156,-2884,-3024,-2621
        .int    -1806,-750,334,1262,1884,2127,1976,1491,786,7
        .int    -708,-1239,-1504,-1484,-1206,-741,-187,347,773,1022
        .int    1072,927,639,269,-108,-426,-632,-705,-646,-485
        .int    -259,-20,190,334,400,383,298,174,39,-82
        .int    -167,-210,-203,-164,-102,-33,23,62,82,75
```

```
*PROGRAM : IPC96.ASM
*INTERPOLATING FILTER COEFF. FOR fd = 40Hz
*Tap length = 96
*This is calculated using Matlab
*Scaleup by 15 , Q15 notation
*16x6 interpolating filter bank coeff

        .ps     2080h
        .int    1742,7455,11505,7858,1987,151
        .int    1518,7045,11469,8253,2251,174
        .int    1314,6632,11399,8636,2535,202
        .int    1130,6218,11294,9005,2839,238
        .int    965,5806,11155,9357,3160,283
        .int    819,5398,10984,9689,3499,338
        .int    691,4997,10781,10000,3854,405
        .int    580,4605,10548,10287,4223,485
        .int    485,4223,10287,10548,4605,580
        .int    405,3854,10000,10781,4997,691
        .int    338,3499,9689,10984,5398,819
        .int    283,3160,9357,11155,5806,965
        .int    238,2839,9005,11294,6218,1130
        .int    202,2535,8636,11399,6632,1314
        .int    174,2251,8253,11469,7045,1518
        .int    151,1987,7858,11505,7455,1742
```

```
*PROGRAM : IPC352_1.ASM
*INTERPOLATING FILTER COEFF. FOR fd = 20Hz
*Tap length = 352
*This is calculated using Matlab
*32x11 interpolating filter bank coeff

        .ps     2080h
        .int    -132,-19,897,2675,4224,4252,2735,942,-5,-134,-70
        .int    -130,-32,852,2615,4194,4279,2794,988,9,-136,-71
        .int    -128,-43,809,2555,4162,4304,2854,1036,24,-138,-72
        .int    -125,-54,766,2495,4129,4327,2913,1084,40,-139,-73
        .int    -123,-65,725,2434,4094,4349,2971,1133,57,-140,-74
        .int    -121,-74,684,2374,4058,4369,3029,1183,75,-141,-75
        .int    -118,-83,645,2314,4021,4387,3087,1233,94,-142,-77
        .int    -116,-91,606,2253,3982,4404,3144,1285,114,-143,-78
        .int    -113,-98,569,2193,3942,4418,3200,1337,134,-143,-80
        .int    -111,-105,533,2133,3900,4431,3256,1390,156,-143,-82
        .int    -108,-111,497,2074,3857,4442,3311,1444,179,-143,-83
```

```
        .int    -106,-117,463,2014,3813,4451,3365,1499,203,-142,-85
        .int    -103,-121,430,1955,3767,4459,3419,1554,227,-141,-87
        .int    -101,-126,398,1896,3721,4464,3472,1610,253,-140,-89
        .int    -98,-130,367,1838,3673,4468,3523,1666,280,-138,-91
        .int    -96,-133,337,1780,3624,4470,3574,1723,308,-136,-94
        .int    -94,-136,308,1723,3574,4470,3624,1780,337,-133,-96
        .int    -91,-138,280,1666,3523,4468,3673,1838,367,-130,-98
        .int    -89,-140,253,1610,3472,4464,3721,1896,398,-126,-101
        .int    -87,-141,227,1554,3419,4459,3767,1955,430,-121,-103
        .int    -85,-142,203,1499,3365,4451,3813,2014,463,-117,-106
        .int    -83,-143,179,1444,3311,4442,3857,2074,497,-111,-108
        .int    -82,-143,156,1390,3256,4431,3900,2133,533,-105,-111
        .int    -80,-143,134,1337,3200,4418,3942,2193,569,-98,-113
        .int    -78,-143,114,1285,3144,4404,3982,2253,606,-91,-116
        .int    -77,-142,94,1233,3087,4387,4021,2314,645,-83,-118
        .int    -75,-141,75,1183,3029,4369,4058,2374,684,-74,-121
        .int    -74,-140,57,1133,2971,4349,4094,2434,725,-65,-123
        .int    -73,-139,40,1084,2913,4327,4129,2495,766,-54,-125
        .int    -72,138,24,1036,2854,4304,4162,2555,809,-43,-128
        .int    -71,-136,9,988,2794,4279,4194,2615,852,-32,-130
        .int    -70,-134,-5,942,2735,4252,4224,2675,897,-19,-132


*PROGRAM : PSC2.ASM
*FILTER COEFF. FOR PULSE SHAPING FIR 96 LENGTH FILTER
*This is calculated using freq. response graph in Matlab
*Q14 notation is used here.
*Reverse the order of coeff. so as for MAC inst. to work correctly

        .ps     2020h
obj1    .int    157,-81,-319,1143,-2631,17365,654,421,-399,145,52,-100
obj2    .int    150,-168,-156,1070,-3091,15681,3389,-362,-242,218,-39,-74
obj3    .int    108,-207,19,795,-2887,13109,6596,-1299,47,226,-128,-21
obj4    .int    45,-191,157,419,-2214,9958,9958,-2214,419,157,-191,45
obj5    .int    -21,-128,226,47,-1299,6596,13109,-2887,795,19,-207,108
obj6    .int    -74,-39,218,-242,-362,3389,15681,-3091,1070,-156,-168,150
obj7    .int    -100,52,145,-399,421,654,17365,-2631,1143,-319,-81,157
obj8    .int    -96,123,33,-417,936,-1388,17951,-1388,936,-417,33,123


*PROGRAM : IF3.ASM
* RECEIVED SIGNAL SAMPLE GENERATING PROGRAM
* Doppler frequency=40Hz.
* Additional Files Reqd.: SHIFT7.ASM+DSC2.ASM+IPC96.ASM
* DSP TMS320C50

        .mmregs
*Table for prws generation simulating circular buffer (CB)
        .ds     0300h                   ;Start address for shift reg table
        .include "shift7.asm"           ;Table of length (31*3)=93 words
I_mask  .word   508                     ;Mask for I Channel
Q_mask  .word   65024                   ;Mask for Q Channel
ipfact  .word   0                       ;initial status for interpolation

* 121 Coeff. for doppler shift FIR filter
        .ps     0990h
        .include "dsc2.asm"             ;Table of Coeff for Doppler Shift filter

*Address value for ipfact used for fading interpolation
aipfact .set    35fh                    ;if ipfact=0 interpolation is over

*Address values for Iprws,Qprws in Data mem.
I_ch    .set    360h                    ;I channel in upper seven bits
Q_ch    .set    3dbh                    ;Q channel in upper seven bits

*Address for first Coeff of DS filter in Prog. mem.
```

```
coeffp          .set        0990h              ;Program memory address for DS filter coeff.

* Ichannel data page for filtering by DS filter
ds_Ich          .set        3dah               ;DS filter output for I channel
Ist_Ich         .set        361h               ;First delay element for I channel
Lst_Ich         .set        3d8h               ;361h+dec(120) Last delay element for I channel
                .ds         361h
                .space      780h               ;Initialize delay elements with zeros
                                               ;780h = no. of bits = (120 * 16)hex

* Qchannel data page for filtering by DS filter
ds_Qch          .set        455h               ;DS filter output for I channel
Ist_Qch         .set        3dch               ;First delay element for I channel
Lst_Qch         .set        453h               ;362h+dec(120) Last delay element for I channel
                .ds         3dch
                .space      780h               ;Initialize delay elements with zeros
                                               ;780h = no. of bits = (120 * 16)hex

*96 point for fd=40Hz
                .ps         2080h
                .include "ipc96.asm"   ;Table of coeff. for channel interpolating filter

* Q channel data page for interpolating filter delay line
Ist_dsQ         .set        456h
Lst_dsQ         .set        45ah       ;6 taps       ;458h >4 taps for MP ::for 11 taps 45fh
RF_Q            .set        461h
                .ds         456h
                .space      0b0h

* I channel data page for interpolating filter delay line
ds_Ich1         .set        465h
Ist_dsI         .set        466h
Lst_dsI         .set        46ah       ;6 taps       ;468h >4 taps for MP:: for 11 taps46fh
RF_I            .set        471h
                .ds         466h
                .space      0b0h
                .ds         47ah
ip_count .word              16                       ;16=40Hz;32=20Hz;64=10Hz
ip_index .word 16
ini_adr  .word  8320  ;2080h      ;initial address for filter coeff
sub_fil  .word  8320  ;2080h      ;subfilter coeff. address

* Data page 9 for Transmit Signal generation
* from address 4a0 to 4ffh only.
D_mask          .set        4a0h
Data            .set        4a1h
E_tbl           .set        4a2h
E_data          .set        4a3h
prv_ph          .set        4a4h
M_tbl           .set        4a5h

M_DI            .set        4a6h
Ist_DI          .set        4a7h
Lst_DI          .set        4b1h
MOD_I           .set        4b3h

M_DQ            .set        4b6h
Ist_DQ          .set        4b7h
Lst_DQ          .set        4c1h
MOD_Q           .set        4c3h

RX_I            .set        4c4h
RX_Q            .set        4c5h

adshift         .set        4c6h
adfreq          .set        4c7h

                .ds         4a0h
```

```
            .word      3                        ;Data Mask =3
            .word      0                        ;Data value
            .word      8192                     ;dma:4a2>E_tbl(2000h)
            .word      0000h                    ;dma:4a3h>E_data
            .word      0000h                    ;dma:4a4h>prv_ph
            .word      8208                     ;dma:4a5h>M_tbl(2010h)

            .ds        4a7h                     ;Delay line for data I
            .space     0c0h                     ;Initialize to zero

            .ds        4b7h                     ;Delay line for data Q
            .space     0c0h                     ;Initialize to zero

*Initial setting for Doppler Shift Value
            .ds        4c6h
dshift      .word      2          ;= dfreq   ;To set initial value of doppler shift/freq.
dfreq       .word      2          ;= dshift  ;80Hz=1; 40Hz=2; 20Hz=4; 10Hz=8.

*Look-up Table for encoder in program memory
            .ps        2000h
            .word      0001,0003,0007,0005

*Look up table for Modulator in Q14 notation

            .ps        2010h
            .word      16384,0
            .word      11585,11585
            .word      0,16384
            .word      -11585,11585
            .word      -16384,0
            .word      -11585,-11585
            .word      0,-16384
            .word      11585,-11585

            .ps        2020h
            .include "psc2.asm"   ;Table of Coeff for Pulse Shaping filter

*Set up the address for interrupt service routine vector
            .ps        00806h
            rete

            .ps        00b00h
            .entry
            SETC       INTM                     ; Disable interrupts
            LDP        #0                       ; Set data page pointer
            OPL        #0834h,PMST
            LACC       #0
            SAMM       CWSR                     ; Set software wait state to 0
            SAMM       PDWSR
            CLRC       OVM                      ; OVM = 0
            SPM        0
            SETC       SXM
            LDP        DXR
            SPLK       #0h,DBMR                 ;Used for comparison for fd change
            SPLK       #06h,IMR
            SPLK       #20h,TCR                 ; To generate 10 MHz from Tout
            SPLK       #01h,PRD                 ; for AIC master clock
            LACC       #00c0h                   ; Reset tx/rx port
            SACL       SPC
            CLRC       INTM                     ; enable

*PRWS generation
            lar        ar6,#0300h

S_no1
loop        idle
            ldp        #9h
            lacc       RX_I
```

```
*           lacc        MOD_I
*           ldp         #8h
*           lacc        RF_I
            ldp         #58h
            sacl        0
            ldp         #9h
            lacc        RX_Q
*           lacc        MOD_Q
*           ldp         #8h
*           lacc        RF_Q
            ldp         #58h
            sacl        1

****************PRWS GENERATION*********************
            mar         *,ar6
            lacc        *+                  ;load acc. with TAP:2
            lar         ar6,*               ;Get address for TAP:30
            xor         *                   ;XOR for M-sequence PRWS method
            sacb                            ;Pseudo Random Word Seq.
            mar         *,ar6
            sacl        *+                  ;Store result in TAP:30
            mar         *+,ar6              ;Update address for New TAP:2
            lar         ar6,*
***************END of PRWS generation*********************
*Check if new value for fading to be generated or to continue
*with interpolation
            ldp         #6h
            cpl         aipfact             ;chk if ipfact=0(DBMR) set TC=1 else TC=0
            bcnd        ip_cont,NTC         ;If TC=0 Branch to interpolation else new sample

            ldp         #6h
            lacc        I_mask
            andb
            sacl        I_ch,7              ;Store Channel I value
            lacc        I_ch                ;scaling Channel I value
            sfr
*           sfr
            sacl        I_ch
            lacc        Q_mask
            andb
            ldp         #7h
            sacl        Q_ch                ;Store Channel Q value

*As the DS filter is overflowing reduce the mag. of
*its input. One SFR command is OK but two SFR reduces any
*remote chance of overflow.
*We have used setc SXM mode so as to maintain the sign
            lacc        Q_ch
            sfr
*           sfr
            sacl        Q_ch

*---------------I Channel FIR DOPPLER SHIFT FILTER--------
*Doppler shift Filter 120 length FIR filter
            lacc        #0h                 ;Zero Accumulator
            zpr
            mar         *,ar3
            lar         ar3,#Lst_Ich
DSF_I       rpt         #120                ;FIR filter for Doppler Shift
            macd        coeffp,*-
            apac
            ldp         #7h
            sach        ds_Ich,1    ;not really reqd. to save here. CHK.??
            ldp         #8h
            sach        ds_Ich1,1   ;store on two locations
*-----------------END of I channel Doppler shift filter-------

*--------------------Q Channel FIR DOPPLER SHIFT FILTER------
```

```
*Doppler shift Filter: 120 length FIR filter
        ldp     #8h                     ;data page no.
        lacc    #0h                     ;Zero Accumulator
        zpr
*       mar     *,ar3
        lar     ar3,#Lst_Qch
DSF_Q   rpt     #120                    ;FIR filter for Doppler Shift
        macd    coeffp,*-
        apac
        ldp     #8h
        sach    ds_Qch,1

*--------------END of Q channel Doppler shift filter--------

*Data generation
ip_cont ldp     #9h
        lacc    D_mask
        andb
        sacl    Data                    ;Store Data value

*Data encoding
        lacc    E_tbl
        add     Data
        tblr    E_data

*Previous phase (state) reference
        lacc    E_data
        add     prv_ph
        and     #0007h
        sacl    prv_ph

*Data modulating
        sfl
        add     M_tbl
        tblr    M_DI
        add     #1
        tblr    M_DQ

*----- Pulse shaping + Interpolating for Data Q ---------------------
        lacc    #0h
        zpr
        mar     *,ar5
        lar     ar5,#Lst_DQ
PS_Q1   rpt     #11
        mac     obj1,*-
        apac
        sach    MOD_Q,2                 ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
        lacc    #0h
        zpr
        lar     ar5,#Lst_DI
PS_I1   rpt     #11
        mac     obj1,*-
        apac
        sach    MOD_I,2                 ,store as Q14 output
*------END of Pulse shaping-----------------------------------

        CALL    CH_IP

*Multiply (complex) transmitted signal and flat fading

        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpy     MOD_I                   ;Preg=RF_I*MOD_I
        ldp     #8h
        lt      RF_Q
```

```
        ldp     #9h
        mpya    MOD_Q           ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
        mpys    MOD_I           ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
        sach    RX_I,1          ;store as Q14
        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpya    MOD_Q           ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
        apac                    ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
        sach    RX_Q,1

S_no2   idle
        ldp     #9h
        lacc    RX_I
*       lacc    MOD_I
*       ldp     #8h
*       lacc    RF_I
        ldp     #58h
        sacl    0
        ldp     #9h
        lacc    RX_Q
*       lacc    MOD_Q
*       ldp     #8h
*       lacc    RF_Q
        ldp     #58h
        sacl    1

*----- Pulse shaping + Interpolating for Data Q ---------------------
        ldp     #9h
        lacc    #0h
        zpr
        mar     *,ar5
        lar     ar5,#Lst_DQ
PS_Q2   rpt     #11
        mac     obj2,*-
        apac
        sach    MOD_Q,2         ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
        lacc    #0h
        zpr
        lar     ar5,#Lst_DI
PS_I2   rpt     #11
        mac     obj2,*-
        apac
        sach    MOD_I,2         ,store as Q14 output
*------END of Pulse shaping-----------------------------------

        CALL    CH_IP
*Multiply (complex) transmitted signal and flat fading

        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpy     MOD_I           ;Preg=RF_I*MOD_I
        ldp     #8h
        lt      RF_Q
        ldp     #9h
        mpya    MOD_Q           ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
        mpys    MOD_I           ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
        sach    RX_I,1          ;store as Q14
        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpya    MOD_Q           ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
```

```
        apac                            ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
        sach    RX_Q,1

S_no3   idle
        ldp     #9h
        lacc    RX_I
*       lacc    MOD_I
*       ldp     #8h
*       lacc    RF_I
        ldp     #58h
        sacl    0
        ldp     #9h
        lacc    RX_Q
*       lacc    MOD_Q
*       ldp     #8h
*       lacc    RF_Q
        ldp     #58h
        sacl    1

*----- Pulse shaping + Interpolating for Data Q ---------------------
        ldp     #9h
        lacc    #0h
        zpr
        mar     *,ar5
        lar     ar5,#Lst_DQ
PS_Q3   rpt     #11
        mac     obj3,*-
        apac
        sach    MOD_Q,2         ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
        lacc    #0h
        zpr
        lar     ar5,#Lst_DI
PS_I3   rpt     #11
        mac     obj3,*-
        apac
        sach    MOD_I,2         ,store as Q14 output
*------END of Pulse shaping-----------------------------------

        CALL    CH_IP

*Multiply (complex) transmitted signal and flat fading

        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpy     MOD_I           ;Preg=RF_I*MOD_I
        ldp     #8h
        lt      RF_Q
        ldp     #9h
        mpya    MOD_Q           ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
        mpys    MOD_I           ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
        sach    RX_I,1          ;store as Q14
        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpya    MOD_Q           ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
        apac                    ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
        sach    RX_Q,1

S_no4   idle
        ldp     #9h
        lacc    RX_I
*       lacc    MOD_I
```

```
*           ldp         #8h
*           lacc        RF_I
            ldp         #58h
            sacl        0
            ldp         #9h
            lacc        RX_Q
*           lacc        MOD_Q
*           ldp         #8h
*           lacc        RF_Q
            ldp         #58h
            sacl        1

*----- Pulse shaping + Interpolating for Data Q ---------------------
            ldp         #9h
            lacc        #0h
            zpr
            mar         *,ar5
            lar         ar5,#Lst_DQ
PS_Q4       rpt         #11
            mac         obj4,*-
            apac
            sach        MOD_Q,2             ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
            lacc        #0h
            zpr
            lar         ar5,#Lst_DI
PS_I4       rpt         #11
            mac         obj4,*-
            apac
            sach        MOD_I,2             ,store as Q14 output
*------ END of Pulse shaping-----------------------------------------

            CALL        CH_IP
*Multiply (complex) transmitted signal and flat fading

            lacc        #0h
            ldp         #8h
            lt          RF_I
            ldp         #9h
            mpy         MOD_I               ;Preg=RF_I*MOD_I
            ldp         #8h
            lt          RF_Q
            ldp         #9h
            mpya        MOD_Q               ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
            mpys        MOD_I               ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
            sach        RX_I,1              ;store as Q14
            lacc        #0h
            ldp         #8h
            lt          RF_I
            ldp         #9h
            mpya        MOD_Q               ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
            apac                            ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
            sach        RX_Q,1

S_no5       idle
            ldp         #9h
            lacc        RX_I
*           lacc        MOD_I
*           ldp         #8h
*           lacc        RF_I
            ldp         #58h
            sacl        0
            ldp         #9h
            lacc        RX_Q
*           lacc        MOD_Q
*           ldp         #8h
*           lacc        RF_Q
            ldp         #58h
```

```
        sacl    1

*----- Pulse shaping + Interpolating for Data Q ----------------------
        ldp     #9h
        lacc    #0h
        zpr
        mar     *,ar5
        lar     ar5,#Lst_DQ
PS_Q5   rpt     #11
        mac     obj5,*-
        apac
        sach    MOD_Q,2             ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
        lacc    #0h
        zpr
        lar     ar5,#Lst_DI
PS_I5   rpt     #11
        mac     obj5,*-
        apac
        sach    MOD_I,2             ,store as Q14 output

*------END of Pulse shaping-----------------------------------

        CALL    CH_IP

*Multiply (complex) transmitted signal and flat fading

        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpy     MOD_I               ;Preg=RF_I*MOD_I
        ldp     #8h
        lt      RF_Q
        ldp     #9h
        mpya    MOD_Q               ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
        mpys    MOD_I               ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
        sach    RX_I,1              ;store as Q14
        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpya    MOD_Q               ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
        apac                        ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
        sach    RX_Q,1


S_no6   idle
        ldp     #9h
        lacc    RX_I
*       lacc    MOD_I
*       ldp     #8h
*       lacc    RF_I
        ldp     #58h
        sacl    0
        ldp     #9h
        lacc    RX_Q
*       lacc    MOD_Q
*       ldp     #8h
*       lacc    RF_Q
        ldp     #58h
        sacl    1

*----- Pulse shaping + Interpolating for Data Q ---------------------
        ldp     #9h
        lacc    #0h
        zpr
        mar     *,ar5
```

```
        lar     ar5,#Lst_DQ
PS_Q6   rpt     #11
        mac     obj6,*-
        apac
        sach    MOD_Q,2         ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
        lacc    #0h
        zpr
        lar     ar5,#Lst_DI
PS_I6   rpt     #11
        mac     obj6,*-
        apac
        sach    MOD_I,2         ,store as Q14 output

*------END of Pulse shaping----------------------------------

        CALL    CH_IP

*Multiply (complex) transmitted signal and flat fading

        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpy     MOD_I           ;Preg=RF_I*MOD_I
        ldp     #8h
        lt      RF_Q
        ldp     #9h
        mpya    MOD_Q           ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
        mpys    MOD_I           ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
        sach    RX_I,1          ;store as Q14
        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpya    MOD_Q           ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
        apac                    ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
        sach    RX_Q,1


S_no7   idle
        ldp     #9h
        lacc    RX_I
*       lacc    MOD_I
*       ldp     #8h
*       lacc    RF_I
        ldp     #58h
        sacl    0
        ldp     #9h
        lacc    RX_Q
*       lacc    MOD_Q
*       ldp     #8h
*       lacc    RF_Q
        ldp     #58h
        sacl    1

*----- Pulse shaping + Interpolating for Data Q ---------------------
        ldp     #9h
        lacc    #0h
        zpr
        mar     *,ar5
        lar     ar5,#Lst_DQ
PS_Q7   rpt     #11
        mac     obj7,*-
        apac
        sach    MOD_Q,2         ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
        lacc    #0h
```

```
        zpr
        lar     ar5,#Lst_DI
PS_I7   rpt     #11
        mac     obj7,*-
        apac
        sach    MOD_I,2         ;store as Q14 output

*------END of Pulse shaping---------------------------------

        CALL    CH_IP

*Multiply (complex) transmitted signal and flat fading

        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpy     MOD_I           ;Preg=RF_I*MOD_I
        ldp     #8h
        lt      RF_Q
        ldp     #9h
        mpya    MOD_Q           ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
        mpys    MOD_I           ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
        sach    RX_I,1          ;store as Q14
        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpya    MOD_Q           ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
        apac                    ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
        sach    RX_Q,1

S_no8   idle
        ldp     #9h
        lacc    RX_I
*       lacc    MOD_I
*       ldp     #8h
*       lacc    RF_I
        ldp     #58h
        sacl    0
        ldp     #9h
        lacc    RX_Q
*       lacc    MOD_Q
*       ldp     #8h
*       lacc    RF_Q
        ldp     #58h
        sacl    1

*----- Pulse shaping + Interpolating for Data Q ---------------------
        ldp     #9h
        lacc    #0h
        zpr
        mar     *,ar5
        lar     ar5,#Lst_DQ
PS_Q8   rpt     #11
        macd    obj8,*-
        apac
        sach    MOD_Q,2         ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
        lacc    #0h
        zpr
        lar     ar5,#Lst_DI
PS_I8   rpt     #11
        macd    obj8,*-
        apac
        sach    MOD_I,2         ,store as Q14 output
*------END of Pulse shaping-----------------------------------
```

```
        CALL    CH_IP

*Multiply (complex) transmitted signal and flat fading

        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpy     MOD_I           ;Preg=RF_I*MOD_I
        ldp     #8h
        lt      RF_Q
        ldp     #9h
        mpya    MOD_Q           ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
        mpys    MOD_I           ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
        sach    RX_I,1          ;store as Q14
        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpya    MOD_Q           ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
        apac                    ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
        sach    RX_Q,1

*       ldp     #9h
        lacc    adshift
        sub     #1h
        sacl    adshift         ;dhift=dshift-1
        bcnd    ip_over,eq      ;if dshift=0 branch to ip_over
        lacc    #1h             ;else set ipfact=1 i.e. continue
        ldp     #6h
        sacl    aipfact         ;interplation for fading
        b       loop            ;and generate new data symbol
ip_over lacc    adfreq          ;set dshift to initial value
        sacl    adshift         ;to continue the cycle
        lacc    #0h             ;set ipfact=0 to generate new fading
        ldp     #6h
        sacl    aipfact         ;point as well
        b       loop

*Subroutine for multiphase interpolation filtering for Channel I and Q

CH_IP   ldp     #8h
        lacc    sub_fil
        samm    bmar
        lacc    #0h
*-----Interpolating for Channel Q ---------------------
        zpr
        mar     *,ar4
        lar     ar4,#Lst_dsQ
        rpt     #5
        mads    *-
        apac
        sach    RF_Q,1          ,store as Q15 output
*-----Interpolating for Channel I ---------------------
        lacc    sub_fil
        samm    bmar
        lacc    #0h
        zpr
        lar     ar4,#Lst_dsI
        rpt     #5
        [illegible]     *-
        [illegible]
        [illegible]     RF_I,1          ,store as Q15 output
*------[illegible]annel Interpolation---------------------
        [illegible]cc   ip_index        ;previous subfilter count
        sub     #1h             ;decrement by one
        sacl    ip_index        ;new subfilter number
        bcnd    ip_last,eq      ;when filtering is complete branch to last
```

```
          lacc    sub_fil              ;previous address of subfilter coeff
          add     #6                   ;add 4
          sacl    sub_fil              ;new subfilter address
          ret

ip_last   lar     ar4,#Lst_dsQ         ;After last subfilter
          rpt     #6
          dmov    *-                   ;move data up the delay line
          lar     ar4,#Lst_dsI
          rpt     #6
          dmov    *-
          lacc    ini_adr              ;re-initialize the first subfilter address
          sacl    sub_fil
          lacc    ip_count  ;re-initialize the subfilter count
          sacl    ip_index
          ret

          .end


* PROGRAM : IF4_20.ASM
* RECEIVED SIGNAL SAMPLE GENERATING PROGRAM
* Doppler shift fd=20Hz
* Change channel interpolating to 32*11 taps
* Additional Files Reqd.: SHIFT7.ASM+DSC2.ASM+IPC352_1.ASM

          .mmregs
*Table for prws generation simulating circular buffer (CB)
          .ds       0300h              ;Start address for shift reg table
          .include "shift7.asm"  ;Table of length (31*3)=93 words
I_mask    .word     508                ;Mask for I Channel
Q_mask    .word     65024              ;Mask for Q Channel
ipfact    .word     0                  ;initial status for interpolation

* 121 coeff. for doppler shift FIR filter
          .ps       0990h
          .include "dsc2.asm"          ;Table of Coeff for Doppler Shift filter

*Address value for ipfact used for fading interpolation
aipfact  set 350h                      ;if ipfact=0 interpolation is over

*Address values for Iprws,Qprws in Data mem.
I_ch      .set      360h               ;I channel in upper seven bits
Q_ch      .set      3dbh               ;Q channel in upper seven bits

*Address for first Coeff of DS filter in Prog. mem.
coeffp    .set      0990h              ;Program memory address for DS filter coeff.

* Ichannel data page for filtering by DS filter
ds_Ich    .set      3dah               ;DS filter output for I channel
Ist_Ich   .set      361h               ;First delay element for I channel
Lst_Ich   .set      3d8h               ;361h+dec(120) Last delay element for I channel
          .ds       361h
          .space    780h               ;Initialize delay elements with zeros
                                       ;780h = no. of bits = (120 * 16)hex

* Qchannel data page for filtering by DS filter
ds_Qch    .set      455h               ;DS filter output for I channel
Ist_Qch   .set      3dch               ;First delay element for I channel
Lst_Qch.set         453h               ;362h+dec(120) Last delay element for I channel
          .ds       3dch
          .space    780h               ;Initialize delay elements with zeros
                                       ;780h = no. of bits = (120 * 16)hex

*32*11 point for fd=20Hz
          .ps       2080h
          .include "ipc352_1.asm"      ;Table of coeff. for channel interpolating filter
```

```
* Q channel data page for interpolating filter delay line
Ist_dsQ    .set      456h
Lst_dsQ    .set      45fh       ;45ah 6 taps;458h >4 taps for MP ::for 11 taps 45fh
RF_Q       .set      461h
           .ds       456h
           .space    0b0h

* I channel data page for interpolating filter delay line
ds_Ich1    .set      465h
Ist_dsI    .set      466h
Lst_dsI    .set      46fh       ;46ah 6 taps;468h >4 taps for MP:: for 11 taps46fh
RF_I       .set      471h
           .ds       466h
           .space    0b0h
           .ds       47ah
ip_count   .word     32                    ;16=40Hz;32=20Hz;64=10Hz
ip_index   .word     32
ini_adr    .word     8320       ;2080h     ;initial address for filter coeff
sub_fil    .word     8320       ;2080h     ;subfilter coeff. address

* Data page 9 for Transmit Signal generation
* from address 4a0 to 4ffh only.
D_mask     .set      4a0h
Data       .set      4a1h
E_tbl      .set      4a2h
E_data     .set      4a3h
prv_ph     .set      4a4h
M_tbl      .set      4a5h

M_DI       .set      4a6h
Ist_DI     .set      4a7h
Lst_DI     .set      4b1h
MOD_I      .set      4b3h

M_DQ       .set      4b6h
Ist_DQ     .set      4b7h
Lst_DQ     .set      4c1h
MOD_Q      .set      4c3h

RX_I       .set      4c4h
RX_Q       .set      4c5h

adshift .set 4c6h
adfreq     .set      4c7h

           .ds       4a0h
           .word     3                     ;Data Mask =3
           .word     0                     ;Data value
           .word     8192                  ;dma:4a2>E_tbl(2000h)
           .word     0000h                 ;dma:4a3h>E_data
           .word     0000h                 ;dma:4a4h>prv_ph
           .word     8208                  ;dma:4a5h>M_tbl(2010h)

           .ds       4a7h                  ;Delay line for data I
           .space    0c0h                  ;Initialize to zero

           .ds       4b7h                  ;Delay line for data Q
           .space    0c0h                  ;Initialize to zero

*Initial setting for Doppler Shift Value
           .ds       4c6h
dshift     .word     4          ;= dfreq   ;To set initial value of doppler shift/freq.
dfreq      .word     4          ;= dshift  ;80Hz=1; 40Hz=2; 20Hz=4; 10Hz=8.

*Look-up Table for encoder in program memory
           .ps       2000h
           .word     0001,0003,0007,0005
```

```
*Look up table for Modulator in Q14 notation

        .ps     2010h
        .word   16384,0
        .word   11585,11585
        .word   0,16384
        .word   -11585,11585
        .word   -16384,0
        .word   -11585,-11585
        .word   0,-16384
        .word   11585,-11585

        .ps     2020h
        .include "psc2.asm"          ;Table of Coeff for Pulse Shaping filter

*Set up the address for interrupt service routine vector
        .ps     00806h
        retc

        .ps     00b00h
        .entry
        SETC    INTM                 ; Disable interrupts
        LDP     #0                   ; Set data page pointer
        OPL     #0834h,PMST
        LACC    #0
        SAMM    CWSR                 ; Set software wait state to 0
        SAMM    PDWSR
        CLRC    OVM                  ; OVM = 0
        SPM     0
        SETC    SXM
        LDP     DXR
        SPLK    #0h,DBMR             ;Used for comparison for fd change
        SPLK    #06h,IMR
        SPLK    #20h,TCR             ; To generate 10 MHz from Tout
        SPLK    #01h,PRD             ; for AIC master clock
        LACC    #00c0h               ; Reset tx/rx port
        SACL    SPC
        CLRC    INTM                 ; enable

*PRWS generation
        lar     ar6,#0300h

S_no1
loop    idle
        ldp     #9h
        lacc    RX_I
*       lacc    MOD_I
*       ldp     #8h
*       lacc    RF_I
        ldp     #58h
        sacl    0
        ldp     #9h
        lacc    RX_Q
*       lacc    MOD_Q
*       ldp     #8h
*       lacc    RF_Q
        ldp     #58h
        sacl    1

*************PRWS GENERATION******************
        mar     *,ar6
        lacc    *+                   ;load acc. with TAP:2
        lar     ar6,*                ;Get address for TAP:30
        xor     *                    ;XOR for M-sequence PRWS method
        sacb                         ;Pseudo Random Word Seq.
        mar     *,ar6
        sacl    *+                   ;Store result in TAP:30
```

```
            mar         *+,ar6                  ;Update address for New TAP:2
            lar         ar6,*
***********END of PRWS generation******************
*Check if new value for fading to be generated or to continue
*with interpolation
            ldp         #6h
            cpl         aipfact                 ;chk if ipfact=0(DBMR) set TC=1 else TC=0
            bcnd        ip_cont,NTC             ;If TC=0 Branch to interpolation else new sample

            ldp         #6h
            lacc        I_mask
            andb
            sacl        I_ch,7                  ;Store Channel I value
            lacc        I_ch                    ;scaling Channel I value
            sfr
*           sfr
            sacl        I_ch
            lacc        Q_mask
            andb
            ldp         #7h
            sacl        Q_ch                    ;Store Channel Q value

*As the DS filter is overflowing reduce the mag. of
*its input. One SFR command is OK but two SFR reduces any
*remote chance of overflow.
*We have used setc SXM mode so as to maintain the sign
            lacc        Q_ch
            sfr
*           sfr
            sacl        Q_ch

*--------------------I Channel FIR DOPPLER SHIFT FILTER-------
*Doppler shift Filter 120 length FIR filter
            lacc        #0h                     ;Zero Accumulator
            zpr
            mar         *,ar3
            lar         ar3,#Lst_Ich
DSF_I       rpt         #120                    ;FIR filter for Doppler Shift
            macd        coeffp,*-
            apac
            ldp         #7h
            sach        ds_Ich,1                ;not really reqd. to save here.
            ldp         #8h
            sach        ds_Ich1,1               ;store on two locations
*----------------END of I channel Doppler shift filter-------

*--------------------Q Channel FIR DOPPLER SHIFT FILTER------
*Doppler shift Filter 120 length FIR filter
            ldp         #8h                     ;data page no.
            lacc        #0h                     ;Zero Accumulator
            zpr
*           mar         *,ar3
            lar         ar3,#Lst_Qch
DSF_Q       rpt         #120                    ;FIR filter for Doppler Shift
            macd        coeffp,*-
            apac
            ldp         #8h
            sach        ds_Qch,1

*-------------END of Q channel Doppler shift filter--------

*Data generation
ip_cont     ldp         #9h
            lacc        D_mask
            andb
            sacl        Data                    ;Store Data value

*Data encoding
```

```
        lacc    E_tbl
        add     Data
        tblr    E_data

*Previous phase (state) reference
        lacc    E_data
        add     prv_ph
        and     #0007h
        sacl    prv_ph

*Data modulating
        sfl
        add     M_tbl
        tblr    M_DI
        add     #1
        tblr    M_DQ

*----- Pulse shaping + Interpolating for Data Q ---------------------
        lacc    #0h
        zpr
        mar     *,ar5
        lar     ar5,#Lst_DQ
PS_Q1   rpt     #11
        mac     obj1,*-
        apac
        sach    MOD_Q,2         ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
        lacc    #0h
        zpr
        lar     ar5,#Lst_DI
PS_I1   rpt     #11
        mac     obj1,*-
        apac
        sach    MOD_I,2         ,store as Q14 output
*------END of Pulse shaping-----------------------------------

        CALL    CH_IP

*Multiply (complex) transmitted signal and flat fading

        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpy     MOD_I           ;Preg=RF_I*MOD_I
        ldp     #8h
        lt      RF_Q
        ldp     #9h
        mpya    MOD_Q           ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
        mpys    MOD_I           ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
        sach    RX_I,1          ;store as Q14
        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpya    MOD_Q           ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
        apac                    ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
        sach    RX_Q,1


S_no2   idle
        ldp     #9h
        lacc    RX_I
*       lacc    MOD_I
*       ldp     #8h
*       lacc    RF_I
        ldp     #58h
        sacl    0
```

```
          ldp       #9h
          lacc      RX_Q
*         lacc      MOD_Q
*         ldp       #8h
*         lacc      RF_Q
          ldp       #58h
          sacl      1

*----- Pulse shaping + Interpolating for Data Q ---------------------
          ldp       #9h
          lacc      #0h
          zpr
          mar       *,ar5
          lar       ar5,#Lst_DQ
PS_Q2     rpt       #11
          mac       obj2,*-
          apac
          sach      MOD_Q,2            ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
          lacc      #0h
          zpr
          lar       ar5,#Lst_DI
PS_I2     rpt       #11
          mac       ob;,.*-
          apac
          sach      MOD_I,2            ,store as Q14 output
*------END of Pulse shaping-----------------------------------

          CALL      CH_IP
*Multiply (complex) transmitted signal and flat fading

          lacc      #0h
          ldp       #8h
          lt        RF_I
          ldp       #9h
          mpy       MOD_I              ;Preg=RF_I*MOD_I
          ldp       #8h
          lt        RF_Q
          ldp       #9h
          mpya      MOD_Q              ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
          mpys      MOD_I              ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
          sach      RX_I,1             ;store as Q14
          lacc      #0h
          ldp       #8h
          lt        RF_I
          ldp       #9h
          mpya      MOD_Q              ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
          apac                         ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
          sach      RX_Q,1


S_no3     idle
          ldp       #9h
          lacc      RX_I
*         lacc      MOD_I
*         ldp       #8h
*         lacc      RF_I
          ldp       #58h
          sacl      0
          ldp       #9h
          lacc      RX_Q
*         lacc      MOD_Q
*         ldp       #8h
*         lacc      RF_Q
          ldp       #58h
          sacl      1
```

```
*----- Pulse shaping + Interpolating for Data Q ---------------------
          ldp       #9h
          lacc      #0h
          zpr
          mar       *,ar5
          lar       ar5,#Lst_DQ
PS_Q3     rpt       #11
          mac       obj3,*-
          apac
          sach      MOD_Q,2             ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
          lacc      #0h
          zpr
          lar       ar5,#Lst_DI
PS_I3     rpt       #11
          mac       obj3,*-
          apac
          sach      MOD_I,2             ,store as Q14 output
*------END of Pulse shaping-----------------------------------

          CALL      CH_IP

*Multiply (complex) transmitted signal and flat fading

          lacc      #0h
          ldp       #8h
          lt        RF_I
          ldp       #9h
          mpy       MOD_I               ;Preg=RF_I*MOD_I
          ldp       #8h
          lt        RF_Q
          ldp       #9h
          mpya      MOD_Q               ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
          mpys      MOD_I               ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
          sach      RX_I,1              ;store as Q14
          lacc      #0h
          ldp       #8h
          lt        RF_I
          ldp       #9h
          mpya      MOD_Q               ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
          apac                          ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
          sach      RX_Q,1

S_no4     idle
          ldp       #9h
          lacc      RX_I
*         lacc      MOD_I
*         ldp       #8h
*         lacc      RF_I
          ldp       #58h
          sacl      0
          ldp       #9h
          lacc      RX_Q
*         lacc      MOD_Q
*         ldp       #8h
*         lacc      RF_Q
          ldp       #58h
          sacl      1

*----- Pulse shaping + Interpolating for Data Q ---------------------
          ldp       #9h
          lacc      #0h
          zpr
          mar       *,ar5
          lar       ar5,#Lst_DQ
PS_Q4     rpt       #11
          mac       obj4,*-
          apac
```

```
        sach     MOD_Q,2            ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
        lacc     #0h
        zpr
        lar      ar5,#Lst_DI
PS_I4   rpt      #11
        mac      obj4,*-
        apac
        sach     MOD_I,2            ,store as Q14 output
*------END of Pulse shaping-----------------------------------------

        CALL     CH_IP
*Multiply (complex) transmitted signal and flat fading

        lacc     #0h
        ldp      #8h
        lt       RF_I
        ldp      #9h
        mpy      MOD_I              ;Preg=RF_I*MOD_I
        ldp      #8h
        lt       RF_Q
        ldp      #9h
        mpya     MOD_Q              ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
        mpys     MOD_I              ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
        sach     RX_I,1             ;store as Q14
        lacc     #0h
        ldp      #8h
        lt       RF_I
        ldp      #9h
        mpya     MOD_Q              ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
        apac                        ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
        sach     RX_Q,1

S_no5   idle
        ldp      #9h
        lacc     RX_I
*       lacc     MOD_I
*       ldp      #8h
*       lacc     RF_I
        ldp      #58h
        sacl     0
        ldp      #9h
        lacc     RX_Q
*       lacc     MOD_Q
*       ldp      #8h
*       lacc     RF_Q
        ldp      #58h
        sacl     1

*----- Pulse shaping + Interpolating for Data Q ---------------------
        ldp      #9h
        lacc     #0h
        zpr
        mar      *,ar5
        lar      ar5,#Lst_DQ
PS_Q5   rpt      #11
        mac      obj5,*-
        apac
        sach     MOD_Q,2            ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
        lacc     #0h
        zpr
        lar      ar5,#Lst_DI
PS_I5   rpt      #11
        mac      obj5,*-
        apac
        sach     MOD_I,2            ,store as Q14 output
```

```
*------END of Pulse shaping----------------------------------

        CALL     CH_IP

*Multiply (complex) transmitted signal and flat fading

        lacc     #0h
        ldp      #8h
        lt       RF_I
        ldp      #9h
        mpy      MOD_I            ;Preg=RF_I*MOD_I
        ldp      #8h
        lt       RF_Q
        ldp      #9h
        mpya     MOD_Q            ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
        mpys     MOD_I            ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
        sach     RX_I,1           ;store as Q14
        lacc     #0h
        ldp      #8h
        lt       RF_I
        ldp      #9h
        mpya     MOD_Q            ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
        apac                      ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
        sach     RX_Q,1


S_no6   idle
        ldp      #9h
        lacc     RX_I
*       lacc     MOD_I
*       ldp      #8h
*       lacc     RF_I
        ldp      #58h
        sacl     0
        ldp      #9h
        lacc     RX_Q
*       lacc     MOD_Q
*       ldp      #8h
*       lacc     RF_Q
        ldp      #58h
        sacl     1

*----- Pulse shaping + Interpolating for Data Q ---------------------
        ldp      #9h
        lacc     #0h
        zpr
        mar      *,ar5
        lar      ar5,#Lst_DQ
PS_Q6   rpt      #11
        mac      obj6,*-
        apac
        sach     MOD_Q,2          ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
        lacc     #0h
        zpr
        lar      ar5,#Lst_DI
PS_I6   rpt      #11
        mac      obj6,*-
        apac
        sach     MOD_I,2          ,store as Q14 output

*------END of Pulse shaping----------------------------------

        CALL     CH_IP

*Multiply (complex) transmitted signal and flat fading

        lacc     #0h
```

```
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpy     MOD_I                   ;Preg=RF_I*MOD_I
        ldp     #8h
        lt      RF_Q
        ldp     #9h
        mpya    MOD_Q                   ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
        mpys    MOD_I                   ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
        sach    RX_I,1                  ;store as Q14
        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpya    MOD_Q                   ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
        apac                            ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
        sach    RX_Q,1


S_no7   idle
        ldp     #9h
        lacc    RX_I
*       lacc    MOD_I
*       ldp     #8h
*       lacc    RF_I
        ldp     #58h
        sacl    0
        ldp     #9h
        lacc    RX_Q
*       lacc    MOD_Q
*       ldp     #8h
*       lacc    RF_Q
        ldp     #58h
        sacl    1

*----- Pulse shaping + Interpolating for Data Q ---------------------
        ldp     #9h
        lacc    #0h
        zpr
        mar     *,ar5
        lar     ar5,#Lst_DQ
PS_Q7   rpt     #11
        mac     obj7,*-
        apac
        sach    MOD_Q,2             ,store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
        lacc    #0h
        zpr
        lar     ar5,#Lst_DI
PS_I7   rpt     #11
        mac     obj7,*-
        apac
        sach    MOD_I,2             ,store as Q14 output

*------END of Pulse shaping----------------------------------

        CALL    CH_IP

*Multiply (complex) transmitted signal anu flat fading

        lacc    #0h
        ldp     #8h
        lt      RF_I
        ldp     #9h
        mpy     MOD_I                   ;Preg=RF_I*MOD_I
        ldp     #8h
        lt      RF_Q
        ldp     #9h
```

```
          mpya      MOD_Q              ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
          mpys      MOD_I              ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
          sach      RX_I,1             ;store as Q14
          lacc      #0h
          ldp       #8h
          lt        RF_I
          ldp       #9h
          mpya      MOD_Q              ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
          apac                         ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
          sach      RX_Q,1

S_no8     idle
          ldp       #9h
          lacc      RX_I
*         lacc      MOD_I
*         ldp       #8h
*         lacc      RF_I
          ldp       #58h
          sacl      0
          ldp       #9h
          lacc      RX_Q
*         lacc      MOD_Q
*         ldp       #8h
*         lacc      RF_Q
          ldp       #58h
          sacl      1

*----- Pulse shaping + Interpolating for Data Q ---------------------
          ldp       #9h
          lacc      #0h
          zpr
          mar       *,ar5
          lar       ar5,#Lst_DQ
PS_Q8     rpt       #11
          macd      obj8,*-
          apac
          sach      MOD_Q,2            ;store as Q14 output
*----- Pulse shaping + Interpolating for Data I ---------------------
          lacc      #0h
          zpr
          lar       ar5,#Lst_DI
PS_I8     rpt       #11
          macd      obj8,*-
          apac
          sach      MOD_I,2            ;store as Q14 output
*------END of Pulse shaping-----------------------------------

          CALL      CH_IP

*Multiply (complex) transmitted signal and flat fading

          lacc      #0h
          ldp       #8h
          lt        RF_I
          ldp       #9h
          mpy       MOD_I              ;Preg=RF_I*MOD_I
          ldp       #8h
          lt        RF_Q
          ldp       #9h
          mpya      MOD_Q              ;Preg=RF_Q*MOD_Q ; Acc=RF_I*MOD_I
          mpys      MOD_I              ;Preg=RF_Q*MOD_I ; Acc=RF_I*MOD_I-RF_Q*MOD_Q
          sach      RX_I,1             ;store as Q14
          lacc      #0h
          ldp       #8h
          lt        RF_I
          ldp       #9h
          mpya      MOD_Q              ;Preg=RF_I*MOD_Q ; Acc=RF_Q*MOD_I
          apac                         ;Acc=RF_Q*MOD_I+RF_I*MOD_Q
```

```
        sach    RX_Q,1

*       ldp     #9h
        lacc    adshift
        sub     #1h
        sacl    adshift         ;dhift=dshift-1
        bcnd    ip_over,eq      ;if dshift=0 branch to ip_over
        lacc    #1h             ;else set ipfact=1 i.e. continue
        ldp     #6h
        sacl    aipfact         ;interplation for fading
        b       loop            ;and generate new data symbol
ip_over lacc    adfreq          ;set dshift to initial value
        sacl    adshift         ;to continue the cycle
        lacc    #0h             ;set ipfact=0 to generate new fading
        ldp     #6h
        sacl    aipfact         ;point as well
        b       loop

*Subroutine for multiphase interpolation filtering for Channel I and Q

CH_IP   ldp     #8h
        lacc    sub_fil
        samm    bmar
        lacc    #0h
*-----Interpolating for Channel Q ---------------------
        zpr
        mar     *,ar4
        lar     ar4,#Lst_dsQ
        rpt     #10
        mads    *-
        apac
        sach    RF_Q,1          ,store as Q15 output
*-----Interpolating for Channel I ---------------------
        lacc    sub_fil
        samm    bmar
        lacc    #0h
        zpr
        lar     ar4,#Lst_dsI
        rpt     #10
        mads    *-
        apac
        sach    RF_I,1          ,store as Q15 output
*------END of Channel Interpolation---------------------
        lacc    ip_index        ;previous subfilter count
        sub     #1h             ;decrement by one
        sacl    ip_index        ;new subfilter number
        bcnd    ip_last,eq      ;when filtering is complete branch to last
        lacc    sub_fil         ;previous address of subfilter coeff
        add     #11             ;add 11
        sacl    sub_fil         ;new subfilter address
        ret

ip_last lar     ar4,#Lst_dsQ    ;After last subfilter
        rpt     #10
        dmov    *-              ;move data up the delay line
        lar     ar4,#Lst_dsI
        rpt     #10
        dmov    *-
        lacc    ini_adr         ;re-initialize the first subfilter address
        sacl    sub_fil
        lacc    ip_count        ;re-initialize the subfilter count
        sacl    ip_index
        ret

        .end
```

```
*PROGRAM :  map.cmd
*C30 Linker Command File containing Memory Map

/* All Memory is RAM */

MEMORY                    /* This describes the hardware */
{
/* External SRAM on the Main Board */

        VECTS:  origin=000000h      length=000C0h      /* Part of Bank0 */
        EPROG:  origin=0000d0h      length=00380h      /* Zero-wait */
        EDTBLK: origin=000450h      length=00330h      /* Zero-wait */
        MAP:    origin=000780h      length=00080h      /* Zero-wait  */

/*      PROG:   origin=000160h      length=00300h       Zero-wait */
/*      MAP:    origin=000460h      length=00200h      Zero-wait */
/*      DTBLK:  origin=000660h      length=001A0h       Zero-wait */


/* The gap in between PROG DTBLK is used for buffering
   since this ragion of memory will be transferred to
   809800h, this gap is reserved for the buffer */

        BANK0:  origin=000800h      length=0F800h      /* Zero-wait */
        BANK1:  origin=010000h      length=10000h      /* SRAM upgrade option */
        BANK2:  origin=020000h      length=10000h      /* SRAM upgrade option */
        BANK3:  origin=030000h      length=0F400h      /* One-wait */

/* Bank 3 is dual-ported between the 'C30 and the PC.
   The length shown is for the default 64Kx4 devices, but 16Kx4 can be used.
   In both cases the top C00h locations are reserved for monitor use.
   If the monitor is not used, you can have this area. */

/* Cached DRAM Memory Expansion on Daughter Board */

        EXPAND: origin=400000h      length=400000h     /* One of various options */

/* On-chip */

        BLK0_P:  origin=809800h     length=300h
        BLK0_M:  origin=809B00h     length=300h
        BLK1_D:  origin=809E00h     length=200h
}

SECTIONS                     /* Allocates uses to the hardware */
{
        .init:     {}        load=VECTS
        .ctext:    {}        load=EPROG
        .cdata:    {}        load=EDTBLK
        .map:      {}        load=MAP
/*      .text:     {}        load=PROG
        .data:     {}        load=DTBLK
        .dual:     {}        load=BANK3         */
}


*PROGRAM : b1.asm
*CHANNEL ESTIMATION FILTER (AVERAGING FILTER) COFFICIENT
*Design using Remez exchange algorithm in Matlab

        .float     0,0,0,0,0,0,0,0,0,0
        .float     -3.926372438188776e-004
        .float     -1.813169595538090e-003
        .float      2.030365302177151e-003
```

```
        .float   3.246944093698858e-003
        .float   2.866728976386388e-003
        .float   -1.195512611524607e-003
        .float   -7.147002615347359e-003
        .float   -1.017372560841225e-002
        .float   -5.340408466161841e-003
        .float   7.628598999227749e-003
        .float   2.159887944285561e-002
        .float   2.461030015014358e-002
        .float   7.854643868360649e-003
        .float   -2.536674816506901e-002
        .float   -5.641569517189035e-002
        .float   -5.812463639649386e-002
        .float   -9.718022814264368e-003
        .float   8.828031503043306e-002
        .float   2.089504950612768e-001
        .float   3.085997315447801e-001
        .float   0
        .float   3.085997315447801e-001
        .float   2.089504950612768e-001
        .float   8.828031503043306e-002
        .float   -9.718022814264368e-003
        .float   -5.812463639649386e-002
        .float   -5.641569517189035e-002
        .float   -2.536674816506901e-002
        .float   7.854643868360649e-003
        .float   2.461030015014358e-002
        .float   2.159887944285561e-002
        .float   7.628598999227749e-003
        .float   -5.340408466161841e-003
        .float   -1.017372560841225e-002
        .float   -7.147002615347359e-003
        .float   -1.195512611524607e-003
        .float   2.866728976386388e-003
        .float   3.246944093698858e-003
        .float   2.030365302177151e-003
        .float   -1.813169595538090e-003
        .float   -3.926372438188776e-004
        .float   0,0,0,0,0,0,0,0,0,0


*PROGRAM : h1.asm
*Header used in ADDN1.ASM
primwd          .set     0800h
expwd           .set     0000h
primary         .set     4
expanse         .set     0
BPF_TL          .set     96
z_isi1          .set     576h
z_isi2          .set     577h
SEED_LEN        .set     31
AVF_TL          .set     61
DIBUF_LEN       .set     31
NSDLEN          .set     28             ;length of noise seed generator
NSCALE1         .set     1.0e-06        ;Factor K to change S/N (K<1)

        .sect   ".init"
RESET   .word   start
INT0    .word   start
INT1    .word   TXDATA
INT2    .word   start
INT3    .word   start
XINT0   .word   start
RINT0   .word   start
XINT1   .word   start
RINT1   .word   start
TINT0   .word   start
```

```
TINT1    .word    start
DINT     .word    start
*To store output in memory if debugging is reqd.
         .sect    ".map"
         .space   128
         .sect    ".edata"
BUS_CTL  .word    00808060h          ;450h Primary and Expansion Bus Address
*        .data
BUFFER   .word    00040000h          ;451h
ADCADR   .word    00804000h          ;452h Address Channel A Analog interface
TIMECTL  .word    00808030h          ;453h TIMER 1 CONTROL REGISTER
PERIOD   .word    00808038h          ;454h TIMER 1 PERIOD REGISTER
RSTCTRL  .word    601h               ;455h TIMER 1 CONTROL REG. TO STOP TIMER
SETCTRL  .word    6c1h               ;456h TIMER 1 CONTROL REG. TO START TIMER
COUNT    .word    868                ;457h for 9600 samples/sec
                                     ;   6944 for 1200 samples/sec    ;TIMER 1 PERIOD
INC      .word    1h                 ;458h  SW wait for DSP C50
VALUE    .word    9000000            ;459h  SW wait for DSP C50
VALUE2   .word    500                ;45ah  Debugging Loop
VALUE3   .word    0                  ;45bh  RAMP GENERATE
INC3     .word    1000000h ;45ch  RAMP GENARATE
zero     .word    0000h              ;45dh
one      .word    1                  ;45eh
SEED2    .word    460h               ;45fh seed2 address
SEED     .int     1,0,3,0,0,1,0,0,2,1,1,2,1,3,3
         .int     0,3,1,2,1,2,1,2,3,3,2,1,3,0,1,3
                                     ;460h to 47eh seed values
         .space   1                  ;47fh
                                     ;BP FILTER COEFF. from 480h to 4dfh
* BPF filter coeff
         .float   -100,-74,-21,45,108,150,157,123
         .float   52,-39,-128,-191,-207,-168,-81,33
         .float   145,218,226,157,19,-156,-319,-417
         .float   -399,-242,47,419,795,1070,1143,936
         .float   421,-362,-1299,-2214,-2887,-3091,-2631,-1388
         .float   654,3389,6596,9958,13109,15681,17365,17951
         .float   17365,15681,13109,9958,6596,3389,654,-1388
         .float   -2631,-3091,-2887,-2214,-1299,-362,421,936
         .float   1143,1070,795,419,47,-242,-399,-417
         .float   -319,-156,19,157,226,218,145,33
         .float   -81,-168,-207,-191,-128,-39,52,123
         .float   157,150,108,45,-21,-74,-100,-96
         .space   29                 ;4e0h to 4fch
BPF_COF  .word    480h               ;4fdh Received Filter (BPF) Coeff. start address
BPI_DLS  .word    500h               ;4feh BPF Delay line start for I signal
BPI_DLE  .word    55fh               ;4ffh BPF Delay line end for I signal
DL_I     .space   96                 ;500h to 55fh          Delay line = 96 for I signal
I1       .float   0.0                ;560h (560hto563h)>BPF filter output array for Ik,Ikm1,Qk,Qkm1
Q1       .float   0.0                ;561h
I2       .float   0.0                ;562h
Q2       .float   0.0                ;563h
SIGN_U   .word    0080000000h        ;564h To extract sign of U for first decision
SIGN_V   .word    0040000000h        ;565h To extract sign of V for first decision
         .space   10                 ;566h to 56fh
DATA0    .word    0h                 ;570h First Decision Data 00
DATA2    .word    2h                 ;571h First Decision Data 10
DATA1    .word    1h                 ;572h First Decision Data 01
DATA3    .word    3h                 ;573h First Decision Data 11
D1       .word    0h                 ;574h First Decision for Data
         .space   1          ;575h
Lz_isi1  .word    0h                 ;576h Store zero isi point(obj 8) for scope output
Lz_isi2  .word    0h                 ;577h Store zero isi point(obj16) for scope output
CONST2   .word    562h               ;578h
D1_TBL   .word    570h               ;579h Decision first look-up table address
STORE    .word    0780h              ;57ah Store in .map for debugging
D1_DATA  .word    0                  ;57bh Store data (delay=11) to compare with D1
D1_ERR   .word    0                  ;57ch Number of errors after first decision
D1_DLY   .word    261        ;fading buid up time of 250pts;11;57dh Delay of 11 before BER calculation for D1
```

```
DATA_PT word     0                    ;57ch Number of Data points used for D1 error calculation
        .space   1                    ;57fh
DL_Q    .space   96                   ;580h to 5dfh  Delay line = 96 for Q signal
BPQ_DLS .word    580h                 ;5c0h
BPQ_DLE.word     5dfh                 ;5c1h
CONSTEL          .word     560h               ;5c2h
Up_dly  .word    0h                   ;5c3h
Vp_dly  .word    0h                   ;5c4h
DII_O   .word    0                    ;5c5h to 5cbh Constellation points for shift...
DII_1   .word    1                    ;5c6h ...as per index Q
DII_3   .word    3                    ;5c7h
DII_2   .word    2                    ;5c8h
RPT_0   .word    0                    ;5c9h
RPT_1   .word    1                    ;5cah
RPT_3   .word    3                    ;5cbh
D2      .word    0                    ;5cch Second decision value
        .space   3                    ;5cdh to 5cfh
DI0_5f0  .word   5c5h                 ;5f0h       Base address + first decision of 0
DI1_5f0  .word   5c6h                 ;5f1h
DI2_5f0  .word   5c8h                 ;5f2h
DI3_5f0  .word   5c7h                 ;5f3h
BASE_ADR .word   5f0h                 ;5f4h  Base address for d1 addition
D2_DATA.word     0                    ;5f5h Store data (delay=41) to compare with D2
D2_ERR  .word    0                    ;5f6h Number of errors after second decision
D2_DLY  .word    291  ;fading buit up delay of 250pts ;41;5f7h Delay of 41 before BER calculation for D2
SYMB_PT          .word     0                  ;5f8h Number of Symbol points used for D2 error calculation
        .space   6                    ;5f9h to 5fch

DATA_BUF .word   600h                 ;5ffh
DATA_ST.space    31                   ;600h to 61ch for data storage
DI_BUF  .word    620h                 ;61fh First element of 30 word long buffer for first decision storage
DI_ST   .space   31                   ;620h to 63ch for first decision storage
DI_DLY  .word    0                    ;63fh Prev. 30th D1 so as to shift as per D2 (delay = av. filter length)

*Averaging Filter Coeff.          ;640h to 67ch
        .include "b1.asm"

AVF_COF.word     640h                 ;67dh Averaging Filter (AVF) Coeff. start address
AVU_DLS          .word     680h               ;67eh AVF Delay line start for Up signal
AVU_DLE .word    6bch                 ;67fh AVF Delay line end for Up signal
DL_Up   .space   61                   ;680h to 6bch Delay line = 61 for Up signal
        .space   1                    ;6bdh
AVV_DLS          .word     6c0h               ;6beh AVF Delay line start for Vp signal
AVV_DLE .word    6fch                 ;6bfh AVF Delay line end for Vp signal
DL_Vp   .space   61                   ;6c0h to 6fch Delay line = 61 for Vp signal
        .space   2                    ;6fdh & 6feh
D_BUF   .word    700h                 ;6ffh pointer for data storage location
D_ST    .space   42                   ;700h to 729h data sorage for D2 BERT
*To add noise I and Q part
        .space   2                    ;72ah,72bh
NI      .float   0                    ;72ch noise I part value after scaling
NQ      .float   0                    ;72dh noise Q part value after scaling
POWI    .float 0          ;72eh
POWQ    .float 0          ;72fh
POWSP   .int   0          ;730h
NSCALE  .float   0.1                  ;731h Factor K to change S/N (K<1)
        .space   13                   ;732h to 73eh
NSDLOC  .word    740h                 ;73fh location for noise gen. seed
*seed value for noise I and Q  ; 740h to 75bh
NSEED   .int     0c31171dfh,0c0bcdafdh,00af7a460h,0aec859efh
        .int     03018ca1dh,0c06da366h,077576b2ch,066d3f110h
        .int     09592d096h,038a1b32dh,05d6272d5h,0d1bd74f2h
        .int     00fc6aa5dh,030dd8927h,0abb33287h,0451ca49eh
        .int     05924ccadh,02d768b06h,0302a2020h,05e932b43h
        .int     072f618f5h,0bd32b343h,066b7d4e8h,0567fd695h
        .int     0a24af876h,08ddb5499h,01f25d897h,0e84d36bah
RSP     .word    STACK                ;75ch Define Stack Location
STACK   .word    0                    ;75dh Begining of the stack
```

```
*PROGRAM : N1.ASM
*TO GENERATE, ADD, AND MEASURE NOISE

NGEN    PUSH    BK
        PUSH    AR2
        PUSH    IR0
        PUSHF   R6
        PUSHF   R7
        LDI     NSDLEN,BK                           ;28 length long prbs generator
*GENERATE NOISE
        LDI     @NSDLOC,AR2                         ;pointer to seed 2 (tap)
*       STI     AR2,*AR0++(1)       ;chk
        LDI     25,IR0
        LDI     0,R0
        XOR     *AR2++(IR0)%,R0,R0                  ;seed 2 xor seed 27 => seed 27
        XOR     *AR2%,R0,R0
        STI     R0,*AR2++(2)%                       ;store result over old seed 30
*       STI     R0,*AR0++(1)        ;chk
        STI     AR2,@NSDLOC                         ;store new seed 2 address
*Separate NI and NQ
        LDI     R0,R1
*       STI     R0,*AR0++(1)        ;chk
        LSH     16,R0
        ASH     -16,R0
        ASH     -16,R1
*       STI     R0,*AR0++(1)        ;chk
*       STI     R1,*AR0++(1)        ;chk
        FLOAT   R0,R7
        FLOAT   R1,R6
*       STF     R7,*AR0++(1)        ;chk
*       STF     R6,*AR0++(1)        ;chk
*Scaling so as to change SNR by changing noise level
        LDF     @NSCALE,R4
        MPYF3   R7,R4,R0
        MPYF3   R6,R4,R1
*       STF     R0,*AR0++(1)        ;chk
*       STF     R1,*AR0++(1)        ;chk
        STF     R0,@NI                              ;store noise I in memory
        STF     R1,@NQ                              ;store noise Q in memory
        POPF    R7
        POPF    R6
        POP     IR0
        POP     AR2
        POP     BK
        RETS

*Add noise to the signal at the input of receiver
ADDN    LDF     @NI,R5                              ;R5=nI and R7=rxI
        LDF     @NQ,R4                              ;R4=nQ and R6=rxQ
*       STF     R5,*AR0++(1)        ;chk
*       STF     R4,*AR0++(1)        ;chk
*To make noise = zero
        LDF     @zero,R5
        LDF     @zero,R4
*To make signal= zero
*       LDF     @zero,R7
*       LDF     @zero,R6
*Chk if pop and push are making R6,R7 work fine
*       STF     R7,*AR0++(1)        ;chk
*       STF     R6,*AR0++(1)        ;chk
*Add signal I and Q with noise I and Q
        ADDF3   R7,R5,R0            ;RxsnI
        ADDF3   R6,R4,R1            ;RxsnQ
*       STF     R0,*AR0++(1)        ;chk
*       STF     R1,*AR0++(1)        ;chk
```

```
        STF     R0,*AR5++%                  ;store signal in BPF delay line
        STF     R1,*AR3++%                  ;store signal in BPF delay line
        RETS

*To calculate Noise Power and Signal Power so as to find SNR
CSNR    MPYF3   R2,R2,R0
        MPYF3   R3,R3,R1
*       STF     R2,*AR0++(1)      ;chk
*       STF     R0,*AR0++(1)      ;chk
*       STF     R3,*AR0++(1)      ;chk
*       STF     R1,*AR0++(1)      ;chk
        LDF     @POWI,R4
        ADDF3   R0,R4,R4
*       STF     R4,*AR0++(1)      ;chk
        STF     R4,@POWI
        LDF     @POWQ,R4
        ADDF3   R1,R4,R4
*       STF     R4,*AR0++(1)      ;chk
        STF     R4,@POWQ
*Count the length of span  to calculate power ratio
        LDI     @POWSP,R4
        ADDI    @one,R4
*       STI     R4,*AR0++(1)      ;chk
        STI     R4,@POWSP
        RETS



* PROGRAM : ADDNI ASM
* RECEIVER ALGO[illegible]
        .include "[illegible]sm"
* External Progra[illegible]ess 0D0h
        .sect
* SET UP PRIMA[illegible] EXPANSION BUS WAIT STATE
start   LDP     BUS_CTL
        LDI     @BUS_CTL,ar0
        LDI     primwd,r5
        LDI     expwd,r6
        STI     r5,*+ar0(primary)
        STI     r6,*+ar0(expanse)
* Set up Stack Pointer
        LDI     @RSP,SP
*SET UP DAC CHANNEL A,B
        LDI     @ADCADR,AR0
        LDI     0,R7
        STI     R7,*+AR0(0)
||      STI     R7,*+AR0(1)
* Set up BPF filter parameters
LOOP    LDI     BPF_TL,BK
        LDI     @BPQ_DLE,AR3                ;Load pointer for Q signal delay line
        LDI     @BPI_DLE,AR5                ;Load pointer for I signal delay line
        LDI     @BPF_COF,AR4                ;Load pointer for Band-pass filter Coeff.
        LDI     @STORE,AR0
*Wait for C50 to stabilize
*Load C30 software; before starting C30 software
*load C50 software and run C30 software
*Use this time to Reset C50 and start C50 software
        LDI     @VALUE,AR6
        SUBI    @INC,AR6
        STI     AR6,@VALUE
        CMPI    0,AR6
        BZ      okstart
        BR      LOOP
okstart
*To start c50 at the same point always
* SET UP TIMER 1 to set sampling rate
        LDI     @TIMECTL,AR2
        LDI     @RSTCTRL,R2
```

```
          STI      R2,*AR2             ;RESET TIMER CONTROL REGISTER
          LDI      @PERIOD,AR6
          LDI      @COUNT,R2
          STI      R2,*AR6             ;SET TIMER 1 PERIOD REGISTER (PERIOD)
          LDI      @SETCTRL,R2
          STI      R2,*AR2             ;SET TIMER 1 CONTROL REGISTER (START)
* SET UP INTERRUPTS
          LDI      0H,IF
          LDI      2H,IE
          OR       2000H,ST
* WAIT FOR INTERRUPTS
* Use very first interrupt, so as to neglect the first buffer read
* of arbitrary data
          idle
* Continue with receiver logic
WAIT
obj1      idle
          LDI      @BUFFER,AR1                      ;Read the C50 Buffer Output
          LDI      *AR1,R4
*         STI      R4,*AR0++(1)        ;chk
          LDIU     R4,AR7
          LDIU     R4,AR6
          LSH      16,AR7
          ASH      -16,AR7                          ;Store I signal separately
          ASH      -16,AR6                          ;Store Q signal sparately
          FLOAT    AR7,R7
          FLOAT    AR6,R6
*         STF      R7,*AR0++(1)        ;chk
*         STF      R6,*AR0++(1)        ;chk
          CALL     NGEN
          CALL     ADDN
*         STF      R7,*AR5++%                       ;Load I signal to [illegible] delay line
*         STF      R6,*AR3++%                       ;Load Q signal to [illegible]B delay line
*B.P. Rx. Filter for I signal
          LDF      @zero,R0            ;Initialize intermediate storage registers
          LDF      @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
*for I signal
          RPTS     BPF_TL-1
          MPYF3    *AR5++%,*AR4++%,R0
||        ADDF3    R0,R2,R2
          ADDF     R0,R2                            ;R2 has BP I filter output
*B.P. Rx. Filter for Q signal
          LDF      @zero,R1            ;Initialize intermediate storage registers
          LDF      @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
          RPTS     BPF_TL-1
          MPYF3    *AR3++%,*AR4++%,R1
||        ADDF3    R1,R3,R3
          ADDF     R1,R3                            ;R3 has BP Q filter output
          CALL     CSNR
          CALL     DATA

obj2      idle
          LDI      @BUFFER,AR1                      ;Read the C50 Buffer Output
          LDI      *AR1,R4
*         STI      R4,*AR0++(1)        ;chk
          LDIU     R4,AR7
          LDIU     R4,AR6
          LSH      16,AR7
          ASH      -16,AR7                          ;Store I signal separately
          ASH      -16,AR6                          ;Store Q signal sparately
          FLOAT    AR7,R7
          FLOAT    AR6,R6
*         STF      R7,*AR0++(1)        ;chk
*         STF      R6,*AR0++(1)        ;chk
          CALL     NGEN
```

```
        CALL    ADDN
*       STF     R7,*AR5++%              ;Load I signal to CB delay line
*       STF     R6,*AR3++%              ;Load Q signal to CB delay line
                                        ;B.P. Rx. Filter for I signal
        LDF     @zero,R0        ;Initialize intermediate storage registers
        LDF     @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
        RPTS    BPF_TL-1
        MPYF3   *AR5++%,*AR4++%,R0
||      ADDF3   R0,R2,R2
        ADDF    R0,R2                   ;R2 has BP I filter output
*B.P. Rx. Filter for Q signal
        LDF     @zero,R1        ;Initialize intermediate storage registers
        LDF     @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
        RPTS    BPF_TL-1
        MPYF3   *AR3++%,*AR4++%,R1
||      ADDF3   R1,R3,R3
        ADDF    R1,R3                   ;R3 has BP Q filter output
        CALL    CSNR

obj3    idle
        LDI     @BUFFER,AR1             ;Read the C50 Buffer Output
        LDI     *AR1,R4
*       STI     R4,*AR0++(1)    ;chk
        LDIU    R4,AR7
        LDIU    R4,AR6
        LSH     16,AR7
        ASH     -16,AR7                 ;Store I signal separately
        ASH     -16,AR6                 ;Store Q signal sparately
        FLOAT   AR7,R7
        FLOAT   AR6,R6
*       STF     R7,*AR0++(1)    ;chk
*       STF     R6,*AR0++(1)    ;chk
        CALL    NGEN
        CALL    ADDN
*       STF     R7,*AR5++%              ;Load I signal to CB delay line
*       STF     R6,*AR3++%              ;Load Q signal to CB delay line
                                        ;B.P. Rx. Filter for I signal
        LDF     @zero,R0        ;Initialize intermediate storage registers
        LDF     @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
        RPTS    BPF_TL-1
        MPYF3   *AR5++%,*AR4++%,R0
||      ADDF3   R0,R2,R2
        ADDF    R0,R2                   ;R2 has BP I filter output
*B.P. Rx. Filter for Q signal
        LDFU    @zero,R1        ;Initialize intermediate storage registers
        LDFU    @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
        RPTS    BPF_TL-1
        MPYF3   *AR3++%,*AR4++%,R1
||      ADDF3   R1,R3,R3
        ADDF    R1,R3                   ;R3 has BP Q filter output
        CALL    CSNR

obj4    idle
        LDI     @BUFFER,AR1             ;Read the C50 Buffer Output
        LDI     *AR1,R4
*       STI     R4,*AR0++(1)    ;chk
        LDIU    R4,AR7
        LDIU    R4,AR6
        LSH     16,AR7
        ASH     -16,AR7                 ;Store I signal separately
        ASH     -16,AR6                 ;Store Q signal sparately
        FLOAT   AR7,R7
```

```
        FLOAT   AR6,R6
*       STF     R7,*AR0++(1)        ;chk
*       STF     R6,*AR0++(1)        ;chk
        CALL    NGEN
        CALL    ADDN
*       STF     R7,*AR5++%                      ;Load I signal to CB delay line
*       STF     R6,*AR3++%                      ;Load Q signal to CB delay line
                                                ;B.P. Rx. Filter for I signal
        LDF     @zero,R0            ;Initialize intermediate storage registers
        LDF     @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
        RPTS    BPF_TL-1
        MPYF3   *AR5++%,*AR4++%,R0
||      ADDF3   R0,R2,R2
        ADDF    R0,R2                           ;R2 has BP I filter output
*B.P. Rx. Filter for Q signal
        LDF     @zero,R1            ;Initialize intermediate storage registers
        LDF     @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
        RPTS    BPF_TL-1
        MPYF3   *AR3++%,*AR4++%,R1
||      ADDF3   R1,R3,R3
        ADDF    R1,R3                           ;R3 has BP Q filter output
        CALL    CSNR

obj5    idle
        LDI     @BUFFER,AR1                     ;Read the C50 Buffer Output
        LDI     *AR1,R4
*       STI     R4,*AR0++(1)        ;chk
        LDIU    R4,AR7
        LDIU    R4,AR6
        LSH     16,AR7
        ASH     -16,AR7                         ;Store I signal separately
        ASH     -16,AR6                         ;Store Q signal sparately
        FLOAT   AR7,R7
        FLOAT   AR6,R6
*       STF     R7,*AR0++(1)        ;chk
*       STF     R6,*AR0++(1)        ;chk
        CALL    NGEN
        CALL    ADDN
*       STF     R7,*AR5++%                      ;Load I signal to CB delay line
*       STF     R6,*AR3++%                      ;Load Q signal to CB delay line
                                                ;B.P. Rx. Filter for I signal
        LDF     @zero,R0            ;Initialize intermediate storage registers
        LDF     @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
        RPTS    BPF_TL-1
        MPYF3   *AR5++%,*AR4++%,R0
||      ADDF3   R0,R2,R2
        ADDF    R0,R2                           ;R2 has BP I filter output
*B.P. Rx. Filter for Q signal
        LDF     @zero,R1            ;Initialize intermediate storage registers
        LDF     @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
        RPTS    BPF_TL-1
        MPYF3   *AR3++%,*AR4++%,R1
||      ADDF3   R1,R3,R3
        ADDF    R1,R3                           ;R3 has BP Q filter output
        CALL    CSNR

obj6    idle
        LDI     @BUFFER,AR1                     ;Read the C50 Buffer Output
        LDI     *AR1,R4
        LDIU    R4,AR7
        LDIU    R4,AR6
        LSH     16,AR7
```

```
        ASH     -16,AR7                         ;Store I signal separately
        ASH     -16,AR6                         ;Store Q signal sparately
        FLOAT   AR7,R7
        FLOAT   AR6,R6
*       STF     R7,*AR0++(1)        ;chk
*       STF     R6,*AR0++(1)        ;chk
        CALL    NGEN
        CALL    ADDN
*       STF     R7,*AR5++%                      ;Load I signal to CB delay line
*       STF     R6,*AR3++%                      ;Load Q signal to CB delay line
                                                ;B.P. Rx. Filter for I signal
        LDF     @zero,R0            ;Initialize intermediate storage registers
        LDF     @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
        RPTS    BPF_TL-1
        MPYF3   *AR5++%,*AR4++%,R0
||      ADDF3   R0,R2,R2
        ADDF    R0,R2                           ;R2 has BP I filter output
*B.P. Rx. Filter for Q signal
        LDF     @zero,R1            ;Initialize intermediate storage registers
        LDF     @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
        RPTS    BPF_TL-1
        MPYF3   *AR3++%,*AR4++%,R1
||      ADDF3   R1,R3,R3
        ADDF    R1,R3                           ;R3 has BP Q filter output
        CALL    CSNR

obj7    idle
        LDI     @BUFFER,AR1                     ;Read the C50 Buffer Output
        LDI     *AR1,R4
        LDIU    R4,AR7
        LDIU    R4,AR6
        LSH     16,AR7
        ASH     -16,AR7                         ;Store I signal separately
        ASH     -16,AR6                         ;Store Q signal sparately
        FLOAT   AR7,R7
        FLOAT   AR6,R6
*       STF     R7,*AR0++(1)        ;chk
*       STF     R6,*AR0++(1)        ;chk
        CALL    NGEN
        CALL    ADDN
*       STF     R7,*AR5++%                      ;Load I signal to CB delay line
*       STF     R6,*AR3++%                      ;Load Q signal to CB delay line
                                                ;B.P. Rx. Filter for I signal
        LDF     @zero,R0            ;Initialize intermediate storage registers
        LDF     @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
        RPTS    BPF_TL-1
        MPYF3   *AR5++%,*AR4++%,R0
||      ADDF3   R0,R2,R2
        ADDF    R0,R2                           ;R2 has BP I filter output
*B.P. Rx. Filter for Q signal
        LDF     @zero,R1            ;Initialize intermediate storage registers
        LDF     @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
        RPTS    BPF_TL-1
        MPYF3   *AR3++%,*AR4++%,R1
||      ADDF3   R1,R3,R3
        ADDF    R1,R3                           ;R3 has BP Q filter output
        CALL    CSNR

obj8    idle
        LDI     @CONSTEL,AR2                    ;Pointer to store Q1,I1
        LDI     @BUFFER,AR1                     ;Read the C50 Buffer Output
        LDI     *AR1,R4
```

```
        LDIU    R4,AR7
        LDIU    R4,AR6
        LSH     16,AR7
        ASH     -16,AR7                          ;Store I signal separately
        ASH     -16,AR6                          ;Store Q signal sparately
        FLOAT   AR7,R7
        FLOAT   AR6,R6
*       STF     R7,*AR0++(1)        ;chk
*       STF     R6,*AR0++(1)        ;chk
        CALL    NGEN
        CALL    ADDN
*       STF     R7,*AR5++%                       ;Load I signal to CB delay line
*       STF     R6,*AR3++%                       ;Load Q signal to CB delay line
                                                 ;B.P. Rx. Filter for I signal
        LDF     @zero,R0            ;Initialize intermediate storage registers
        LDF     @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
        RPTS    BPF_TL-1
        MPYF3   *AR5++%,*AR4++%,R0
||      ADDF3   R0,R2,R2
        ADDF    R0,R2                            ;R2 has BP I filter output
        STF     R2,@z_isi1                       ;store zero isi pt. in memory for scope output
*       FLOAT   R2,R6
*       STF     R6,*AR2++(1)                     ;Store I1 for U and V calculation in memory
*       STF     R2,*AR0++(1)        ;chk
        STF     R2,*AR2++(1)                     ;Store I1 for U and V calculation in memory

*B.P. Rx. Filter for Q signal
        LDF     @zero,R1            ;Initialize intermediate storage registers
        LDF     @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
        RPTS    BPF_TL-1
        MPYF3   *AR3++%,*AR4++%,R1
||      ADDF3   R1,R3,R3
        ADDF    R1,R3                            ;R3 has BP Q filter output
        STF     R3,@z_isi2                       ;store zero isi pt. in memory for scope output
*       FLOAT   R3,R7
*       STF     R7,*AR2++(1)                     ;Store Q1 for U and V calculation
*       STF     R3,*AR0++(1)        ;chk
        STF     R3,*AR2++(1)                     ;Store Q1 for U and V calculation
        CALL    CSNR

*Calculate U and V for first Decision
        LDI     2,IR0
        MPYF3   *AR2,*-AR2(IR0),R5
        MPYF3   *+AR2(1),*-AR2(1),R4
        ADDF3   R4,R5,R6            ;U = Q1*Q2+I1*I2
*       STF     R6,*AR0++(1)        ;chk
        LDIN    2,R0                             ;extract sign of U and store
        LDINN   0,R0                             ;value to calculate index for D1
        MPYF3   *+AR2(1),*-AR2(IR0),R5
        MPYF3   *-AR2(1),*AR2,R4
        SUBF3   R5,R4,R7            ;V = I2*Q1-I1*Q2
*       STF     R7,*AR0++(1)        ;chk
        LDIN    1,R1                             ;extract sign of V and store
        LDINN   0,R1                             ;value to calculate index for D1
*First Decision based on U and V
        ADDI3   R0,R1,IR1           ;Index for look-up table for first decision
        LDI     @D1_TBL,AR2                      ;based on sign of U and V
        LDI     *+AR2(IR1),R5                    ;D1 = First Decision
        STI     R5,@D1                           ;store D1 in memory
*       STI     R5,*AR0++(1)        ;chk
        CALL    BER
        CALL    STORE_D1
*Extract Channel Characteristics i.e. Up and Vp
        NEGF    R6,R4                            ;R6=U and R4=-U
        NEGF    R7,R5                            ;R7=V and R5=-V
```

```
        LDF     R6,R6                  ;extract sign of U for Up and Vp
        LDFNN   R6,R0                  ;for +ive U, Up1=R0=U
        LDFNN   R7,R1                  ;for +ive U, Vp1=R1=V
        LDFN    R4,R0                  ;for -ive U, Up1=R0=-U
        LDFN    R5,R1                  ;for -ive U, Vp1=R1=-V
        LDF     R7,R7                  ;extract sign of V for Up and Vp
        LDFNN   R7,R2                  ;for +ive V, Up2=R2=V
        LDFNN   R4,R3                  ;for +ive V, Vp2=R3=-U
        LDFN    R5,R2                  ;for -ive V, Up2=R2=-V
        LDFN    R6,R3                  ;for -ive V, Vp2=R3=U
        ADDF3   R0,R2,R4      ;Up=Up1+Up2=R0+R2 based on sign of U and V or D1
        ADDF3   R1,R3,R5      ;Vp=Vp1+Vp2=R1+R3 based on sign of U and V or D1
*       STF     R4,*AR0++(1)  ;chk
*       STF     R5,*AR0++(1)  ;chk
*Averaging filter for Up and Vp
        CALL    AVF                    ;Call Averaging Filter Subroutine
obj9    idle
        LDI     @BUFFER,AR1            ;Read the C50 Buffer Output
        LDI     *AR1,R4
        LDIU    R4,AR7
        LDIU    R4,AR6
        LSH     16,AR7
        ASH     -16,AR7                ;Store I signal separately
        ASH     -16,AR6                ;Store Q signal sparately
        FLOAT   AR7,R7
        FLOAT   AR6,R6
*       STF     R7,*AR0++(1)  ;chk
*       STF     R6,*AR0++(1)  ;chk
        CALL    NGEN
        CALL    ADDN
*       STF     R7,*AR5++%             ;Load I signal to CB delay line
*       STF     R6,*AR3++%             ;Load Q signal to CB delay line
                                       ;B.P. Rx. Filter for I signal
        LDF     @zero,R0      ;Initialize intermediate storage registers
        LDF     @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
        RPTS    BPF_TL-1
        MPYF3   *AR5++%,*AR4++%,R0
||      ADDF3   R0,R2,R2
        ADDF    R0,R2                  ;R2 has BP I filter output
*B.P. Rx. Filter for Q signal
        LDF     @zero,R1      ;Initialize intermediate storage registers
        LDF     @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
        RPTS    BPF_TL-1
        MPYF3   *AR3++%,*AR4++%,R1
||      ADDF3   R1,R3,R3
        ADDF    R1,R3                  ;R3 has BP Q filter output
        CALL    CSNR
        CALL    DATA

obj10   idle
        LDI     @BUFFER,AR1            ;Read the C50 Buffer Output
        LDI     *AR1,R4
        LDIU    R4,AR7
        LDIU    R4,AR6
        LSH     16,AR7
        ASH     -16,AR7                ;Store I signal separately
        ASH     -16,AR6                ;Store Q signal sparately
        FLOAT   AR7,R7
        FLOAT   AR6,R6
*       STF     R7,*AR0++(1)  ;chk
*       STF     R6,*AR0++(1)  ;chk
        CALL    NGEN
        CALL    ADDN
*       STF     R7,*AR5++%             ;Load I signal to CB delay line
*       STF     R6,*AR3++%             ;Load Q signal to CB delay line
```

```
                                                        ;B.P. Rx. Filter for I signal
        LDF     @zero,R0                ;Initialize intermediate storage registers
        LDF     @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
        RPTS    BPF_TL-1
        MPYF3   *AR5++%,*AR4++%,R0
||      ADDF3   R0,R2,R2
        ADDF    R0,R2                           ;R2 has BP I filter output
*B.P. Rx. Filter for Q signal
        LDF     @zero,R1                ;Initialize intermediate storage registers
        LDF     @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
        RPTS    BPF_TL-1
        MPYF3   *AR3++%,*AR4++%,R1
||      ADDF3   R1,R3,R3
        ADDF    R1,R3                           ;R3 has BP Q filter output
        CALL    CSNR

obj11   idle
        LDI     @BUFFER,AR1                     ;Read the C50 Buffer Output
        LDI     *AR1,R4
        LDIU    R4,AR7
        LDIU    R4,AR6
        LSH     16,AR7
        ASH     -16,AR7                         ;Store I signal separately
        ASH     -16,AR6                         ;Store Q signal sparately
        FLOAT   AR7,R7
        FLOAT   AR6,R6
*       STF     R7,*AR0++(1)        ;chk
*       STF     R6,*AR0++(1)        ;chk
        CALL    NGEN
        CALL    ADDN
*       STF     R7,*AR5++%                      ;Load I signal to CB delay line
*       STF     R6,*AR3++%                      ;Load Q signal to CB delay line
                                                ;B.P. Rx. Filter for I signal
        LDF     @zero,R0                ;Initialize intermediate storage registers
        LDF     @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
        RPTS    BPF_TL-1
        MPYF3   *AR5++%,*AR4++%,R0
||      ADDF3   R0,R2,R2
        ADDF    R0,R2                           ;R2 has BP I filter output
*B.P. Rx. Filter for Q signal
        LDF     @zero,R1                ;Initialize intermediate storage registers
        LDF     @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
        RPTS    BPF_TL-1
        MPYF3   *AR3++%,*AR4++%,R1
||      ADDF3   R1,R3,R3
        ADDF    R1,R3                           ;R3 has BP Q filter output
        CALL    CSNR

obj12   idle
        LDI     @BUFFER,AR1                     ;Read the C50 Buffer Output
        LDI     *AR1,R4
        LDIU    R4,AR7
        LDIU    R4,AR6
        LSH     16,AR7
        ASH     -16,AR7                         ;Store I signal separately
        ASH     -16,AR6                         ;Store Q signal sparately
        FLOAT   AR7,R7
        FLOAT   AR6,R6
*       STF     R7,*AR0++(1)        ;chk
*       STF     R6,*AR0++(1)        ;chk
        CALL    NGEN
        CALL    ADDN
```

```
*         STF       R7,*AR5++%                     ;Load I signal to CB delay line
*         STF       R6,*AR3++%                     ;Load Q signal to CB delay line
                                                   ;B.P. Rx. Filter for I signal
          LDF       @zero,R0              ;Initialize intermediate storage registers
          LDF       @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
          RPTS      BPF_TL-1
          MPYF3     *AR5++%,*AR4++%,R0
||        ADDF3     R0,R2,R2
          ADDF      R0,R2                          ;R2 has BP I filter output
*B.P. Rx. Filter for Q signal
          LDF       @zero,R1              ;Initialize intermediate storage registers
          LDF       @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
          RPTS      BPF_TL-1
          MPYF3     *AR3++%,*AR4++%,R1
||        ADDF3     R1,R3,R3
          ADDF      R1,R3                          ;R3 has BP Q filter output
          CALL      CSNR

obj13     idle
          LDI       @BUFFER,AR1                    ;Read the C50 Buffer Output
          LDI       *AR1,R4
          LDIU      R4,AR7
          LDIU      R4,AR6
          LSH       16,AR7
          ASH       -16,AR7                        ;Store I signal separately
          ASH       -16,AR6                        ;Store Q signal sparately
          FLOAT     AR7,R7
          FLOAT     AR6,R6
*         STF       R7,*AR0++(1)          ;chk
*         STF       R6,*AR0++(1)          ;chk
          CALL      NGEN
          CALL      ADDN
*         STF       R7,*AR5++%                     ;Load I signal to CB delay line
*         STF       R6,*AR3++%                     ;Load Q signal to CB delay line
                                                   ;B.P. Rx. Filter for I signal
          LDF       @zero,R0              ;Initialize intermediate storage registers
          LDF       @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
          RPTS      BPF_TL-1
          MPYF3     *AR5++%,*AR4++%,R0
||        ADDF3     R0,R2,R2
          ADDF      R0,R2                          ;R2 has BP I filter output
*B.P. Rx. Filter for Q signal
          LDF       @zero,R1              ;Initialize intermediate storage registers
          LDF       @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
          RPTS      BPF_TL-1
          MPYF3     *AR3++%,*AR4++%,R1
||        ADDF3     R1,R3,R3
          ADDF      R1,R3                          ;R3 has BP Q filter output
          CALL      CSNR

obj14     idle
          LDI       @BUFFER,AR1                    ;Read the C50 Buffer Output
          LDI       *AR1,R4
          LDIU      R4,AR7
          LDIU      R4,AR6
          LSH       16,AR7
          ASH       -16,AR7                        ;Store I signal separately
          ASH       -16,AR6                        ;Store Q signal sparately
          FLOAT     AR7,R7
          FLOAT     AR6,R6
*         STF       R7,*AR0++(1)          ;chk
*         STF       R6,*AR0++(1)          ;chk
```

```
         CALL    NGEN
         CALL    ADDN
*        STF     R7,*AR5++%                   ;Load I signal to CB delay line
*        STF     R6,*AR3++%                   ;Load Q signal to CB delay line
                                              ;B.P. Rx. Filter for I signal
         LDF     @zero,R0           ;Initialize intermediate storage registers
         LDF     @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
         RPTS    BPF_TL-1
         MPYF3   *AR5++%,*AR4++%,R0
||       ADDF3   R0,R2,R2
         ADDF    R0,R2                        ;R2 has BP I filter output
*B.P. Rx. Filter for Q signal
*without array storage for now
         LDF     @zero,R1           ;Initialize intermediate storage registers
         LDF     @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
         RPTS    BPF_TL-1
         MPYF3   *AR3++%,*AR4++%,R1
||       ADDF3   R1,R3,R3
         ADDF    R1,R3                        ;R3 has BP Q filter output
         CALL    CSNR

obj15    idle
         LDI     @BUFFER,AR1                  ;Read the C50 Buffer Output
         LDI     *AR1,R4
         LDIU    R4,AR7
         LDIU    R4,AR6
         LSH     16,AR7
         ASH     -16,AR7                      ;Store I signal separately
         ASH     -16,AR6                      ;Store Q signal sparately
         FLOAT   AR7,R7
         FLOAT   AR6,R6
*        STF     R7,*AR0++(1)       ;chk
*        STF     R6,*AR0++(1)       ;chk
         CALL    NGEN
         CALL    ADDN
*        STF     R7,*AR5++%                   ;Load I signal to CB delay line
*        STF     R6,*AR3++%                   ;Load Q signal to CB delay line
                                              ;B.P. Rx. Filter for I signal
         LDF     @zero,R0           ;Initialize intermediate storage registers
         LDF     @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
         RPTS    BPF_TL-1
         MPYF3   *AR5++%,*AR4++%,R0
||       ADDF3   R0,R2,R2
         ADDF    R0,R2                        ;R2 has BP I filter output
*B.P. Rx. Filter for Q signal
         LDF     @zero,R1           ;Initialize intermediate storage registers
         LDF     @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
         RPTS    BPF_TL-1
         MPYF3   *AR3++%,*AR4++%,R1
||       ADDF3   R1,R3,R3
         ADDF    R1,R3                        ;R3 has BP Q filter output
         CALL    CSNR

obj16    idle
         LDI     @CONST2,AR2                  ;Pointer to store Q2,I2
         LDI     @BUFFER,AR1                  ;Read the C50 Buffer Output
         LDI     *AR1,R4
         LDIU    R4,AR7
         LDIU    R4,AR6
         LSH     16,AR7
         ASH     -16,AR7                      ;Store I signal separately
         ASH     -16,AR6                      ;Store Q signal sparately
```

```
        FLOAT   AR7,R7
        FLOAT   AR6,R6
*       STF     R7,*AR0++(1)      ;chk
*       STF     R6,*AR0++(1)      ;chk
        CALL    NGEN
        CALL    ADDN
*       STF     R7,*AR5++%                      ;Load I signal to CB delay line
*       STF     R6,*AR3++%                      ;Load Q signal to CB delay line
                                                ;B.P. Rx. Filter for I signal
        LDF     @zero,R0          ;Initialize intermediate storage registers
        LDF     @zero,R2
*FIR Filter for square root raised cosine received band-pass filter
        RPTS    BPF_TL-1
        MPYF3   *AR5++%,*AR4++%,R0
||      ADDF3   R0,R2,R2
        ADDF    R0,R2                           ;R2 has BP I filter output
        STF     R2,@z_isi1                      ;store zero isi pt. in memory for scope output
*       FLOAT   R2,R6
*       STF     R6,*AR2                         ;Store I2 for U and V calculation in memory
*       STF     R2,*AR0++(1)      ;chk
        STF     R2,*AR2                         ;Store I2 for U and V calculation in memory

*B.P. Rx. Filter for Q signal
        LDF     @zero,R1          ;Initialize intermediate storage registers
        LDF     @zero,R3
*FIR Filter for square root raised cosine received band-pass filter
*for Q signal
        RPTS    BPF_TL-1
        MPYF3   *AR3++%,*AR4++%,R1
||      ADDF3   R1,R3,R3
        ADDF    R1,R3                           ;R3 has BP Q filter output
        STF     R3,@z_isi2                      ;store zero isi pt. in memory for scope output
*       FLOAT   R3,R7
*       STF     R7,*+AR2(1)                     ;Store Q2 for U and V calculation in memory
*       STF     R3,*AR0++(1)      ;chk
        STF     R3,*+AR2(1)                     ;Store Q2 for U and V calculation in memory
        CALL    CSNR

*Calculate U and V for first decision
        LDI     2,IR0
        MPYF3   *AR2,*-AR2(IR0),R5
        MPYF3   *+AR2(1),*-AR2(1),R4
        ADDF3   R4,R5,R6          ;U = Q1*Q2+I1*I2
*       STF     R6,*AR0++(1)      ;chk
        LDIN    2,R0                            ;extract sign of U and store
        LDINN   0,R0                            ;value to calculate index
        MPYF3   *+AR2(1),*-AR2(IR0),R5
        MPYF3   *-AR2(1),*AR2,R4
        SUBF3   R4,R5,R7          ;V = I2*Q1-I1*Q2
*       STF     R7,*AR0++(1)      ;chk
        LDIN    1,R1                            ;extract sign of V and store
        LDINN   0,R1                            ;value to calculate index
*First Decision based on U and V
        ADDI3   R0,R1,IR1         ;Index for look-up table for first decision
        LDI     @D1_TBL,AR2                     ;based on sign of U and V
        LDI     *+AR2(IR1),R5                   ;D1 = First Decision
        STI     R5,@D1                          ;store D1 in memory
*       STI     R5,*AR0++(1)      ;chk
        CALL    BER                             ;calculate error rate
        CALL    STORE_D1                        ;store 30 D1 values in an array

*Extract Channel Characteristics i.e. Up and Vp
        NEGF    R6,R4                           ;R6=U and R4=-U
        NEGF    R7,R5                           ;R7=V and R5=-V
        LDF     R6,R6                           ;extract sign of U for Up and Vp
        LDFNN   R6,R0                           ;for +ive U, Up1=R0=U
        LDFNN   R7,R1                           ;for +ive U, Vp1=R1=V
        LDFN    R4,R0                           ;for -ive U, Up1=R0=-U
```

```
        LDFN    R5,R1                        ;for -ive U, Vp1=R1=-V
        LDF     R7,R7                        ;extract sign of V for Up and Vp
        LDFNN   R7,R2                        ;for +ive V, Up2=R2=V
        LDFNN   R4,R3                        ;for +ive V, Vp2=R3=-U
        LDFN    R5,R2                        ;for -ive V, Up2=R2=-V
        LDFN    R6,R3                        ;for -ive V, Vp2=R3=U
        ADDF3   R0,R2,R4             ;Up=Up1+Up2=R0+R2 based on sign of U and V or D1
        ADDF3   R1,R3,R5             ;Vp=Vp1+Vp2=R1+R3 based on sign of U and V or D1
*       STF     R4,*AR0++(1)         ;chk
*       STF     R5,*AR0++(1)         ;chk
*Averaging filter for Up and Vp
        CALL    AVF                          ;Call Averaging Filter Subroutine

*CONTROL LOOP NUMBER BY VALUE COUNT
*       LDI     @VALUE2,R6
*       SUBI    @INC,R6
*       STI     R6,@VALUE2
*       CMPI    0,R6
*       BZ      LWAIT
*LWAIT  BR      LWAIT


        BR      WAIT
*INTERRUPT SERVICE ROUTINE
*Output Received data on channel A and B
*       .align               ;Cache ??setting

TXDATA  PUSH    ST
        PUSH    AR0
        LDI     @ADCADR,AR0
*Output zero isi constellation pts.
*       LDI     @z_isi1,R2
*       LDI     @z_isi2,R3
*       STI     R2,*AR0
*       STI     R3,*+AR0(1)

*Prog. to generate RAMP OUTPUT
*       LDI     @VALUE3,AR6
*       ADDI    @INC3,AR6
*       STI     AR6,@VALUE3
*       STI     AR6,*AR0
*       STI     AR6,*+AR0(1)
        POP     AR0
        POP     ST
        RETI

*Data Symbol generator and storage buffer
*Subroutine to generate same data as in C50 for BER calculations
DATA    PUSH    BK                   ;length of generator and storage buffer
        PUSH    AR2                  ;seed no. pointer for generator
        PUSH    IR0                  ;disp= 28 as seed2 + 28 = seed 30
        PUSH    R0                   ;intermediate for data generation
        PUSH    AR3                  ;Pointer for storage location
        PUSH    R1
        LDI     SEED_LEN,BK          ;31 length long prbs generator + storage buffer
*Generator
        LDI     @SEED2,AR2           ;point to seed 2
        LDI     28,IR0
        LDI     0,R0
        XOR     *AR2++(IR0)%,R0,R0          ;seed 2 xor seed 30 => seed 30
        XOR     *AR2%,R0,R0
        STI     R0,*AR2++(2)%               ;store result in old seed 30
*       STI     R0,*AR0++(1)                ;chk the data
        STI     AR2,@SEED2                  ;store new seed 2 address

*Storage for D1 BERT
        LDI     @DATA_BUF,AR3  ;point to first storage
        STI     R0,*AR3++(1)%  ;store data in buffer
```

```
          STI       AR3,@DATA_BUF    ;store new address for storage
          LDI       *AR3--(12)%,R1   ;Dummy storage so as to point to
          LDI       *AR3,R1          ;last 11th input
          STI       R1,@D1_DATA      ;Store in memory for D1 comparison
          CALL      ST_DATA
          POP       R1
          POP       AR3
          POP       R0
          POP       IR0
          POP       AR2
          POP       BK
          RETS


*Storage for D2 BERT
*R1 has data
ST_DATA PUSH        BK
          PUSH      AR3
          LDI       42,BK            ;42 length long CB for storage of data
          LDI       @D_BUF,AR3       ;point to first storage of
*         STI       AR3,*AR0++(1)    ;chk
          STI       R0,*AR3++(1)%    ;store data and point to last 41st element
*         STI       R0,*AR0++(1)     ;chk
          STI       AR3,@D_BUF       ;store new address for storage
          LDI       *AR3,R1          ;store last 41st input
*         STI       R1,*AR0++(1)     ;chk
          STI       R1,@D2_DATA      ;store in memory for D2 comparison
          POP       AR3
          POP       BK
          RETS

*BER Calculations for D1
*Ignore first 11 pts. and then compare D1 with D1 Data
*increase error count by one if not matching, zero otherwise.
BER       PUSH      R1
          PUSH      R2
          PUSH      R0
*Ignore first 11 points because of delay due to BP filter
          LDI       @D1_DLY,R0       ;delay count =11
          SUBI      @one,R0          ;decrement
          STI       R0,@D1_DLY       ;new delay count
          BNN       DLY
          NEGI      R0,R1
          STI       R1,@DATA_PT      ;Total number of Data Symbols for error calculation
*Find number of errors in first decision
          LDI       @D1_ERR,R0       ;initial symbol error count
          LDI       @D1,R1           ;first decision value
*         STI       R1,*AR0++(1)
          LDI       @D1_DATA,R2      ;transmitted data value
          CMPI      R1,R2            ;compare for symbol error
          BZ        NO_ERR           ;no increment if same
          ADDI      @one,R0          ;increment error count
          LDI       @DATA_PT,R1      ;Data point in error
*         STI       R1,*AR0++(1)     ;stored in buffer for analysis
NO_ERR    STI       R0,@D1_ERR       ;store new symbol error count
DLY       POP       R0
          POP       R2
          POP       R1
          RETS
*Averaging filter subroutine for Up and Vp filtering and get
*Upf and Vpf at the output.
AVF       PUSH      BK
          PUSH      AR3
          PUSH      AR4
          PUSH      AR5
          PUSH      IR1
          LDI       AVF_TL,BK                  ;Length = 61 for averaging filter CB
          LDI       @AVU_DLE,AR5               ;Up delay line end address
```

```
        LDI     @AVF_COF,AR4                    ;Averaging filter coeff. start address
        LDI     @AVV_DLE,AR3                    ;Vp delay line end address
        LDI     30,IR1                          ;Index to point to middle of delay line
*       STI     AR5,*AR0++(1)       ;chk
        LDF     *AR5--(IR1)%,R6                 ;Dummy storage to change pointer using index
*       STI     AR5,*AR0++(1)       ;chk
        LDF     *AR3--(IR1)%,R7                 ;Dummy storage to change pointer using index
        LDF     *AR5++(IR1)%,R6     ;Change pointer back to end of delay line
        LDF     *AR3++(IR1)%,R7     ;after storing middle point i.e. Up(-31),Vp(-31)
        STF     R6,@Up_dly
        STF     R7,@Vp_dly
*       STF     R6,*AR0++(1)        ;chk
*       STF     R7,*AR0++(1)        ;chk
        STF     R4,*AR5++%                      ;Store new Up to buffer end, pt. to start for filter
        STF     R5,*AR3++%                      ;Store new Vp to buffer end, pt. to start for filter
        LDFU    0,R0
        LDFU    0,R2
        RPTS    AVF_TL-1                        ;FIR filter for Up
        MPYF3   *AR5++%,*AR4++%,R0
||      ADDF3   R0,R2,R2
        ADDF    R0,R2                           ;Upf = filter output
*       STF     R2,*AR0++(1)        ;chk
        LDFU    0,R1
        LDFU    0,R3
        RPTS    AVF_TL-1                        ;FIR filter for Vp
        MPYF3   *AR3++%,*AR4++%,R1
||      ADDF3   R1,R3,R3
        ADDF    R1,R3                           ;Vpf = filter output
*       STF     R3,*AR0++(1)        ;chk
        STI     AR3,@AVV_DLE                    ;Store new Vp storage address for next initial pt.
        STI     AR5,@AVU_DLE                    ;Store new Up storage address for next initial pt.
*       STI     AR3,*AR0++(1)       ;chk
*       STI     AR5,*AR0++(1)       ;chk
*Calculate C and D to find phase change because of channel
*Note R2=Upf, R3=Vpf, R6=Up_dly ,R7=Vp_dly
* C = Upf*Up_dly + Vpf*Vp_dly
* D = Vpf*Up_dly - Upf*Vp_dly
*       STF     R2,*AR0++(1)        ;chk
*       STF     R6,*AR0++(1)        ;chk
*       STF     R3,*AR0++(1)        ;chk
*       STF     R7,*AR0++(1)        ;chk

        MPYF3   R2,R6,R0            ;Upf*Up_dly
        MPYF3   R3,R7,R1            ;Vpf*Vp_dly
        ADDF3   R0,R1,R4            ;>> C = Upf*Up_dly + Vpf*Vp_dly

*       STF     R4,*AR0++(1)        ;chk
        MPYF3   R3,R6,R0            ;Vpf*Up_dly
        MPYF3   R2,R7,R1            ;Upf*Vp_dly
        SUBF3   R1,R0,R5            ;>> D = Vpf*Up_dly - Upf*Vp_dly
*       STF     R5,*AR0++(1)        ;chk

*Extract sign of C and D to calculate index Q
*C=R4 and D=R5
*Look-up Table for index calculation (Q)
* K=C>0  L=|C|>|D|  M=D>0           Q          Shift of D1
* ---------------------------------------------------------------------
* True       True    don't care     00         No shift
* False      True    don't care     10         180 degree shift(2 Quad a.c.)
* don't care False   True           11         -90 degree shift(3 Quad a.c.)
* don't care False   False          01         +90 degree shift(1 Quad a.c.)
        LDF     R4,R4                          ;extract sign of C
        LDIN    2,R0                           ;C<0  R0=2
        LDINN   0,R0                           ;C>=0 R0=0
        LDF     R5,R5                          ;extract sign of D
        LDIN    0,R1                           ;D<0  R1=0
        LDINN   2,R1                           ;D>=0 R1=2
        ABSF    R4,R4                          ;magnitude of C
```

```
          ABSF     R5,R5                        ;magnitude of D
          CMPF     R5,R4                        ;see |C|-|D| for relative mag.
          LDIN     1,R3                         ;|C|<|D|
          LDIN     R1,R0                        ;replace R0 by R1 (old R0 is don't care)
          LDINN    0,R3                         ;|C|>|D| R3=0
          ADDI3    R0,R3,R3          ;Index Q for second Decision
                                                ;R0+R3=Index Q
*         STI      R3,*AR0++(1)      ;chk
*         STI      R3,*AR0++(1)      ;chk
* Refer back to averaging filter delay line pointer, extract location of
* Up_dly and Vp_dly and change their value based on index Q.

          LDF      @Up_dly,R4                   ;R4=Up(old) Old value of Up (extracted channel info.)
          LDF      @Vp_dly,R5                   ;R5=Vp(old) Old value of Vp
*         STF      R4,*AR0++(1)      ;chk
*         STF      R5,*AR0++(1)      ;chk
          NEGF     R4,R6                        ;R6=-Up(old)
          NEGF     R5,R7                        ;R7=-Vp(old)
          CMPI     0,R3                         ;for Q=0 i.e. no shift
          LDFZ     R4,R0                        ;Up(new)=Up(old)
          LDFZ     R5,R1                        ;Vp(new)=Vp(old)
          CMPI     2,R3                         ;for Q=2 i.e. 180deg data shift
          LDFZ     R6,R0                        ;Up(new)=-Up(old)
          LDFZ     R7,R1                        ;Vp(new)=-Vp(old)
          CMPI     1,R[illegible]               ;for Q=1 i.e. +90deg data shift
          LDFZ     R5[illegible]                ;Up(new)=Vp(old)
          LDFZ     R[illegible]                 ;Vp(new)=-Up(old)
          CMPI     3,R3                         ;for Q=3 i.e. -90deg data shift
          LDFZ     R7,R0                        ;Up(new)=-Vp(old)
          LDFZ     R4,R1                        ;Vp(new)=Up(old)
          LDI      31,IR1                       ;Index to point to store new Up_dly
          LDF      *AR5--(IR1)%,R6              ;Dummy storage to adjust pointer to old Up_dly
          LDF      *AR3--(IR1)%,R7              ;Dummy storage to adjust pointer to old Vp_dly
*         STF      R6,*AR0++(1)      ;CHK
*         STF      R7,*AR0++(1)      ;CHK
          STF      R0,*AR5                      ;Store new Up_dly to replace old Up_dly
          STF      R1,*AR3                      ;Store new Vp_dly to replace old Vp_dly
*         STF      R0,*AR0++(1)      ;CHK
*         STF      R1,*AR0++(1)      ;CHK
*         STI      AR5,*AR0++(1)     ;CHK
*         STI      AR3,*AR0++(1)     ;CHK
          CALL     CORRECT
          CALL     DII_BER
          POP      IR1
          POP      AR5
          POP      AR4
          POP      AR3
          POP      BK
          RETS


* Store first decision previous 31 values so as to use later during
* correction

STORE_D1
          PUSH     BK
          PUSH     AR3
          LDI      DIBUF_LEN,BK      ;30 length long D1 value storage buffer
          LDI      @DI_BUF,AR3       ;point to first storage
*         STI      AR3,*AR0++(1)     ;chk
          LDI      @D1,R0
*         STI      R0,*AR0++(1)      ;chk
          STI      R0,*AR3++(1)%     ;store data in buffer
          STI      AR3,@DI_BUF       ;store new address for storage
          LDI      *AR3,R1           ;last 30th input
          STI      R1,@DI_DLY        ;Store in memory for D1 comparison
```

```
*         STI      R1,*AR0++(1)        ;chk
          POP      AR3
          POP      BK
          RETS

*Correct first decision based on index Q so as to have second decision
*Note R3 has index Q value
CORRECT PUSH     AR2
          PUSH     AR3
          PUSH     IR1
          LDI      R3,IR1                          ;Index Q for quadrant shift
*         STI      R3,*AR0++(1)        ;chk
          LDI      @DI_DLY,R4
*         STI      R4,*AR0++(1)        ;chk
          LDI      @BASE_ADR,R5
          ADDI     R4,R5,AR2           ;first decision to be shifted
*         STI      AR2,*AR0++(1)       ;chk
          LDI      *AR2,AR3                        ;Address pointer i.e. DI+Base adr. => Adr. pointer
*         STI      AR3,*AR0++(1)       ;chk
          LDI      *AR3++(IR1),R1                  ;Dummy storage to change address pointer as per Q
*         STI      AR3,*AR0++(1)       ;chk
          LDI      *AR3,R1                         ;Second decision i.e. shifted first decision
*         STI      R1,*AR0++(1)        ;chk
          STI      R1,@D2                          ;Store second decision value
          POP      IR1
          POP      AR3
          POP      AR2
          RETS


*BER Calculations for D2
*Ignore first 41 pts. and then compare D2 with D2 Data
*increase error count by one if not matching, zero otherwise.
DII_BER  PUSH     R1
          PUSH     R2
          PUSH     R0
*Ignore first 41 points because of delay due to BP filter
          LDI      @D2_DLY,R0          ;delay count =41
*         STI      R0,*AR0++(1)        ;chk
          SUBI     @one,R0             ;decrement
          STI      R0,@D2_DLY          ;new delay count
          BNN      DLY_ON
          NEGI     R0,R1
          STI      R1,@SYMB_PT         ;Total number of Data Symbols for error calculation
*         STI      R1,*AR0++(1)        ;chk
*Find number of errors in second decision
          LDI      @D2_ERR,R0          ;initial symbol error count
          LDI      @D2,R1              ;second decision value
*         STI      R1,*AR0++(1)        ;chk
          LDI      @D2_DATA,R2         ;transmitted data value
*         STI      R2,*AR0++(1)        ;chk
          CMPI     R1,R2               ;compare for symbol error
          BZ       ERR_FRE             ;no increment if same
          ADDI     @one,R0             ;increment error count
          LDI      @SYMB_PT,R1         ;Data point in error
*         STI      R1,*AR0++(1)        ;stored in buffer for analysis
ERR_FRE STI      R0,@D2_ERR          ;store new symbol error count
*         STI      R0,*AR0++(1)        ;chk
DLY_ON   POP      R0
          POP      R2
          POP      R1
          RETS

*Module to generate,add,measure noise to signal at receiver antenna
          .include "N1.ASM"
          .end
```

*TABLE FOR NOISE FACTOR (NSCALE) USED FOR VARIOUS SNR

SNR(dB)	NSCALE for fd=40Hz	NSCALE for fd=20Hz
70	1.199802e-4	0.585050e-4
60	0.379411e-3	0.185009e-3
50	1.199802e-3	0.585050e-3
40	0.379411e-2	0.185009e-2
30	1.199802e-2	0.585050e-2
20	0.379411e-1	0.185009e-1
10	1.199802e-1	0.585050e-1