DESIGN AND IMPLEMENTATION OF A PILOT SIGNAL SCANNING RECEIVER FOR CDMA PERSONAL COMMUNICATION SERVICES SYSTEMS

by

T. Keith Blankenship

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
Electrical Engineering

Approved:

May 1998
Blacksburg, Virginia

Design and Implementation of a Pilot Signal Scanning Receiver for CDMA Personal Communication Services Systems

by

T. Keith Blankenship

Committee Chairman: Theodore S. Rappaport

Electrical Engineering

Abstract

In cellular and personal communications services (PCS) systems based on code division multiple access (CDMA) a pilot signal is used on the forward link for synchronization, coherent detection, soft handoff, maintaining orthogonality between base stations, and, in the future, position location. It is critical that the percentage of power allocated to the pilot signal transmitted by each base station be fixed properly to ensure the ability of the CDMA network to support subscriber demand.

This thesis reports on the design and implementation of a prototype receiver for measuring pilot signals in CDMA PCS systems. Since the pseudonoise (PN) signal of the pilot channel is *a priori* information, the receiver searches for pilot signals by digitally correlating the received signal with this known, locally generated pilot signal. By systematically changing the phase of this locally generated pilot signal, the receiver scans the received signal to identify all possible signs of pilot signal activity. Large values of correlation indicate the presence of a pilot signal at the particular phase of the locally generated pilot signal. The receiver can also detect multipath components of the pilot signal transmitted from a given base station.

One issue associated with this receiver is its ability to keep the signal power within the dynamic range of the analog-to-digital (A/D) converter at its input. This necessitated the design of an automatic gain control (AGC) mechanism, which is digitally implemented in this receiver.

Simulation studies were undertaken to assist in the design and implementation of the pilot signal scanning receiver. These simulations were used to quantify how various non-idealities related to the radio frequency (RF) front-end and A/D converter adversely affect the ability of the digital signal processing algorithms to detect and measure pilot signals.

Because the period of the pilot signal is relatively long, methods were developed to keep the receiver's update period as small as possible without compromising its detection ability. Furthermore, the high sampling rate required strains the ability of the digital logic to produce outputs at a rate commensurate with real-time operation. This thesis presents techniques that allow the pilot signal scanning receiver to achieve real-time operation. These techniques involve the judicious use of partial correlations and windowing the received signal to decrease the transfer rate from the A/D converter to the digital signal processor. This thesis provides a comprehensive discussion of these and other issues associated with the actual hardware implementation of the pilot signal scanning receiver.

Acknowledgements

The highest praise belongs to the God the Father and His Son Jesus Christ, Engineers of the universe, whose laws we lesser engineers attempt to understand and use for the good of mankind, for granting me life and health to complete this thesis.

I thank my parents for their unselfish devotion to me throughout my life and for setting an example to me of living by one's principles. I thank my sister/roommate for her cooking and cleaning over the past one and one-half years, and my girlfriend, Yufei Wu, for her encouragement and support.

My advisor, Prof. Rappaport, gave me another chance at making good of my life and arranged for me to lead this interesting project. For this, for befriending me, and for providing technical suggestions at crucial moments throughout the course of the project, he has earned my gratitude. Profs. Reed, Tranter, and Sowerby also provided technical suggestions at various points in the project. Thanks to Dr. Bob Boyle for his meticulous reading of the thesis draft and for helping me to sort out various issues related to the sampling rate. Thanks to Profs. Brown and deWolf, members of my graduate committee, for their readings and suggestions on this thesis.

The funding for this thesis was graciously provided by Tektronix, Inc. through the administration of Dr. Mihir Ravel. He and several other Tektronix engineers, including Steve Jachim, Wayne Musik, Steve Pepper, Mike Sekora, and Ben Ward, provided invaluable suggestions to the project.

It was truly a blessing to have been associated with so many wonderful students at the Mobile and Portable Radio Research Group (MPRG) at Virginia Tech. Many of them helped me with difficulties related to electronics, computing, simulation, and system design throughout the course of this project, and I would desire that they know how much I appreciate their time and courtesy. By name, they are as follows: Paulo Cardieri, Neiyer Correal, Francis Dominique, Greg Durgin, Brian Fox, Zhong Hu, Donna Krizman, Kevin Krizman, Nitin Mangalvedhe, Scott McCulley, Rias Muhamed, Bill Newhall, Steve Nicoloso, Kim Phillips, Roger Skidmore, Steve Swanchara, and Hao Xu. The staff at MPRG, especially Aurelia Scharnhorst, is top-notch and has been a great assistance during the past few years.

Special thanks are due to Neal Patwari for his diligent work on the analog tuner portion of the pilot signal scanning receiver and for reviewing the draft of this thesis. Many thanks also go to Charles Zhou for helping me solve the noise problems I was experiencing with the high-speed digital interface between the A/D converter and the digital signal processor (DSP) board.

Torrence Robinson of Texas Instruments arranged the donation of two TMS320C541 digital signal processor evaluation modules (EVMs) and Brad Brannon of Analog Devices arranged the donation of two AD9220 A/D converter evaluation boards for implementing the pilot signal scanning receiver, and hence deserve my thanks.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Principles of Cellular Radio Networks

Figure 1.1 illustrates a simplified cellular radio network [1]. Such a network consists of many base stations which serve all mobile units in a given geographical region. The system is full duplex, which means that at any given instance in time communication can occur both on the forward link (from base station to mobile unit) and on the reverse link (from mobile unit to base station). The link between the antenna of the base station and the antenna of the mobile unit, which relies on the propagation of electromagnetic radiation, is often referred to as the mobile radio channel, and can be analyzed with methods similar to that used for analyzing other information bearing channels in electrical engineering.

Users of the cellular system access the network in one of three ways: frequency division multiple access (FDMA), time division multiple access (TDMA), or code division multiple access (CDMA). In FDMA, each user is assigned a unique pair of narrow frequency bands – one for transmission, one for reception. In TDMA, each user is not only assigned a unique pair of narrow frequency bands, but is also assigned a time slot at which transmission and reception are to occur. In CDMA, each user is assigned a unique code which is carried by the user's signal. This unique code allows the base station and mobile unit to distinguish the each other's signals from all other signals. In addition, the frequencies at which transmission and reception occur are different.

Although the multiple access scheme employed in a cellular system determines how the network is implemented, the key tradeoff common to all multiple access schemes is that of radio coverage versus interference. In FDMA and TDMA, the same frequency is allocated

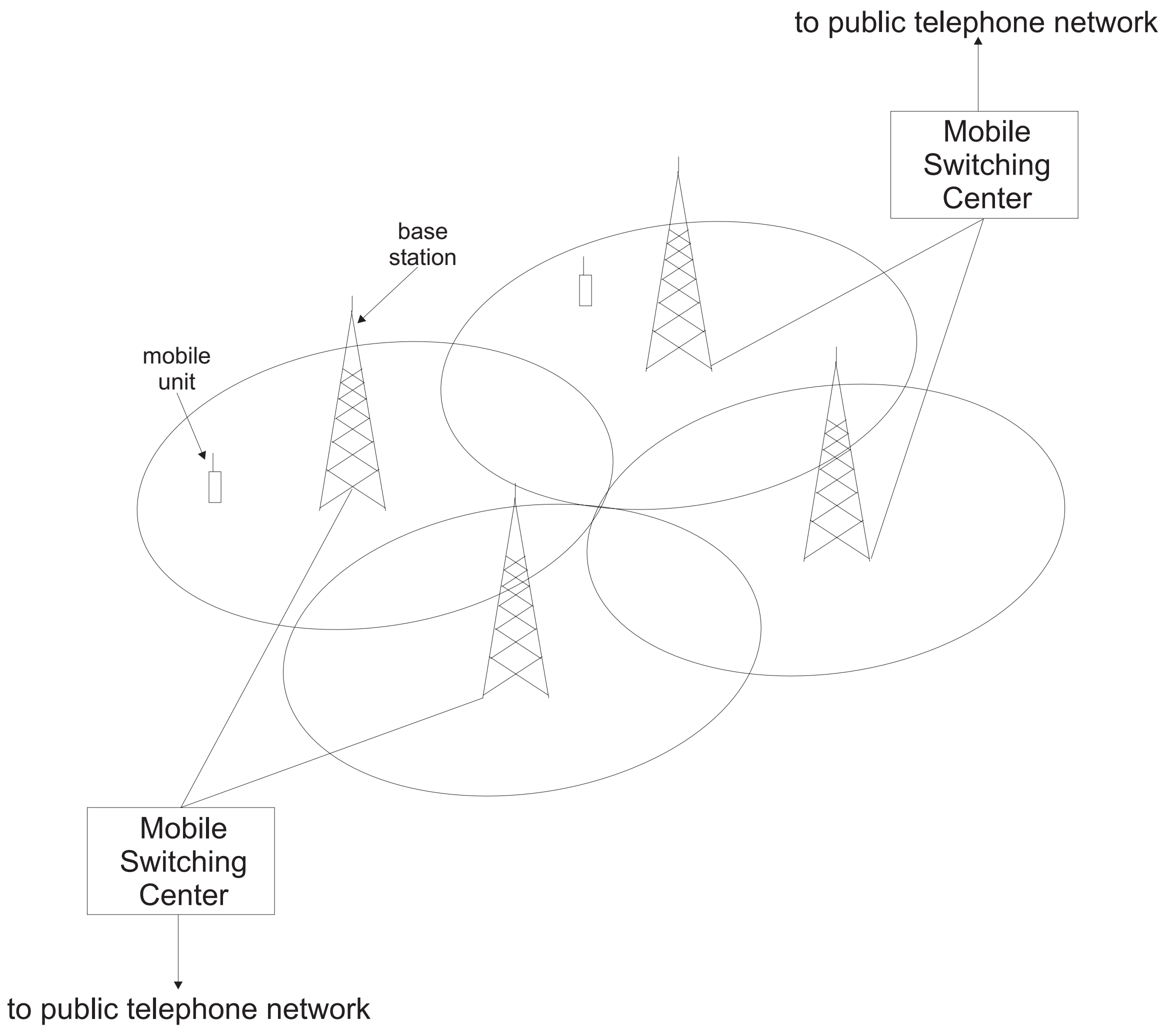

Figure 1.1: A simplified cellular radio network.

to two different users only if these users are sufficiently separated in distance. This scheme exploits the natural loss in signal power which occurs over distance to ensure that the two users do not interfere with each other.

Adjusting the power transmitted by each base station and the antenna orientations also assists in controlling the mutual interference experienced by two users of the same frequency. Ideally, we would like each base station to service a very large area, which can be accomplished by increasing the transmitted power. However, this is not feasible due to the limited nature of the radio spectrum, since user demand for channels may exceed the availability of channels. The solution adopted in cellular radio is to use a smaller transmitted power, employ a greater number of base stations, and reuse the channels at diverse geographical locations. This practice is commonly referred to "frequency reuse."

In CDMA, since all users transmit on the same frequency, there is no need to implement frequency reuse. Each base station is designed to accommodate a maximum number of users, and base stations are situated based upon expected user demand and radio coverage. Base stations can be distinguished from each other through the use of timing information. However, interference is still a concern, since any power received by a mobile unit (base station) from an undesired base station (mobile unit) increases the interference level in the receiver. Thus, it is necessary to adjust the transmitted power levels in such a way as to control these interference levels, yet provide sufficient radio coverage with as few base stations as possible.

1.2 The Prevalence of CDMA in the PCS Bands

Code division multiple access (CDMA) has become the preferred multiple access technique for commercial radio systems in the licensed broadband Personal Communication Services (PCS) frequency bands in the United States [2]. Table 1.1, which lists the number of licensees that have committed to each PCS technology for each PCS frequency block, clearly shows that CDMA is the market leader. In fact, in the A and B blocks, which were first licensed by the Federal Communications Commission (FCC) and where most licensees already have systems in operation, CDMA dominates all other PCS technologies. In Appendix A, a brief overview of the fundamentals of PCS is provided.

The popularity of CDMA in commercial cellular and PCS is a consequence of its potential for achieving high capacity systems, where capacity is the number of users per unit bandwidth which can be accommodated by the system. An information theoretic analysis of multiple

Table 1.1: The number of PCS licensees selecting various multiple access technologies in different PCS frequency blocks.

PCS Block(s)	**CDMA**	**CDMA/ GSM**	**GSM**	**GSM/ IS-661**	**TDMA**	**PACS**	**N/A**
A & B	704	11	296	33	310	4	13
C	278	22	189	0	1	23	130
D, E, & F	283	0	343	0	270	14	780

access channels which are binary symmetric with additive white Gaussian noise (AWGN) indicates that the channel capacity of CDMA, in terms of bits per second, equals or exceeds that of either frequency division multiple access (FDMA) or time division multiple access (TDMA). This is because with CDMA all subscribers use all of the bandwidth at all times. With appropriate receiver techniques, interference from other users of the channel can be subtractively cancelled – the concept behind multiuser receivers for CDMA [3, 4].

However, because of the severe frequency-selective fading due to multipath (see Chapter 3), mobile radio channels are not simply AWGN channels. As a result, the actual performance gain of CDMA in a mobile radio channel is less than those touted in simplified AWGN analyses. Moreover, complicated receiver techniques, such as the RAKE receiver[1], and system design techniques, such as power control, are required so that CDMA may operate in mobile radio channels.

These issues have lead to a lack of consensus among radio engineers concerning preferred cellular and PCS standards. Nevertheless, as shown in Table 1.1, CDMA is standing the test of time and is indeed becoming the preferred air interface for cellular and PCS providers. In fact, many standardization groups, such as IMT-2000 and UMTS, are currently scrutinizing CDMA-based air interfaces for third-generation cellular and PCS systems. [5, 6, 7, 8]

1.3 Role of the Pilot Channel in CDMA

Cellular and PCS band CDMA systems comply with the IS-95A and J-STD-008 standards, respectively. These standards dictate that a "pilot signal," which is a carrier modulated by a

[1]Special RAKE receivers are commonly employed in CDMA communications. These receivers constructively combine multipath components to increase signal-to-noise ratios in the decision-making circuits in the receiver. This type of receiver is made possible due to the inherently wide bandwidth of the CDMA signal.

high chip rate (1.2288 MHz) pulse-shaped pseudonoise (PN) sequence, be transmitted on the forward link (base station to mobile unit). Further, all mobile units have *a priori* knowledge of this periodic PN sequence. Since the PN sequence is periodic, with a period of 2^{15} chips, the pilot signal can be spoken of in terms of its phase. In this section, the multiple purposes of the pilot signal are summarized.

1.3.1 Mobile Unit Synchronization

The conjugate despreading operation required for demodulation of a CDMA signal necessitates that a CDMA receiver generate a time-synchronous (with that of the received signal) version of the spreading sequences. One major purpose of the pilot signal is to assist mobile units in acquiring this requisite time synchronization with the received signal. The power of the pilot signal is usually greater than that of any channel in the system for this reason. The mobile unit achieves synchronization with the pilot signal of the serving base station by searching for large correlations between the receiver signal and the locally generated pilot signal. These correlations may be determined using digital signal processing techniques, which are described in some detail in Chapter 5.

1.3.2 Handoff

Handoff occurs when a call is transferred between two base stations or whenever two or more base stations can support (i.e., provide a reliable link to) the same call. The latter is called soft handoff, when two base station can support the same call, or soft-soft handoff, when three base stations can support the same call. These types of handoffs are made possible by a RAKE receiver, where up to three correlators are available in the mobile unit to receive and diversity combine [1, p. 336-338] CDMA signals. Mobile units measure the received signal strength of the pilot signals of different base stations, thereby determining which base station(s) should control the call.

1.3.3 Provision of a Coherent Reference

After the pilot signal at radio or intermediate frequency is despread in the receiver, an unmodulated carrier remains. In addition to this carrier, the voltage sum of all other CDMA channels in the system appears as wideband noise. A filter can be used to extract a "clean" carrier, which can be fed forward for use in coherent demodulation of the desired channel.

1.3.4 Position Location

In the United States, regulations requiring cellular and PCS providers to locate the positions of mobile units in their respective service areas are being enforced by the FCC. Consequently, both the wireless industry and academic communities are scrambling to develop and evaluate various position location algorithms. One possible approach that is currently under investigation for CDMA systems requires that the mobile unit measure the phases of the pilot signals from different base stations to determine its location.

1.4 Issues Related to the Pilot Signal

1.4.1 Search Windows

A search window is a range of pilot signal phases over which usable multipath components from the home base station or from other base stations are sought by a mobile unit [9, p. 6-172–6-173]. Search windows are used to ensure that the mobile unit operates efficiently by searching for a pilot signal only at phases at which it should be expected. The search window size is a system parameter which must be fixed by system deployment engineers by means of analysis or measurements. If the search window is fixed too large, mobile unit power is wasted in searching for non-existent pilot signals or multipath components. If the search window is fixed too small, multipath components which could be used with success by the mobile unit's RAKE receiver for diversity combining may be missed.

1.4.2 Pilot Pollution

The tradeoff between coverage and interference is fundamental to cellular radio engineering. Coverage is maximized by making the transmitted signal power as large as possible. However, intercell and intracell interference is minimized by making the transmitted signal power as small as possible. "Pilot pollution" at some geographical location occurs when pilot signals from too many base stations are radiated into that location. This phenomenon increases interference levels and complicates the handoff procedure in that region, since the mobile unit may have difficulty determining which base station(s) should control the call. Therefore, to optimize a CDMA network for handoff performance, system engineers must take care to minimize the number of pilot signals being radiated to each location in the network.

On the other hand, if the pilot signal is to be used for position location, then a large number of distinguishable pilot signals is desirable. Clearly, more pilot signals radiating into a given location implies better performance for any position location algorithm in which mobile units determine their own location using the base station locations and the relative phase offsets of their respective pilot signals.

Thus, system deployment engineers are faced with a tradeoff. A large number of distinguishable pilot signals will facilitate position location at the expense of interference. However, a small number of distinguishable pilot signals will minimize interference and facilitate handoff at the expense of degrading the ability of the mobile units to determine their location. Ultimately, system deployment engineers must determine proper pilot signal power levels to achieve specific requirements.

1.4.3 Pilot Signal PN Offsets

All base stations in a cellular CDMA network are synchronized in time via the Global Positioning System (GPS) (see Section 2.8). This allows the phases of pilot signals at different base stations to be staggered in such a way that pilot signals from different base stations should never arrive in phase at a mobile unit. The result of this practice is that the multipath components from base stations tend to arrive in clusters, and no two clusters should ever overlap each other in phase. This scheme of PN offsets is part of the design space of a CDMA network, and is what allows a frequency reuse pattern of unity in a CDMA network [1].

1.4.4 Island Cells

Island cells occur when the base station for that cell, for various reasons, loses synchronicity with the CDMA system. Although calls may take place normally within the cell when this happens, call handoff into or out of the cell may be disrupted, and the plan of PN offsets is disturbed.

1.5 Measurements in CDMA Network Deployment

Cellular network planning and deployment typically relies on both computational modeling of the radio wave propagation and drive testing. Widely used computational modeling of the mobile radio channel in cellular system planning and deployment is limited to predicting the received power level as a function of distance from the base station. Present-day drive testing involves transmitting a signal at the site of the base station and measuring the received power level or call quality at diverse geographical locations throughout the network. Such modeling and measurements help fix network parameters, including antenna gains, antenna orientations, and transmitted power levels.

While such an approach to cellular network planning and deployment has been adequate for narrowband FDMA and TDMA systems, deployment of cellular and PCS systems using CDMA requires information on the signal timing and multipath. Ray tracing techniques [10, 11] offer hope for a computational method of determining such information. However, a topic regarding measurements in CDMA network deployment is addressed in this thesis.

Sections 1.3 and 1.4 above discussed the role of the pilot channel in the CDMA system and the necessity of fixing network parameters based on the number, amplitudes, and spread of phase delays of the pilot signals received by the mobile unit. This thesis details the design and implementation of a receiver, whose purpose is measurement of this information.

Conceptually, it is expected that this receiver should continuously scan for the existence of pilot signals at all possible received phases. Further, it is expected that a pilot signal can be detected by fixing the phase of a locally generated pilot signal (i.e., local to the receiver), and correlating this locally generated pilot signal with the received signal. A large degree of correlation would indicate the presence of a received pilot signal having the same phase as the locally generated pilot signal. Scanning of all possible received pilot signal phases would be accomplished by systematically adjusting the phase of the locally generated pilot signal. Due to the periodicity of the pilot signal, this scanning process would repeat itself after some period, which is referred to as the update period, T_{update}, in this thesis. If the output of the receiver (i.e., the degree of correlation) were to be plotted over time, a graph similar to that found in Figure 1.2 would be expected. Ideally, it would be possible to determine which base station transmitted each pilot signal detected.

A pilot signal scanning receiver, such as has just been conceptualized, would engender expectations typical of any radio receiver regarding sensitivity (the ability to detect "weak" signals) and selectivity (the ability to extract the signal of interest in the presence of strong interference). However, a pilot signal scanning receiver would also have some special requirements, which are listed as follows.

- Delay accuracy. This is the accuracy to which the received phase of a single pilot signal can be determined.
- Delay resolution. This is the minimum phase difference between two detected pilot

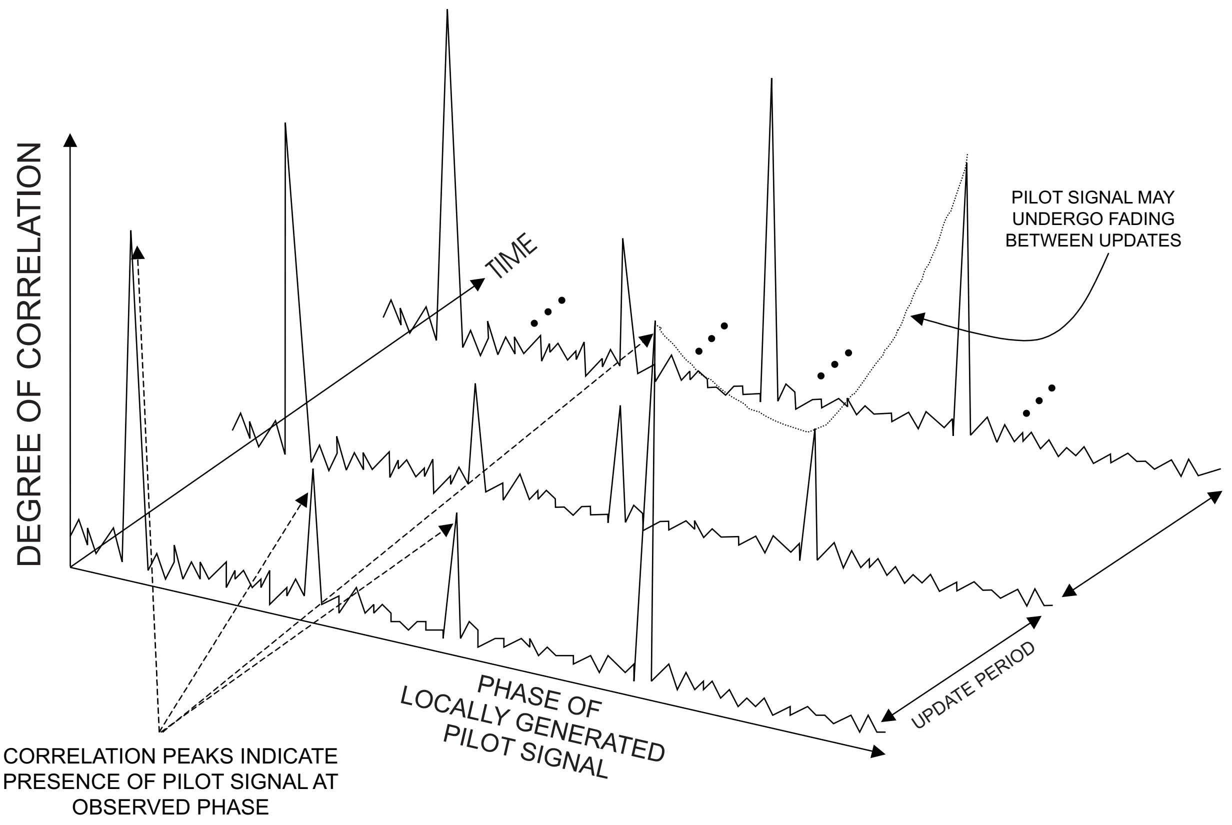
DEGREE OF CORRELATION
TIME
PILOT SIGNAL MAY UNDERGO FADING BETWEEN UPDATES
UPDATE PERIOD
PHASE OF LOCALLY GENERATED PILOT SIGNAL
CORRELATION PEAKS INDICATE PRESENCE OF PILOT SIGNAL AT OBSERVED PHASE

Figure 1.2: Conceptually, the output of the receiver should fill a plot of the degree of correlation versus phase of the locally generated pilot signal. The degree of correlation may change over time. The time required to gather information on all possible received phases is the update period, T_{update}.

signals which still allows the two pilot signals to be distinguished from each other.

- Amplitude accuracy. If a correlation peak is detected, its height can be converted into a measurement of the amplitude of the pilot signal (see Appendix E) to some degree of accuracy.
- Dynamic range. Interference from other CDMA channels in the system, partial correlation, and thermal noise induce non-trivial correlation outputs when no pilot signal at a given phase is present. The ability of the receiver to distinguish the true correlation peaks from these non-trivial correlation outputs is quantified in this thesis as the dynamic range.
- Update rate. T_{update} should be as small as possible. In fact, if the receiver is mobile, then T_{update} should be less than the radio channel coherence time [1] (see Section 3.2).
- Drift. If the radio channel is static, no drift should be observed in the phase of a pilot signal. However, clock instabilities in the receiver may induce such artificial drifts. The receiver should be designed to keep the effects of these drifts as small as possible.

1.6 Thesis Overview

The design and implementation of a pilot signal scanning receiver for PCS CDMA networks is presented in this thesis. The design of this receiver requires a precise knowledge of the CDMA forward link signal. Therefore, this thesis begins with a coverage of this topic in Chapter 2. A logical subsequent step would be to characterize the radio channel through which the signal must propagate on its way to the receiver. Thus, this topic occupies the whole of Chapter 3. These two chapters are fundamental to understanding the nature of the received signal.

In the infancy of the research presented in this thesis, it was decided that correlation would be used to detect the pilot signals, and that this correlation would be implemented in digital electronics using a digital signal processor (DSP). This is because using a DSP provides a measure of flexibility, since some of the receiver functions are programmed in software. A digital approach also provides immunity to temperature fluctuations, thereby increasing the repeatability of the measurements. Furthermore, the implementation of the receiver was expected to employ algorithms in which decisions involving several alternatives would occur. Radio receivers which use digital techniques have some special architectures and design requirements; therefore, in Chapter 4 these issues are discussed.

In Chapter 5, the digital signal processing algorithms developed for the receiver are presented. Specifically, two different algorithms for correlation and an algorithm for automatic gain control (AGC) are presented. The emphasis in Chapter 5 is on understanding the mathematical and logical operations performed in each of these algorithms.

Extensive details are provided on the physical design of the receiver in Chapter 6. In particular, the following are justified:

- overall receiver architecture,
- selection of sampling frequency,
- selection of final intermediate frequency (IF),
- selection of period for partial correlations,
- downconversion architecture, and
- gain and noise figure requirements.

A detailed study, by means of computational modeling and simulation of the signal and receiver, is made of the effects upon the receiver dynamic range of the non-idealities introduced by the use of real-world hardwares. These include the following:

- thermal noise,
- sampling phase offset,
- sampling frequency offset,
- sampling frequency jitter,
- quantization and fixed point effects.
- frequency offset, and
- local oscillator (LO) phase noise.

A presentation of the effects upon the receiver dynamic range of the interference introduced by the inclusion of multipath components and signals from multiple base stations concludes Chapter 6.

An implementation in real-world hardware of the pilot signal scanning receiver designed in this thesis was undertaken. The details of this are presented in Chapter 7. Specifically, the analog-to-digital (A/D) converter, A/D-DSP interface, and DSP hardware are discussed. To deal with data transfer and memory limitations, a novel approach to scanning the received signal for pilot signals was developed, which is discussed in Chapter 7. The actual implementation of the AGC algorithm is also detailed in Chapter 7.

This thesis presents a concept along with the design and analysis leading up to a hardware prototype which proves the feasibility of that concept. However, there are several factors which must be taken into account if the prototype is to be developed into a receiver which is truly capable of meeting the needs of system deployment engineers. These are outlined in Chapter 8. This thesis is summarized and its major contributions trumpeted in Chapter 9.

Several appendices are included which provide information or mathematical derivations so lengthy that their inclusion in the main text would disrupt the logical flow. In Appendix A, the PCS spectrum and its organization are presented. In Appendix B, a detailed explanation of the method by which the simulation of the received signal and the receiver were accomplished is given. In Appendix C, tables are given which present the values of all adjustable parameters for all simulations for which results are provided in the main body of this thesis. One of the correlation algorithms required the design of a digital bandpass filter, which is the topic of Appendix D. The method by which the correlation output is used to determine the input amplitude of the pilot signal is presented in Appendix E. The the third-order intercept and spurious free dynamic range (SFDR) are important figures-of-merit of a receiver, and are discussed in Appendix F. For reference, the simulation source code is provided for the interested in Appendix G, and the digital signal processor assembly language source code is provided in Appendix H.

Chapter 2

The CDMA Forward Link

The application of CDMA to cellular radio is embodied in two industry standards. The IS-95A standard [9] covers CDMA for the cellular frequency band, and the ANSI J-STD-008 standard [12] covers CDMA for the PCS frequency band. The two standards differ in their frequency bands and network level details. However, the physical layer signaling in the two standards, which is of interest in this thesis, is identical.

In this chapter, the details of the CDMA forward link signal relevant to the design of a pilot signal scanning receiver are provided. In particular, in this chapter the signal transmitted by a single base station is described. This signal is modeled as a sequence of complex weights. These complex weights scale a train of pulses, and the weighted pulse train modulates sinusoidal carriers in quadrature. This model is particularly useful for computational simulations of the forward link signal, which are described in Appendix B. The actual received signal is complicated by contributions from multipath and other base stations, and the methods for modeling these effects are also described in Appendix B.

2.1 The Information Sequences

A block diagram of the mathematical operations performed in generating the CDMA forward link signal is shown in Figure 2.3. The first step in generating the baseband signal, which is not illustrated in Figure 2.3, is to synthesize the human voice using a vocoder [13]. The CDMA standards do not specify which vocoding algorithms are to be used. Following the voice coding, the signal is subjected to forward link error correction encoding algorithms. The bit rate of the signal emerging from the error correction encoding algorithms is 19.2 kbps.

For the purposes of this project, it sufficed to ignore the details of the voice coding and error correction encoding operations by modeling the signal emerging from the error correction encoding as a sequence of random 1's and ⇔1's. The bit stream for the i^{th} channel is thus represented here by the sequence

$$\{d_{ik}\} \quad = \quad \{\pm 1\}\,, \; k = \ldots, \Leftrightarrow 2, \Leftrightarrow 1, 0, 1, 2, \ldots, \tag{2.1}$$

where the likelihood that $d_{ik} = 1$ equals the likelihood that $d_{ik} = \Leftrightarrow 1$.

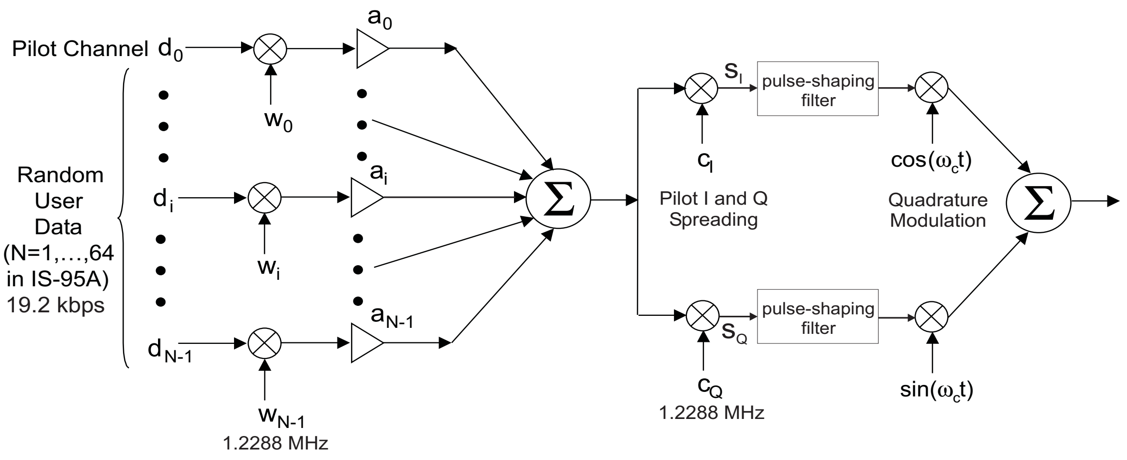

Figure 2.3: The forward link in cellular CDMA systems.

2.2 The Walsh Functions

As illustrated in Figure 2.3, each random bit stream modulates a unique Walsh function. The i^{th} Walsh function is represented here by the periodic sequence

$$\{w_{il}\}\,, \; l = \ldots, \Leftrightarrow 2, \Leftrightarrow 1, 0, 1, 2, \ldots, \tag{2.2}$$

where $w_{il} = w_{i(l+64n)}$ and n is an arbitrary integer. (Table 2.2 gives the values for one period of the Walsh functions used in the CDMA standards.)

There are a total of 64 mutually orthogonal Walsh functions, implying that

$$\frac{1}{64}\sum_{l=0}^{63} w_{il} w_{jl} = \delta_{ij}, \tag{2.3}$$

where δ_{ij} is the Kronecker delta. The Walsh functions can therefore provide "channelization" for the CDMA forward link; that is, the Walsh functions are what allow one user's signal to be distinguished from all others in the system. This channelization is perfect when all signals arrive synchronously at the receiver. However, the multipath present in a mobile radio environment destroys this perfect channelization by introducing non-synchronous signals at the receiver.

Each entity of a given Walsh function is referred to as a "chip," and the duration of each bit modulating a Walsh function is exactly one period of the Walsh function. Thus, there are 64 chips per bit. Since the bit rate is 19.2 kbps, the chip rate of the Walsh functions is thus 1.2288 MHz. Therefore, the product of the i^{th} data stream and the i^{th} Walsh function is represented here as the sequence $d_{i\lfloor k/64 \rfloor} w_{ik}, \; k = \ldots, -2, -1, 0, 1, 2 \ldots$

2.3 The Power of Individual CDMA Channels

On the CDMA forward link, each channel carries a fraction of the total transmitted power. This is accounted for in the production of a single sequence, b_k, which is a weighted summation over all active channels:

$$\{b_k\} = \sum_{i=0}^{N_{ch}-1} a_i d_{i\lfloor k/64 \rfloor} w_{ik}, \tag{2.4}$$

where a_i is the weight assigned to the i^{th} channel and N_{ch} is the number of active channels. Due to their random natures, the signals from any two channels will prove to be uncorrelated. Hence, for a signal having a power of unity, the weights a_i satisfy the criterion

$$\sum_{i=0}^{N_{ch}-1} a_i^2 = 1. \tag{2.5}$$

The IS-97A [14] and IS-98A [15] standards are companions to the IS-95A standard. These companion standards specify a set of tests to which base station and mobile unit hardware should be subjected to ensure adequate functionality in an IS-95A network. A number of measurement quantities are also defined in these companion standards. In this thesis, the measurement quantity E_c/I_0, where E_c is defined as the energy of a single chip of a certain CDMA channel and I_0 is the power spectral density of the received signal from all sources, is of interest. In a loose sense, E_c/I_0 for a certain channel can be thought of as the fraction of the total received signal power carried by that channel.

In a similar vein, the measurement quantity E_c/I_{0r} is defined as the ratio of the energy of a single chip of a certain CDMA channel, E_c, to the power spectral density of the signal

Table 2.2: The Walsh functions for cellular CDMA.

		Chip Number																																																															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1
	2	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1
	3	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1
	4	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1
	5	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1
	6	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1
	7	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1
	8	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
	9	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1
	10	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1
	11	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1
	12	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1
	13	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1
	14	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1
	15	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1
	16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	17	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1
	18	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1
	19	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1
	20	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1
W	21	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1
a	22	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1
l	23	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1
s	24	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1
h	25	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1
	26	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1
F	27	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1
u	28	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
n	29	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1
c	30	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1
t	31	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1
i	32	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
o	33	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1
n	34	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1
	35	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1
N	36	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1
u	37	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1
m	38	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1
b	39	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1
e	40	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1
r	41	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1
	42	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1
	43	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1
	44	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1
	45	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1
	46	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1
	47	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1
	48	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	49	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1
	50	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1
	51	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1
	52	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1
	53	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1
	54	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1
	55	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1
	56	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
	57	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1
	58	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1
	59	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1
	60	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1
	61	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1
	62	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1
	63	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1

transmitted by a single base station, I_{0r}. The following relationship must thus be satisfied:

$$\left.\frac{E_c}{I_{0r}}\right|_{pilot} + \left.\frac{E_c}{I_{0r}}\right|_{sync} + \left.\frac{E_c}{I_{0r}}\right|_{paging} + \left.\frac{E_c}{I_{0r}}\right|_{traffic} = 1. \tag{2.6}$$

(The channel types are discussed below in Section 2.7.) E_c/I_{0r} for a certain channel can be thought of as the fraction of the total transmitted signal power carried by that channel.

Thus, E_c/I_{0r} for the i^{th} channel, $E_c/I_{0r}|_i$, can be related to the set of weights $\{a_i\}$, according to

$$\left.\frac{E_c}{I_{0r}}\right|_i = \frac{a_i^2}{\sum\limits_{i=0}^{N_{ch}-1} a_i^2}. \tag{2.7}$$

From this discussion, it can be deduced that I_0 can be related to I_{0r} of the i^{th} base station, I_{0r}^i, according to

$$I_0 = \sum_{i=1}^{B} \widehat{I_{0r}^i}, \tag{2.8}$$

where $\widehat{I_{0r}^i}$ is an attenuated version of I_{0r} to account for propagation losses and B is the number of base stations contributing to the received signal. Equation 2.8 neglects the signal power contributed by multipath.

2.4 The Pilot PN Sequences

Following the weighted summation of all channels, the complex-valued sequence $\{s_{Ik} + js_{Qk}\}$ is produced by multiplication of the sequence $\{b_k\}$ with the pilot PN sequences $\{c_{Ik}\}$ and $\{c_{Qk}\}$

$$\{s_{Ik} + js_{Qk}\} \quad = \quad \{b_k c_{Ik} + j b_k c_{Qk}\}\,. \tag{2.9}$$

The pilot PN sequences $\{c_{Ik}\}$ and $\{c_{Qk}\}$ are derived from the binary sequences $i(n) \in \{0,1\}$ and $q(n) \in \{0,1\}$ which satisfy the following linear recursion relations:

$$\begin{aligned} i(n) \quad = \quad & i(n \Leftrightarrow 15) \oplus i(n \Leftrightarrow 10) \oplus i(n \Leftrightarrow 8) \oplus i(n \Leftrightarrow 7) \oplus \\ & i(n \Leftrightarrow 6) \oplus i(n \Leftrightarrow 2), \end{aligned} \tag{2.10}$$

and

$$\begin{aligned} q(n) \quad = \quad & q(n \Leftrightarrow 15) \oplus q(n \Leftrightarrow 12) \oplus q(n \Leftrightarrow 11) \oplus q(n \Leftrightarrow 10) \oplus q(n \Leftrightarrow 9) \oplus \\ & q(n \Leftrightarrow 5) \oplus q(n \Leftrightarrow 4) \oplus q(n \Leftrightarrow 3), \end{aligned} \tag{2.11}$$

where the symbol $\oplus$ represents the exclusive OR operation. These sequences are m-sequences [16, pp. 58–64], and have a repetition period of $2^{15} \Leftrightarrow 1$. Thus $i(n) = i[n + m(2^{15} \Leftrightarrow 1)]$ and $q(n) = q[n + m(2^{15} \Leftrightarrow 1)]$, where m is any integer.

Since $i(n)$ and $q(n)$ are m-sequences, they each have one run of 14 consecutive 0's [16, pp. 60–61]. The pilot PN sequences $\{c_{Ik}\}$ and $\{c_{Qk}\}$ are obtained from the sequences $i(n)$ and $q(n)$ by inserting an extra 0 after this run of 14 consecutive 0's. Thus, each pilot PN sequence has exactly one run of 15 consecutive 0's and one run of 15 consecutive 1's.

The extra 0 is inserted so that there are an integer number of periods of the Walsh functions for every period of the pilot PN sequence. (Unfortunately, insertion of the extra 0 ruins the ideal autocorrelation properties of the sequences $i(n)$ and $q(n)$.) Like the Walsh functions, each entity of the pilot PN sequences is referred to as a "chip," and the chip rate of the pilot PN sequences is 1.2288 MHz. Thus, since the period of the Walsh functions is 64 chips, there are exactly 512 periods of a Walsh function for every period of the pilot PN sequence.

The pilot PN sequences are mapped into bipolar voltage sequences by making the substitutions $0 \rightarrow 1$ and $1 \rightarrow \Leftrightarrow 1$. The pilot PN sequences $c_{Ik} = c_{I(k+2^{15}n)}$ and $c_{Qk} = c_{Q(k+2^{15}n)}$, where n is an arbitrary integer, result.

2.5 Baseband Filtering

The complex sequence $\{s_{Ik} + js_{Qk}\}$ is pulse-shaped using a 48-tap finite impulse response (FIR) pulse shaping filter. The coefficients for this filter are provided in the CDMA standards, and are tabulated in Table 2.3 for reference. The delay between the taps of this filter is one-quarter of a chip period. Plots of the impulse and frequency responses of the pulse shaping filter are provided in Figures 2.4 and 2.5, respectively. In a practical CDMA transmitter, the output of the pulse shaping filter would be passed through a digital-to-analog (D/A) converter, followed by an analog lowpass filter, to produce an analog signal for transmission.

The pulse shaping filter is not a root raised cosine rolloff filter, and it does not produce nontrivial inter-symbol interference (ISI) at the centers of the pulses. The filter is used for at least two reasons. First, the filter restricts the baseband bandwidth of the signal to 615 kHz, as can be seen in Figure 2.5. Hence, the RF bandwidth of the signal is 1.23 MHz. One of the early requirements of cellular CDMA was that the standard be designed so that the system could be phased in incrementally. It was reasoned that, since 1.23 MHz occupies exactly

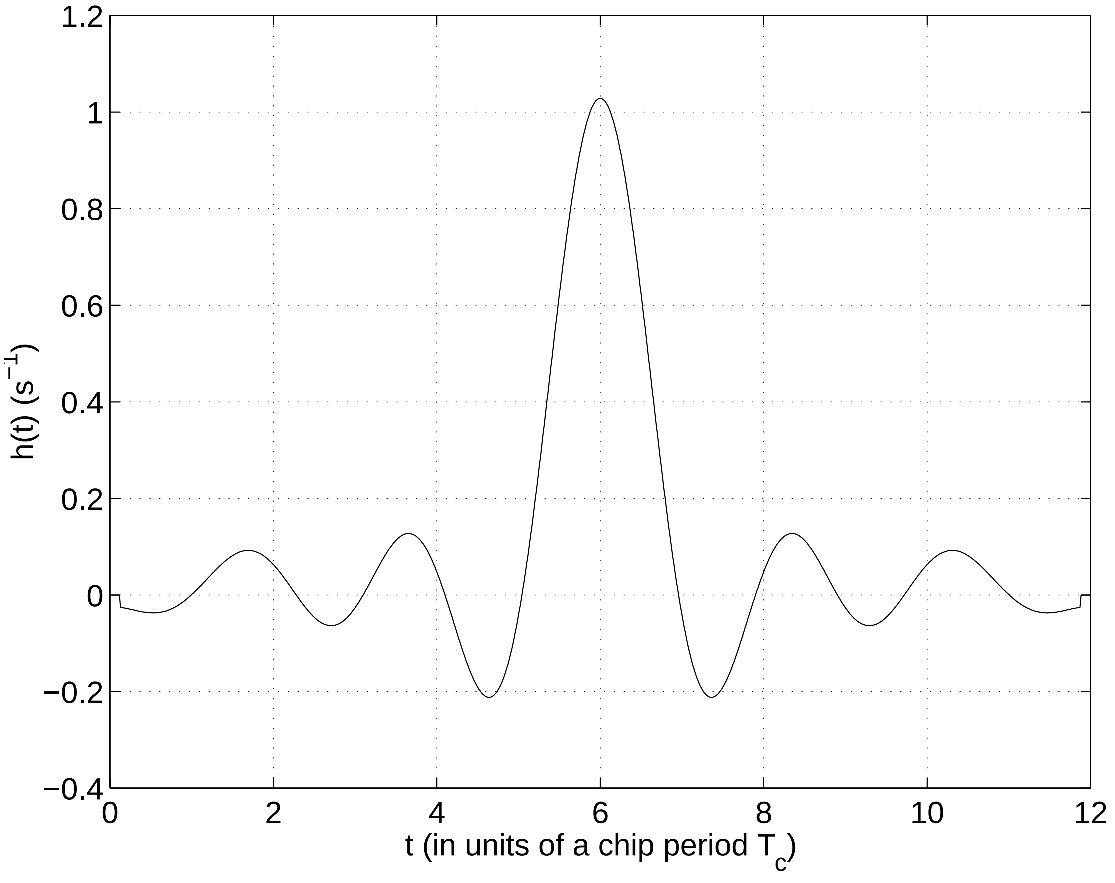

Figure 2.4: The impulse response of the CDMA pulse shaping filter.

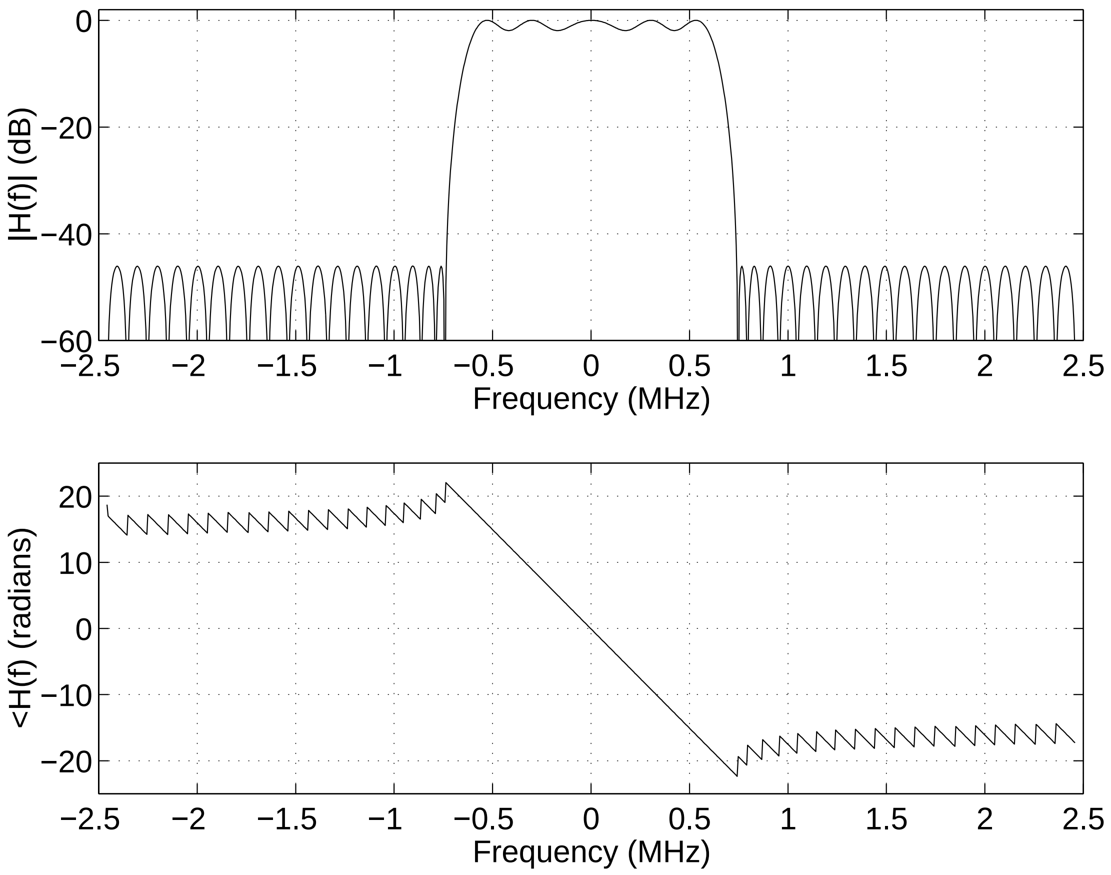

Figure 2.5: The amplitude and phase of the frequency response of the CDMA pulse shaping filter. These were obtained by computing the discrete Fourier transform of the zero-padded (up to 1024 samples) impulse response.

Table 2.3: The coefficients of the 48-tap FIR pulse shaping filter, as specified in the IS-95A standard.

Tap Number n	Tap Value $h(n)$	Tap Number n	Tap Value $h(n)$
0,47	⇔0.025288315	12,35	0.007874526
1,46	⇔0.034167931	13,34	0.084368728
2,45	⇔0.035752323	14,33	0.126869306
3,44	⇔0.016733702	15,32	0.094528345
4,43	0.021602514	16,31	⇔0.012839661
5,42	0.064938487	17,30	⇔0.143477028
6,41	0.091002137	18,29	⇔0.211829088
7,40	0.081894974	19,28	⇔0.140513128
8,39	0.037071157	20,27	0.094601918
9,38	⇔0.021998074	21,26	0.441387140
10,37	⇔0.060716277	22,25	0.785875640
11,36	⇔0.051178658	23,24	1.000000000

41 channels of bandwidth 30 kHz (the channel bandwidth of the original AMPS system) [1, Chapter 1], the transition to cellular CDMA would thereby be facilitated.

Second, in [17, p. 182], the U.S. FCC rules stipulate that any emissions outside a service provider's frequency block must be attenuated by $43 + 10\log(P)$ dB, where P is the power in Watts of the unmodulated carrier. The IS-95A pulse shaping filter helps to provide compliance with this regulation. In Figure 2.5, it can be seen that the filter stopband provides about 46 dB of attenuation. In addition, the J-STD-008 standard provides for guard bands at the edge of each frequency block to ensure compliance with the regulation above.

Notice that when the number of active CDMA channels is large, the spectrum of $\{s_{Ik} + js_{Qk}\}$ should be white. Therefore, it is expected that the spectrum of the pulse-shaped baseband signal will assume a shape similar to that of the pulse shaping filter itself.

2.6 Modulation

The continuous pulse-shaped baseband complex envelope, $\hat{s}\,(t)$, can be represented as

$$\hat{s}\,(t) \;=\; \sum_{k=-\infty}^{\infty} (s_{Ik} + js_{Qk})p(t \Leftrightarrow kT_c), \tag{2.12}$$

where $p(t)$ is the functional representation of the analog IS-95A pulse (i.e., after D/A conversion and lowpass filtering) and T_c is the chip period. Quadrature carriers are amplitude modulated by this baseband complex envelope, producing a composite transmitted signal, $s(t)$, as follows:

$$\begin{aligned} s(t) &= \Re\left\{\hat{s}(t)e^{-j\omega_c t}\right\} && (2.13)\\ &= \sum_{k=-\infty}^{\infty}\left[s_{Ik}p(t \Leftrightarrow kT_c)\cos(2\pi f_c t) + s_{Qk}p(t \Leftrightarrow kT_c)\sin(2\pi f_c t)\right], && (2.14)\end{aligned}$$

where f_c is the carrier frequency and $\Re\{\bullet\}$ denotes the real part of the argument $\{\bullet\}$.

The CDMA signal is sometimes mistakenly believed to employ quadrature phase shift keying (QPSK) modulation. However, in classical QPSK the information stream is split, with half transmitted over the I channel and half transmitted over the Q channel. In the forward link signal described in this chapter, however, all the information is transmitted over both channels. Therefore, the signal is not truly a QPSK signal, but rather binary phase shift keying (BPSK) on quadrature channels. Transmitting all the data over both channels is a diversity scheme, since the decision-making circuits in the receiver have the option of using the information from two demodulated information streams rather than one.

2.7 The CDMA Channels

2.7.1 The Pilot Channel

In the CDMA standards, the pilot channel is assigned Walsh function number zero and given an input bit stream of $d_{0k} = 1 \ \forall\ k$. Hence, the pilot channel matches the description given in Chapter 1 as a carrier modulated by a high chip rate pulse-shaped PN sequence.

Equation 2.12 can be manipulated to exhibit the pilot signals explicitly, giving

$$\hat{s}(t) = \left[a_0 P_I(t) + u_I(t)\right] + j\left[a_0 P_Q(t) + u_Q(t)\right], \qquad (2.15)$$

where

$$P_I(t) = \sum_{i=-\infty}^{\infty} c_{Ii}\,p\left(t \Leftrightarrow iT_c\right) \qquad (2.16)$$

and

$$P_Q(t) = \sum_{i=-\infty}^{\infty} c_{Qi}\,p\left(t \Leftrightarrow iT_c\right) \qquad (2.17)$$

are the I- and Q-channel pilot signals, respectively, and a_0 is the amplitude of the pilot signal. The signals $u_I(t)$ and $u_Q(t)$ depend upon the linear combination of all other channels in use by the base station. For the purposes of this thesis, Equation 2.15 is a useful representation of the baseband signal. The pilot signal scanning receiver is designed to detect the signals $P_I(t)$ and $P_Q(t)$, and the signals $u_I(t)$ and $u_Q(t)$ act as sources of interference to this detection process.

2.7.2 The Sync Channel

Time and frame synchronization information are provided to the mobile units on the sync channel. In particular, the base station transmits its pilot PN offset on the sync channel. To prevent transmission errors, convolutional encoding, symbol repetition, and block interleaving are used on the sync channel. The sync channel is assigned Walsh function number 32 (see Table 2.2).

2.7.3 The Paging Channel

Prior to assignment to a traffic channel, control information is delivered to a mobile unit on the paging channel. When a traffic channel number is assigned, this number is passed to the mobile unit over the paging channel. The paging channel is also integral to the handoff procedure in CDMA. In particular, the search window and pilot PN offset information of neighboring cells are passed to the mobile unit over the paging channel. In addition to identical error correction techniques as the sync channel, the pilot channel undergoes an additional spreading by another PN sequence known as the "long code." The purpose of this additional spreading is for data scrambling. There are a total of seven paging channels, assuming Walsh functions numbered one through seven (see Table 2.2). However, the primary paging channel is assigned Walsh function number one.

2.7.4 The Traffic Channels

Voice signals are transmitted over the traffic channel. Since there are one pilot channel, one sync channel, and up to seven paging channels out of a total of 64 possible, there are 55 traffic channels. The same types of error correction techniques employed on the sync and paging channels, in addition to data scrambling using long code spreading, are used on the traffic channels. The traffic channels are assigned Walsh functions numbered eight to 31 and 33 to 63.

2.8 System Synchronization and PN Offsets

It is mandated in the CDMA standards that all base stations maintain accurate time standards and which assume that January 6, 1980 at 00:00:00 hours UTC (Universal Coordinated Time) is time zero. It is presumed that, at this precise instant, the pilot PN sequences of a base station with offset zero made the transition to one from the final zero of the run of 15 consecutive zeros in the pilot PN sequences.

Following this definition of system time, the phase of the PN sequences at each base station is offset by some multiple of 64 chips. This is done so that pilot signals from different base stations will never arrive in phase at a mobile unit, thus preventing interference. Practically, this scheme means that the multipath components from each base station are clustered in phase. PN offset planning is a system design problem, the solution of which is unique to each system.

Chapter 3

The Mobile Radio Channel

In this chapter, details of the mobile radio channel which were utilized in the design of the pilot signal scanning receiver are presented. The effects of the mobile radio channel are categorized as either "large scale" or "small scale," both of which are discussed in this chapter. The focus of the large scale fading section is a discussion of the phenomenon at 1900 MHz.

3.1 Large Scale Path Loss Models for Propagation at 1900 MHz

Path loss is the apparent decrease in power experienced by a signal as it traverses the space between the transmitter and receiver. For the case of mobile radio, a widely used formula for the average path loss in dB, $\overline{\mathrm{PL}}(d)$, over a transmitter-receiver (T-R) separation d is

$$\overline{\mathrm{PL}}(d) \quad = \quad \overline{\mathrm{PL}}(d_0) + 10n\log_{10}\left(\frac{d}{d_0}\right), \tag{3.18}$$

where $\overline{\mathrm{PL}}(d_0)$ is the average path loss in dB at some reference T-R separation d_0 and n is the "path loss exponent." [1] In the absence of actual measurements of $\overline{\mathrm{PL}}(d_0)$, the free space path loss in dB over T-R separation d_0 may be used:

$$\overline{\mathrm{PL}}(d_0) = 20\log_{10}\left(\frac{4\pi d_0}{\lambda}\right), \tag{3.19}$$

where λ is the wavelength of the carrier.

The path loss in dB at a T-R separation of d is typically distributed according to a Gaussian distribution about the average value $\overline{\mathrm{PL}}(d)$. This phenomenon is often referred to as log-normal fading [1].

For this path loss model, the parameter n completely characterizes the path loss in a given environment, and is generally determined by statistical linear regression. Note that $n = 2$ for propagation through free space. However, for mobile radio propagation, n will be different from two, in general.

In [18], [19], and [20], the model of Equation 3.18 was used with $d_0 = 300$ m along with measured values of $\overline{\mathrm{PL}}(d_0)$. The results found in Table 3.4 were reported. Equation 3.18 was also used in [21] to model path loss in cases where the line-of-sight (LOS) between the transmitter and receiver is obstructed. Values of n between 2.6 and 2.7 were reported in this work for $d_0 = 1$ m and $\overline{\mathrm{PL}}(d_0) = 38$ dB. Strictly speaking, the results are valid for T-R separations of less than 2 km.

Table 3.4: Path loss parameters for the model used in [18], [19], and [20]. A reference distance of $d_0 = 300$ m was used. The model is strictly valid for T-R separations of 1 km or less. X_σ represents the spread of the measurements about the mean value.

Source	n	X_σ **(dB)**	$\overline{\mathbf{PL}}(d_0)$ **(dB)**
[18]	4.8	11.4	85.0
[19]	5.0	6.9	83.3
[20]	3.8–4.1	not given	76.3–80.7

A slightly more complicated model is also used in [21] to model path loss in cases where the LOS between the transmitter and receiver is not obstructed. This model predicts the average path loss for this case at T-R separation d, PL(d), as follows:

$$\overline{\mathrm{PL}}(d) = \begin{cases} (10n_1)\log_{10}(d) + p_1, & 1\ \mathrm{m} < d < d_f \\ (10n_2)\log_{10}(d/d_f) + (10n_1)\log_{10}(d_f) + p_1, & d > d_f, \end{cases} \tag{3.20}$$

where n_1 and n_2 are experimental parameters, p_1 is the free space path loss at 1 m,

$$p_1 = 20\log_{10}\left(\frac{4\pi d}{300/1900}\right) = 38.0\ \mathrm{dB}, \tag{3.21}$$

and d_f is the Fresnel zone distance [21, p. 490], [1, p. 91–94]. The values found in Table 3.5 were reported.

Table 3.5: Path loss parameters for the model found in [21]. The model is strictly valid for unobstructed LOS cases with T-R separations of 2 km or less.

Antenna Height (m)	n_1	n_2	d_f **(m)**	X_σ **(dB)**
3.7	2.18	3.29	159	8.76
8.5	2.17	3.36	366	7.88
13.3	2.07	4.16	573	8.77

With a value for the path loss, the power at the input of the receiver (i.e., at the receiving antenna terminals) in dBm, P_r, can be computed using the classical link budget equation:

$$P_r \quad = \quad G_t + G_r + P_t \Leftrightarrow \overline{\mathrm{PL}}(d), \tag{3.22}$$

where G_t is the gain in dB of the transmitting antenna, G_r is the gain in dB of the receiving antenna, and P_t is the output power of the transmitter (i.e., at the transmitting antenna terminals) in dBm.

3.2 Small Scale Fading Model

Path loss, as described above, measures the apparent decrease in average power as the T-R separation changes on the order of tens, hundreds, and thousands of meters. Small scale models, however, are concerned with channel variations as the T-R separation changes on the the order of a few wavelengths of the carrier frequency.

The key aspect to describing small-scale phenomenon is "multipath" – a term descriptive of the reception of multiple delayed, attenuated, and phase-shifted versions of a single transmitted signal. An intuitive, physically appealing, and analytically tractable method of dealing with this phenomenon is to assign the mobile radio channel a tapped-delay line baseband channel impulse response [1]:

$$h(t,\tau) \quad = \quad \sum_{i=1}^{N(t)} \alpha_i(t)\delta(\tau \Leftrightarrow \tau_i(t))e^{j\theta_i(t)}, \tag{3.23}$$

where $N(t)$ is the number of multipath components present at time t, $\alpha_i(t)$ is the amplitude at time t of the i^{th} multipath component, $\tau_i(t)$ is the delay at time t of the i^{th} multipath component, $\theta_i(t)$ is the phase shift at time t suffered by the carrier of the i^{th} multipath component, and $\delta(\bullet)$ is the Dirac delta function. Notice that in Equation 3.23, the variable

τ is used to index the change in time of the input signal, whereas the variable t is used to index the change in time of the channel.

To exercize Equation 3.23, consider two extreme examples. First, suppose of signal with a constant baseband complex envelope (an impulse in the frequency domain) is transmitted through the channel, or $x(\tau) = K$, where K is a constant. Then the channel output, $y(t, \tau)$, is

$$y(t,\tau) = x(\tau) * h(t,\tau) = K \sum_{i=1}^{N} \alpha_i(t) e^{j\theta_i(t)}, \tag{3.24}$$

so the phases of the carriers interfere with each other. Since the phases $\theta_i(t)$ are time-varying, $|y(t)|$ is time-varying as well. If N is large, and $\alpha_i \approx \alpha_j$, $\forall\ i, j$, then $|y(t)|$ is distributed according to the Rayleigh distribution. However, if $\alpha_p \gg \alpha_i$, $\forall\ i \neq p$, then $|y(t)|$ is distributed according to the Ricean distribution. For these cases, the output of the channel is a replica of the input, with the exception that the power of the output signal varies with time.

Second, suppose an impulse (a flat spectrum in the frequency domain) is transmitted through the channel, or $x(\tau) = K\delta(\tau)$, where K is a constant. Then the channel output, $y(t, \tau)$, is

$$y(t,\tau) = h(t,\tau) = K \sum_{i=1}^{N} \alpha_i(t) \delta\left(\tau \Leftrightarrow \tau_i(t)\right) e^{j\theta_i(t)}. \tag{3.25}$$

Whereas the spectrum of the input signal was flat, at the output of the channel it contains nulls at the same frequencies at which nulls occur in the channel frequency response. This observation forms the basis of practical mobile radio channel sounding, where a very high bandwidth signal is used to probe the channel, causing the spectrum of the output signal to assume the characteristics of the channel frequency response [1, p. 153–159], [22, p. 224–225].

The bandwidth of a real-world communications signal lies between these two extremes. What the two extreme examples above show is that it is important to consider the "spread" of the delays of the multipath components, σ_τ, in relation to the duration of a modulation symbol, T_{sym}. When $\sigma_\tau \ll T_{sym}$, the complex envelopes of the multipath components add together coherently, whereas the carriers experience time-varying constructive and destructive interference. As a result, the signal power experiences extreme variations. In a frequency domain description, since $\sigma_\tau \ll T_{sym}$, the nulls in the frequency response of the channel will lie well outside the bandwidth of the signal. The channel appears to be flat over the bandwidth of the signal, and the signal is thus said to experience "flat fading."

However, when $\sigma_\tau \gg T_{sym}$, the multipath components produce intersymbol interference (ISI) in the receiver [23]. If the input signal has a small autocorrelation, $R_{xx}(\tau)$ for $\tau \neq 0$, then the multipath components can be distinguished and coherently combined in the receiver. This is the concept underlying the RAKE receiver, and is an advantage to using high bandwidth signaling in mobile radio. In a frequency domain description, since $\sigma_\tau \gg T_{sym}$, the nulls in the frequency response of the channel will lie well inside the bandwidth of the signal. Therefore, in this case, the signal is said to experience "frequency-selective fading."

An alternate means of describing the channel is to consider the correlation of the amplitudes of two tones separated in frequency by Δf. As Δf grows from 0 Hz, the amplitude correlation decreases due to the frequency-selective fading phenomenon. The largest value of Δf for which the amplitude correlation exceeds some threshold (0.5 or 0.9 are typical values of this threshold) is referred to as the "coherence bandwidth" of the channel.

Equation 3.23 shows that the channel impulse response is time-varying. The channel coherence time, T_0, is the time over which the channel can be assumed to be static [1], [24, p. 144]. A good rule-of-thumb is that T_0 is related to the maximum Doppler shift, f_d, through the relation $T_0 = 1/f_d$ [1], [24, p. 144]. If $T_0 > T_{sym}$, then "slow fading" is said to occur. However, if $T_0 < T_{sym}$ then "fast fading" is said to occur. Most cases in terrestrial mobile radio exhibit slow fading.

Chapter 4

Digital Receiver Architectures

The term "digital receiver" implies that some parts of the receiver are implemented with digital circuits or microprocessing hardware. This, in turn, implies that the signal must be sampled, which is usually accomplished with a sample-and-hold amplifier, possibly followed by an analog-to-digital (A/D) converter to digitize the signal.

The main advantages of using a digital receiver are the ease of reconfigurability using software and the insensitivity to changes in temperature or other environmental factors. The main disadvantages to using a digital receiver are the limitation in processing bandwidth with current digital technology and the (sometimes) excessive power consumption.

As mentioned in Chapter 1, a premise of this thesis is that the pilot signal scanning receiver is to be implemented digitally. This chapter presents a study of digital receiver architectures, with a focus on the trade-offs encountered in designing with a particular digital receiver architecture, and concludes with two case studies of digital receiver architectures.

4.1 Architecture Alternatives

Many ways exist to construct a digital receiver. This section summarizes the radio receiver literature by presenting four features which characterize any particular digital receiver architecture. These are the following:

1) number of intermediate frequencies (IFs),
2) downconversion method,
3) location of the A/D converter in the receiver chain, and
4) relationship of the sampling frequency to the center frequency of the signal.

A study of this sort also provides a starting point for the selection of a basic receiver architecture when undertaking a new design.

4.1.1 Number of IFs

The circuits in the receiver which extract the baseband information may not function at the frequencies at which transmission occurs. If this is the case, then the receiver must translate the signal to a lower frequency band at which the circuits will function so that the baseband information can be extracted.

Translation of the signal to a lower frequency band can take place over several stages. A receiver in which this occurs possesses multiple IFs. For instance, a receiver may translate a received signal with a center frequency of 2 GHz first to 200 MHz, and then to 20 MHz, at which frequency the baseband information is extracted. In this case, the first IF is 200 MHz, and the second IF is 20 MHz.

One advantage of having multiple IFs is for gain distribution. If a large gain is required prior to the circuits which extract the baseband information, then it is best that this gain be distributed over several IFs. With a large gain at a single frequency, practical amplifiers are prone to oscillate, since small amounts of feedback are inevitable. It is usually better to reduce the gain at each frequency and distribute the gain over several frequencies to reduce the tendency of the amplifiers to oscillate.

Making filtering easier is another advantage of having multiple IFs. Filtering is necessary to maximize the selectivity of the receiver. High frequency filters with narrow bandwidths are notoriously difficult to construct. One solution is to translate the signal to a lower frequency band where the bandwidth required is a larger fraction of the center frequency and hence filters are easier to construct.

Also, with a greater number of IFs, the frequency difference between adjacent IFs can be made smaller. This admits the use of wider (therefore, more easily constructed) image-reject filters. This can understood by recalling that the IF, f_{IF}, is the difference between the LO frequency, f_{LO}, and the center frequency of the signal, f_c [23]:

$$f_{IF} = \begin{cases} f_c \Leftrightarrow f_{LO} & \text{for low side injection} \\ f_{LO} \Leftrightarrow f_c & \text{for high side injection.} \end{cases} \tag{4.26}$$

The image frequency, f_{IM}, lies at

$$f_{IM} = \begin{cases} f_{LO} \Leftrightarrow f_{IF} = f_c \Leftrightarrow 2f_{IF} & \text{for low side injection} \\ f_{LO} + f_{IF} = f_c + 2f_{IF} & \text{for high side injection.} \end{cases} \tag{4.27}$$

These equations show that the image frequency lies farther from the desired frequency for high values of f_{IF}.

One disadvantage of having multiple IFs is that the cost of the receiver is higher due to the increased number of parts. The increased number of parts also results in higher power consumption in the receiver, which is undesirable in the case of a mobile unit.

A receiver which translates the received signal to baseband without going through any IF stages is referred to as a homodyne receiver. On the other hand, a receiver which translates the received signal to a lower frequency band which does go through one or more IF stages is referred to as a heterodyne receiver.

The homodyne receiver is adversely affected by DC offsets [25, p. 131]. DC offsets occur when an accidental signal path exists between the local oscillator port and RF input port of a mixer. DC offsets can also occur when a strong interference signal couples to the mixer both through the RF port and through to the LO port. Heterodyne receivers can deal with DC offsets better than homodyne receivers because the DC offset can be easily filtered out at the first IF without any loss of information. A homodyne receiver which filters out DC offsets must do so at the expense of the loss of information.

4.1.2 Downconversion Method

Several methods for translation of the signal between frequency bands exist. In direct downconversion, a single mixer is used to effect a multiplication between the signal and a single sinusoid. Subsequent filtering is used to remove interference while retaining the signal of interest. If no filtering is performed, then this is termed "block downconversion."

In digital cellular communications, quadrature modulation is used to achieve bandwidth efficiency[1]. In direct downconversion, quadrature components cannot be distinguished. Quadrature downconversion, as illustrated in Figure 4.6, is used to separate the quadrature components using analog circuits. In this downconversion method, the signal power is divided, and two mixers whose local oscillator inputs are 90° out of phase are used to simultaneously frequency-translate the signal and extract the quadrature components.

There are at least two disadvantages to using quadrature downconversion. First, to avoid distortion of the signal, the overall gain and phase shift across the I arm must be exactly matched to that of the Q arm. The local oscillator inputs to the mixers on the I and Q

[1]As discussed in Section 2.6, in the cellular and PCS CDMA standards, quadrature modulation is not used for bandwidth efficiency, but rather for diversity.

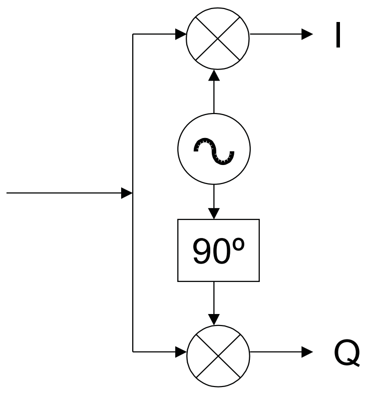

Figure 4.6: Block diagram of the quadrature downconversion method.

arms must also be exactly 90° out of phase. These requirements are difficult to meet in practice. Second, since two of every component are required, quadrature downconversion can be relatively costly.

Usually, image-reject filters are placed just prior to the mixer in a receiver. If it is necessary to select a very low IF, then a very narrow image-reject filter is required. Depending on the frequency range of interest, the bandwidth of the filter can be prohibitively narrow. In such cases, an alternative downconversion method, referred to as image-reject downconversion, is an option.

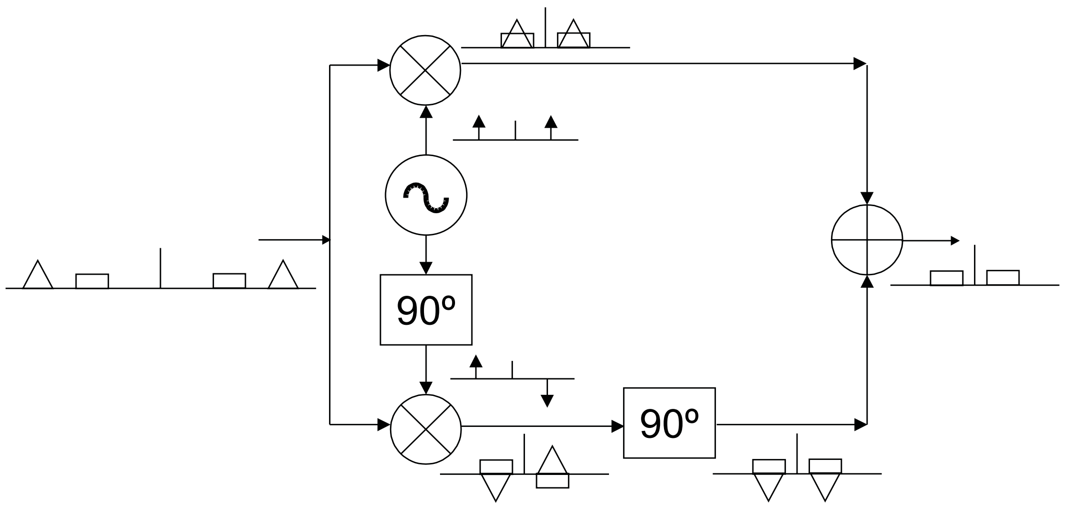

Figure 4.7: Block diagram of the image-reject downconversion method.

Figure 4.7 illustrates the operations involved in the image-reject downconversion. Quadrature downconversion first produces I and Q components with different phase spectra. An additional 90° phase shift in one of the arms is used to alter the spectrum in such a way that at the summing node the undesired image is eliminated.

Image-reject downconversion alleviates the need for an image-reject filter. However, like the quadrature downconversion method, exact matching of the overall gain and phase shift in the two arms is difficult. In practice, image rejection of 30 to 40 dB is feasible [25, p. 143].

4.1.3 Location of the A/D Converter in Receiver Chain

The A/D converter can be used to sample either a baseband signal, an IF signal, or even a RF signal. A receiver which samples the baseband or IF signal contains a mixture of analog and digital components. However, a receiver which samples the RF signal is almost entirely digital.

The difficulty in implementing an IF or RF sampling receiver lies in the practical limitations on the A/D converter. First, due to the lowpass nature of the frequency response of the A/D converter (see Section 4.1.4), the highest frequency component which can be digitized properly is limited. With current technology, while digitization of signals with frequencies above 100 MHz is possible, the cost and power consumption of such A/D converters is usually prohibitive for use in mobile units.

Second, as discussed in Section 6.5.3, sampling jitter also places practical limitations on the highest frequency component which can be admitted at the input of the A/D converter, making sampling of the baseband signal much more practical.

Lastly, the dynamic range of A/D converters is such that a large gain is usually required prior to the A/D converter to increase the signal power to a level which can be digitized. Another thing which should be considered is that it may also be necessary to place narrowband filters in the signal path in order to remove interference, and these filters are more difficult to construct at higher frequencies. If strong interference overloads the A/D, then no amount of digital filtering can be used to remove it.

4.1.4 Relationship of Sampling Frequency to Signal Center Frequency

Recall that ideal sampling is equivalent to multiplication of the analog signal, $m(t)$, by a train of delta functions, $\sum_{i=-\infty}^{\infty} \delta(t \Leftrightarrow iT_s)$, where T_s is the sampling interval. It can be shown [23] that in the frequency domain, the spectrum of the sampled signal, $M_s(f)$, is

$$M_s(f) = M(f) * f_s \sum_{n=-\infty}^{\infty} \delta(f \Leftrightarrow nf_s) \tag{4.28}$$

$$= f_s \sum_{n=-\infty}^{\infty} M(f \Leftrightarrow nf_s), \tag{4.29}$$

where $M(f)$ is the Fourier transform of $m(t)$ and f_s is the sampling frequency. Equation 4.29 is a mathematical expression of the fact that sampling effectively generates images of the original signal at intervals of the sampling frequency.

Figure 4.8 defines some commonly used nomenclature. The regions

$$\left[kf_s, \left(k + \frac{1}{2}\right) f_s\right] \tag{4.30}$$

and

$$\left[\left(k + \frac{1}{2}\right) f_s, (k+1)f_s\right], \tag{4.31}$$

where k is an integer and f_s is the sampling frequency, define "Nyquist zones."

If ideal sampling could be achieved in practice, then any signal could be sampled without aliasing, provided that the entire signal could fit inside a single Nyquist zone. This is essentially an extension of Nyquist's rule that the sampling rate must be twice the bandwidth of the signal being sampled. (Also notice that the sidebands of the image closest to baseband of any signal which may lie in an even-numbered Nyquist zone are reversed upon sampling.)

Unfortunately, due to sample-and-hold amplifiers, settling times, and propagation delays due to electronics, the range of frequencies which practical A/D converters can digitize is limited. This limitation is usually characterized by assigning an "input bandwidth" to the A/D converter, which specifies a lowpass frequency response of the A/D converter. Any signal which is to be digitized must lie completely within this input bandwidth for sampling and digitization without attenuation.

An important relationship in a digital receiver is that of the sampling frequency to the frequency band of the signal being sampled. In this thesis, "classical sampling" refers to

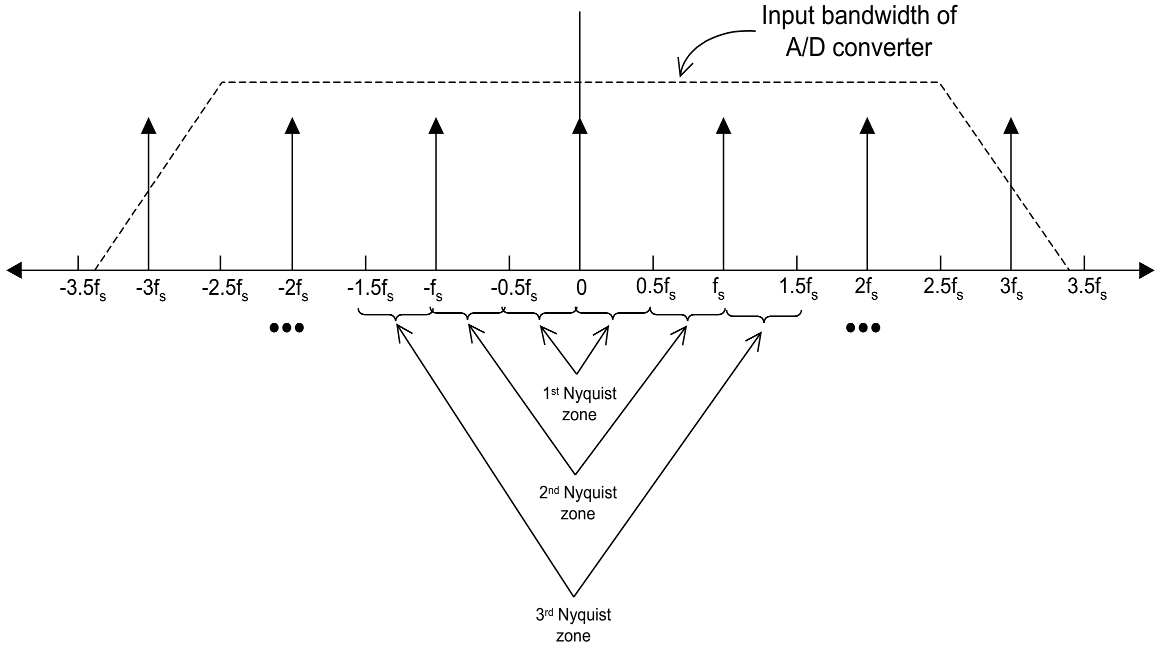

Figure 4.8: For proper digitization without aliasing or attenuation, the input signal to the A/D converter must lie completely in a single Nyquist zone, and must lie within the input bandwidth of the A/D converter.

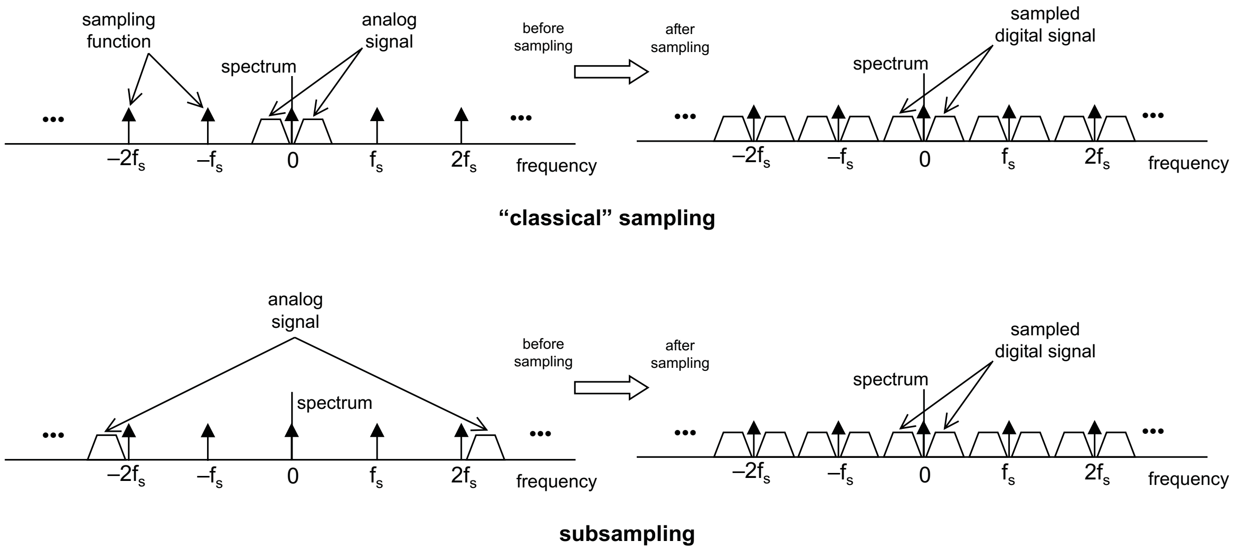

Figure 4.9: The difference between "classical sampling" and "subsampling."

the case where the signal lies entirely in either the first or second Nyquist zone, whereas "subsampling" refers the case where the signal lies elsewhere. Figure 4.9 illustrates this difference.

Figure 4.9 also illustrates that a benefit of using subsampling is that an image of the sampled signal appears near baseband, which shows that the A/D converter essentially plays the role of a downconverter. Subsampling thus allows an entire stage of analog components to be eliminated.

Another benefit is that with direct downconversion and subsampling, the I and Q components can be extracted digitally, avoiding the gain and phase mismatches present in the quadrature downconversion method. This is often referred to as "digital downconversion."

The drawbacks associated with subsampling are the practical limitations on the A/D converter, including the sampling jitter and input bandwidth. Usually, subsampling receivers must use at least one stage of analog downconversion and gain in order to place the signal frequency band within the input bandwidth of the A/D converter and at a frequency at which sampling jitter has little effect. Gain is necessary to amplify the signal so that it falls within the input dynamic range of the A/D converter.

4.2 Digital Receiver Examples

4.2.1 Homodyne Receiver

A homodyne receiver employing classical sampling at baseband is shown in Figure 4.10. Filtering and amplification is provided before the quadrature downconversion to baseband. Two A/D converters appear in this receiver.

4.2.2 Heterodyne Receiver

A dual-conversion heterodyne receiver with IF sampling is shown in Figure 4.11. Two stages of filtering, amplification, and downconversion are used. The heterodyne receiver is useful because it can employ an extensive emount of filtering to remove interference while retaining and amplifying the signal of interest. Also, if a large gain before the A/D is required, it can be distributed over several downconversion stages, which can be used to ensure that even a slight feedback path does not send the amplifiers into oscillation. Notice that only one A/D converter appears in this receiver, and the I and Q components are extracted via

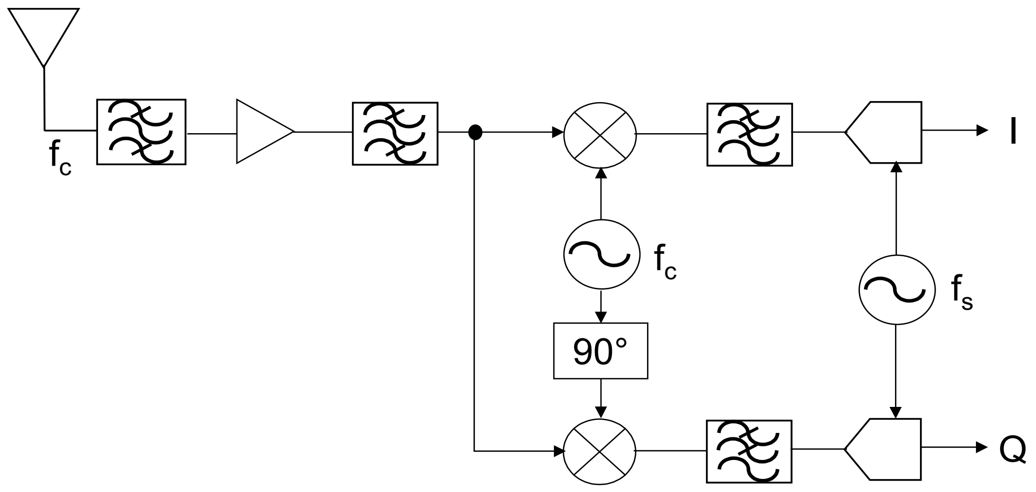

Figure 4.10: A homodyne receiver employing classical sampling at baseband.

digital downconversion. The filter before the A/D converter is usually quite narrow, and is present to extract only the signal of interest, thereby avoiding post-sampling aliasing of either desired or undesired signals into the frequency band of interest. This receiver architecture was selected for the pilot signal scanning receiver described in this thesis.

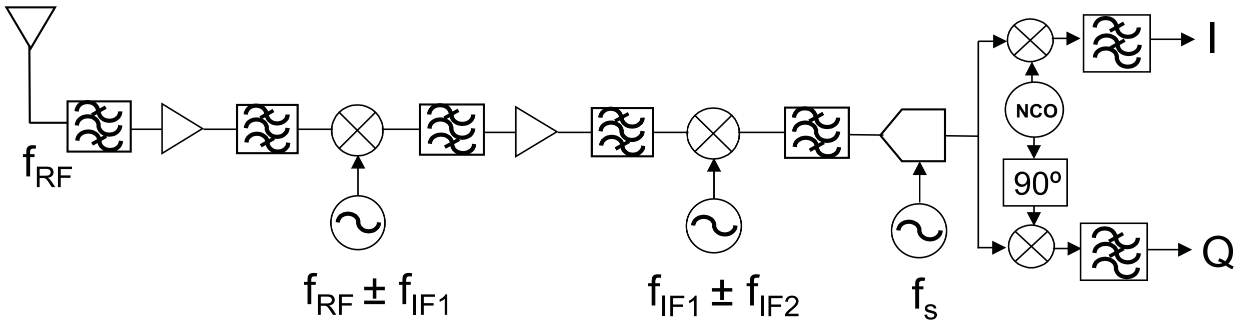

Figure 4.11: A dual-downconversion heterodyne receiver employing IF sampling.

Chapter 5

Digital Signal Processing Algorithms for Pilot Signal Scanning

In this chapter, the digital signal processing algorithms which detect pilot channel signals and control the gain of the circuits prior to the A/D converter are specified. As discussed in Section 4.2.2, a subsampling receiver architecture was adopted in this thesis. Therefore, an implicit assumption is that the algorithms process samples of an IF signal.

5.1 Correlation Algorithms

Correlation is the means by which pilot signals are identified. In this section, two possible means of performing this correlation are presented.

5.1.1 Baseband Correlation Method

The algorithm depicted in Figure 5.12 is referred to in this thesis as the "baseband" correlation method. Let the output of the analog tuner, $r_{IF}(t)$, be the following:

$$r_{IF}(t) = I(t)\cos\left(2\pi f_{IF}t + \theta\right) + Q(t)\sin\left(2\pi f_{IF}t + \theta\right), \tag{5.32}$$

where, from Equation 2.15,

$$I(t) = a_0 P_I(t) + u_I(t), \tag{5.33}$$

$$Q(t) = a_0 P_Q(t) + u_Q(t), \tag{5.34}$$

where a_0 is the amplitude of the pilot signal, $P_I(t)$ and $P_Q(t)$ are the I- and Q-channel pilot signals, and $u_I(t)$ and $u_Q(t)$ are interference terms due to all other CDMA channels in use.

In Equation 5.32, f_{IF} is the intermediate frequency at which the sampling occurs, and θ is an arbitrary constant phase. Assume that the A/D converter samples $r_{IF}(t)$ at times $t_n = n/f_s$ $(n = \ldots, \Leftrightarrow 2, \Leftrightarrow 1, 0, 1, 2, \ldots)$, where f_s is the sampling frequency, to produce the sampled IF signal $r_{IF}(n)$. The image closest to baseband produced by sampling appears a center frequency f_{alias}, where

$$f_{alias} = \underset{\forall\ k \geq 0}{\min} \left(f_{IF} \Leftrightarrow k f_s \right). \tag{5.35}$$

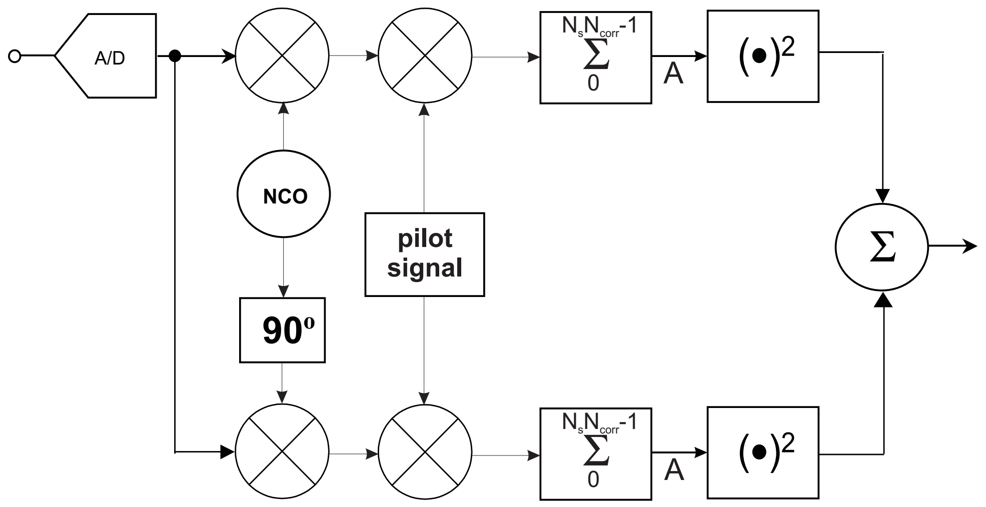

Figure 5.12: The "baseband" method of correlation. The NCO is a numerically controlled oscillator algorithm which generates samples of a sinusoid. The pilot signal is pulse-shaped to match the pilot signal of the received signal. The squared-modulus only needs to be computed after the summation is complete (i.e., once every $N_s N_{corr}$ samples).

The sampled signal is first translated to baseband to produce components $r_c(n)$ and $r_s(n)$ by multiplying it by samples of quadrature sinusoids, produced by the numerically controlled oscillator (NCO) of Figure 5.12:

$$r_c(n) = r_{IF}(n) \cos\left(2\pi f_{IF} \frac{n}{f_s}\right) \tag{5.36}$$

$$r_s(n) = r_{IF}(n) \sin\left(2\pi f_{IF} \frac{n}{f_s}\right). \tag{5.37}$$

Now an important observation is made. If f_{IF} is chosen so that

$$f_{IF} = \frac{2m+1}{4} f_s, \tag{5.38}$$

where m is any non-negative integer, then we have

$$\cos\left(2\pi f_{IF}\frac{n}{f_s}\right) = \cos\left(\frac{(2m+1)\pi}{2}n\right) = \begin{cases} \pm 1 \\ 0 \end{cases} \tag{5.39}$$

$$\sin\left(2\pi f_{IF}\frac{n}{f_s}\right) = \sin\left(\frac{(2m+1)\pi}{2}n\right) = \begin{cases} \pm 1 \\ 0 \end{cases} \tag{5.40}$$

Thus, provided that f_{IF} and f_s are chosen so that Equation 5.38 is satisfied, computation of the products in Equations 5.36 and 5.37 amounts to multiplication of $r_{IF}(n)$ by ± 1 and 0. This elegant method of digital I and Q downconversion, which requires very little computational effort, was proposed in [26] and [27].

Next, the correlation is measured by computing the following four summations:

$$\begin{aligned} r_{cI}(n_o) &= \sum_{i=0}^{N_sN_{corr}-1} r_c(i)P_I(i+n_o) \qquad & r_{cQ}(n_o) &= \sum_{i=0}^{N_sN_{corr}-1} r_c(i)P_Q(i+n_o) \\ r_{sI}(n_o) &= \sum_{i=0}^{N_sN_{corr}-1} r_s(i)P_I(i+n_o) \qquad & r_{sQ}(n_o) &= \sum_{i=0}^{N_sN_{corr}-1} r_s(i)P_Q(i+n_o), \end{aligned} \tag{5.41}$$

where n_o is the offset (in samples) between the received pilot signal and the local pilot signal in the receiver, N_s is the number of samples per chip, and N_{corr} is the number of chips over which the summations are computed. Here, the possibility of performing only a partial correlation over the entire pilot signal has been admitted, so that N_{corr} does not necessarily equal 2^{15} (the number of chips in the pilot PN sequence).

Algebraic and trigonometric manipulation of Equations 5.41 yield the following relations:

$$\begin{aligned} r_{cI}(n_o) &= \frac{a_0}{2}\cos(\theta)R_{II_p}(n_o) + i_{cI}(n_o) \qquad & r_{cQ}(n_o) &= \frac{a_0}{2}\sin(\theta)R_{QQ_p}(n_o) + i_{cQ}(n_o) \\ r_{sI}(n_o) &= \Leftrightarrow\frac{a_0}{2}\sin(\theta)R_{II_p}(n_o) + i_{sI}(n_o) \qquad & r_{sQ}(n_o) &= \frac{a_0}{2}\cos(\theta)R_{QQ_p}(n_o) + i_{sQ}(n_o), \end{aligned} \tag{5.42}$$

where the partial autocorrelation products, $R_{II_p}(n_o)$ and $R_{QQ_p}(n_o)$,

$$R_{II_p}(n_o) = \sum_{i=0}^{N_sN_{corr}-1} P_I(i)P_I(i+n_o), \tag{5.43}$$

and

$$R_{QQ_p}(n_o) = \sum_{i=0}^{N_sN_{corr}-1} P_Q(i)P_Q(i+n_0) \tag{5.44}$$

are the desired part of the signal and the terms $i_{cI}(n_o)$, $i_{cQ}(n_o)$, $i_{sI}(n_o)$, and $i_{sQ}(n_o)$ are undesired interference terms due to other CDMA channels in the system.

Finally, the outputs $O_I^{bb}(n_o)$ and $O_Q^{bb}(n_o)$ are generated by removing the unknown phase θ by computing the squared-modulus:

$$O_I^{bb}(n_o) = r_{cI}^2(n_o) + r_{sI}^2(n_o) = \frac{a_0^2}{4} R_{II_p}^2(n_o) + i_I^{bb}(n_o) \tag{5.45}$$

$$O_Q^{bb}(n_o) = r_{cQ}^2(n_o) + r_{sQ}^2(n_o) = \frac{a_0^2}{4} R_{QQ_p}^2(n_o) + i_Q^{bb}(n_o), \tag{5.46}$$

where $i_I^{bb}(n_o)$ and $i_Q^{bb}(n_o)$ are undesired interference terms. By systematically changing the phase of the local signal, or scanning, the receiver adjusts the value of n_o. When the phase of the local pilot signal is such that $n_o = 0$, large-valued outputs $O_I^{bb}(n_o)$ and $O_Q^{bb}(n_o)$ are observed due to the fact that $R_{II_p}^2(n_o)$ and $R_{QQ_p}^2(n_o)$ are large for $n_o = 0$ and relatively small otherwise. The successful performance of the algorithm further rests on the premise that $i_I^{bb}(n_o) \ll a_0^2 R_{II_p}^2(0)$ and $i_Q^{bb}(n_o) \ll a_0^2 R_{QQ_p}^2(0)$ for all n_o, so that the correlation peak can be easily identified.

The outputs $O_I^{bb}(0)$ and $O_Q^{bb}(0)$ can be used to obtain an estimate of the amplitude of the pilot signal, a_0. This subject is covered in Appendix E.

The number of operations per sample for the baseband correlation algorithm is a quantity of interest. This is best figured using Figure 5.12. Assuming that the relationship between f_s and f_{IF} has been selected to satisfy Equation 5.38, we find that every other sample on both I and Q arms is zero, thus requiring no operations. If multiplication by +1 is not counted as an operation, then an average of $(0+2+0+3)/4 = 1.25$ operations per sample is required on both the I and Q arms. Thus, the average number of operations per sample for the baseband correlation algorithm is 2.5 operations/sample. Notice that the squared-modulus only needs to be computed after the summation is complete, or every $N_s N_{corr}$ samples. Thus, if $N_s N_{corr}$ is large, the squared-modulus operation does not contribute significantly to the operations count.

For implementing the algorithm (and the bandpass correlation algorithm presented next), the idea is to store the samples of the pilot signals, $P_I(t)$ and $P_Q(t)$ (which are referred to here as the pilot signals), in memory. In this manner, the pilot signal only needs to be generated once, and subsequently the samples of the pilot signal can be read from memory for the multiply-accumulate operation required in the correlation algorithms.

It was found that the production of a single output, $O^{bb}(n_o)$, which was the average

$$O^{bb}(n_o) = \frac{1}{2}\left(O_I^{bb}(n_o) + O_Q^{bb}(n_o)\right) \tag{5.47}$$

was effective at reducing the variance of the noise and uncorrelated outputs at the output of the algorithm, and thereby increasing the output signal-to-noise ratio or detection capability of the algorithm.

5.1.2 Bandpass Correlation Method

The algorithm depicted in Figure 5.13 is referred to here as the "bandpass" correlation method. First, the sampled IF signal, $r_{IF}(n)$, is multiplied by the sampled pilot signals, $P_I(n+n_o)$ and $P_Q(n+n_o)$:

$$r_{IF}(n)P_I(n+n_o) = a_0 P_I(n)P_I(n+n_o)\cos\left(2\pi f_{IF}\frac{n}{f_s}+\theta\right)+i_I(n,n_o) \qquad (5.48)$$

$$r_{IF}(n)P_Q(n+n_o) = a_0 P_Q(n)P_Q(n+n_o)\sin\left(2\pi f_{IF}\frac{n}{f_s}+\theta\right)+i_Q(n,n_o) \qquad (5.49)$$

where n_0 is the offset in samples between the received pilot signal and the locally generated pilot signal, a_0 is the amplitude of the received pilot signal, f_{IF} is the final intermediate frequency of the analog tuner, and $i_I(n,n_o)$ and $i_Q(n,n_o)$ are interference terms. Due to the relatively small autocorrelation of the pilot signal for $n_o \neq 0$, the following behavior is expected:

$$r_{IF}(n)P_I(n+n_o) \propto \begin{cases} \text{noise} & \text{for } n_o \neq 0 \\ \cos\left(2\pi f_{IF}\frac{n}{f_s}+\theta\right)+\text{noise} & \text{for } n_o = 0 \end{cases} \qquad (5.50)$$

$$r_{IF}(n)P_Q(n+n_o) \propto \begin{cases} \text{noise} & \text{for } n_o \neq 0 \\ \sin\left(2\pi f_{IF}\frac{n}{f_s}+\theta\right)+\text{noise} & \text{for } n_o = 0 \end{cases} \qquad (5.51)$$

Figure 5.14 shows the magnitudes of simulated spectra of $r_{IF}(n_o)P_I(n+n_o)$ for $n_o=0$ and $n_o=80$ samples. In Figure 5.14, the relationship between f_s and f_{IF} expressed in Equation 5.38 has been satisfied, so that $f_{alias}=f_s/4$. For $n_o=0$, a large spectral component at the frequency $f_s/4$ is clearly identifiable. The spectrum for $n_o=80$ samples appears to be noise-like, with no clearly dominant spectral component.

A narrowband one-pole Butterworth bandpass filter[1] with center frequency f_{alias} is employed to attenuate the noise yet retain any large spectral component present at f_{alias}. To minimize computations, yet provide sufficient filtering, a single-pole infinite impulse response (IIR) filter was selected. The design of this filter is covered in Appendix D.

In terms of a difference equation relating the bandpass filter output, $y(n)$, to the filter input, $x(n)$, it is shown in Appendix D that

$$y(n) = B_0(x(n) \Leftrightarrow x(n \Leftrightarrow 2)) \Leftrightarrow A_1 y(n \Leftrightarrow 1) \Leftrightarrow A_2 y(n \Leftrightarrow 2), \qquad (5.52)$$

[1]Other filter types, e.g., Bessel, could have been used.

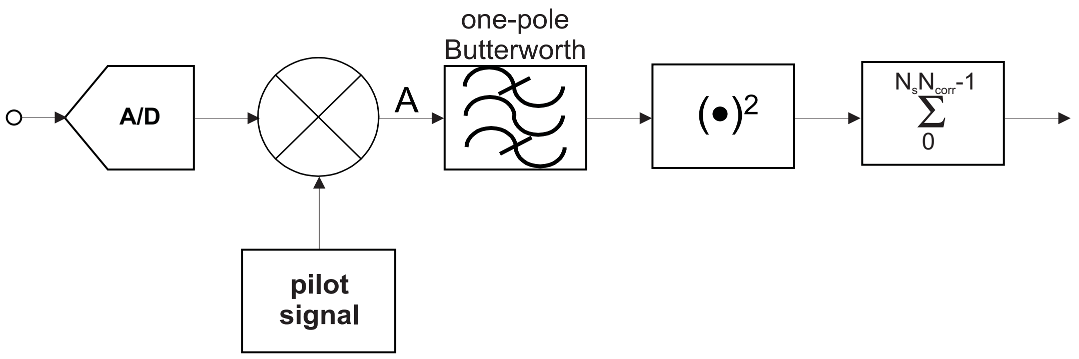

Figure 5.13: The "bandpass" method of correlation. The spectrum of the signal at the point labeled "A" is shown in Figure 5.14.

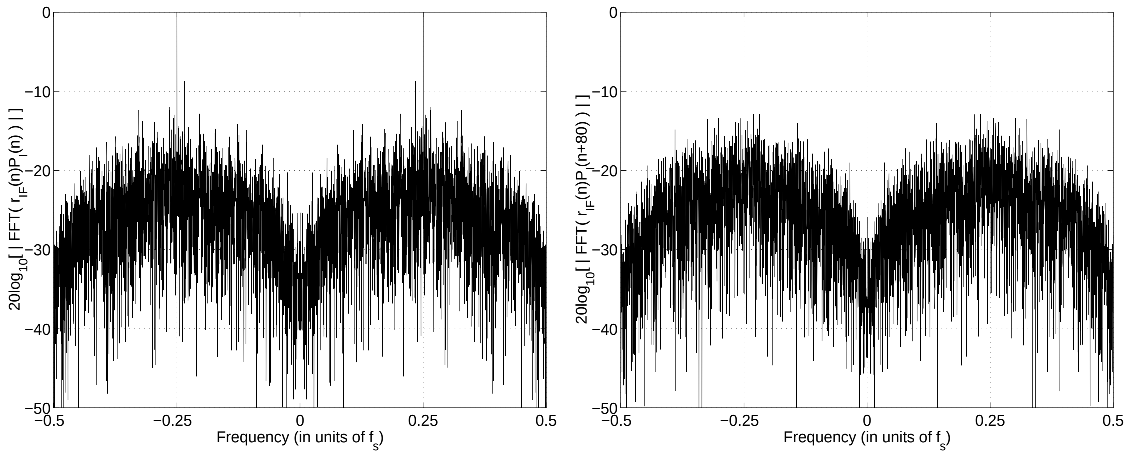

Figure 5.14: The spectrum of $r_{IF}(n)P_I(n+n_o)$ for $n_o = 0$ (left) and $n_o = 80$ samples (right).

with

$$B_0 = \frac{W}{1 + W + \omega_0^2}, \tag{5.53}$$

$$A_1 = \frac{2\left(\omega_0^2 \Leftrightarrow 1\right)}{1 + W + \omega_0^2}, \tag{5.54}$$

$$A_2 = \frac{1 \Leftrightarrow W + \omega_0^2}{1 + W + \omega_0^2}, \tag{5.55}$$

where ω_0^2 and W are defined by Equations D.172, D.173, D.174, D.175, D.178, and D.179.

Since a narrowband filter is being considered, B_0 is typically a very small number, $B_0 \ll 1$, while A_2 is typically a number strictly less than but almost unity. Thus, Equation 5.52 shows that the bandpass filtering generally requires a total of six operations. However, if the center frequency of the filter is chosen so that $\omega_0^2 = 1$, the coefficient A_1 becomes zero, and then the filtering operation only requires four operations. By Equations D.172, D.174, D.175, and D.179, it is found that $\omega_0^2 = 1$ implies that

$$\tan\left(\frac{\omega_{low} T_s}{2}\right) \tan\left(\frac{\omega_{high} T_s}{2}\right) = 1, \tag{5.56}$$

where ω_{low} is the lower cutoff frequency of the filter, ω_{high} is the higher cutoff frequency of the filter, and T_s is the sampling period. Equation 5.56 is satisfied trivially when $\omega_{low} = \omega_{high} = 2\pi f_s/4$, where $f_s = 1/T_s$ is the sampling frequency. Otherwise, we have

$$\omega_{low} < 2\pi f_s/4 \tag{5.57}$$

and

$$\omega_{high} > 2\pi f_s/4. \tag{5.58}$$

Thus, for a narrowband bandpass filter, the center frequency of that filter should reside at $f_s/4$.

The minimum number of computations per sample for bandpass filtering is thus four. Any other choice for the center frequency of the filter other than $f_s/4$ results in a total of six operations per sample. Since the choice of $f_s/4$ as the center frequency of the filter drives the coefficient of $y(n \Leftrightarrow 1)$ to zero, the filtering operation then only depends upon $x(n)$, $x(n \Leftrightarrow 2)$, and $y(n \Leftrightarrow 2)$. Although it is tempting to conclude that the filtering operation can be decimated by filtering only on every other input sample, careful reflection indicates this is not feasible since the phase of the sinusoid to be filtered is unknown *a priori*.

The filtered signal is energy-detected by squaring each sample at the filter output and integrating the result. Mathematically, the outputs of the algorithm, $O_I^{bp}(n_o)$ and $O_Q^{bp}(n_o)$, should then be

$$O_I^{bp}(n_o) = \frac{a_0^2}{2} \sum_{i=0}^{N_s N_{corr}-1} \left(P_I(i) P_I(i+n_o)\right)^2 + i_I^{bp}(n_o) \tag{5.59}$$

$$O_Q^{bp}(n_o) = \frac{a_0^2}{2} \sum_{i=0}^{N_s N_{corr}-1} \left(P_Q(i) P_Q(i+n_o)\right)^2 + i_Q^{bp}(n_o), \tag{5.60}$$

where $i_I^{bp}(n_o)$ and $i_Q^{bp}(n_o)$ are interference terms, which should satisfy the relations

$$i_I^{bp}(n_o) \ll a_0^2 \sum_{i=0}^{N_s N_{corr}-1} \left(P_I(i) P_I(i)\right)^2 \tag{5.61}$$

$$i_Q^{bp}(n_o) \ll a_0^2 \sum_{i=0}^{N_s N_{corr}-1} \left(P_Q(i) P_Q(i)\right)^2 \tag{5.62}$$

for all values of n_o for adequate detection of the correlation peak.

The bandwidth of the bandpass filter is an important part of the design of the bandpass correlation algorithm. At least four factors must be taken into account when choosing the bandwidth:

- offset between the frequency of the local oscillator(s) and the frequency of the carrier, which causes the strong sinusoidal component to appear elsewhere other than the center frequency of the filter, thus causing it to be attenuated,
- the rise time of the filter,
- the energy integration period (in samples), $N_s N_{corr}$, and
- the noise rejection capability of the filter.

To cope with frequency offset, a wider bandwidth is required. Similarly, shorter filter rise time is obtained through the use of a filter with a wider bandwidth. However, if the bandwidth of the filter is too wide, the loss in the noise rejection capability becomes excessive. Another factor which must be taken into account is that if the bandwidth of the filter is too narrow, although the noise rejection capability is enhanced, the energy integration period, $N_s N_{corr}$, must be increased to maintain sufficient energy-detection capability to overcome the rise time of the filter.

Examination of Figure 5.13 along with the discussion above indicates that, provided that the sampling aliases the signal at $f_s/4$, the bandpass detection method requires a total of seven operations per sample. This includes one multiplication by the pilot signal, two multiplications and two additions for the filtering operation, and one multiplication and one addition for the energy detection.

5.2 Automatic Gain Control (AGC)

5.2.1 Discussion

AGC is the process of automatically adjusting the overall gain of the RF/downconversion circuits prior to the detection circuits. The gain is usually adjusted in such a way that the power of the signal at the input of the detection circuits remains constant and falls within the linear range of the detection circuits.

AGC is used for many reasons. First, detection circuits usually have a range of input power levels over which they operate properly. If the input power of the signal is too small, noise overcomes the detection circuits. If the input power of the signal is too large, overload or saturation of the detection circuits occurs.

Second, adjusting the gain extends the dynamic range of receiver. When the T-R separation is large, maximum gain is desired. However, as can be seen from Appendix F, when the T-R separation becomes smaller, the gain must be decreased to avoid generation of excessive intermodulation distortion and blocking due to gain compression.

Third, in mobile receivers, as discussed in Chapter 3, the signal power may undergo extreme excursions. A properly designed AGC system will compensate for these extreme excursions, thereby maintaining a constant quality of communications.

For example, in a digital receiver such as is designed in this thesis, it is important to adjust the signal power so that all likely received signal strength variations fall into one of the quantization levels of the A/D converter. If the signal power is too small, then there are unused quantization levels near the upper and lower portions of the full scale range of the A/D. If the signal power is too large, then the signal overloads the A/D, and excessive quantization distortion results.

To investigate this concept for the receiver under design, forward link unity power IS-95A signals composed of various numbers of active CDMA channels were simulated (see Appendix B). Figure 5.15 shows the probability density functions (PDFs) of these simulated signals for 1, 5, 10, and 20 active channels. Ostensibly, the PDF of the IS-95A signal approaches that of a Gaussian distribution when the number of active channels is large.

In a simple numerical computation, 1024 samples of a random zero-mean Gaussian signal, $s(n)$, having variance σ_s^2, were quantized over a full-scale range of $[\Leftrightarrow V_{FS}/2, V_{FS}/2]$ to produce

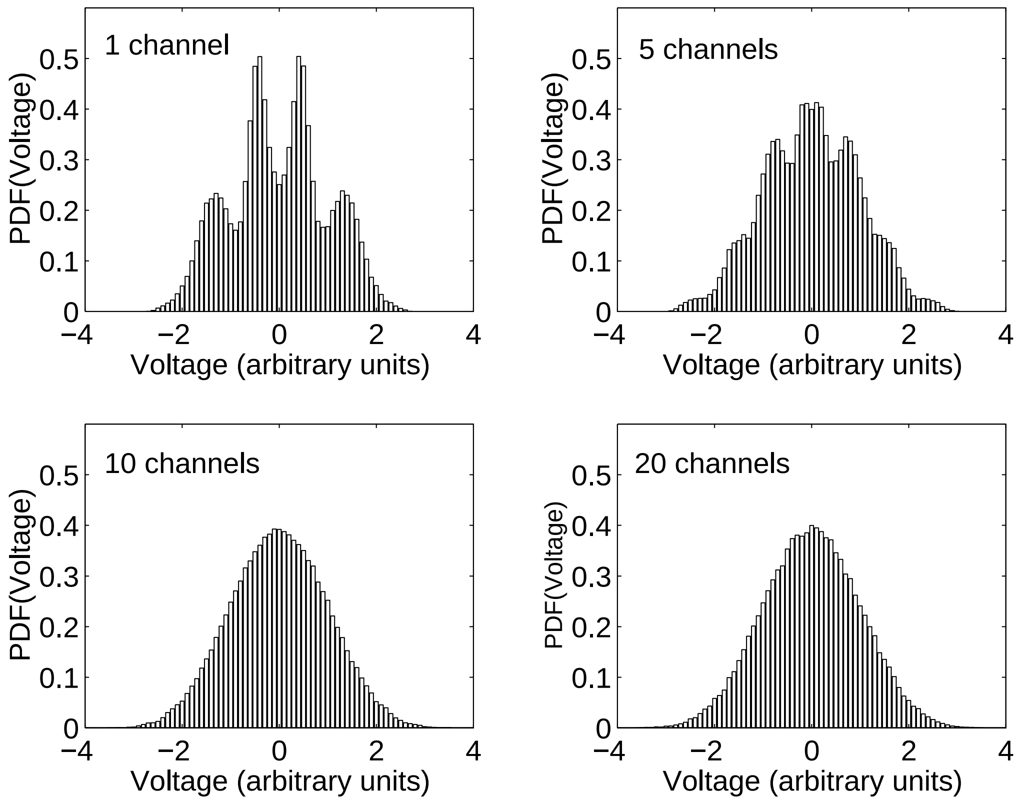

Figure 5.15: Simulated PDFs of unity power forward link IS-95A signals for various numbers of active channels. As the number of active channels grows, the PDF of the signal closely approximates that of a Gaussian distribution.

the signal, $s_q(n)$. The mean-squared quantization error, ϵ^2, was computed as follows:

$$\epsilon^2 = \frac{1}{N}\sum_{n=0}^{N-1}\left(s(n) \Leftrightarrow s_q(n)\right)^2 = \sum_{n=0}^{N-1}\left(S(n) \Leftrightarrow S_q(n)\right)^2, \tag{5.63}$$

where $S(n)$ is the discrete Fourier transform of the sequence $s(n)$,

$$S(n) = \sum_{m=0}^{N-1} s(m)e^{-2\pi jmn/N}, \tag{5.64}$$

and the last equality in Equation 5.63 is the discrete form of Parseval's relation. In Figure 5.16 ϵ^2 is plotted versus σ_s^2/V_{FS}^2 for 4-bit, 8-bit, and 12-bit quantization. It is observed that the mean-squared quantization error for 8-bit quantization exhibits a minimum as long as $\sigma_s^2/V_{FS}^2 < \Leftrightarrow 10$ dB, and for $\sigma_s^2/V_{FS}^2 > \Leftrightarrow 10$ dB the mean-squared error grows monotonically. Thus, in order to keep the noise due to quantization to a minimum, the AGC system should be designed to keep σ_s^2/V_{FS}^2 below about $\Leftrightarrow 10$ dB.

The spurious outputs of the A/D conversion process have not been modeled here. However, it is well known that A/D converters often produce spurious outputs which do not necessarily decrease with decreasing input power. Therefore, it is advisable to keep the input power to the A/D such that σ_s^2/V_{FS}^2 is close to, but does not exceed, $\Leftrightarrow 10$ dB.

5.2.2 Algorithm

In a digital receiver, a convenient way to estimate the signal power is provided by the A/D converter. Letting $s(i)$ represent the i^{th} sample from the A/D converter, a total of N_{agc} such samples may be used to estimate the (unnormalized) signal power, P_{est}, according to the formula

$$P_{est} = \frac{1}{N_{agc}}\sum_{i=1}^{N_{agc}} s^2(i). \tag{5.65}$$

Here it has been assumed that the gain prior to the A/D, G, remains constant over the period over which the N_{agc} samples are taken.

Analyses of system performance, such as those found in Section 6.5.3, assist in determining a proper target power level at the input of the A/D, P_{target}. Given this P_{target}, as well as G and P_{est}, the receiver then generates an updated gain, G', according to the formula

$$G' = G\frac{P_{target}}{P_{est}}. \tag{5.66}$$

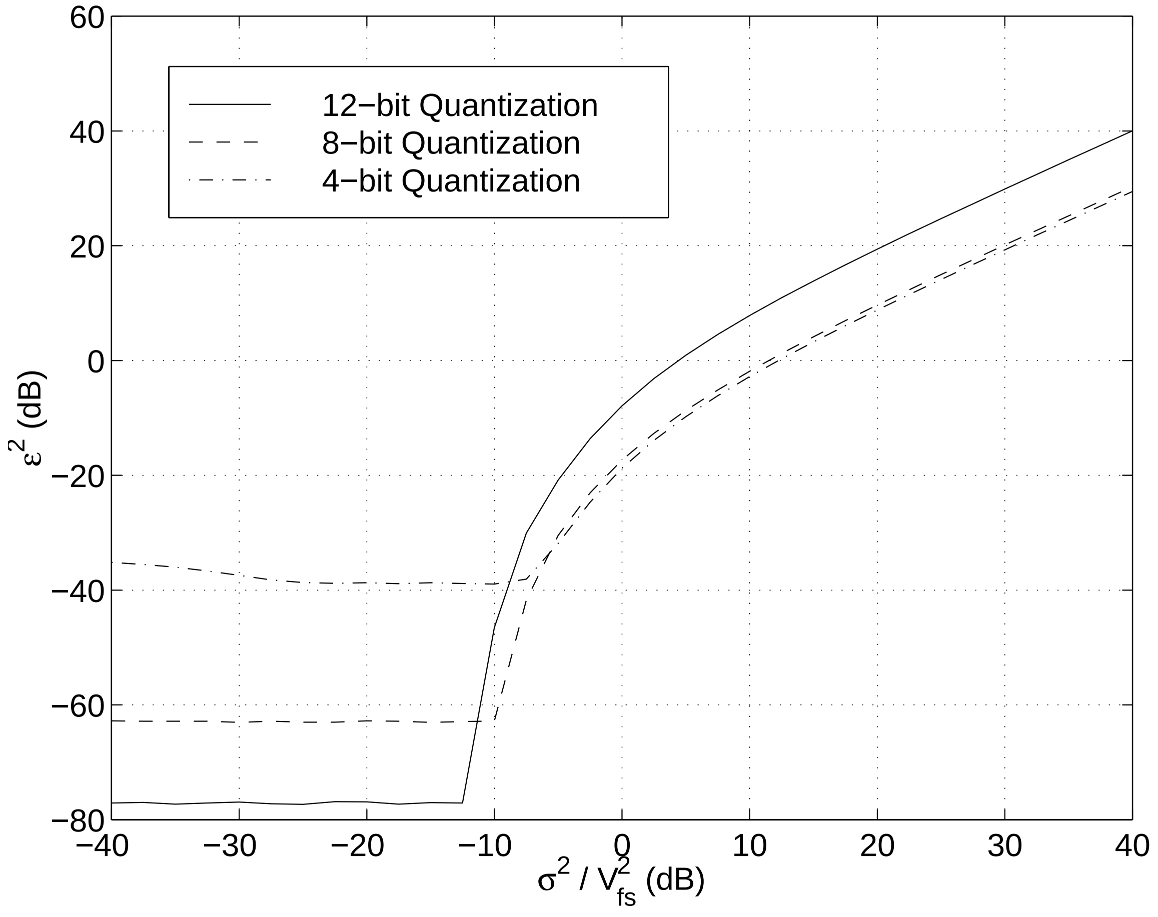

Figure 5.16: The mean-squared quantization error, ϵ^2, for a signal having a Gaussian PDF versus the (normalized) variance of the signal, σ_s^2/V_{FS}^2.

Fortunately, the computationally expensive quotient P_{target}/P_{est} only needs to be computed most frequently once every N_{agc} samples. G' may be transferred into a series of bits which controls circuits that fix the gain of the circuits prior to the A/D converter.

The relationship between the amount of time required to update the gain, T_{agc}, and the fading rate of the signal power is very important in mobile applications if the AGC is to be used to combat signal fading. T_{agc} must be chosen large enough to obtain an accurate estimate of the signal power, yet small enough that the signal power does not change significantly over that period.

Chapter 6

System Design Studies for a Pilot Signal Scanning Receiver

This chapter presents a set of studies, both analytical and simulation, of some of the most important issues which should be addressed in the design of a pilot signal scanning receiver. Due to time constraints, it was not possible to address some issues. These will be identified in Chapter 8.

This chapter begins by positing an abstract model of the receiver under design. This is followed by a discussion of three isolated topics: the time sidelobes of the autocorrelation function of the pilot signal (which limit dynamic range), selection of the sampling frequency of the receiver, and selection of the final IF of the receiver (i.e., the center frequency of the signal at the input of the A/D converter). Next, the performance of the receiver under real-world hardware and channel conditions is presented. This performance is measured in terms of a type of dynamic range, which is defined in this chapter. The final section of this chapter treats the design of the analog portion of the receiver.

6.1 Abstract Model

A block diagram of the receiver under design is provided in Figure 6.17. The received signal is downconverted (from a nominal center frequency of f_c to a nominal center frequency of f_{IF}), filtered, and amplified by the analog tuner. The signal is sampled and digitized by the A/D converter at a nominal rate $f_s < f_{IF}$, making the receiver an application of a common-IF subsampling architecture (see Section 4.1.4). The IF signal is sampled with a n_1-bit A/D converter. This discrete, quantized signal is then routed to a fixed-point DSP which quantizes its locally generated pilot signal to a resolution of n_2 bits. The gain of the analog tuner is controlled by feedback from the DSP to keep the power of the signal at the input of the A/D constant.

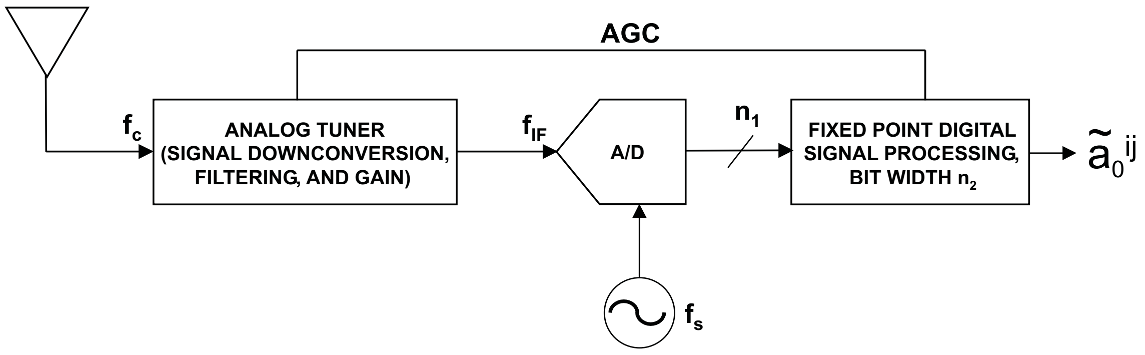
AGC
f_c
ANALOG TUNER (SIGNAL DOWNCONVERSION, FILTERING, AND GAIN)
f_{IF}
A/D
n_1
FIXED POINT DIGITAL SIGNAL PROCESSING, BIT WIDTH n_2
$\tilde{a}_0^{ij}$
f_s

Figure 6.17: The receiver is designed to consist of an analog tuner for signal downconversion, filtering, and gain; an A/D converter to sample and digitize the signal; and a DSP to process the signal and search for correlations. The goal is to extract information on the amplitudes of the pilot signals, a_0^{ij}, from each base station.

Determination of the number of pilot signals (and their amplitudes) arriving at the receiver is the *raison d'etre* of the pilot signal scanning receiver. Consistent with the labeling in Chapter 2 of the amplitude of the transmitted pilot channel signal as a_0, Figure 6.17 shows that the output of the DSP is a record of the estimates of the amplitudes of the j^{th} pilot signal from the i^{th} base station, a_0^{ij}, $i = 1, \ldots, B$; $j = 1, \ldots, M_i$, where N is the number of base stations and M_i is the total number of multipath components from the i^{th} base station.

6.2 Time Sidelobes of the Pilot Signal Autocorrelation Function

Detection of pilot signal activity relies on the sharp peaks that the autocorrelation functions of the pilot signals possess. These autocorrelation functions are labeled $R_{II}(\tau)$ and $R_{QQ}(\tau)$, where

$$R_{II}(\tau) = \frac{1}{T_{pilot}} \int_0^{T_{pilot}} P_I(t) P_I(t+\tau) dt, \tag{6.67}$$

$$R_{QQ}(\tau) = \frac{1}{T_{pilot}} \int_0^{T_{pilot}} P_Q(t) P_Q(t+\tau) dt, \tag{6.68}$$

and T_{pilot} is the period of the pilot signal. $R_{II}(\tau)$ and $R_{QQ}(\tau)$ possess sharp peaks at $\tau = 0$ and are relatively small otherwise.

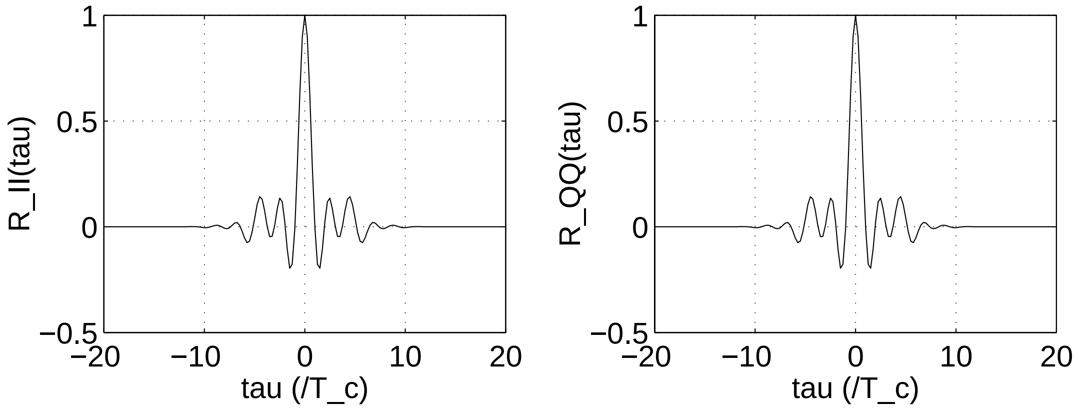

Figure 6.18: The autocorrelation functions $R_{II}(\tau)$ and $R_{QQ}(\tau)$ and their magnitudes for $\Leftrightarrow 20T_c < \tau < 20T_c$, where T_c is a chip period. The sidelobes of the autocorrelation, the largest of which is about 18 dB smaller than the main lobe, serve to obfuscate detection of multipath components which arrive within about 5 chip periods of each other.

The bandlimiting nature of pulse shaping the pilot PN sequence (see Section 2.5), however, introduces time sidelobes in the autocorrelation functions $R_{II}(\tau)$ and $R_{QQ}(\tau)$. The functions $R_{II}(\tau)$ and $R_{QQ}(\tau)$ are plotted in Figure 6.18, where the time sidelobes are clearly observable. The largest sidelobe is only about 18 dB smaller than the peak. The time sidelobes obfuscate the detection of multipath components which arrive within about 5 chip periods of each other.

6.3 Selection of Sampling Frequency, f_s

There are several considerations involved in selecting a proper sampling rate, which are listed as follows:

1) relationship of pre-processed signal bandwidth to sampling frequency (i.e., no aliasing after sampling but before processing),
2) relationship of post-processed signal bandwidth to sampling frequency (i.e., no aliasing neither after sampling nor after processing),
3) relaxation of constraints on pre-sampling filtering,
4) quantization noise,
5) accuracy on determination of phase of pilot signal,
6) data rate into the DSP,
7) storage memory of the DSP,
8) power consumption and cost of available A/D converters, and
9) sampling jitter.

Items one through five in this list all argue in favor of a high sampling rate, whereas items six through nine in this list argue in favor of a low sampling rate. For instance, for quantization noise, a higher sampling rate distributes the quantization noise (which is, to first approximation, white) over a wider bandwidth, which leaves less in-band quantization noise as compared to the noise contributed by a quantizer having a lower sampling rate.

Based upon bandwidth considerations alone, the minimum sampling rate is twice the bandwidth, or two samples per chip, of the analog signal at the input of the A/D converter. This is because in both the baseband and bandpass detection algorithms the bandwidth is not increased as a result of the processing. Nevertheless, a sampling rate of four times the bandwidth of the analog signal, or four samples per chip, was selected to relax the requirements on the anti-aliasing filter prior to the A/D converter and to obtain a better accuracy on the estimate of the phase of any pilot signal detected than could be obtained by using two samples per chip.

6.4 Selection of Final IF, f_{IF}

In Sections 5.1.1 and 5.1.2, it was shown that if f_{IF} and f_s satisfy the relationship given in Equation 5.38, a minimum number of operations for the digital signal processing algorithms for correlation results. Therefore, for the pilot signal scanning receiver it was required that this relationship be satisfied.

During the course of this project, 21.4 MHz was found to be a standard military IF. Therefore, good off-the-shelf filters exist at this frequency. Besides, 21.4 MHz is low enough that custom filters can be made using lumped element components if necessary. Moreover, it was reasoned that 21.4 MHz must be a region of the radio spectrum which is recognized by practicing radio engineers as being relatively free of interference and therefore good for filtering and amplification. In addition, 21.4 MHz was low enough that it was well within the input bandwidth of the A/D converter selected (see Sections 4.1.4 and 7.1.3).

The closest frequency which satisfied Equation 5.38 and was in the 21.4 MHz region was the frequency 20.8896 MHz. Therefore, the frequency $f_{IF} = 20.8896$ MHz was selected as the final IF. This is the nominal center frequency of the signal at the input of the A/D converter.

6.5 Receiver Performance

6.5.1 Performance Metric, D

If a pilot signal at a given phase exists at the receiver input, a peak in the output of the correlation algorithms is observed if the local pilot signal is in phase with this pilot signal. However, when the local pilot signal is not phase with this pilot signal, various factors conspire to induce non-trivial outputs of the correlation algorithms. Successful detection of the existence of the pilot signals depends upon the ability of the algorithms to distinguish the correlation peaks from the non-trivial outputs of correlation algorithms.

A quantity, hereafter referred to as the discrimination metric, D, was defined as follows:

$$D = \frac{S \Leftrightarrow \mu_N}{\sigma_N}, \tag{6.69}$$

where S is the output of the correlation algorithm when there is a pilot signal in phase with the local pilot signal, and μ_N and σ_N are the mean and standard deviation, respectively, of the distribution of the output of the algorithm when there is not a pilot signal in phase with the local pilot signal. Notice that D actually depends upon the amplitude of the pilot signal to be detected, a_0. The quantity D is graphically defined in Figure 6.19.

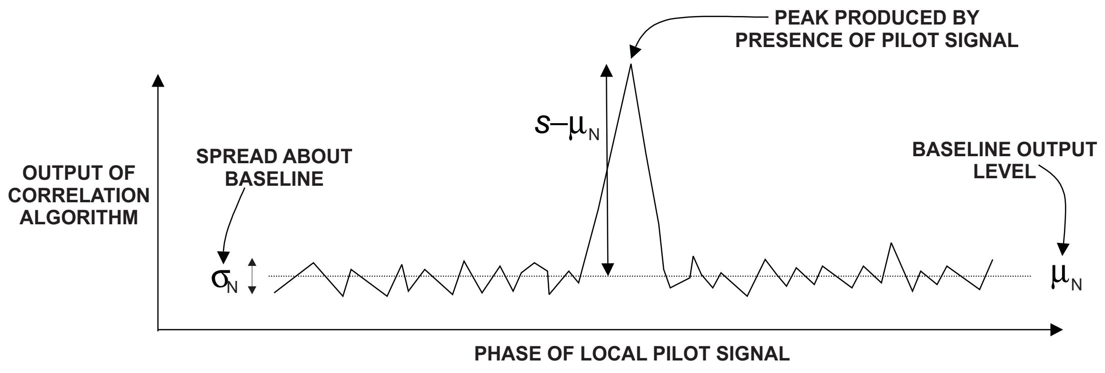
PEAK PRODUCED BY PRESENCE OF PILOT SIGNAL
S−μN
OUTPUT OF CORRELATION ALGORITHM
SPREAD ABOUT BASELINE
BASELINE OUTPUT LEVEL
σN
μN
PHASE OF LOCAL PILOT SIGNAL

Figure 6.19: This figure shows fictitious correlation outputs versus phase of the local pilot signal. A large peak indicates the presence of a signal having the same phase as the local pilot signal. In order to identify it as such, the peak must be well above the other outputs of the correlation algorithm. The quantities S, μ_N, and σ_N in the text have been graphically defined.

6.5.2 Simulation Groundrules

The digital signal processing algorithms described in Chapter 5 were run using simulated samples of IS-95A signals (see Appendix B) for the purpose of determining D in real-world situations. The simulation program continuously generated a stream of samples. For the correlation algorithms, the phase of the (local) pilot signal was fixed for N_{corr} chips ($N_s N_{corr}$ samples, where N_s is the number of samples per chip period) and an output was obtained and recorded. Then the (local) pilot signal was delayed by $1/N_s$ chip, and the correlation process begun anew. This type of scanning procedure was different than that which was actually implemented in hardware (see Section 7.4).

The parameters which were under control in the simulations were as follows:

- correlation period in chips, N_{corr},
- bandwidth of digital bandpass filter in bandpass detection method, BW_{bp}, (see Section 5.1.2),
- input SNR at A/D converter, $\mathrm{SNR}_{A/D}$,
- target signal power for AGC, P_{target},
- final IF, f_{IF},
- fractional frequency offset, δ_f,
- local oscillator RMS phase error, χ_{rms}^{phase},
- fraction of local oscillator mean-square phase error contributed by white noise, ϵ,
- number of samples per chip, N_s,
- fractional sampling frequency offset, δ_s,
- RMS sampling frequency jitter, χ_{rms}^{jitter},
- sampling phase offset, t_0^{offset},
- quantization of received signal (i.e., number of A/D bits), n_1,
- quantization of local pilot signal, n_2,
- number (M), delays ($\tau_i,\ i = 1, \ldots, M$), and amplitudes ($\alpha_i,\ i = 1, \ldots, M$) of multipath components[1],
- number of active CDMA channels per base station[2], N_{ch},
- number of active base stations contributing to received signal, B, and their relative amplitudes, $\beta_i,\ i = 1 \ldots, B$, and
- full scale range of A/D, $[V_{lim}^{lo}, V_{lim}^{hi}]$.

[1]For simplicity, the same multipath parameters were used for each base station.

[2]For simplicity, each base station was assumed to have the same number of active channels.

The radio channel was assumed to be static. The specification given in the IS-98 standard [15] was used to set the following:

$$\left.\frac{E_c}{I_{0r}}\right|_{pilot} = 0.20\ (=)\ \Leftrightarrow 7\ \text{dB} \tag{6.70}$$

$$\left.\frac{E_c}{I_{0r}}\right|_{sync} = 0.03\ (=)\ \Leftrightarrow 16\ \text{dB} \tag{6.71}$$

$$\left.\frac{E_c}{I_{0r}}\right|_{paging} = 0.06\ (=)\ \Leftrightarrow 12\ \text{dB} \tag{6.72}$$

$$\left.\frac{E_c}{I_{0r}}\right|_{traffic} = 0.71, \tag{6.73}$$

where $E_c/I_{0r}\ |_x$ is the ratio of the energy per chip for channel x to the total power spectral density of the signal transmitted by the base station in question (see Section 2.3). For the traffic channels, E_c/I_{0r} was divided evenly over the number of traffic channels in use. From the discussion in Section 2.3, it can be deduced that $a_0 = \sqrt{0.2}$.

6.5.3 Simulation Results

P_{target}

P_{target} is the mean-square voltage of the signal at the input of the A/D converter. It was necessary to determine a value of P_{target} for proper operation of the AGC in the pilot signal scanning receiver. In Figure 6.20, D is plotted versus a range of values of P_{target}. These results show that a relatively constant performance is achieved provided that $\Leftrightarrow 30 < 10\log_{10}(P_{target}) < 30$ for the baseband correlation method, and $\Leftrightarrow 10 < 10\log_{10}(P_{target}) < 30$ for the bandpass correlation method. The reason D decreases for large P_{target} is due to the overload of the A/D converter. The reason D decreases for small P_{target} is that the signal occupies very few of the quantization levels of the A/D converter. Based on these results, a value of $P_{target} = 2\ \text{V}^2$ was selected for implementation in the receiver.

Thermal Noise

The minimum allowable SNR at the input of the A/D converter, $\text{SNR}_{A/D}$, must be known for determining the tradeoff between the receiver noise figure and maximum T-R separation. In Figure 6.21, D is plotted versus a range of $\text{SNR}_{A/D}$. These results show that a constant performance is obtained by ensuring that $\text{SNR}_{A/D} > 5$ dB.

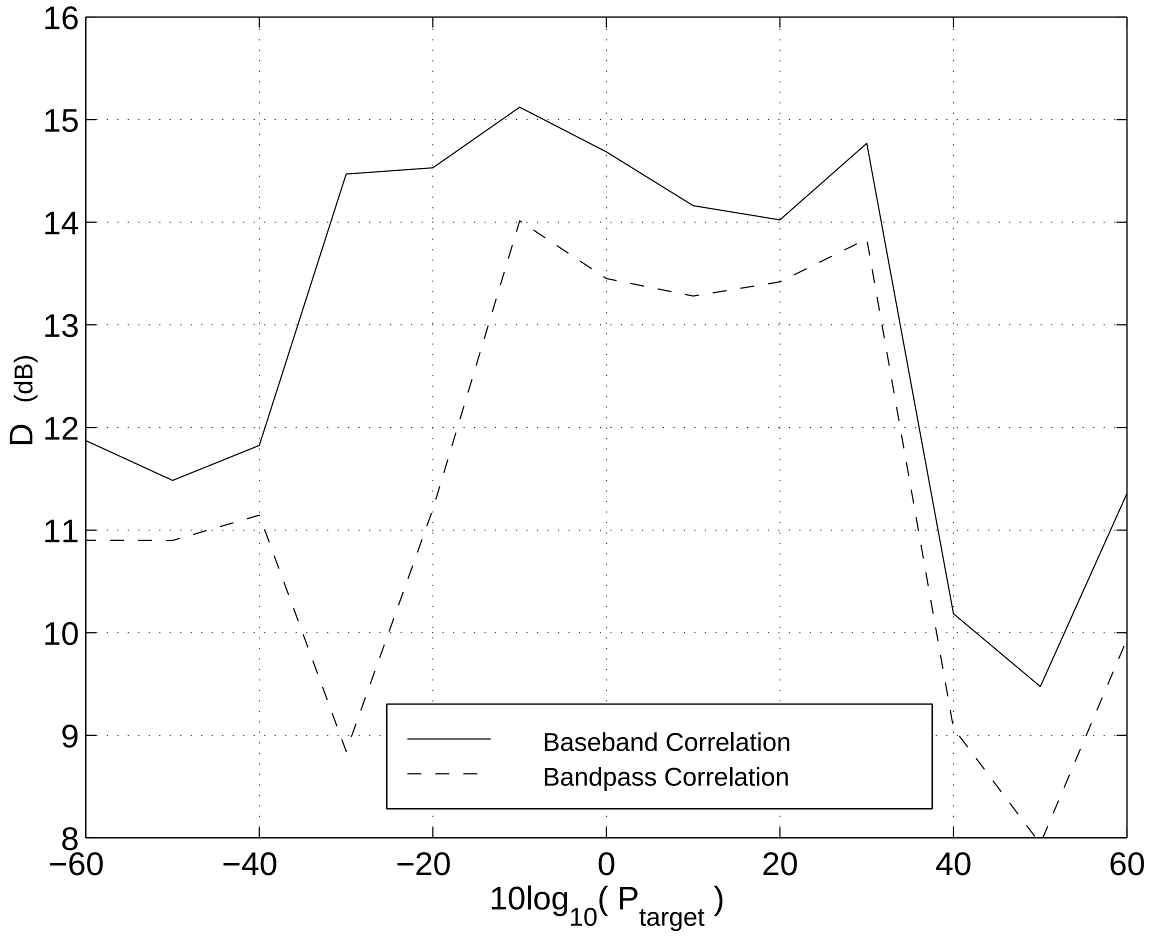

Figure 6.20: The discrimination metric D versus P_{target}. The parameters of the simulations which produced these results are given in Table C.13 of Appendix C.

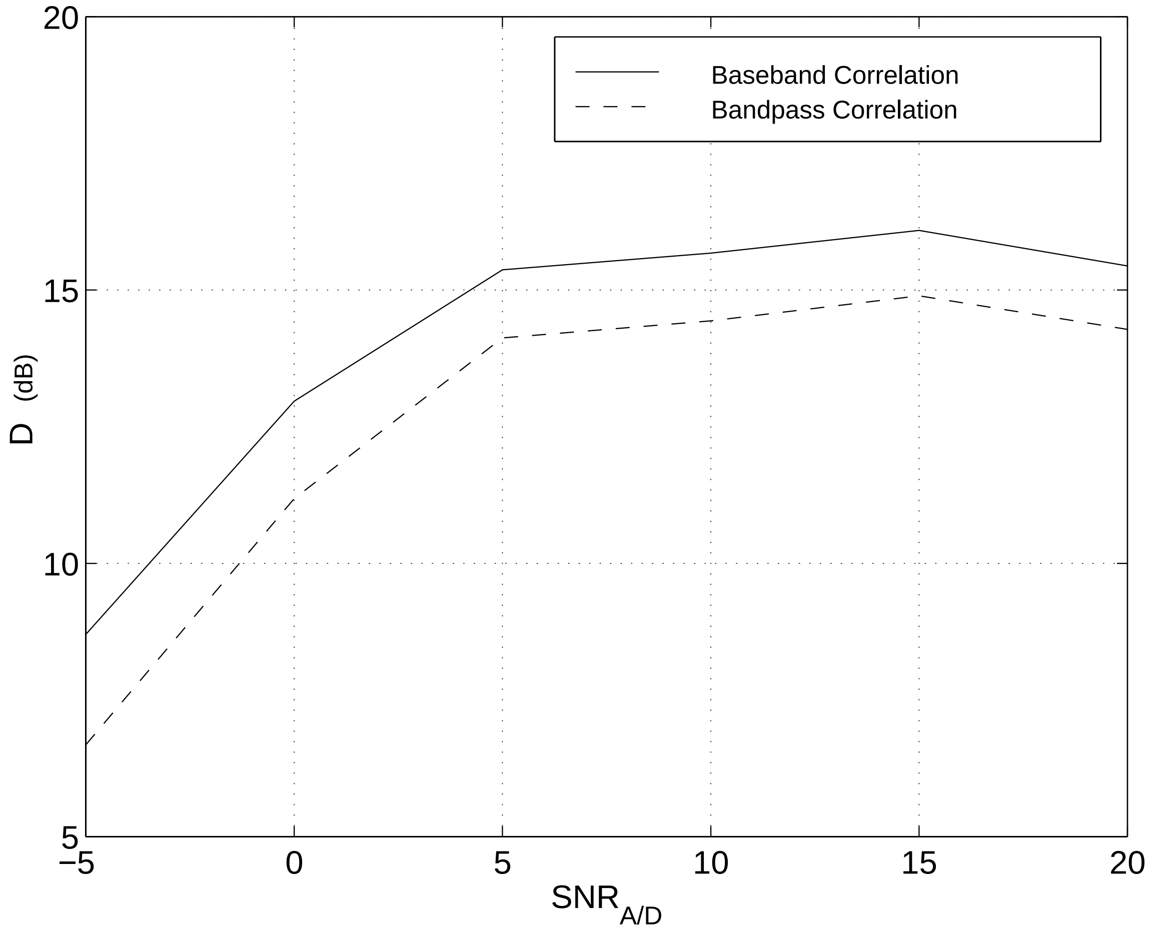

Figure 6.21: The discrimination metric D versus SNR$_{A/D}$, in dB. The parameters of the simulations which produced these results are given in Table C.14 in Appendix C.

Correlation Period, N_{corr}

It is expected that D increases monotonically with increasing N_{corr}, where N_{corr} is the number of chips over which the correlation is performed. However, designing the receiver using large values of N_{corr} leads to undesirably large values of T_{update} and the necessity for more stable local oscillators, as will be shown below. Therefore, it is desirable to keep N_{corr} as small as possible, yet maintain sufficient peak detection capability. The quantity D is plotted versus N_{corr} in Figure 6.22. The results show that adequate performance is attained with $N_{corr} = 1024$ (16 dB peak detection capability for the baseband correlation algorithm). $N_{corr} = 1024$ was also deemed to be acceptable in terms of the value of T_{update} incurred.

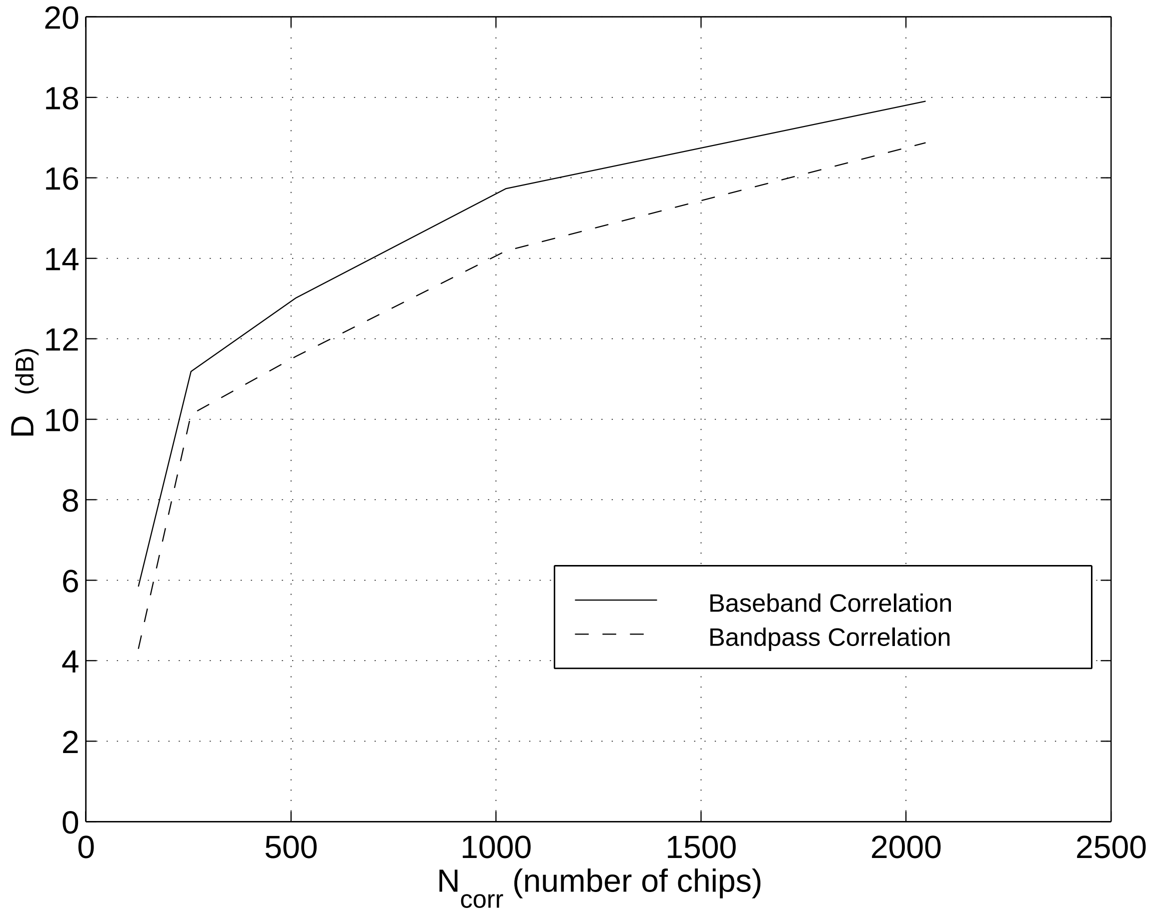

Figure 6.22: The discrimination metric D versus N_{corr}. The parameters of the simulations which produced these results are given in Table C.15 in Appendix C.

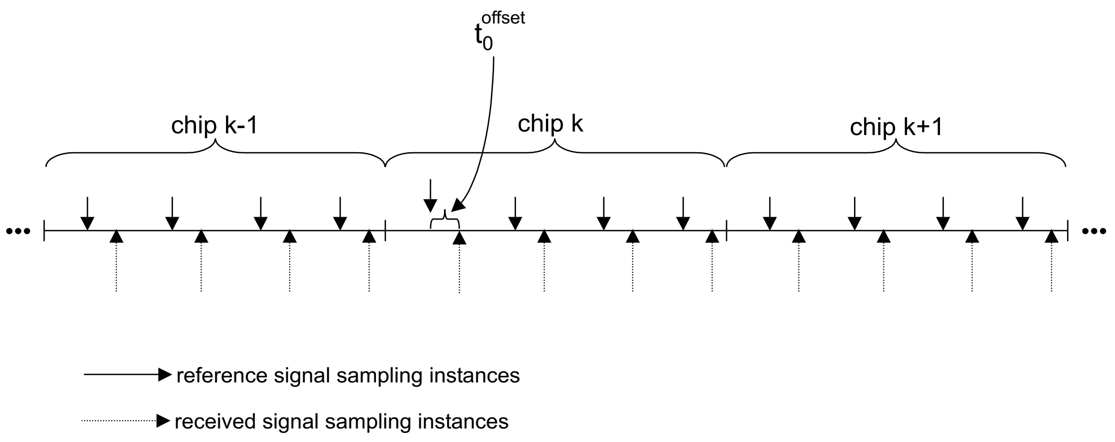
t_0^{offset}
chip k-1
chip k
chip k+1
reference signal sampling instances
received signal sampling instances

Figure 6.23: When the receiver samples the received pilot signal at instances other than those at which the reference pilot signal is sampled, a "sampling phase offset" is said to have occurred.

Sampling Phase Offset

The DSP maintains a sampled version of the pilot signal for the correlation algorithm. As shown in Figure 6.23, sampling phase offset occurs when the A/D samples the received pilot signal at instances other than those at which the DSP's version is sampled. D is plotted versus sampling phase offset, $t_0^{offset} \in [\Leftrightarrow 1/8T_c, 1/8T_c]$, in Figure 6.24. As the effect of this phenomenon is small, it has been neglected in further analyses.

Sampling Frequency Offset

Sampling frequency offset occurs when the sampling clock driving the A/D conversion does not run at the nominal rate. "Good" oscillators typically differ from their nominal frequencies by one to ten parts-per-million (ppm), whereas frequency standards typically differ from their nominal frequencies by ten to 100 parts-per-trillion.

Communications receivers may deal with sampling frequency offset by hardware which tracks and uses feedback of the error induced by the sampling frequency offset to correct the frequency of the voltage controlled clock which drives the analog-to-digital conversion. No attempt at such a feedback loop was attempted in this project, and this matter will be revisited in Chapter 8.

In this section, the effect of sampling frequency offset upon the peak detection capability, quantified by D, is analyzed. However, sampling frequency offset actually introduces a more onerous effect: apparent drift in the received phase of a pilot signal. To understand this

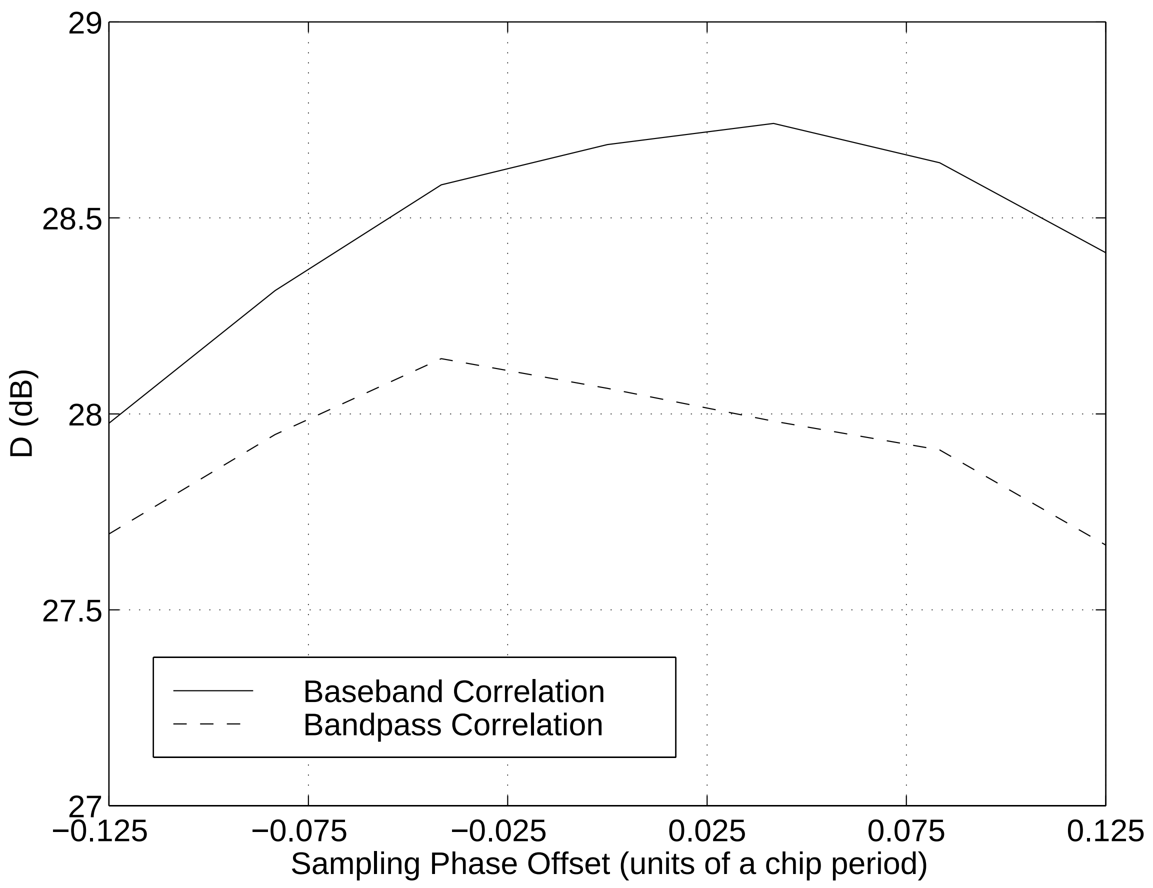

Figure 6.24: The discrimination metric D versus $t_0^{offset} \in [\Leftrightarrow 1/8T_c, 1/8T_c]$. The parameters of the simulations which produced these results are given in Table C.16 in Appendix C.

effect, consider a single pilot signal received through a static radio channel. Assume a receiver attempts to sample exactly one complete period of this pilot signal. Unfortunately, because its sampling clock is running too slow, it samples one complete period of the pilot signal and a small part of the following period of the pilot signal.[3] Because of the accumulation of the error, from the perspective of the receiver, the phase of the received pilot signal drifts in time with respect to the phase of its own pilot signal. Figure 6.25 depicts this phenomenon.

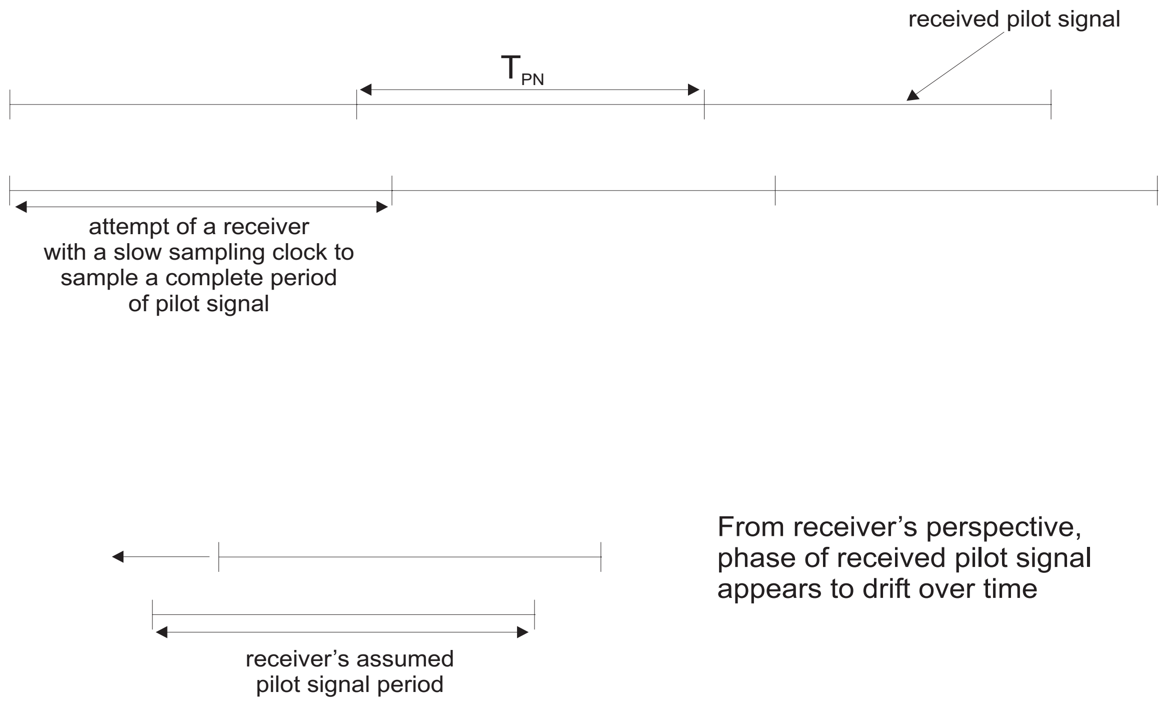
received pilot signal
T_{PN}
attempt of a receiver with a slow sampling clock to sample a complete period of pilot signal
From receiver's perspective, phase of received pilot signal appears to drift over time
receiver's assumed pilot signal period

Figure 6.25: Consider a single received pilot signal in a static radio channel and a receiver with a sampling clock running slower than the nominal rate. Then the receiver always samples a little more than one period of the pilot signal. From the perspective of the receiver, the phase of the received pilot signal appears to drift over time relative to its own.

In Equation B.123 of Appendix B, the fractional sampling frequency offset, δ_s, was defined as

$$\delta_s = \frac{f_s' \Leftrightarrow f_s}{f_s'}, \tag{6.74}$$

where $f_s{}'$ is the true sampling frequency and f_s is the nominal sampling frequency. Using

[3]A digital receiver would assume that one complete period of the pilot signal had been sampled when the appropriate number of samples had been taken. For instance, for a pilot signal with period T_{pilot}, and a sampling period of T_s, a digital receiver would assume one complete period had been sampled after T_{pilot}/T_s samples had been taken.

sampling periods, we have

$$T_s{}' = T_s\left(1 \Leftrightarrow \delta_s\right), \tag{6.75}$$

where $T_s{}'$ is the true sampling period and T_s is the nominal sampling period.

Using these formulas, it can be reasoned that the absolute value of the total phase error in seconds after the update period, T_{update}, is $|\delta_s T_{update}|$. An upper bound on the update period, which assumes only one correlation output is obtained every $N_s N_{corr}$ samples, is

$$T_{update} < T_{corr}\frac{T_{pilot}}{T_s}, \tag{6.76}$$

where T_{corr} is the correlation period in seconds, T_{pilot} is the period of the pilot signal in seconds, and T_s is the sampling period in seconds. Requiring the phase drift in seconds after one update period to be smaller than one sampling period in seconds, we have

$$|\delta_s T_{update}| < T_s. \tag{6.77}$$

Using the upper bound on the update period provided in Equation 6.76, and canceling factors of T_s, we have

$$|\delta_s| < \frac{1}{2^{15} N_s^2 N_{corr}}. \tag{6.78}$$

For $N_s = 4$ and $N_{corr} = 1024$, we have $|\delta_s| < 0.002$ ppm. Such a level a stability is not available with standard off-the-shelf oscillators. Therefore, it is concluded that the receiver needs to provide for some means of controlling for the sampling frequency offset using either a tracking loop, a frequency standard, or possibly some external synchronization signal (e.g., the Global Positioning System (GPS)). Equation 6.78 also shows that using a smaller value of N_{corr} admits the use of a less stable (and therefore less expensive) sampling clock. A smaller value of N_{corr} also decreases the update period T_{update}. The tradeoff incurred is that using a smaller value of N_{corr} leads to a decreased capability to detect correlation peaks, as was discussed in Section 6.5.3.

In Figure 6.26, the discrimination metric D is plotted versus sampling frequency offset. This figure roughly shows that the loss in discrimination capability can be kept below 1 dB if the sampling clock is within 20 ppm of its nominal sampling rate.

Sampling Frequency Jitter

Sampling frequency jitter refers to the uncertainty in the exact sampling instances, and is introduced either by noise in the sample-and-hold amplifier of the A/D converter or by a

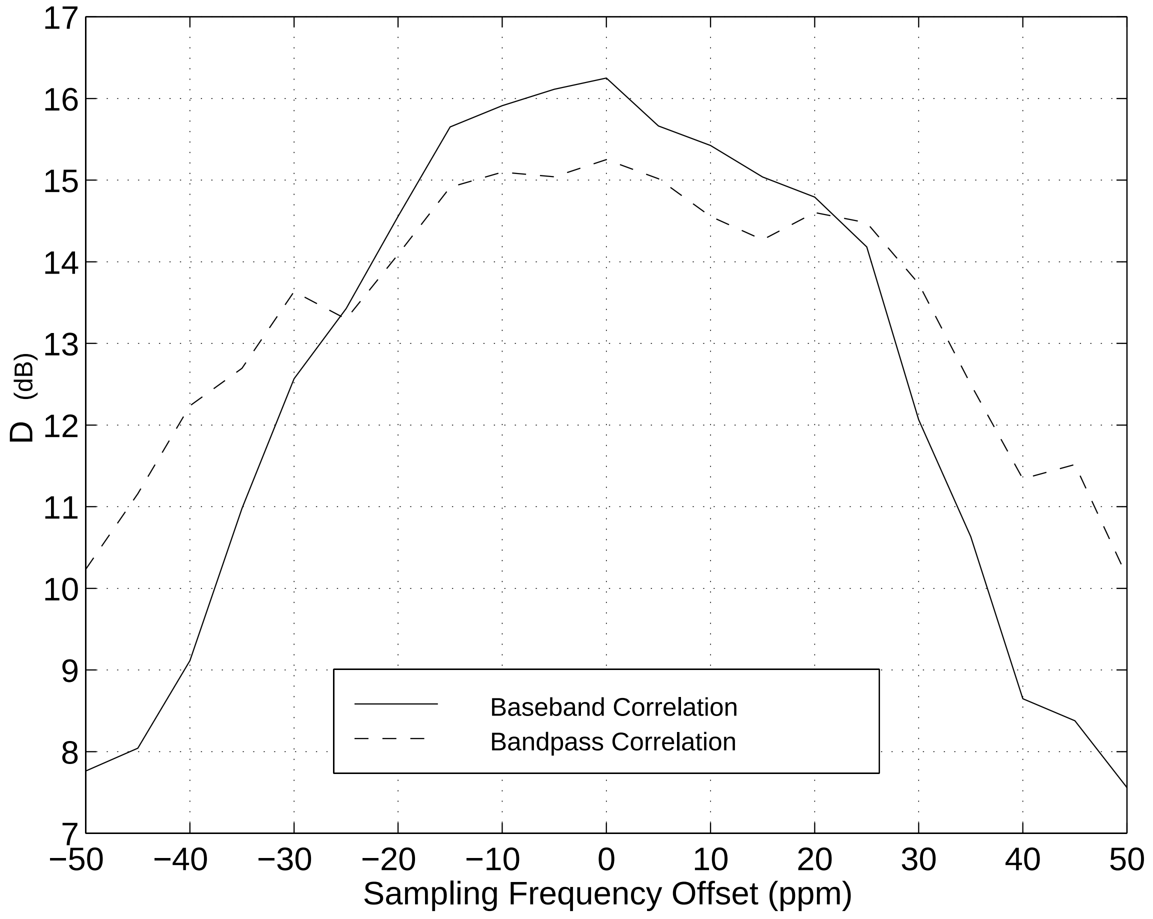

Figure 6.26: The discrimination metric D versus sampling frequency offset. The parameters of the simulations which produced these results are given in Table C.17 in Appendix C.

noisy sampling clock. In this thesis (see Section B.3.2), sampling jitter was modeled by adding a small random component, which was a random number chosen from a zero-mean Gaussian distribution with a specified variance, to the fixed sampling period component.

Sampling frequency jitter has the most adverse effects upon high frequency signals, since for these signals a small difference in time may lead to a large change in the voltage of the signal. To understand how sampling frequency jitter limits the maximum frequency which can be properly digitized, consider the highest frequency component, f_{max}, of a bandlimited signal. Suppose it is required that the signal voltage never change more than one quantization level over the aperture of the sampling uncertainty, which is labeled here as Δt. This requirement is tantamount to requiring that the maximum rate of change of the frequency component f_{max} be less than ΔV/Δt, where ΔV is the voltage difference of a single quantization level, or

$$\Delta \mathrm{V} = \frac{V_{FS}}{2^n}, \tag{6.79}$$

where $V_{FS} = V_{max} \Leftrightarrow V_{min}$ is the full scale voltage of the A/D converter and n is the number of bits of the A/D converter. The maximum rate of change of a full-scale sinusoid of frequency f_{max} occurs as the sinusoid is passing through its average value and is $V_{FS}\pi f_{max}$. Thus,

$$V_{FS}\pi f_{max} < \frac{V_{FS}}{2^n \Delta \mathrm{t}} \tag{6.80}$$

or

$$f_{max} < \frac{1}{2^n \pi \Delta \mathrm{t}}. \tag{6.81}$$

Equation 6.81 is very useful in digital receiver design. It shows that the maximum allowable input frequency to an A/D converter is inversely proportional to the sampling frequency jitter. It further shows that, for a given sampling uncertainty, a smaller number of A/D bits allows the maximum input frequency to be larger. As a design guideline, to keep the effects of sampling frequency jitter to a minimum, the maximum input frequency to the A/D converter should be kept as low as possible.

In Figure 6.27, D is plotted versus sampling frequency jitter in ps. This figure shows that the degradation in discrimination capability can be kept below 1 dB by keeping the sampling frequency jitter below 30 or 40 ps.

Fixed Point Effects

Degradation in discrimination capability may arise from the quantization of the signal incurred during the analog-to-digital conversion process or by the integer arithmetic employed

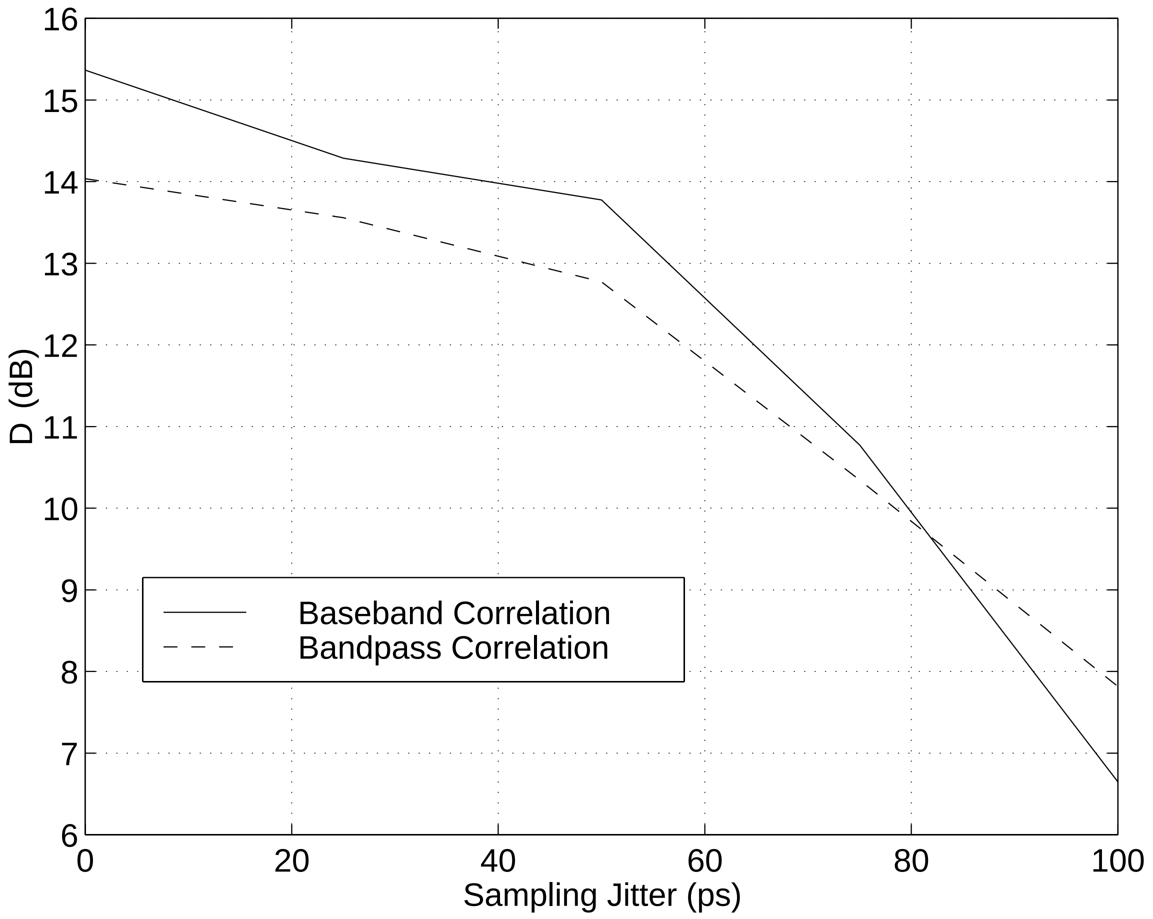

Figure 6.27: The discrimination metric D versus sampling frequency jitter in ps. The parameters of the simulations which produced these results are given in Table C.18 in Appendix C.

by the DSP (see Section 7.3). These effects were studied by varying the numbers of bits used to quantize the simulated signal and the locally generated pilot signal. The effects of these variations on D were examined, and are tabulated in Tables 6.6 and 6.7. The results show that the bandpass detection method is more sensitive to quantizaton effects. This is possibly due to the fact that the bandpass detection method requires more than twice the number of computations of the baseband detection method. Since the error incurred by quantization introduces an independent error on each computation, it is expected that the mean-square error would be largest for the algorithm which requires the greatest number of computations. The bandpass correlation method also experienced problems with overflow which were not experienced with the baseband correlation method[4].

Table 6.6: D (in dB) for different numbers of bits used by the A/D (n_1) and DSP (n_2) for the baseband correlation method. The parameters for the simulations which produced these results can be found in Table C.19 of Appendix C.

	n_1		
n_2	4	8	12
4	15.7	15.7	16.0
8	15.8	16.4	15.7
12	15.8	15.5	16.1

Table 6.7: D (in dB) for different numbers of bits used by the A/D (n_1) and DSP (n_2) for the bandpass correlation method. The symbol "–" in a table entry indicates that no result was obtained due to overflow. The parameters for the simulations which produced these results can be found in Table C.19 of Appendix C.

	n_1		
n_2	4	8	12
4	8.5	13.6	15.4
8	14.0	15.5	–
12	15.1	–	–

[4] As explained in Appendix B, all signal processing computations were performed in fixed point arithmetic to simulate the actual hardware employed for the implementation. An overflow occurred when some number exceeded the range $[-2^{31}, 2^{31}-1]$ during the computations.

Frequency Offset

Oscillator instability and Doppler shift create differences between the frequency of the oscillator used to downconvert the signal in the receiver and the carrier frequency of the received signal. Any such frequency offset results in an imperfect downconversion, with the center frequency of the received signal being translated to a final IF which is different from that which is desired.

D is plotted versus frequency offset in ppm (relative to f_{IF}) in Figure 6.28. This figure essentially illustrates the frequency responses of the two correlation algorithms. The baseband correlation algorithm has nulls in its frequency response. This is due to the integration which is performed in the algorithm (see Figure 5.12). The impulse response of this integration is $\mathrm{rect}\,(t/T_{corr})$, where $T_{corr} = T_c N_{corr}$ and T_c is the chip period. The frequency response of the integration has a $\mathrm{sinc}(\pi f T_{corr})$ dependency, where f is the frequency of the input signal.

Note that the null in the frequency of the baseband algorithm occurs at approximately 60 ppm. This corresponds to a frequency of $(60 \times 10^{-6})(20.8896 \times 10^{6}) \approx 1200$ Hz. This cross-checks with the first null in the frequency response of the integration, which should occur at a frequency of $1/T_{corr} = 1.2288 \times 10^{6}/1024 = 1200$ Hz.

The bandpass correlation method does not have nulls in its frequency response. Following the bandpass filtering required in the algorithm (see Figure 5.13), squaring the bandpass signal always produces a DC component. The degradation in the magnitude of this DC component due to frequency offset occurs because the center frequency of the signal does not lie at the center of the bandpass filter. As a result, the bandpass filter attenuates the signal, resulting in a smaller DC component after the squaring operation.

It is concluded that the bandpass correlation method is preferred over the baseband correlation method in the presence of frequency offset. If the baseband correlation algorithm is implemented, the system design should ensure that the relationship between the correlation period, T_{corr}, and the frequency offset, $\delta_f f_{IF}$, is such that

$$T_{corr} \ll \frac{1}{\delta_f f_{IF}}. \tag{6.82}$$

This ensures that frequency offset does not have a deleterious effect upon the algorithm. An alternative yet equivalent explanation is that ensuring that Equation 6.82 is satisfied ensures that the local oscillator(s) is (are) stable over the correlation period.

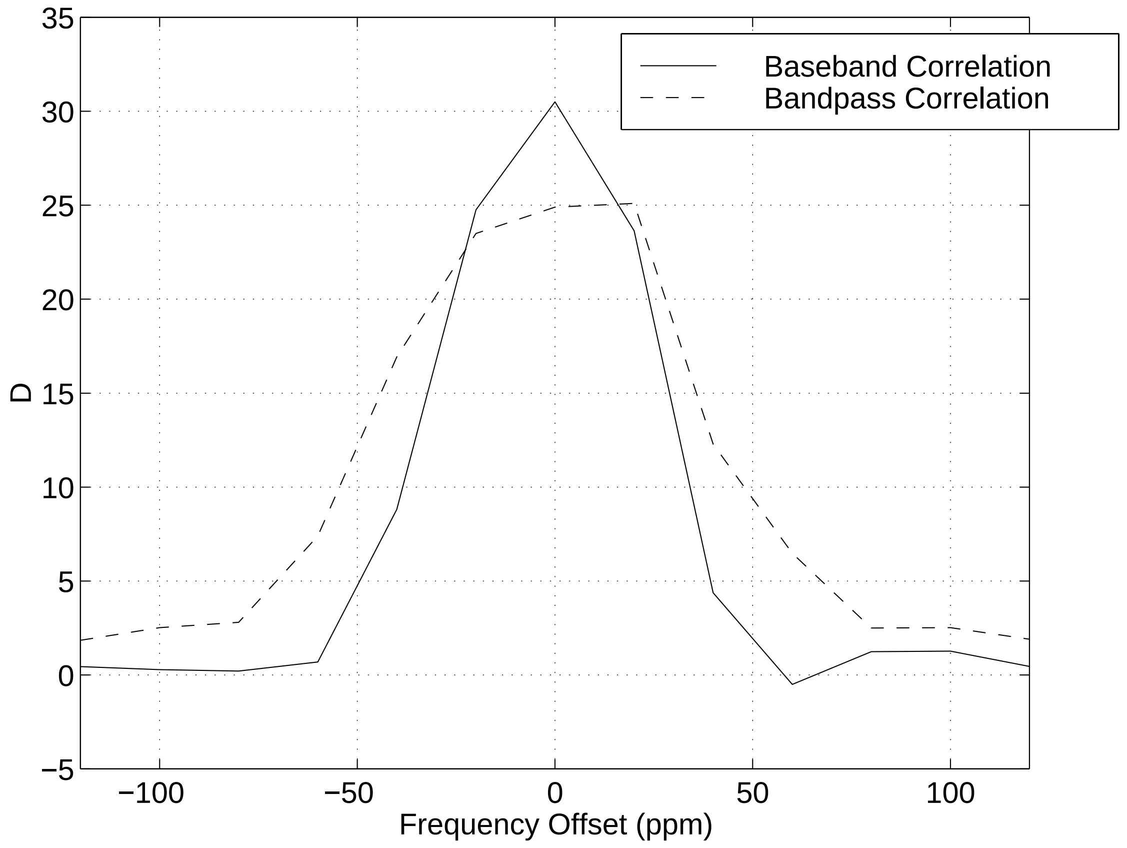

Figure 6.28: The discrimination metric D versus frequency offset in ppm. As discussed in the text, the first null in the response of the baseband correlation algorithm occurs at the expected location. The parameters for the simulations which produced these results can be found in Table C.20 in Appendix C.

Local Oscillator Phase Noise

In general, a local oscillator will produce the following steady-state signal, $c(t)$ [28, p. 650]:

$$c(t) = A_c \left[1 + a(t)\right] \cos\left(\omega_c t + \phi(t) + \frac{\alpha t^2}{2}\right), \tag{6.83}$$

where A_c is a constant, $a(t)$ describes variations in the output amplitude, ω_c is the free-running frequency, $\phi(t)$ is a random process describing the phase jitter of the oscillator, and the term $\alpha t^2/2$ (α constant) describes the long-term frequency drift of the oscillator. In the analysis presented here, the system is analyzed over very short periods of time; therefore, the term $\alpha t^2/2$ is neglected. Furthermore, since oscillators are generally feedback amplifiers operating in saturation mode, it is reasonable to neglect any amplitude instabilities in favor of the instabilities caused by phase jitter. Thus, the following can be written

$$c(t) = A_c \cos\left(\omega_c t + \phi(t)\right). \tag{6.84}$$

The local oscillator phase noise was simulated by generating samples of the random process $\phi(t)$. This simulation depended upon an accurate reproduction of both the spectral properties and the second-order moment of $\phi(t)$. The method by which this was accomplished is discussed in detail in Section B.4.2. D is plotted versus RMS phase error in degrees in Figure 6.29. This figure shows that the discrimination capability falls sharply after the RMS phase error increases above $0.05°$.

Multipath

Due to non-trivial cross-correlations between received multipath components and the locally generated pilot signal, multipath will introduce interference which serves to obfuscate the detection of pilot signals.

Simulations were performed in which the number of multipath components was varied between one and six. The hypothetical multipath profiles in Figure 6.30 were used. In Figure 6.31, the discrimination metric D is plotted versus the number of multipath components. (The metric was computed for the first multipath component.)

Multiple Base Stations

Similar to multipath, signals of other base stations at the input of the receiver will introduce interference due to non-trivial cross-correlations with the locally generated pilot signal. The

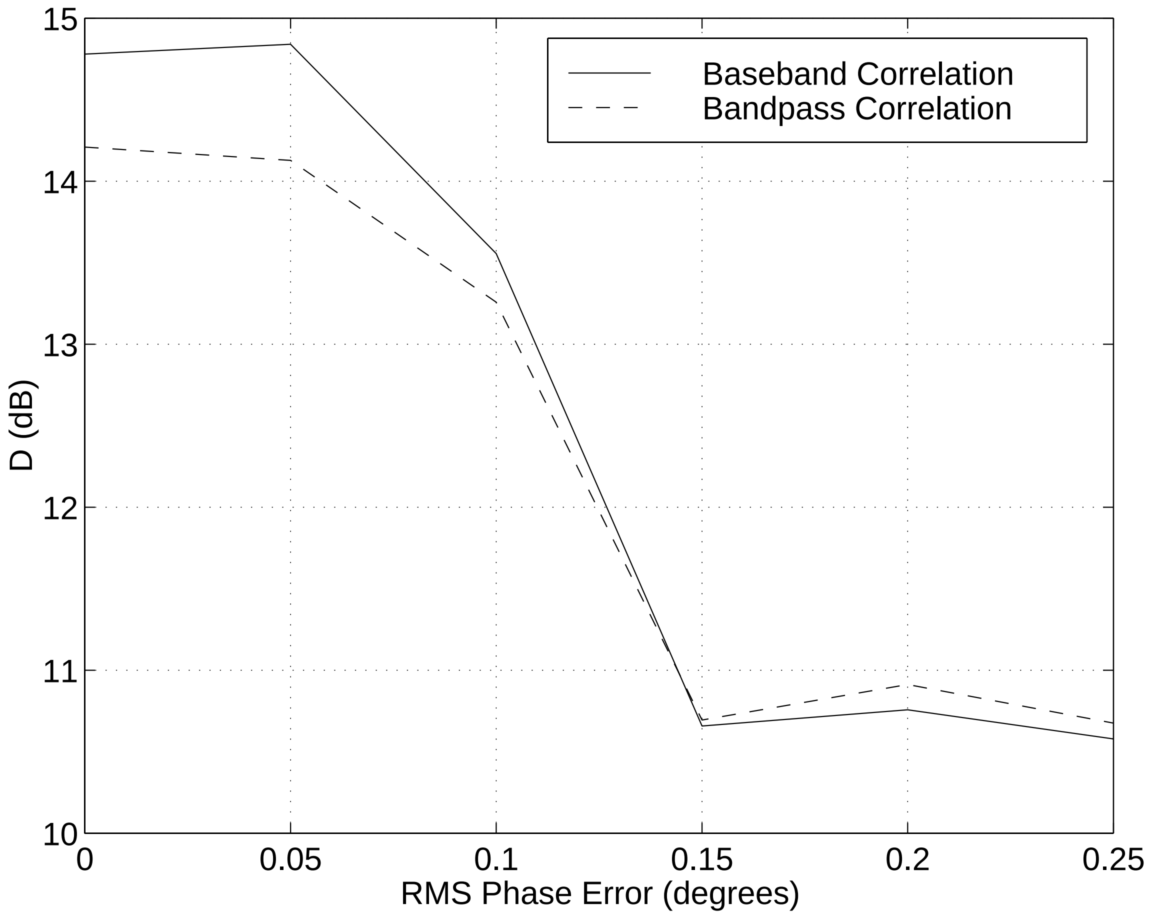

Figure 6.29: The discrimination metric D versus local oscillator RMS phase error in degrees. The parameters for the simulations which produced these results are given in Table C.21 in Appendix C.

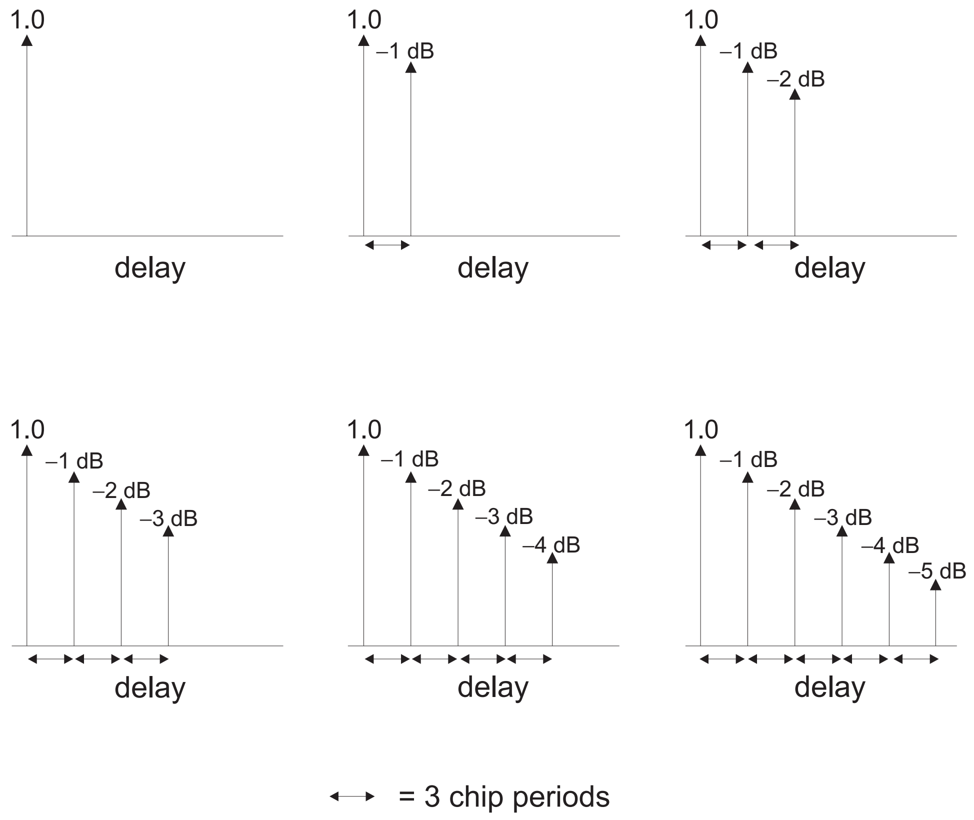

Figure 6.30: The hypothetical multipath profiles used in the simulations to gauge the effect of multipath upon pilot signal detection.

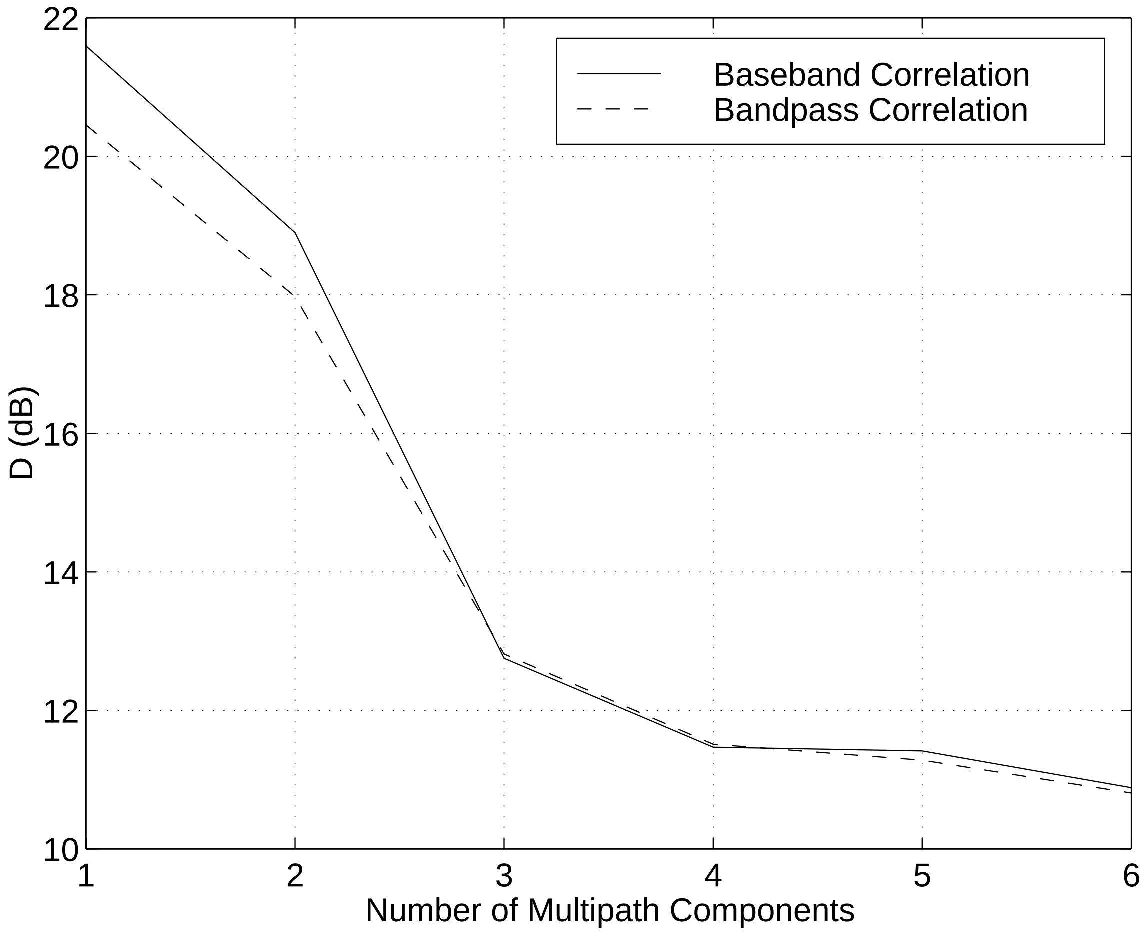

Figure 6.31: The discrimination metric D versus the number of multipath components present in the signal at the input of the receiver. The multipath profiles presented in Figure 6.30 were used in the simulations. The parameters of the simulations which produced these results are given in Table C.22 in Appendix C.

difference between these two cases is that all multipath components from the same base station carry identical multiuser interference, whereas signals from other base stations carry different multiuser interference.

Simulations were performed in which the number of base stations and the number of multipath components from each base station were each varied independently between one and three. The pilot signals from the base stations were spaced 256 chips apart in phase, and the multipath profiles used were identical to the first three provided in Figure 6.30. Results on D for the baseband correlation method are given in Table 6.8, and results on D for the bandpass correlation method are given in Table 6.9.

Table 6.8: The discrimination metric D in dB for the baseband correlation method for different numbers of base stations and different numbers of multipath components from each base station. The parameters of the simulations which produced these results are given in Table C.23 in Appendix C.

Multipath Components	**Base Stations**		
	1	**2**	**3**
1	21.4	18.6	17.2
2	19.0	16.1	12.5
3	12.2	12.0	12.1

Table 6.9: The discrimination metric D in dB for the bandpass correlation method for different numbers of base stations and different numbers of multipath components from each base station. The parameters of the simulations which produced these results are given in Table C.23 in Appendix C.

Multipath Components	**Base Stations**		
	1	**2**	**3**
1	20.2	17.5	16.4
2	18.1	15.1	12.0
3	12.1	11.2	11.5

6.6 Analog Tuner Requirements

6.6.1 Gain

The maximum overall gain required of the analog tuner, G_{tot}, is the difference between the signal power required at the input of the A/D converter ($S_{A/D}$) (in dB) and the minimum detectable signal power (MDS) (in dB) at the receiver input. In Section 6.5.3, it was shown that $\mathrm{SNR}_{A/D} > 5$ dB provides adequate receiver performance. This leads to a MDS of

$$\mathrm{MDS} = N_{th} + \mathrm{NF} + 10 \text{ dB}, \tag{6.85}$$

where N_{th} is the thermal noise floor over the bandwidth of the receiver and NF is the noise figure (in dB) of the receiver. In the following, NF = 15 dB is conservatively estimated. Since the bandwidth of the signal of interest (see Section 2.5) is 1.23 MHz, the thermal noise floor is

$$N_{th} = 10\log_{10}(kTB) = \Leftrightarrow 174 \text{ dBm} + 10\log_{10}\left(1.23 \times 10^6\right) = \Leftrightarrow 113 \text{ dBm}. \tag{6.86}$$

Hence, the MDS is

$$\mathrm{MDS} = N_{th} + \mathrm{NF} + 5 \text{ dB} = \Leftrightarrow 113 \text{ dBm} + 15 \text{ dB} + 5 \text{ dB} = \Leftrightarrow 93 \text{ dBm}. \tag{6.87}$$

From the studies of the AGC presented in Section 5.2.1, a target mean-square voltage at the input of the A/D converter of 2 V^2 was adopted. Since the input resistance to the analog-to-digital converter is 50Ω (see Section 7.1.3), the target signal power at the input of the A/D converter is thus

$$S_{A/D} \;=\; 10\log_{10}\left(\frac{2}{50}\right) = 16 \text{ dBm}. \tag{6.88}$$

Therefore, the maximum overall gain required is

$$G_{tot} = S_{A/D} \Leftrightarrow \mathrm{MDS} = 16 \text{ dBm} \Leftrightarrow (\Leftrightarrow 93 \text{ dBm}) = 109 \text{ dB}. \tag{6.89}$$

6.6.2 Downconversion Architecture

In practice, in a receiver it is best to limit the amount of gain at any one frequency to 30 or 40 dB to thwart the possibility of oscillations which can be caused by small amounts of accidental feedback. Since the receiver required a total gain of 109 dB, a multiple stage downconversion architecture was thus selected. The minimum number of IF's, given the above requirement for maximum overall gain in any one stage, was selected to be two. A dual downconversion architecture was thus adopted. The dual downconversion architecture is a good architecture in terms of both selectivity and sensitivity.

In Section 6.4, the rationale underlying the choice of the final IF, f_{IF}, as 20.8896 MHz was given. For the first IF, it was found during the course of this project that designers of IS-95A-compliant equipment have begun using two different IFs: 210.38 MHz and 85 MHz. Thus, these IFs have become *de facto* standards, and a great deal of components, especially amplifiers and filters, are currently being manufactured for operation at these frequencies. Since a subsampling architecture was desired, the larger of these was selected for the first IF. With this selection, the image reject filter in the first downconversion is less restrictive. In fact, since the center frequency of the desired signal is between 1930 and 1960 MHz, the selection of IFs at 210.38 MHz and 20.8896 MHz allows the image reject filters for both stages of downconversion to have approximately the same fractional bandwidth.

Filters, amplifiers, mixers, and frequency synthesizers were identified which could realize the gain requirement of the dual downconversion architecture adopted, while providing adequate noise figure and third-order intercept performance. The details of this are provided in [29].

6.6.3 Maximum Range and Noise Figure

Following Section 3.1, the received power in dBm at T-R separation d, $P_r(d)$, is

$$P_r(d) \quad = \quad G_t + G_r + P_t \Leftrightarrow \mathrm{PL}(d), \tag{6.90}$$

where G_t is the gain in dB of the transmitting antenna, G_r is the gain in dB of the receiving antenna, P_t is the power in dBm out of the transmitter, and PL(d) is the path loss in dB.

First, the minimum value of G_t is estimated. Base station coverage is usually sectorized so that each antenna illuminates roughly one-third of a cell, and base station antenna heights range between 15 m and 60 m. From Figure 6.32, if a single antenna is to illuminate one-third of a circle of radius R, then its minimum gain, G_t, should be

$$G_t = \frac{4\pi}{\Omega_A} = \frac{4\pi}{\frac{2\pi}{3}\int_0^{\alpha} \sin(\theta)\, \mathrm{d}\theta} = \frac{6}{1 \Leftrightarrow \cos(\alpha)}, \tag{6.91}$$

where

$$\alpha = \tan^{-1}\left(\frac{R}{h_t}\right), \tag{6.92}$$

R is the cell radius, and h_t is the antenna height. Since $R \gg h_t$, we find that $G_t \approx 6\,(\approx)\,8$ dB, which is a conservative estimate of the gain to be expected of the transmitting antenna. For the receiving antenna, a dipole is assumed; therefore, $G_r = 2$ dB.

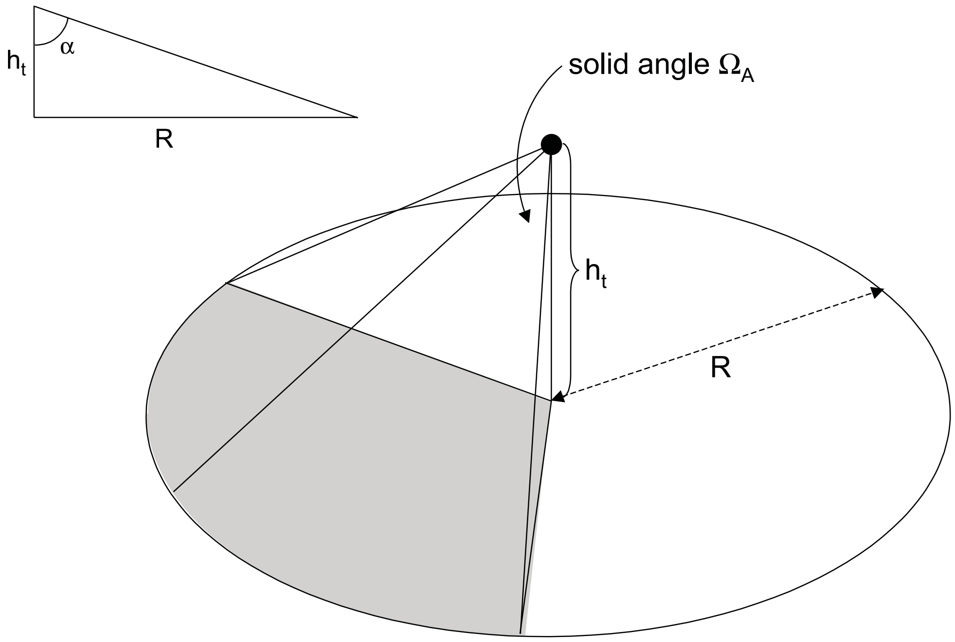

Figure 6.32: Diagram used to compute the minimum expected base station antenna gain for a sectorized cell.

It was assumed that the power out of the transmitter was 2 W, or 33 dBm. Substituting this value, along with the values for the antenna gains, into Equation 6.90, it is found that $P_r(d)$ is

$$P_r(d) \quad = \quad 43 \text{ dBm} \Leftrightarrow \text{PL}(d). \tag{6.93}$$

Since PL(d) is monotonically increasing with distance, the maximum T-R separation, d_{max}, is achieved when $P_r(d) = \text{MDS}$. Thus,

$$N_{th} + \text{NF} + 5 = 43 \text{ dBm} \Leftrightarrow \text{PL}(d_{max}), \tag{6.94}$$

or, substituting $N_{th} = \Leftrightarrow 113$ dBm, it is found that

$$151 \Leftrightarrow \text{NF} = \text{PL}(d_{max}). \tag{6.95}$$

The path loss models of Section 3.1 were used to estimate PL(d_{max}). Specifically, the Devasirvatham model from [19], the Feuerstein obstructed model (OBS) for antenna height 13.3 m, and the Feuerstein line-of-sight model (LOS) for antenna height 13.3 m were used. In Figure 6.33, the maximum T-R separation, d_{max}, is plotted versus receiver noise figure, NF, for these three models.

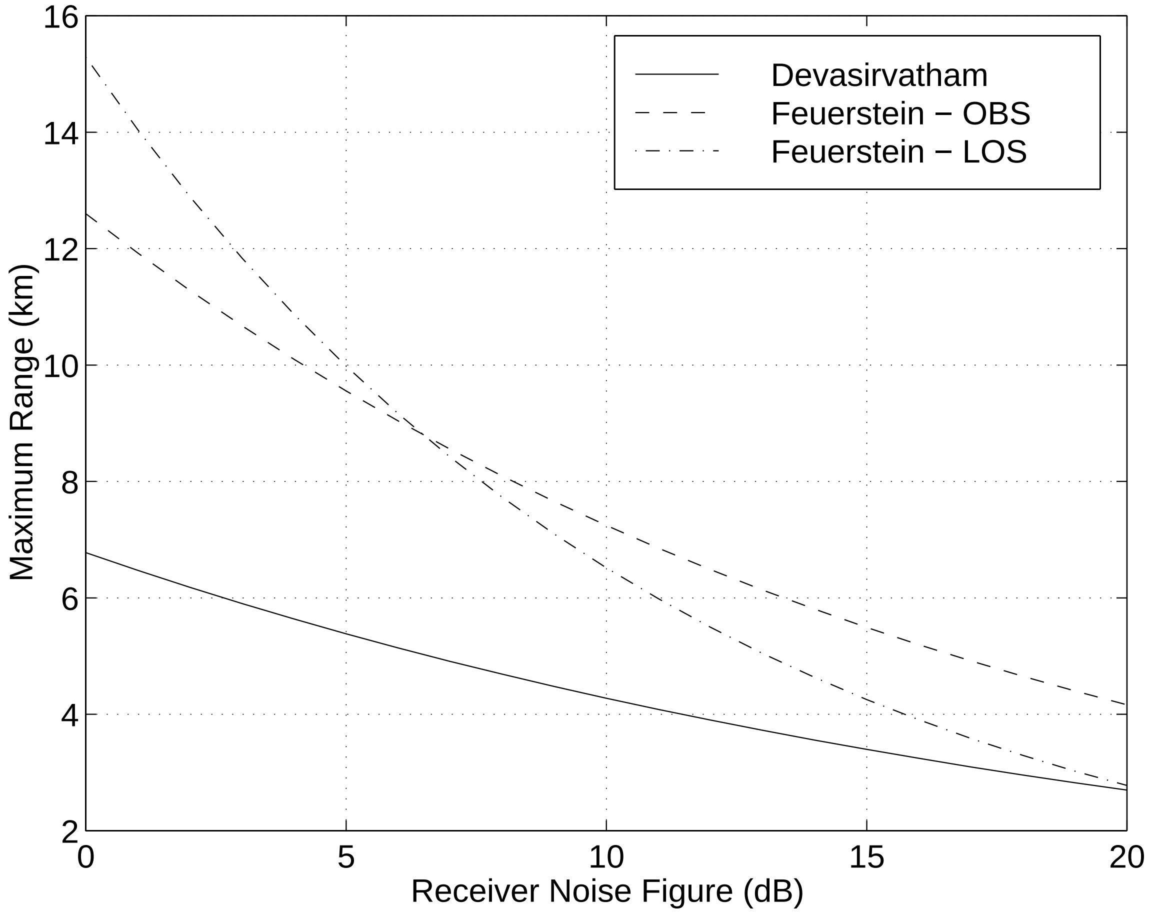

Figure 6.33: The maximum T-R separation versus the receiver noise figure. The path loss models used were those detailed in Chapter 3. It was assumed that the power out of the transmitter was 2 W (33 dBm), that the transmitting antenna gain was 8 dB, and that the receiving antenna gain was 2 dB.

6.6.4 Spurious Free Dynamic Range, SFDR

The spurious free dynamic range (SFDR) of a receiver is the range of input power levels over which (1) third-order intermodulation products never appear above the MDS (as measured at the output of the receiver) and (2) the desired signal never falls below the MDS (as measured at the output of the receiver). This receiver figure-of-merit is discussed in detail in Appendix

F. Setting specifications on the SFDR in this thesis would have required modeling the effect of intermodulation products produced by the analog tuner upon the digital signal processing algorithms. Unfortunately, due to lack of time, such an analysis was not performed. However, in mobile radio applications, a SFDR of 60 dB or more is usually considered adequate [30, p. 83]. In Chapter 8, it is recommended that this line of investigation be followed in future studies.

Chapter 7

Implementation of a Pilot Channel Scanning Receiver

This chapter presents the hardware implementation of the pilot signal scanning receiver studied in previous portions of this thesis. The chapter begins with a discussion of details related to the A/D converter. The A/D was interfaced to the DSP using custom digital circuits, and the design and implementation of these circuits is presented next. This is followed by a discussion of the digital signal processing hardware employed in the project, and a detailed presentation of the actual implementation of the scanning correlation and AGC algorithms on the DSP hardware. The chapter concludes with a discussion of the testing and verification of the receiver.[1]

7.1 Analog-to-Digital Conversion

7.1.1 Key Parameters

In selecting an off-the-shelf A/D converter for the implementation of the pilot signal scanning receiver, the important parameters which were considered were as follows:

- maximum sampling rate – greater than 4.9152 MHz was required,
- input bandwidth – greater than 20.8896 MHz was required, and
- sampling jitter – less than 20 ps RMS was required.

The studies on fixed point effects in Chapter 6 showed that the number of bits of quantization resolution for the baseband correlation method (which was implemented in this thesis) was

[1]Another student on the project, Neal Patwari, implemented the analog tuner. Therefore, this topic is not covered here, but is covered in the companion report [29].

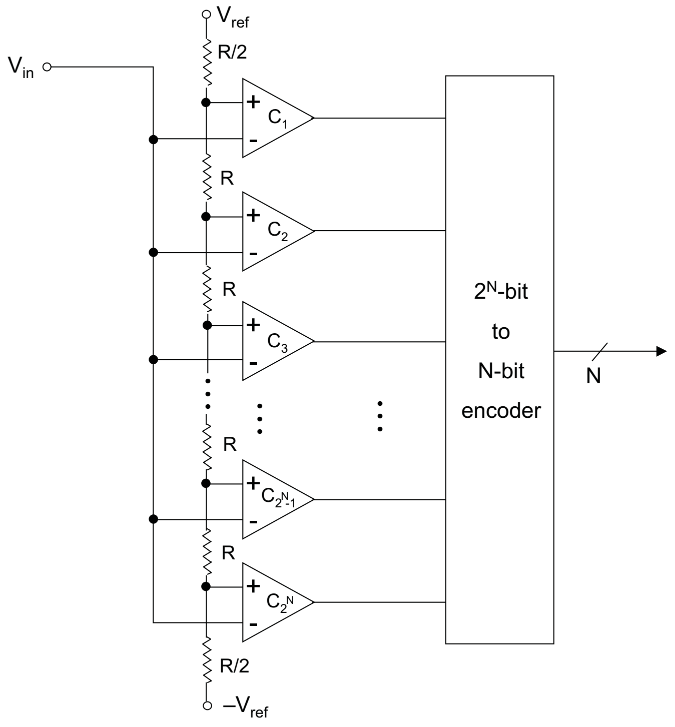

Figure 7.34: A flash A/D converter.

not critical. Furthermore, input voltage full scale range was not deemed important, since any practical voltage range could be accommodated by adjusting the gain of the analog tuner. Lastly, no analyses were undertaken on A/D nonlinearities or spurious products; therefore, no specifications on these parameters could be fixed.

7.1.2 Subranging A/D Converters

The AD9220 A/D converter from Analog Devices was selected for use in this thesis. This A/D converter uses the subranging digitization technique. Subranging A/D converters use two or more flash A/D converters to digitize the signal [31, 32, 33]. A schematic diagram of a flash A/D converter is shown in Figure 7.34. A N-bit flash A/D converter uses a bank of 2^N comparators and a resistor network to quantize the signal. The "thermometer" output[2] is encoded to form a N-bit output.

A schematic diagram of a subranging A/D converter is shown in Figure 7.35. The first flash A/D converter provides a coarse, N_1-bit digitization of the signal. This digitization

[2]The name thermometer is used because the outputs of the comparator network have identical polarity from the lowest one up to a certain point. For all comparators higher than this this point, the outputs are of the opposite polarity. Therefore, the output resembles the mercury in a thermometer.

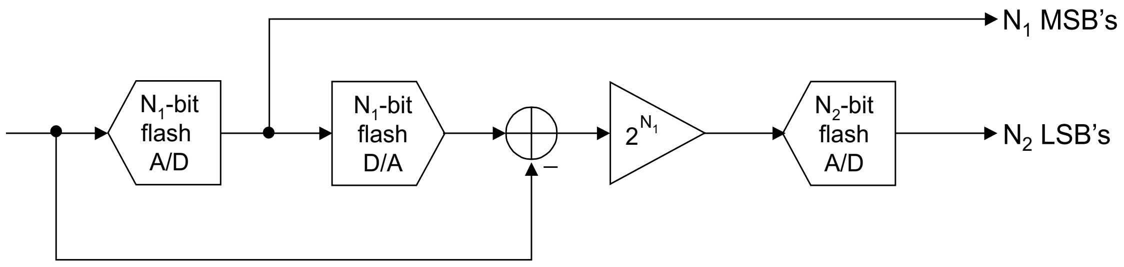

Figure 7.35: A two-step subranging A/D converter.

is converted back into analog form, and an error signal with the original analog signal formed. This error signal is amplified by a factor of 2^{N_1} and then digitized by a second, N_2-bit A/D converter. The digitization of the first A/D converter provides the N_1 most significant bits (MSB's) and the digitization of the second A/D converter provides the N_2 least significant bits (LSB's). The number of comparators needed for digitization with the two-step subranging A/D converter illustrated is $2^{N_1} + 2^{N_2}$. By comparison, to achieve $N_1 + N_2$ bits of resolution using a single flash A/D converter $2^{N_1+N_2}$ comparators would be required. Thus, the number of comparators needed for digitization is significantly reduced in subranging A/D converters, as compared to flash A/D converters of the same resolution.

7.1.3 The AD9220 A/D Converter

The AD9220 from Analog Devices is a 12-bit A/D converter having a maximum sample rate of 10 MHz, an input bandwidth of about 60 MHz, and sampling jitter specification of 4 ps RMS[3]. The A/D was mounted on an evaluation board with accompanying electronics which level-shifted the signal from the range $[\Leftrightarrow 2.5, 2.5]$ V at the input connector to $[0, 5]$ V at the actual input of the A/D. The AD9220 was configured for single-ended AC coupling with an input full scale range of 0 V to 5 V. This configuration, while not the optimum for subsampling applications, is the optimum in terms of noise performance [34].

The output of the A/D is shown in Figure 7.36, where the level shifting at the input required in practice for the AD9220 has been disregarded. (For simplicity, a three-bit A/D is depicted.) Each level between $\Leftrightarrow 2.5$ V and 2.5 V is assigned an integer number. The three-bit converter depicted can be transformed into a two-bit converter by discarding the least significant bit (LSB). In this project, the 12-bit AD9220 was converted into an eight-bit converter by discarding the four LSBs.

[3]This specification does not account for any jitter introduced by the sampling clock.

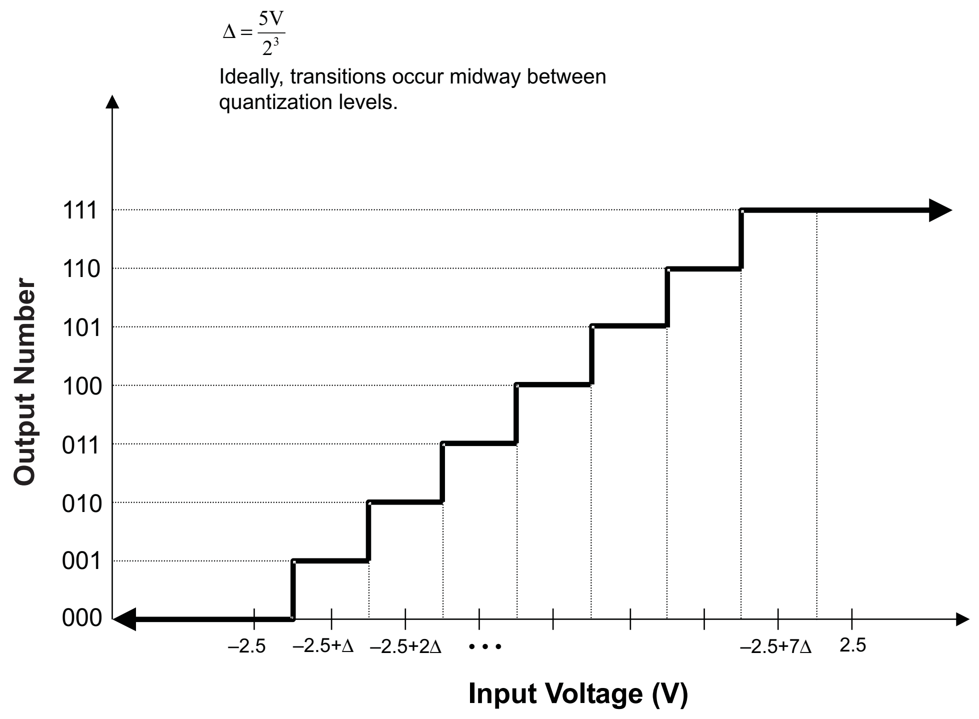

Figure 7.36: The output levels of the A/D converter.

From Figure 7.36, note that the output of the A/D converter is not provided in two's complement notation. Moreover, the eight-bit samples had to feed a 16-bit data bus on the DSP. Conversion of the eight-bit output of the A/D to 16-bit two's complement notation was accomplished by inverting and sign extending the MSB of the A/D output. This conversion was done in hardware using the logic shown in Figure 7.37.

7.2 A/D–DSP Interface

Samples are produced by the A/D converter at a rate of 4.9152 MHz. This rate is much too large to interrupt the DSP to transfer each sample from the A/D converter to the DSP. Therefore, the samples must be buffered and transferred to the DSP in blocks to minimize the impact of the overhead due to interrupts. The A/D–DSP interface circuit which accomplished this buffering and data transfer is the subject of this section.

Figure 7.38 shows the functional blocks of the interface circuit. The circuit has three basic parts: the data storage and retrieval part, the signal windowing part, and the interface memory part. The data storage part accepts the data from the A/D converter and stores it for transferral to the DSP. The signal windowing part "windows" the signal, as discussed

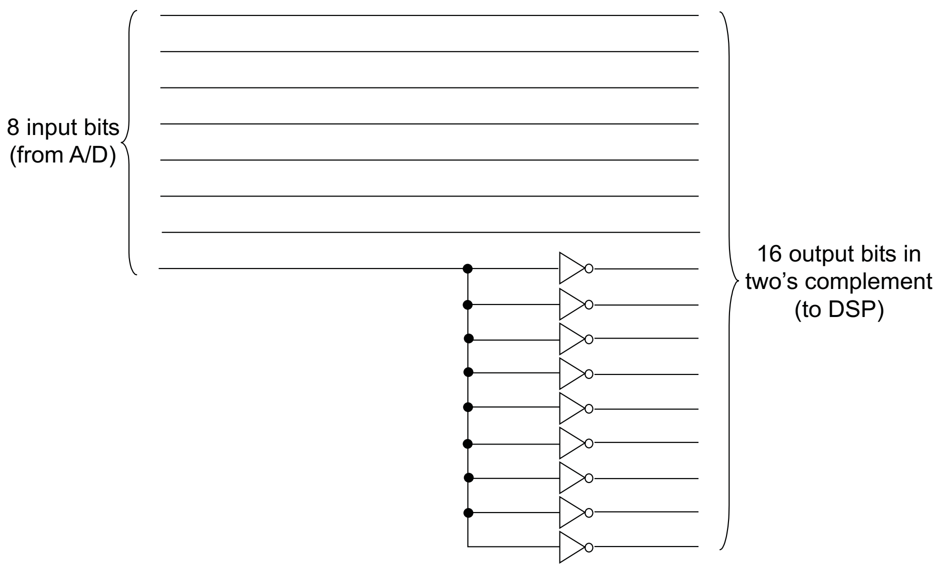

Figure 7.37: The method for converting the eight-bit A/D sample into a 16-bit two's complement notation number.

in Section 7.4.2, so that non-contiguous blocks of data can be captured at regular intervals. The interface memory part is a bank of D-type flip-flops which are used to configure the windowing circuit, to enable and disable the FIFO in the data storage and retrieval part, and to provide the digital signals for the AGC system.

The various portions of the circuit are addressed through the DSP's I/O space addresses (see Section 7.3). The DSP Evaluation Module (EVM) provides address lines for the first 16 addresses of I/O space (addresses `0x0000` through `0x000F` for a total of four address lines). Each time a value is written to or read from one of these addresses the data lines are set to the appropriate values and the $\overline{\texttt{IOSTRB}}$ pin is pulsed low for two or three clock cycles of the processor clock (depending upon the configuration of the I/O wait state). The address lines are tied to the input of a 4-to-16 decoder/demultiplexer, and the $\overline{\texttt{IOSTRB}}$ output is tied to the enable pin of the decoder. Thus, whenever a value is written to or read from locations `0x0000` through `0x000F` in I/O space, the appropriate line on the output of the decoder is pulsed low for two or three cycles of the processor clock. Otherwise, all outputs of the decoder remain high.

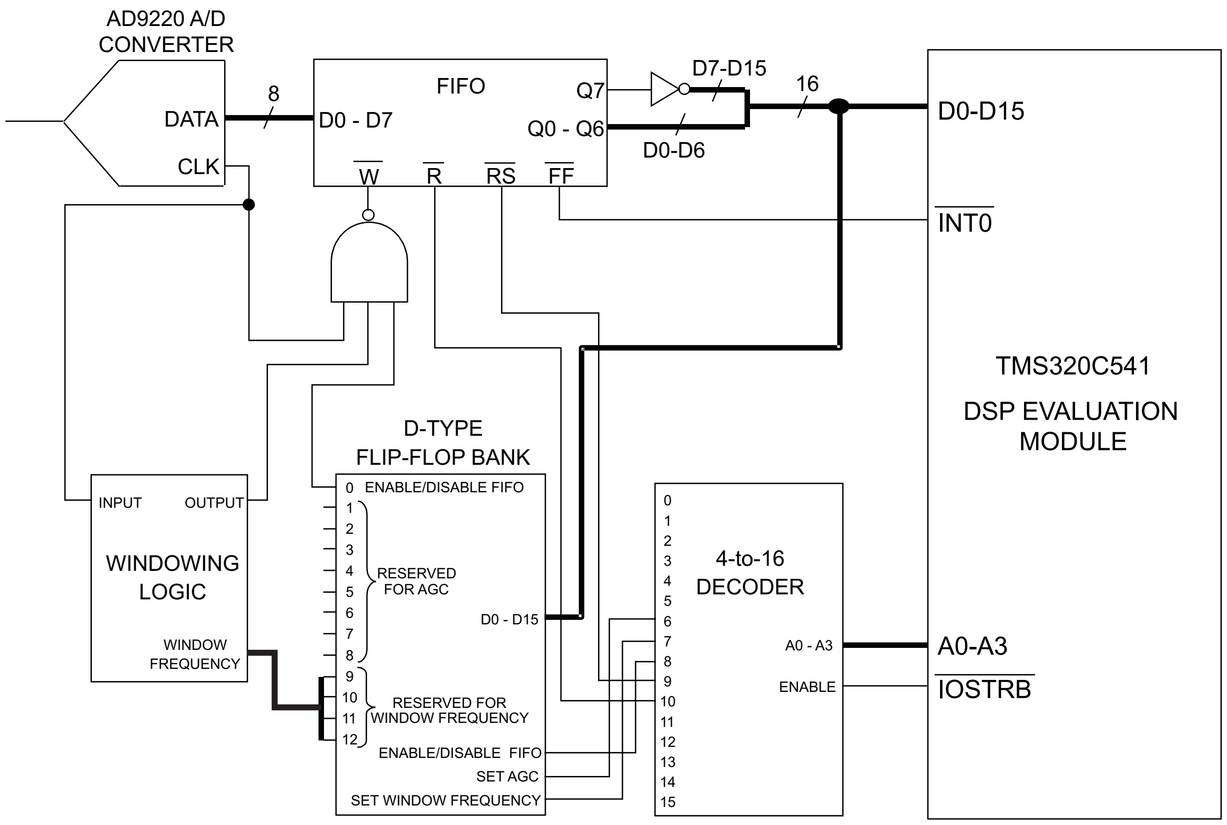

Figure 7.38: The A/D–DSP interface circuit.

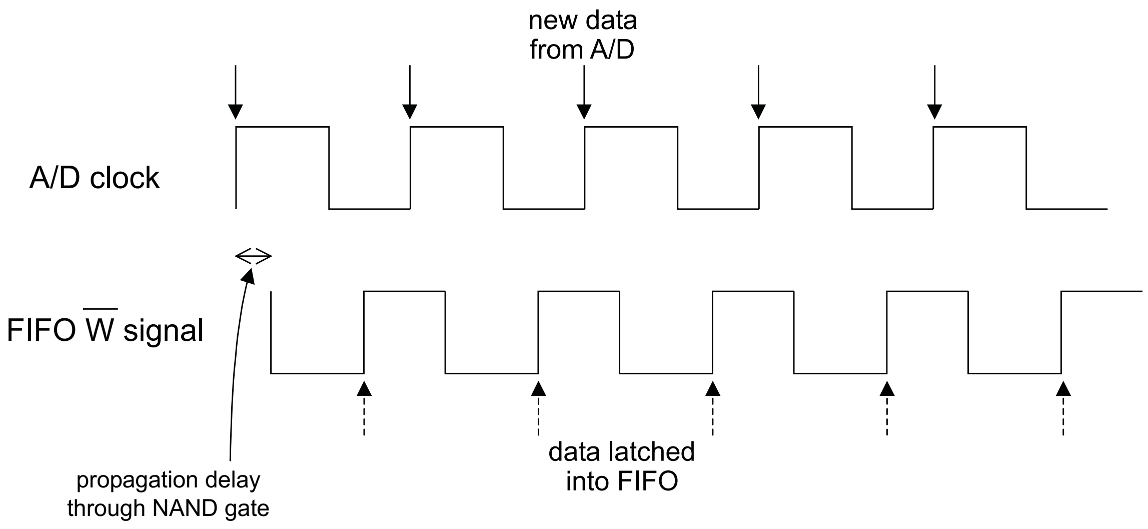

Figure 7.39: The FIFO $\overline{W}$ is a delayed, inverted version of the A/D clock signal. A clock signal appears at the FIFO $\overline{W}$ input only when the FIFO is enabled and when the output of the windowing circuit is high.

7.2.1 Data Storage and Retrieval

The data storage and retrieval part of the interface circuit is centered around the use of an asynchronous FIFO[4]. The eight most significant bits (MSBs) of the output of the A/D converter are tied directly to the data input of the FIFO. When the output of the data windowing circuit is high and the FIFO is enabled, the A/D clock passes through the NAND gate and appears inverted on the FIFO write line ($\overline{W}$).

The reason the clock is NAND-gated with the FIFO enable and window circuits is so that it will appear inverted on the FIFO $\overline{W}$ line. This is because new samples from the A/D are produced on the rising edge of the clock. Writes to the FIFO are also triggered on the rising edge of the clock. Thus, while the clock falls at the output of the A/D, and the data is still valid at the A/D output, it is rising, and hence being written into the FIFO. This ensures that the data has had time to settle before being written to the FIFO. Figure 7.39 shows the timing of the signal at the input of the FIFO.

The FIFO is 4096 samples deep. If the FIFO is empty, and 4096 samples are written into the FIFO before any reads occur, then the full flag ($\overline{FF}$) on the FIFO is driven low. $\overline{FF}$ is tied to the DSP's interrupt 0 pin ($\overline{INT0}$). Thus, when the FIFO is full, an interrupt of the DSP is triggered and the DSP enters the `int0` interrupt service routine. (The `int0`

[4]The term "asynchronous" means that reads and writes to the FIFO are independent and can occur at different rates.

interrupt is a maskable interrupt. To be recognized by the DSP, (1) the interrupt must be unmasked using the assembly command `RSBX INTM`, and (2) the interrupt must be enabled in the interrupt mask register using the assembly command `STM 0x0001, IMR`.)

The read signal on the FIFO ($\overline{\mathrm{R}}$) is tied to address 10 on the decoder. Thus, the FIFO appears to the DSP to be at I/O address `0x000A`. In the `int0` interrupt service routine, 4096 samples are transferred to the DSP from the FIFO using the three command lines

```
STM ips, AR6
RPT #4095
PORTR 0x000A, *AR6+
```

where `ips` is the location in data memory of the first data sample, and `AR6` is a memory-mapped register which points to the data memory location into which a data sample should be written. For this data transfer to work properly, it was found that a total of three wait states for each FIFO access were required, due to the propagation delay and settling time of data arriving from the FIFO at the input of the DSP. (The three wait states were inserted by setting the software wait state register `SWWSR` at memory location `0x0029` to the value `0x3000`.) The output of the FIFO and the inverting buffers for sign extension and two's complement conversion (see Section 7.1.3) were tri-state devices, so they were tied directly to the data lines of the DSP. For this data transfer arrangement, it was found that the DSP could read data continuously from the FIFO at a rate of about 200 ns per sample.

7.2.2 Windowing Circuit

The implementation of the correlation algorithm, which is discussed in Section 7.4, relies on capturing blocks of samples at regularly spaced intervals in time. This was accomplished by creating a circuit which produced a high output for $T_{block} = N_{corr}T_s$ (the duration of a block of samples), where N_{corr} is the number of chips over which the correlation is performed and T_s is the sampling interval, and a low output for nT_{block}, where $n \geq 0$ is an integer. The input of the circuit was the A/D clock, and the output of the circuit, as shown in Figure 7.38, is used as one input to a multi-NAND gate. When the output of the circuit is high (and the other inputs to the NAND are high), the NAND gate passes the A/D clock (inverted). When the output of the circuit is low, the NAND output remains high, and writes to the FIFO are disabled.

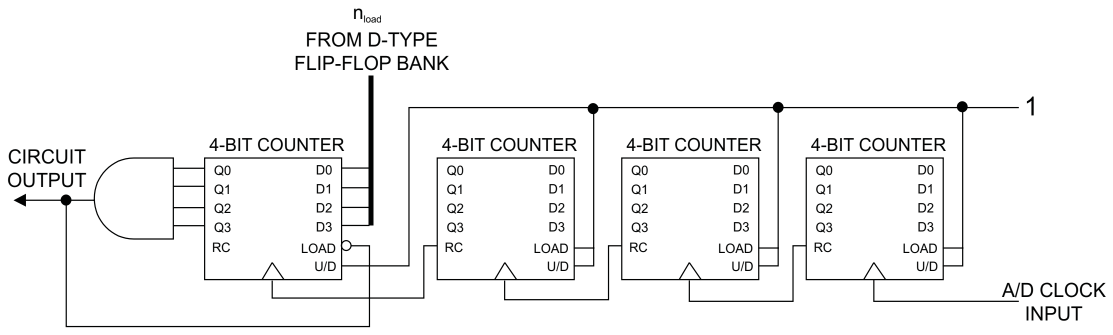

Figure 7.40: The windowing circuit. The purpose of the window circuit is basically to count samples, producing a high output for $T_{block} = N_{corr}T_s$ and a low output for nT_{block}. The output of this circuit alternately enables and disables the write line to the FIFO.

Figure 7.40 depicts the windowing circuit. The A/D clock is used as the clock input to a four-bit counter. The ripple carry output of the four-bit counter [5] drives the clock input of a second counter. Likewise, the ripple carry output of the second four-bit counter drives the clock input of a third four-bit counter. These three counters essentially form a single 12-bit counter, and thus counts multiples of $2^{12} = 4096$ samples.

The ripple carry from the third four-bit counter drives the clock input of a fourth four-bit counter. The circuit output is generated by this fourth four-bit counter. The four load inputs to the counter are taken from four of the D-type flip-flops in the D-type flip-flop bank, which hold a four-bit binary number, n_{load}. The four counter outputs are AND-gated together. The output of this AND gate is the actual circuit output. This output is inverted and fed back into the load signal of the counter. Thus, the fourth four-bit counter continuously counts from n_{load} to 15, and a high output of the four-input AND gate is produced when the output of the fourth counter is equal to 15. This configuration produces the desired output.

7.2.3 D-Type Flip-Flop Bank

The purpose of the bank of D-type flip-flops is to hold data for the enablement or disablement of the FIFO, for the digital control of the AGC, and for setting the number n_{load} for the windowing circuit discussed in Section 7.2.2. The implementation of the bank of D-type flip-flops, shown in Figure 7.41, is straightforward. Four hex flip-flops are employed. The data lines from the DSP are ties to the data inputs of the flip-flops. The clock inputs of the flip-flops are tied to the sixth, seventh, and eighth outputs of the decoder. Therefore,

[5]The ripple carry output is low as long as all outputs of the counter are high and high otherwise.

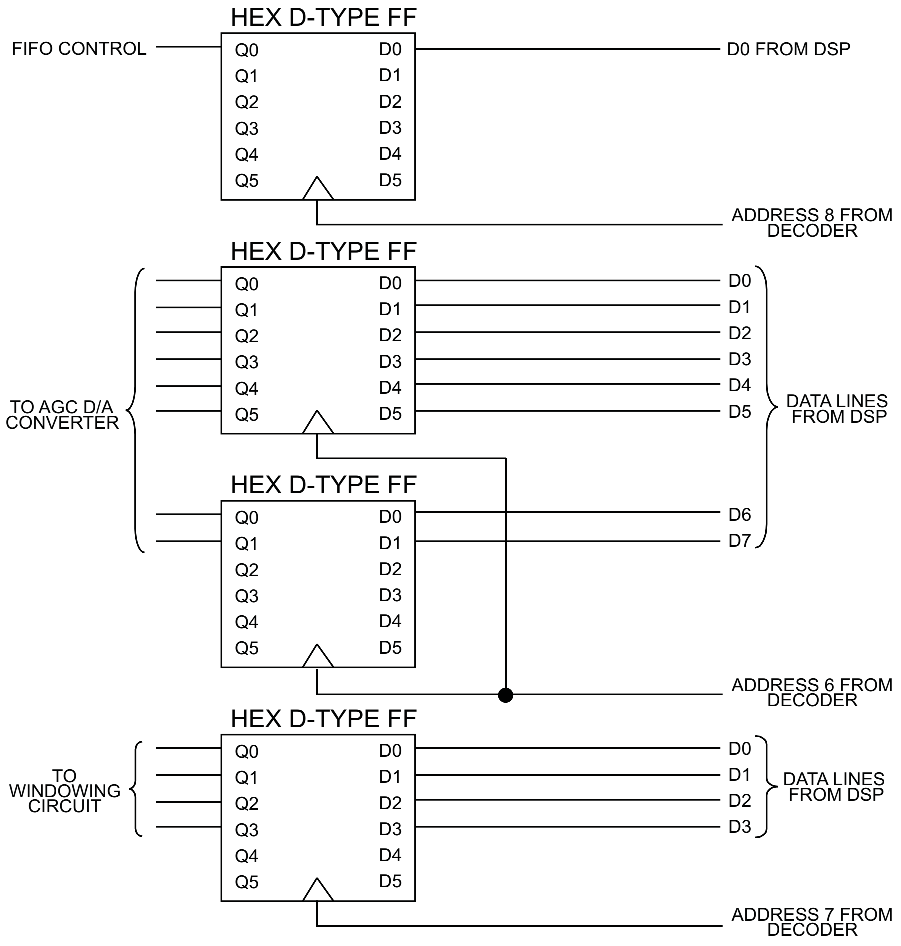

Figure 7.41: The connections for the D-type flip-flop bank.

the flip-flops appear to the DSP to be at I/O addresses `0x0006`, `0x0007`, and `0x0008`. The flip-flops are set via software with the assembly language instruction

```
PORTW data_mem, port,
```

where `data_mem` is a data memory address and `port` is an I/O address. From Figure 7.38, it is clear that `port` can be one of the following: `0x0006`, `0x0007`, or `0x0008`.

The data outputs of the flip-flops are tied to the parts of the interface circuit over which control is desired. One data output is tied to one input of the multi-NAND gate whose output is the FIFO. Thus, writing a 0 to this flip-flop will cause all writes to the FIFO to be disabled. Four data outputs from one flip-flop chip serve to hold the value of n_{load} for the windowing circuit discussed in Section 7.2.2. Finally, eight of the data outputs, spanning two flip-flop chips, are used to hold the value of the digital word which is written to a D/A converter to set the gain of the analog tuner portion of the receiver.

7.3 Digital Signal Processing Hardware

The DSP used in this thesis was the Texas Instruments TMS320C541, which is a 16-bit fixed point processor that runs at a clock rate of 40 MHz. Three address spaces, each having a total of $2^{16} = 65536$ addresses exist in the C54: program, data, and input/output (I/O). The program and address spaces are tied to physical memory locations. The TMS320C541 has 33792 words of on-chip program memory (addresses `0x0000` through `0x13FF` and `0x9000` through `0xFFFF`), and 13312 words of on-chip data memory (addresses `0x0000` through `0x13FF` and `0xE000` through `0xFFFF`). Although there are several ways to address the memory in software, in this thesis the indirect addressing mode [35, p. 5-10–5-24] was found to be most directly useful. The TMS320C541 has one 17-bit $\times$ 17-bit multiplier, one 40-bit adder, and two 40-bit accumulators. A multiply-accumulate or multiply-subtract operation (MAC or MAS) can be performed in a single clock cycle. The TMS320C541 can also accomplish no overhead looping using the block repeat counter (BRC) so that several lines of code can be executed repeatedly without flushing the instruction pipeline.

For this thesis, the DSP was used as mounted on an evaluation module (EVM) printed circuit board [36]. The EVM contained two 128K SRAM chips which filled out the entire program and memory address spaces. (Off-chip memory at the on-chip address spaces was accessed by setting the `OVLY`, `MP/MC`, or `DROM` bits as appropriate.) The EVM also contained a 64-pin header which provided user access to the data bus lines, the four least significant bits (LSBs) of the addresses bus lines, interrupt lines, and other control signals. The EVM resided in a host personal computer (PC), to which it was interfaced via the ISA bus. The board was powered over the ISA bus, software was downloaded into the program memory via the ISA bus as well, and the ISA bus was used to transfer data between the PC and DSP during real-time operation.

7.4 Implementation of the Correlation Algorithm

7.4.1 Signal Scaling to Avoid Overflow

When computations are done with fixed point DSPs, extra care must be taken to avoid overflow. In this thesis, this was accomplished by limiting the number of bits to which the received signal and the local reference signal were quantized. The numbers of bits of quantization were limited in such a way that the output of the algorithm would never cause an overflow of the 40-bit accumulator. This approach is more computationally efficient than attempting to simulate floating point arithmetic on the fixed point processor.

Figure 7.42 shows how the correlation algorithm was implemented. The DSP was used to perform the fast multiply-accumulate operations required for the baseband detection method. The accumulation was performed using the accumulators on the DSP. The result stored in the accumulator was transferred to the PC, which performed the square and sum operation in floating point arithmetic. This method is more efficient than performing the squaring operation on the DSP, since squaring doubles the number of bits occupied. Therefore, this method allows more bits to be used to quantize the signals than if the squaring operation were done on the DSP.

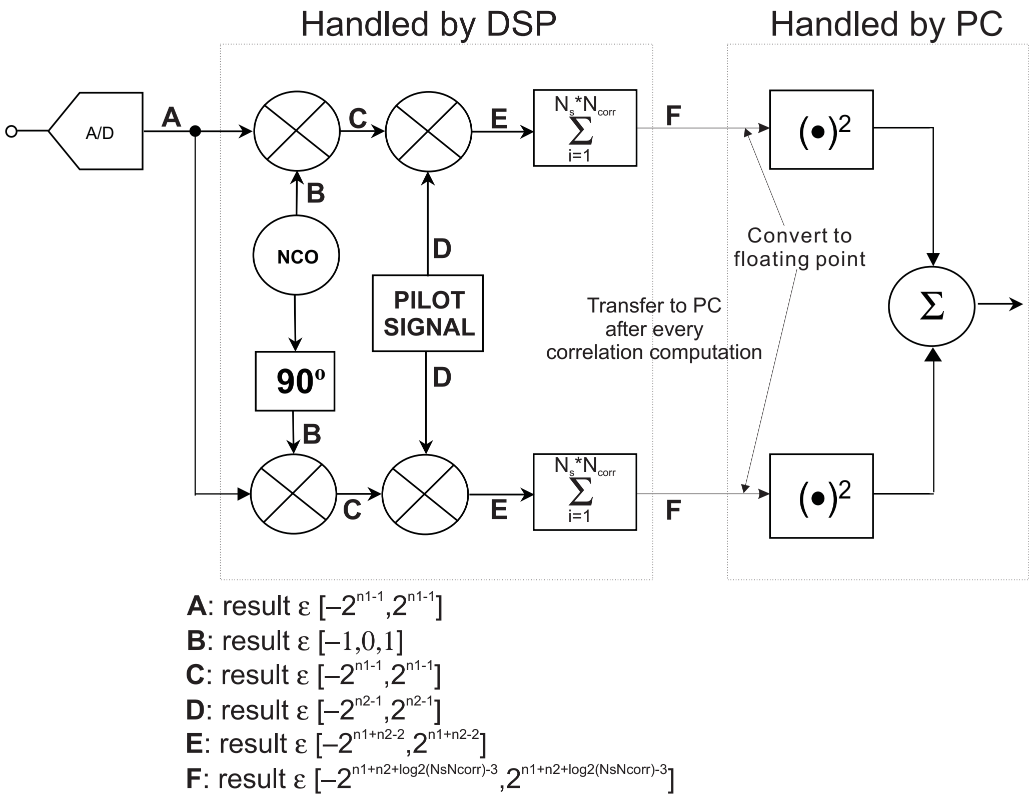
Handled by DSP
Handled by PC
A/D
A
B
C
D
E
F
NCO
90°
PILOT SIGNAL
$\sum_{i=1}^{N_s*N_{corr}}$
$(\bullet)^2$
Σ
Convert to floating point
Transfer to PC after every correlation computation
A: result ε $[-2^{n1-1}, 2^{n1-1}]$
B: result ε $[-1,0,1]$
C: result ε $[-2^{n1-1}, 2^{n1-1}]$
D: result ε $[-2^{n2-1}, 2^{n2-1}]$
E: result ε $[-2^{n1+n2-2}, 2^{n1+n2-2}]$
F: result ε $[-2^{n1+n2+\log2(NsNcorr)-3}, 2^{n1+n2+\log2(NsNcorr)-3}]$

Figure 7.42: The approximate range of integers at various points in the baseband correlation algorithm.

As shown in Figure 7.42, the received signal is quantized with n_1 bits of resolution and the local reference signal is quantized with n_2 bits of resolution. Figure 7.42 illustrates the approximate range of integers at various steps in the baseband correlation algorithm. The result at point F in the figure is stored in the DSP accumulator. Although the accumulator is 40 bits in width, the most efficient transfer occurs if the result occupies only 32 bits, since the DSP has a bus width of 16 bits. Thus, from Figure Figure 7.42, it is deduced that to avoid results greater than 32 bits in width before transfer to the PC, n_1 and n_2 should satisfy the relation

$$n_1 + n_2 + \log_2 (N_s N_{corr}) \Leftrightarrow 3 \leq 31, \tag{7.96}$$

where N_s is the number of samples per chip and N_{corr} is the number of chips over which the correlation is performed. For $N_s = 4$ and $N_{corr} = 1024$, as was used in this thesis, it is found that

$$n_1 + n_2 \leq 22. \tag{7.97}$$

Similarly, in the AGC computation (see Section 7.5) each sample is squared and a running summation over $\log_2 (N_s N_{agc})$ samples is maintained. Thus, the greatest possible output of this operation is $2^{2(n_1 - 1) + \log_2(N_s N_{agc})}$. In order to avoid overflow, then, it must be ensured that

$$2 (n_1 \Leftrightarrow 1) + \log_2 (N_s N_{agc}) < 31, \tag{7.98}$$

or

$$\log_2 (N_{agc}) < 31 \Leftrightarrow 2 (n_1 \Leftrightarrow 1) \Leftrightarrow \log_2 (N_s) . \tag{7.99}$$

For example, if $n_1 = 12$ bits and $N_s = 4$ samples/chip, then it must be ensured that $N_{agc} < 128$ chips.

7.4.2 Scanning Algorithm

Previous chapters have shown that pilot signals can be detected by testing for partial correlations between the received signal and the locally generated pilot signal. Implementation of the ideas presented earlier in this thesis in hardware had to be accomplished under the constraints imposed by data transfer rates, memory, and processing bandwidth of the DSP, while keeping the dynamic range as high as possible and the update rate as low as possible.

Figure 7.43 provides a starting point for understanding the implementation of scanning correlation algorithm. An entire period of the pilot signal is stored in memory, and one block of data samples is acquired each period of the received signal. This block of samples is "scanned" through the pilot signal stored in memory, and at each offset the partial correlation

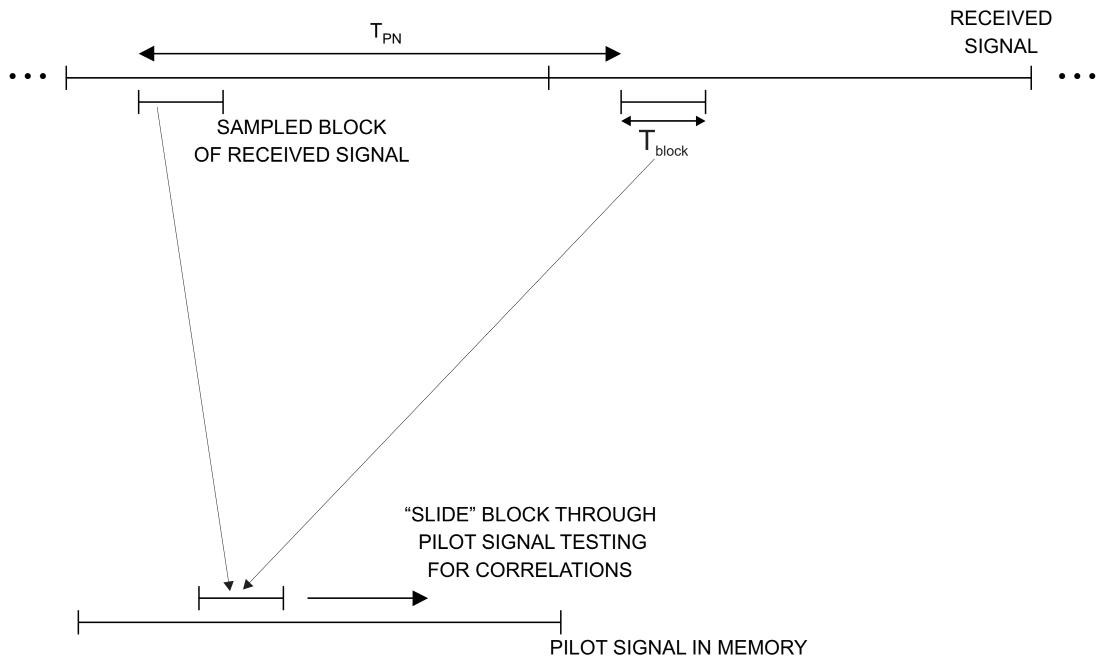
T_{PN}
RECEIVED SIGNAL
SAMPLED BLOCK OF RECEIVED SIGNAL
T_{block}
"SLIDE" BLOCK THROUGH PILOT SIGNAL TESTING FOR CORRELATIONS
PILOT SIGNAL IN MEMORY

Figure 7.43: Partial correlations can be obtained by sliding a small block of the sampled received signal through the pilot signal contained in memory.

is computed. This approach produces the desired output.[6] The data transfer rate for this method is reasonable – only $N_{block} = T_{block}/T_s$ (where T_{block} is the time duration of a block of samples and T_s is the sampling period) samples once per period of the pilot signal. Notice, however, that this method requires that $N_s * 2^{15}$ (where N_s is the number of samples per chip period) samples of the pilot signal be stored in memory, which is excessive given the hardware selected for use in this thesis.[7]

To reduce the amount of memory required for storage of the pilot signals, the approach graphically illustrated in Figure 7.44 was adopted. Blocks of data samples are captured more than once per cycle of the received pilot signal, which increases the data transfer rate to the DSP to N_{block} samples every $T_{block} (< T_{pilot})$. However, this method does allow a smaller fraction of the pilot signal to be stored in memory. Figure 7.44 shows a special case where one-eighth of the received pilot signal is sampled four times per cycle of the pilot signal. In this case, only three-eighths of the pilot signal needs to be maintained in memory.

[6]Thanks to Ben Ward of Tektronix for pointing out this possibility to me.

[7]The possibility exists for storing the entire pilot signal in memory by packing multiple low resolution samples into single memory locations. However, the processing power taken for unpacking the samples from memory for use in correlation is undesirable.

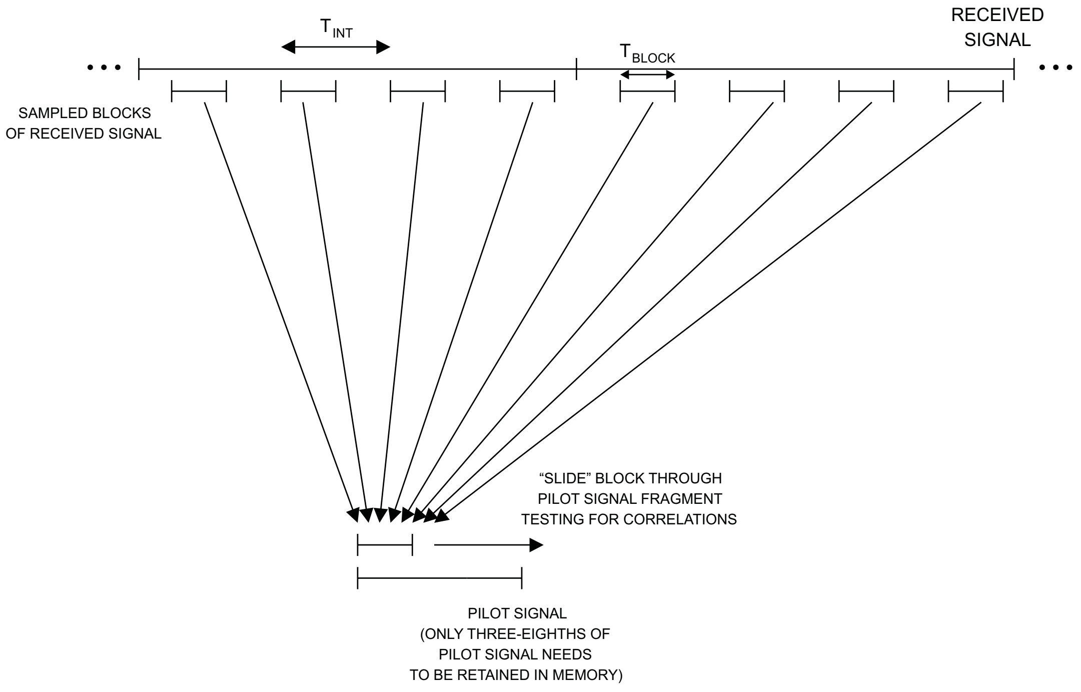
T_INT
T_BLOCK
RECEIVED SIGNAL
SAMPLED BLOCKS OF RECEIVED SIGNAL
"SLIDE" BLOCK THROUGH PILOT SIGNAL FRAGMENT TESTING FOR CORRELATIONS
PILOT SIGNAL (ONLY THREE-EIGHTHS OF PILOT SIGNAL NEEDS TO BE RETAINED IN MEMORY)

Figure 7.44: Partial correlations can be obtained by scanning a block of the sampled received signal through a fragment the pilot signal contained in memory. The idea is similar to Figure 7.43, but blocks of samples of the received signal are captured more frequently, which allows the digital signal processor to maintain a smaller portion of the pilot signal in memory.

In general, with this method, if T_{int} is the time interval between the first sample of adjacent blocks of data samples, T_{block} is the time duration of a block of samples, and T_{pilot} is the time duration of a single period of the pilot signal, then the fraction of the pilot signal which must be stored in memory, γ_{pilot}, is

$$\gamma_{pilot} = \frac{T_{int} + T_{block}}{T_{pilot}}. \tag{7.100}$$

For instance, in the example of Figure 7.44, $T_{int} = T_{pilot}/4$ and $T_{block} = T_{pilot}/8$. This leads to $\gamma_{pilot} = 3/8$, as mentioned above and in Figure 7.44.

Consider the following argument to show that the method of Figure 7.44 is capable of producing correlations at all possible phase offsets between the locally generated pilot signal and the received pilot signal. The phase offset in samples between the locally generated pilot signal and the sampled received pilot signal, $\Delta\psi$, can be expressed

$$\Delta\psi = s_1^l \Leftrightarrow s_1^r, \tag{7.101}$$

where s_1^r is the first sample of the received pilot signal at which a correlation is computed and s_1^l is the first sample of the locally generated pilot signal at which a correlation is computed. In order to prove that the method under consideration produces the desired results, it must be shown that correlations at all possible phase offsets can be computed, or that $\Delta\psi \in [0, N_{pilot} \Leftrightarrow 1]$, where N_{pilot} is the number of samples in one period of the pilot signal.

For the example under consideration, for simplicity, suppose that the receiver stores samples such that $s_1^l \in [1, N_{pilot}/4]$. Furthermore, assume that on the first block of samples captured $s_1^r = 1$. Then, for the first block of samples, the range of phase offsets for which correlations can be computed, $\Delta\psi_1$, is $\Delta\psi_1 \in [0, N_{pilot}/4 \Leftrightarrow 1]$.

For the second block of samples captured, $s_1^r = N_{pilot}/4 + 1$. Then, for the second block of samples, the range of phase offsets for which correlations can be computed, $\Delta\psi_2$, is $\Delta\psi_2 \in [\Leftrightarrow N_{pilot}/4, \Leftrightarrow 1]$. Similarly, for the third block of samples, $s_1^r = N_{pilot}/2 + 1$, and the range of phase offsets for which correlations can be computed, $\Delta\psi_3$, is $\Delta\psi_3 \in [\Leftrightarrow N_{pilot}/2, \Leftrightarrow N_{pilot}/4 \Leftrightarrow 1]$. Finally, for the fourth block of samples, $s_1^r = 3N_{pilot}/4 + 1$, and the range of phase offsets for which correlations can be computed, $\Delta\psi_4$, is $\Delta\psi_4 \in [\Leftrightarrow 3N_{pilot}/4, \Leftrightarrow N_{pilot}/2 \Leftrightarrow 1]$.

Due to the periodicity of the pilot signal, adding any multiple of N_{pilot} to the phase offset $\Delta\psi$ does not alter the phase offset being referenced. Thus, we have

$$0 \leq \Delta\psi_1 \leq N_{pilot}/4 \Leftrightarrow 1 \tag{7.102}$$

$$\Leftrightarrow N_{pilot}/4 \leq \Delta\psi_2 \leq \Leftrightarrow 1 \quad \Leftrightarrow \quad 3N_{pilot}/4 \leq \Delta\psi_2 \leq N_{pilot} \Leftrightarrow 1 \tag{7.103}$$

$$\Leftrightarrow N_{pilot}/2 \leq \Delta\psi_3 \leq \Leftrightarrow N_{pilot}/4 \Leftrightarrow 1 \quad \Leftrightarrow \quad N_{pilot}/2 \leq \Delta\psi_3 \leq 3N_{pilot}/4 \Leftrightarrow 1 \tag{7.104}$$

$$\Leftrightarrow 3N_{pilot}/4 \leq \Delta\psi_4 \leq \Leftrightarrow N_{pilot}/2 \Leftrightarrow 1 \quad \Leftrightarrow \quad N_{pilot}/4 \leq \Delta\psi_4 \leq N_{pilot}/2 \Leftrightarrow 1. \tag{7.105}$$

Combining these results, $\Delta\psi = \Delta\psi_1 \cup \Delta\psi_2 \cup \Delta\psi_3 \cup \Delta\psi_4$, it is observed that $0 \leq \Delta\psi \leq N_{pilot} \Leftrightarrow 1$, which shows that the method under consideration is indeed capable of producing correlations at all possible phase offsets between the locally generated pilot signal and the received pilot signal.

Notice that, with this method, if the correlation output is plotted versus the phase offset between the locally generated and received pilot signals (see Section 1.5) then the graph is not filled out with monotonically increasing phase offset. This is because the DSP cannot compute the correlation rapidly enough to test for correlations between all possible phase offsets between the block of data samples and the pilot signal fragment over the duration T_{int}. Returning to the example above (Figure 7.44), suppose that the DSP only has enough processing power to compute the correlations for one-fourth of all possible phase offsets between the block of data samples and the pilot signal fragment for each block of data captured (that is, over the duration T_{int}). Then the correlation plot will be filled out according to the diagram given in Figure 7.45, where the phase offset is defined in Equation 7.101.

In this thesis, T_{block} was chosen to be a time duration equivalent to 1024 chips, or 4096 samples. This was shown to provide about 15 dB of dynamic range (see Chapter 6) in typical scenarios. The choice of the parameter T_{int} was driven by memory constraints. There is enough memory on the DSP for 16 blocks of 4096 samples. One of these blocks must be reserved for the data samples of the received signal. Although not necessary, it is desirable to choose T_{int} such that T_{pilot} an integer multiple of T_{int}, since it simplifies the software programming task. To store fragments of both the I and Q pilot signals, the best compromise was to choose $T_{int} = 4T_{block}$, which meant that eight blocks of samples were captured per period of the pilot signal. This required that $2 \times 5 \times 4096$ words of memory be reserved for storage of both the I and Q pilot signal fragments. 70% of the data memory on the DSP was thus utilized.

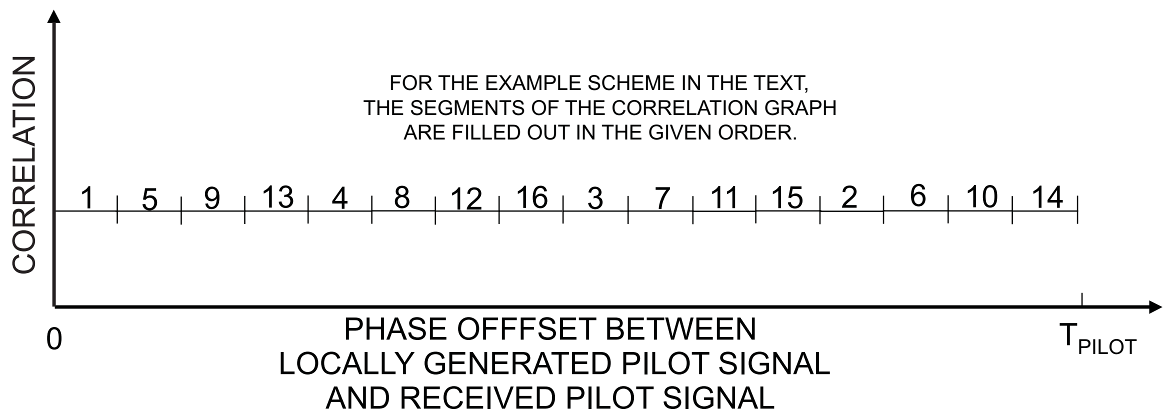
CORRELATION
FOR THE EXAMPLE SCHEME IN THE TEXT, THE SEGMENTS OF THE CORRELATION GRAPH ARE FILLED OUT IN THE GIVEN ORDER.
1
5
9
13
4
8
12
16
3
7
11
15
2
6
10
14
0
PHASE OFFFSET BETWEEN LOCALLY GENERATED PILOT SIGNAL AND RECEIVED PILOT SIGNAL
T_{PILOT}

Figure 7.45: The sequence by which the correlation plot will be filled out when the receiver captures data four times per cycle of the received pilot sequence and there is enough processing power to correlate over one-fourth of all possible phase shifts between the block of data samples and the pilot signal fragment for each segment of data captured.

7.4.3 Update Rate

Figure 7.46 shows a timing diagram of one cycle data acquisition and correlation in the of the digital portion of the receiver. During a given cycle, data is captured and placed in a FIFO. When the FIFO is full, the DSP is interrupted and the data is transferred from the FIFO into the DSP's data memory on the EVM. When the transfer is complete, the DSP begins computing the correlations and information for the AGC. As the results are generated, they are transferred over to the PC for further analysis and graphical display.[8]

With this method of data acquisition and processing, the total amount of time just for computing correlations was about $3T_{block}$. It was found empirically that during this processing time the results of eight partial correlations can be computed. Four of these partial correlations are done using the I-channel pilot signal, and four of the partial correlations are done using the Q-channel pilot signal. Since there are a total of eight data acquisition cycles per period of the received pilot signal (see Figure 7.46), then a total of $4 \times 8 = 32$ correlations can be computed per period of the pilot signal. A total of 4×2^{15} such correlations are desired; therefore, a total of $4 \times 2^{15}/32 = 4096$ periods of the pilot signal are required to complete the scan of all possible code phases. Each period of the pilot signal requires $26\frac{2}{3}$ ms; hence, a total of $4096 \times 26\frac{2}{3} \times 10^{-3} \approx 109.23$ s are required for one complete scan of all possible code phases. The update period, T_{update}, is thus $T_{update} = 109.23$ s.

[8]Actually, the results are placed in a FIFO on-board the EVM, and the DSP signals the PC to read from the FIFO.

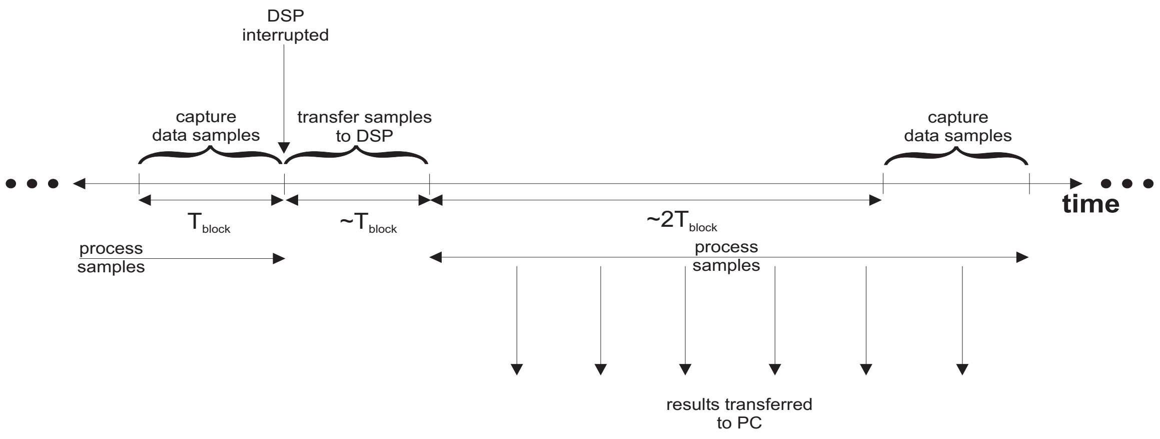

Figure 7.46: The basic activities of the digital portion of the receiver. The receiver captures data and stores it in a FIFO. When the FIFO is full, the DSP is interrupted and the data transfer takes place. When the transfer is complete, the data processing takes place with results transferred to the PC periodically, until the DSP is interrupted again.

7.5 Implementation of the AGC Algorithm

The hardware implementation of the AGC algorithm presented in Chapter 5 was accomplished as illustrated in Figure 7.47 [29]. The AGC amplifiers in the RF front-end chain were digitally controllable for gains between ⇔120 dB and 60 dB. The gain control voltage was between 0.1 V (for ⇔120 dB gain) and 2.7 V (for 60 dB gain). The transfer function between gain and control voltage was approximately linear.

The control voltage was set by writing an eight-bit binary number to a digital-to-analog (D/A) converter, the output of which lay between 0 V and 3 V. Therefore, the binary numbers between

$$\left\lceil 0.1\frac{256}{3}\right\rceil = 8 \tag{7.106}$$

and

$$\left\lfloor 2.7\frac{256}{3}\right\rfloor = 230 \tag{7.107}$$

were used to control the output voltage. As discussed in Section 7.2, the eight D-type flip-flops which were dedicated for the AGC were at I/O address `0x0006`. Therefore, the AGC was controlled using the assembly command

```
    PORTW 0x0006, agcnum
```

where `agcnum` was a number between 5 and 114.

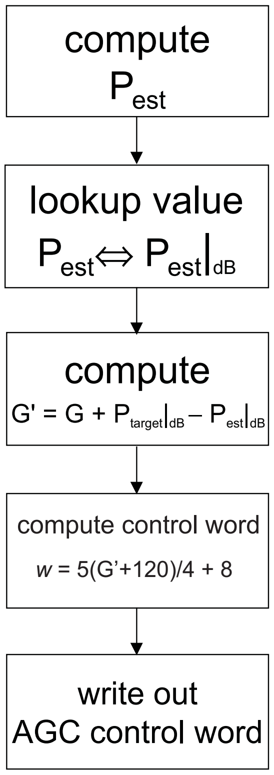

Figure 7.47: A flowchart for the AGC algorithm.

For each block of samples received from the A/D, the DSP computed P_{est} according to the following formula:

$$P_{est} = \frac{1}{N_{corr}} \sum_{i=1}^{N_{corr}} s_i^2, \tag{7.108}$$

where s_i is the i^{th} sample and $N_{corr} (= 4096)$ is the number of samples used in the computation. (Since $N_{corr} = 4096$, the division by N_{corr} was actually accomplished by a right shift of 12 bits.)

For ease of computation, Equation 5.66 was converted to dB values according to the following equation:

$$G'|_{\mathrm{dB}} = G|_{\mathrm{dB}} + P_{target}|_{\mathrm{dB}} \Leftrightarrow P_{est}|_{\mathrm{dB}}, \tag{7.109}$$

where

$$\begin{array}{ll} G'|_{\mathrm{dB}} = 10\log_{10}(G') & G|_{\mathrm{dB}} = 10\log_{10}(G) \\ P_{target}|_{\mathrm{dB}} = 10\log_{10}(P_{target}) & P_{est}|_{\mathrm{dB}} = 10\log_{10}(P_{est}). \end{array} \tag{7.110}$$

The quantity $P_{est}|_{\mathrm{dB}}$ was computed through the use of a lookup table. The lookup table used is given in Table 7.5. Whenever a given P_{est} was computed in the AGC algorithm, the

Table 7.10: The lookup table for the AGC algorithm. Whenever a P_{est} is computed by the DSP, the index of the closest value is found in the lookup table. The index is used to find the value in dB of P_{est} from a corresponding table.

Computed P_{est}	Lookup P_{est} (dB)	Computed P_{est}	Lookup P_{est} (dB)
1	1	200	23
2	3	251	24
3	5	316	25
4	6	398	26
5	7	501	27
6	8	631	28
8	9	794	29
10	10	1000	30
13	11	1259	31
16	12	1585	32
20	13	1995	33
25	14	2512	34
32	15	3162	35
40	16	3981	36
50	17	5012	37
63	18	6310	38
79	19	7943	39
100	20	10000	40
126	21	12589	41
158	22	15849	42

lookup table given in Table 7.5 was scanned and the closest value to the computed P_{est} was determined. The index of this value was used as the index into a corresponding table of values of $P_{est}|_{\mathrm{dB}}$. The value so determined is inserted into Equation 7.109 and the value of $G'|_{\mathrm{dB}}$ computed.

With the value of $G'|_{\mathrm{dB}}$ determined, the value of the digital control word, w, was computed as follows:

$$w = \frac{5\,(\,G'|_{\mathrm{dB}} + 120)}{4} + 8. \tag{7.111}$$

Notice that the division by 256 in Equation 7.111 can be accomplished by a right shift of two bits. The value w was written to I/O address `0x0006` to control the voltage of the AGC amplifier to values between 0.1 V and 2.7 V.

Chapter 8

Potential Future Developments

In Chapter 7, a prototype implementation of a scanning digital correlator receiver was demonstrated. Nevertheless, for practical field use of this system as a pilot signal detector for system deployment, more work is required. Moreover, there are several ways in which the receiver could be analyzed on theoretic and simulation grounds.

8.1 Implementation Improvements

8.1.1 Determination of Transmitting Base Station

To be useful in system deployment applications, a pilot signal scanning receiver should be able to identify the base station from which each pilot signal detected was transmitted. No attempt was made to develop a method for making this determination in this thesis. However, there are at least two methods by which this could be accomplished.

Integration of Global Positioning System Data

The Global Positioning System (GPS) is a constellation of satellites used for precise measurements of time and position [37, 38]. Several manufacturers market GPS receivers which provide users with precise time information. Such a receiver could be integrated into the pilot signal scanning receiver, which would synchronize the pilot signal scanning receiver with the PCS network. Thus, the pilot signal scanning receiver would have a phase reference by which to determine the absolute phase of the received pilot signal. A knowledge of the PN offset plan in the region under measurement would allow the receiver to determine the transmitting base station.

Decoding the Sync Channel

As described in Section 2.7.2, the PN offset of the transmitted signal is carried over the sync channel. If this channel were decoded in the receiver, then the PN offset information would be available. Due to the implementation of the pilot signal scanning receiver discussed in Section 7.4.2, the signal was not sampled continuously, since this would have overloaded the transfer to the DSP. Therefore, with the implementation in this thesis, it is not possible to decode the received signal. This difficulty must be overcome if the transmitting base station is to be determined using the information transmitted over the sync channel.

8.1.2 Dynamic Range Improvement

In the receiver designed and implemented in this thesis, a dynamic range of about 15 dB was optimistically demonstrated. Whether this dynamic range is large enough for field use was not addressed in this thesis. This topic should be addressed in future studies.

8.1.3 Update Rate Improvement

It was shown in Chapter 7 that the update rate of the pilot signal scanning receiver as implemented is 109 s. Since the pilot signal scanning receiver is essentially a channel measurement instrument, one update period, T_{update}, should be less than the channel coherence time. Thus, the pilot signal scanning receiver designed and implemented in this thesis is not suitable for mobile application.

As designed in this thesis, the update rate of the pilot signal scanning receiver is limited by the processing bandwidth of the DSP. A direct method of increasing the update rate is to increase this bandwidth. Worthy of investigation to this end is that several vendors (Ariel, DSP Research, Pentek, Spectrum Signal Processing, and White Mountain DSP, among others) market DSP boards consisting of multiple DSP chips which operate in parallel, thereby increasing the processing bandwidth.

8.1.4 Application Specific Integrated Circuit Implementation

The correlation, AGC, and scanning algorithms presented in this thesis are all repetitive in nature. A prototype receiver which implements these algorithms using a DSP is a good first

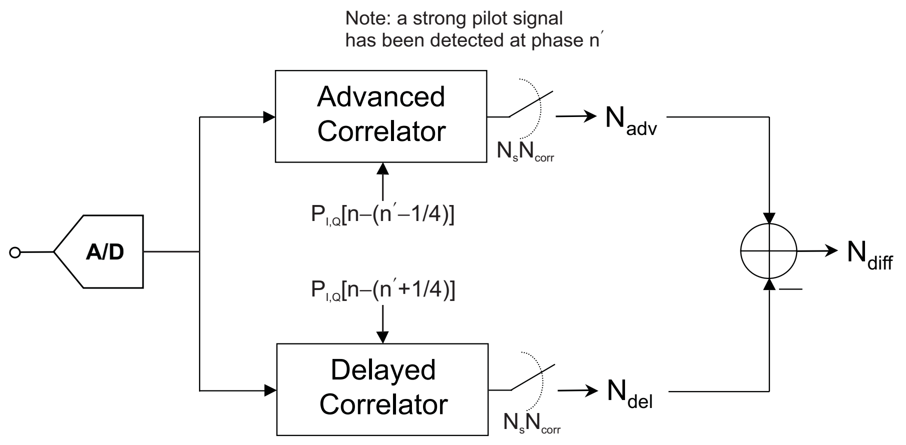

Figure 8.48: A preliminary algorithm for correction of sampling frequency offset.

step to prove that the ideas are valid. However, DSPs are most useful when the algorithms implemented require extensive branching or decisions between several alternatives. Since the algorithms in this thesis are not of that nature, a more efficient implementation (in terms of bandwidth and power consumption) of the digital signal processing would be to create a single chip which performs the algorithms described in this thesis. This chip would need an arithmetic logic unit (ALU), memory, and logic to accomplish the scanning action. The analysis and implementation presented in this thesis would be a steppingstone to such a development.

8.1.5 Sampling Frequency Offset Correction

As discussed in Section 6.5.3, sampling frequency offset induces and undesired artificial drift in the observed phase of a pilot signal. If a pilot signal is initially detected at some phase, then this drift can be detected using delayed and advanced correlation as shown in Figure 8.48. The output of the algorithm, N_{diff}, can be used either (1) to develop an analog voltage to adjust the frequency of a voltage-controlled clock driving the A/D converter, or (2) to correct for the phase drift digitally by changing the phase reference used by the DSP in such a way that the detected pilot signal always maintains the same phase. Both methods must make assumptions concerning the stationarity of the channel, namely that any drift in the observed phase of a pilot signal is due solely to the sampling frequency offset and not to the channel.

8.2 Further Simulations and Theoretical Studies

The purpose of Chapter 2 in conjunction with Appendix B was to show that the IS-95A and receiver were modeled adequately so that simulation could be used to predict the performance of the receiver. However, some of the results of the simulations have limited explanations of a theoretical nature, which future investigations should strive to rectify.

Some effects were not studied in this thesis. These include effects due to the analog tuner: intermodulation products[1] and non-ideal filtering. Also included are effects due to the A/D converter: spurious products, integral nonlinearities, and differential nonlinearities. Channel effects, including narrowband jammers[2] and a non-static channel (see Sections 6.5.3 and B.5), should also be studied. Furthermore, if the receiver is to be used in a mobile application, then investigations into the appropriate period for the integration time (N_{agc}) of the AGC system would be required.

[1]In [15, p. 9-43], the CDMA standard recommends that an IS-95A receiver be tested with two equal power (-40 dBm) tones separated by ± 900 kHz and ± 1700 kHz from the carrier frequency.

[2]In [15, p. 9-43], the CDMA standard recommends that an IS-95A receiver be tested with an interfering tone having an input power of -30 dBm and separated by ± 900 kHz from the carrier frequency.

Chapter 9

Summary and Contributions

9.1 Summary

CDMA is becoming the technology of choice for new and next generation cellular and PCS systems in the U.S. Resolution of the coverage/interference dichotomy, search window settings, handoff, pilot pollution, position location, PN offset planning, and island cells dictate that a system deployment engineer have information on the number, amplitudes, and delays of pilot signal at diverse geographical locations throughout a CDMA network.

This thesis presented the design and implementation of a prototype receiver capable of making the requisite pilot signal measurements. Chapters 2 and 3, in conjunction with Appendix B, discussed the details of the forward link signal and propagation channel and the method by which channel and receiver distortions were modeled for computational simulations. Chapter 4 discussed design considerations for digital receivers.

The receiver designed in this thesis uses digital correlation to detect pilot signals. The phase of the digital signal in the receiver is continually adjusted to detect pilot signals of all possible received phases. An analog tuner is needed to amplify, filter, and translate the signal to a frequency band at which it can be sampled.

In Chapter 5 two digital correlation algorithms were stipulated. In one, the baseband correlation method, the sampled signal is digitally downconverted to baseband where the correlation takes place. In the other, the bandpass correlation method, the sampled IF signal is multiplied by the pilot signal, and this is followed by digital bandpass filtering and digital energy detection. Chapter 5 also presented a digital signal processing algorithm for the automatic gain control of the analog tuner.

The main chapter dealing with the design of the receiver was Chapter 6. Rationale for the choices of sampling frequency and final IF frequency were discussed first. A performance measure, D, which was identified as a type of dynamic range, was introduced. This performance measure was used to determine the appropriate signal power at the input of the A/D converter, gauge the effect of thermal noise at the input of the A/D converter, select an appropriate correlation period, and to study the effects of sampling phase offset, sampling frequency offset, sampling frequency jitter, signal quantization, frequency offset, local oscillator phase noise, multipath, and multiple base stations. The gain and downconversion architecture of the analog tuner were also addressed, as was the trade-off between the maximum range and noise figure of the receiver.

A hardware implementation of a pilot signal scanning receiver was undertaken in this thesis, which was the topic of Chapter 7. Details on the A/D converter and the circuit which was constructed to interface the A/D to the DSP board were described. The hardware implementations of the scanning correlation (including signal windowing to decrease the data transfer rate to the DSP) and AGC algorithms was presented. It was demonstrated that the pilot signal scanning receiver could actually detect real-world pilot signals. In Chapter 8, some steps which must be taken to make the pilot signal scanning receiver viable for practical field use were discussed.

9.2 Results

A dual downconversion analog tuner, with IFs at 210.38 MHz and 20.8896 MHz, was chosen. It was determined that the maximum gain of this analog tuner should be 109 dB. The target signal power at the input of the A/D converter was selected to be 16 dBm (for an A/D converter input impedance of 50Ω). It was also found that the signal-to-noise ratio at the input of the A/D converter should be kept greater than 5 dB for adequate performance.

The receiver samples the IF signal, having a center frequency of 20.8896 MHz, four times per chip period, which amounts to a sampling rate of 4.9152 MHz. A correlation period of 1024 chips was found to provide adequate performance. Under heavily loaded conditions, which were simulated in this thesis, with an ideal receiver a discrimination metric, D, of 15 dB can be consistently enjoyed.

However, various factors conspire to decrease D. It was found that a sampling frequency offset of 10 ppm from the nominal can decrease D by 1 dB. To keep the artificial drift induced by sampling frequency offset small, it was determined that the relation $|\delta_s| < 1/2^{15} N_s^2 N_{corr}$

should be satisfied, where δ_s is the fractional sampling frequency offset, N_s is the number of samples per chip, and N_{corr} is the number of chips over which the correlation is performed. Furthermore, the RMS sampling jitter should be kept below 20 ps (40 ps) for the baseband (bandpass) detection method to keep D within 1 dB of it nominal maximum value. Sampling phase offset was found to cause no significant degradation in D.

It was found that the baseband detection method is not as sensitive to quantization effects in the receiver as the bandpass detection method. To keep the bandpass detection method from overflowing a 32-bit accumulator, it was found that the relation $n_1 + n_2 \leq 22$ should be satisfied, where n_1 is the number of bits to which the received signal is quantized and n_2 is the number of bits to which the locally generated pilot signal in the DSP is quantized. To keep the AGC algorithm from overflowing a 32-bit accumulator, it was found that the relation $\log_2(N_{agc}) < 31 \Leftrightarrow 2(n_1 \Leftrightarrow 1) \Leftrightarrow \log_2(N_s)$ must be statisfied.

An interesting result concerns the effect of frequency offset upon D. Because of their fundamentally different frequency responses, the baseband detection method exhibits nulls in the curve of D versus frequency, whereas the bandpass detection method does not. In order to keep the effect of the frequency offset small, it was determined that the relation $T_{corr} \ll 1/\delta_f f_{IF}$ should be satisfied, where T_{corr} is the correlation period in seconds and $\delta_f f_{IF}$ is the absolute frequency offset in Hz. This result indicates that if the frequency offset is large, it is necessary to correlate over a shorter period of time to reduce the effects of the frequency offset. However, shorter correlation periods decrease the D as well.

Lastly, it was found that the local oscillator RMS phase error should be kept smaller than $0.1°$ to incur degradations in D of less than 1 dB.

9.3 Contributions

This thesis has been an exercize in system design engineering. Chapter 4 presented a study of digital receiver architectures, essentially systematizing a method of analyzing these architectures. Chapter 4 should thus be useful to engineers and researchers desiring information on some of the fundamental tradeoffs incurred in digital receiver engineering.

In Chapter 5, two digital methods of pilot signal detection were identified, and ways by which these methods become computationally efficient for implementation on a DSP were developed. In Chapter 6, these two methods of pilot signal detection were studied to quantify their performances under some very specific receiver and channel distortions. These results should be useful to those studying pilot signal acquisition in real-world IS-95A receivers.

In Chapter 5, a digitally-based means of AGC was specified. Engineers involved in the design of digital receivers may find the concepts presented on digitally-based AGC useful.

This thesis has made contributions to the modeling of receiver distortions for computational simulation. In particular, the technique used in this thesis for modeling sampling jitter when using shaped pulses and modeling local oscillator phase noise should be of interest to students and researchers alike. These techniques are detailed in Appendix B. The ability to model these distortions makes it possible to study, via computational simulation, the complex interactions and tradeoffs between the RF and digital portions of the receiver. Such studies are very important in receiver system design engineering.

In the implementation of the receiver, a method was developed to accomplish pilot signal scanning detection which uses partial correlations and windowing of the signal. This method simultaneously reduces the data transfer rate from the A/D converter to the DSP and the memory requirements of the DSP. A digital circuit was developed to window the signal and perform the acquisition of the samples from the A/D converter. These results, along with the suggestions in Chapter 8, should be of benefit to those wishing to implement a practical pilot signal scanning receiver suitable for applications to system deployment engineering.

Appendix A

PCS Fundamentals

Since its inception, the intention of the cellular radio services was to provide a wireless connection, mainly for people in their automobiles, to the public switched telephone network (PSTN). With the new Personal Communication Services (PCS), the intention is to extend this connection, so that people can connect to the PSTN anywhere, not just from their automobiles. In addition to this promise of ubiquitous service, in order to remain competitive with the cellular services, PCS is also striving to maintain a higher quality of service (i.e., more clear-sounding phone calls and fewer dropped calls) and a greater range of services (e.g., call waiting, call forwarding, voice mail, Internet traffic, fax). An idea prevalent in the adoption of PCS, which also accounts for its name, is that each person will have a single phone number which will follow that person wherever he or she goes, so that phone numbers will no longer necessarily be associated with a fixed location in space as is true currently. Rather, when a number is called, the system will be able to locate and ring the person being called, regardless of the region of the country in which the person resides or whether the person is indoors or outdoors.

Between December 5, 1994 and January 14, 1997, the U.S. government auctioned the rights to use parts of the radio frequency spectrum between 1850 MHz and 1990 MHz for broadband PCS. This spectrum was divided into six blocks. Blocks A and B were designated for use in Major Trading Areas (MTAs), while blocks C, D, E, and F were designated for use in Basic Trading Areas (BTAs)[1]. Table A.11 gives the forward link and reverse link frequency assignments for the various blocks [40, p. 233].

[1]In [39, p. 38–39] the U.S. is divided into 47 MTAs and 487 BTAs.

Table A.11: The forward link and reverse link frequency assignments for broadband PCS.

Block	**Forward Link**	**Reverse Link**
A	1930–1945 MHz	1850–1865 MHz
B	1950–1965 MHz	1870–1885 MHz
C	1975–1990 MHz	1895–1910 MHz
D	1945–1950 MHz	1865–1870 MHz
E	1965–1970 MHz	1885–1890 MHz
F	1970–1975 MHz	1890–1895 MHz

In [12, p. 1-2–1-3], the frequency blocks are subdivided into channels 50 kHz in bandwidth. The center frequencies of the reverse link channels are the set of frequencies

$$\{1850.000 + 0.050q \text{ MHz};\ q = 0, 1, \ldots, 1199\}\,, \tag{A.112}$$

and the center frequencies of the forward link channels are the set of frequencies

$$\{1930.000 + 0.050q \text{ MHz};\ q = 0, 1, \ldots, 1199\}\,. \tag{A.113}$$

Furthermore, transmission on any channel having a center frequency 1.2 MHz from the extremities of the frequency blocks is permitted only under special circumstances.

The auction winners have agreed to abide by the rules set forth in [40, p. 232], which stipulate the following:

- A, B, and C block licensees must provide adequate service to one-third of the population in their licensed area within five years of licensing, and to two-thirds of the population in their licensed area within 10 years of licensing, and
- D, E, and F block licensees must provide adequate service to at least one-quarter of the population in their licensed area within five years of licensing.

Failure to meet these regulations will result in license forfeiture.

The rules on power output in [40] dictate that the peak output power of a broadband PCS base station transmitter must never exceed 100 W, and the peak equivalent isotropically radiated power (EIRP) from a base station must never exceed 1640 W, for antenna heights up to 300 m. For antennas heights greater than this, a corresponding reduction in allowed EIRP is enforced. Mobile units are allowed to transmit up to 2 W EIRP.

The FCC regulations in [40, p. 233] mandate that, to mitigate interference outside a service provider's locale, the median field strength at any location on the border of the area of PCS service must not exceed 47 dBμV/m (or about ⇔95 dBm when received through an isotropic antenna), unless special permission is obtained. In some regions, the spectrum between 1850 MHz and 1990 MHz was previously licensed for fixed point microwave services. In [40, p. 233–238], detailed instructions are provided to PCS licensees on assisting previous owners of this spectrum in relocating their services to a different band in the radio spectrum to avoid interference.

Appendix B

System Simulation

B.1 Purpose of the Simulations

The reason for attempting a simulation was to understand the interactions and tradeoffs between uncontrollable and controllable receiver and channel parameters and how those parameters affect the ability of the receiver to detect correlation peaks. The method for doing this was to first produce realistic samples of a signal as would appear at the output of the A/D converter. Then the correlation algorithms were run on these samples and the output was analyzed. This was repeated for many different combinations of simulation parameters.

This appendix provides a detailed explanation of the method by which the realistic samples at the output of the A/D converter were produced, and can be read along with the computer source code found in Appendix G. A logical approach to understanding the simulation methodology is to first explain how samples of a single multipath component from a single base station are generated. This method can then be extended to the generation of several multipath components from a single base station and several multipath components from several base stations. This appendix concludes with a discussion of the following assorted topics: how the signal power is defined and computed, how the AGC is simulated, how the quantization and fixed point arithmetic is handled, details of the bandpass filter for the bandpass detection method, the various types of "noise" in the simulation, and how thermal noise was incorporated into the simulation.

B.2 Generation of Samples of a Single Multipath Component from a Single Base Station

In Chapter 2, it was shown that the forward link IS-95A baseband signal could be represented by two sequences of numbers, $\{s_{Ik}, k = \Leftrightarrow\infty, \ldots, \infty\}$ (I-channel) and $\{s_{Qk}, k = \Leftrightarrow\infty, \ldots, \infty\}$ (Q-channel), which consisted of contributions from the various channels of the IS-95A system. The elements of these sequences are the weights of two trains of IS-95A pulses, thus forming the baseband signals $I(t)$ and $Q(t)$:

$$I(t) = \sum_{k=-\infty}^{\infty} s_{Ik} p(t \Leftrightarrow kT_c) \tag{B.114}$$

$$Q(t) = \sum_{k=-\infty}^{\infty} s_{Qk} p(t \Leftrightarrow kT_c). \tag{B.115}$$

(The IS-95A pulse, $p(t)$, was discussed in Section 2.5.) These two trains of weighted pulses amplitude-modulate two orthogonal sinusoidal signals, forming the transmitted signal $r(t)$:

$$r(t) = I(t)\cos(2\pi f_c t) + Q(t)\sin(2\pi f_c t), \tag{B.116}$$

where f_c is the carrier frequency. The simulation is based upon this representation of the IS-95A signal.

In order to produce the random data for the channels requiring such, a random number generator `ran2.c` [41, pp. 280–282], which is an implementation of L'Ecuyer's method [42], was used to produce random numbers which were uniformly distributed on $[0, 1]$. These random numbers were used to produce data bits having magnitudes ± 1.

It was necessary to produce samples of the signal at arbitrary sampling instants. In the simulation program, these arbitrary sampling instants were produced by tracking the number of the main chip being sampled and the time within the chip at which the sample occurred. Referring to Figure B.49, which shows chips numbered $k \Leftrightarrow 1$, k, and $k+1$, a variable $t_0 \in [\Leftrightarrow 0.5, 0.5]$ is used. The chip being sampled is numbered k and t_0 is the time within the chip at which the sample occurred. The periodicity of the pilot signal was handled by letting k be a non-negative integer, modulo 2^{15}.

After downconversion and filtering by the analog tuner, the signal was assumed to take the form

$$r_{IF}(t) = I(t)\cos(2\pi f'_{IF} t + \theta + \phi(t)) + Q(t)\sin(2\pi f'_{IF} t + \theta + \phi(t)), \tag{B.117}$$

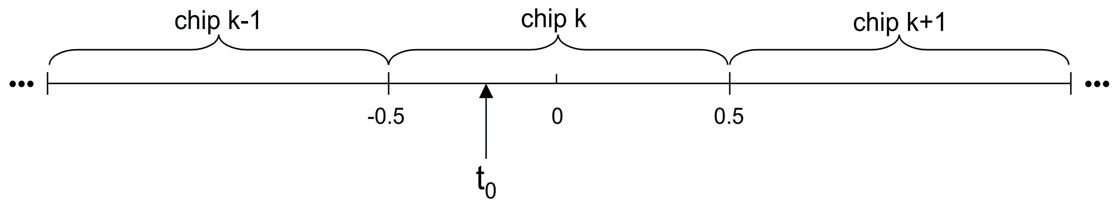

Figure B.49: The variables used to represent the sampling times in the simulation program.

where f'_{IF} is the actual second IF frequency (f_{IF} would be the desired second IF frequency), $\theta \in [0, 2\pi)$ is a constant phase, and $\phi(t)$ is the phase noise produced by the local oscillators.

The method by which the pulse shaping was accounted for can be understood by considering Figure B.50. Pulse shaping spreads the energy of a single chip over several chip periods. This means that at a given sampling time during chip k, not only is the k^{th} chip being sampled, but so are several chips both previous and subsequent to the k^{th} chip.

By this argument, along with the assumption that the transmitted pulses have negligible jitter in pulse spacing, the sample during the k^{th} chip period at time t_0, $s(n, t_0, \theta)$, can be computed as follows:

$$\begin{aligned} s(k, t_0, \theta) &= \sum_{i=-6}^{6} s_{I(k+i)} p(t_0 \Leftrightarrow iT_c) \cos(2\pi f'_{IF} t' + \theta + \phi(t')) \\ &+ \; s_{Q(k+i)} p(t_0 \Leftrightarrow iT_c) \sin(2\pi f'_{IF} t' + \theta + \phi(t')), \end{aligned} \tag{B.118}$$

where t' is the simulated time at which the sample is taken and $t_0 \in [\Leftrightarrow 0.5, 0.5]$. The parameter i takes a maximum value of six since the pulse $p(t)$ is assumed to be zero for $|t| > 5.5T_c$ (see Section 2.5).

As discussed in Section 2.5, a D/A conversion followed by lowpass filtering in the transmitter generates the continuous pulses, $p(t)$. For computational purposes, the value of the continuous pulse $p(t)$ at an arbitrary time was determined by assuming that the filter taps given in Table 2.3 represented the samples of the pulse taken at the instants $\Leftrightarrow 0.125 + mT_c/4$, $(m = \Leftrightarrow 23, \Leftrightarrow 22, \ldots, 23, 24)$, where T_c is the chip period. The value of the pulse at an arbitrary sampling time was determined by cubic spline interpolation on the coordinates given in Table 2.3. The cubic spline interpolation was computed using the `splint.c` program given in [41, p. 116], which uses the program `spline.c` given in [41, p. 115] to compute the second order derivates at the specified points. (This was done using the so-called natural boundary conditions, which specifies that the first order derivative at the end points of the tabulated function are zero.)

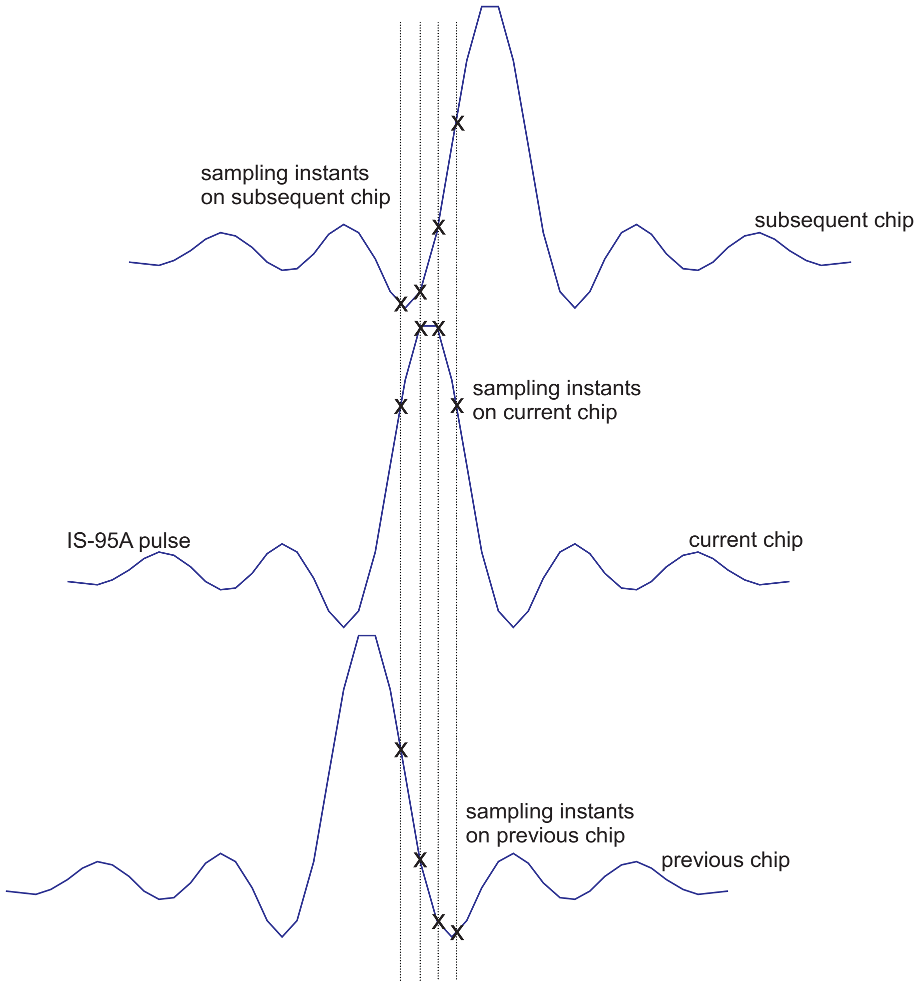

Figure B.50: At a given sampling instant, current, "previous," and "future" pulses are sampled, since pulse shaping in the transmitter spreads the energy of the pulses over several chip periods.

B.3 Advancement of the Time Parameter

Equation B.118 gives a closed-form expression, in terms of the random signals s_{Ii}, s_{Qi}, and the random process $\phi(t)$, for the sample of a single multipath component from a single base station. In the simulation, after a sample was generated at sampling instant t_0, an updated t_0, t_0', was determined as follows:

$$t_0' = t_0 + \frac{1 \Leftrightarrow \delta_s + \delta_j}{N_s} \tag{B.119}$$

where N_s is the number of samples per chip, δ_s is the (constant) sampling frequency offset component (see Section B.3.1), and δ_j is the (random) sampling frequency jitter component (see Section B.3.2). If the generated $t_0' < 0.5$, then the value of k remained the same. However, if $t_0' \geq 0.5$, then t_0' was reset to $t_0' \Leftrightarrow 1$ and k was advanced by unity, modulo 2^{15}.

B.3.1 Sampling Frequency Offset

The nominal time interval between samples, T_s, is determined by the nominal number of samples per chip, N_s, according to

$$T_s = \frac{T_c}{N_s}, \tag{B.120}$$

where T_c is the time duration of a chip and N_s is the number of samples per chip. Thus, the nominal sampling frequency is

$$f_s = \frac{1}{T_s} = \frac{N_s}{T_c}. \tag{B.121}$$

With sampling frequency offset, the time interval between samples is either increased or decreased, depending upon whether the receiver samples too fast or too slow.

Sampling frequency offset was modeled by adding a small constant to the nominal sampling interval, to make the actual nominal sampling duration, T_s', the following:

$$T_s' = T_s \left(1 \Leftrightarrow \delta_s\right). \tag{B.122}$$

Here, δ_s is the sampling frequency offset, and is defined as

$$\delta_s = \frac{f_s' \Leftrightarrow f_s}{f_s'}, \tag{B.123}$$

where f_s' is the actual nominal sampling frequency and f_s is the desired nominal sampling frequency.

B.3.2 Sampling Jitter

The small random component, δ_j which modeled the sampling jitter was assumed to be distributed according to a Gaussian distribution having a specified variance $(\chi_{rms}^{jitter})^2$. The Gaussian distribution was generated using the Box-Mueller method [43, pp. 175–176], and the uniform distribution required of this method was generated using the `ran2.c` routine from [41, pp. 280–282].

B.4 Advancement of the Phase Parameter

B.4.1 LO Frequency Offset

In Equation B.118, the frequency offset between the local oscillator and the carrier frequency of the signal was modeled by letting the center frequency of the IF signal equal f'_{IF}, where f'_{IF} differs from the desired center frequency, f_{IF}, according to the equation

$$f'_{IF} = f_{IF}\left(1 + \delta_f\right). \tag{B.124}$$

The LO frequency offset can be due to either the first LO, the second LO, or both.

B.4.2 LO Phase Noise

Finding the RMS Phase Error

In Equation 6.84 is a mathematical expression of the fact that real-world oscillators have random phase fluctuations. If these phase fluctuations are not too great, then Equation 6.84 can be rewritten as follows:

$$\begin{aligned} c(t) &= A_c \cos\left(\omega_c t\right) \cos\left(\phi(t)\right) \Leftrightarrow A_c \sin\left(\omega_c t\right) \sin\left(\phi(t)\right) && \text{(B.125)} \\ &\approx A_c \cos\left(\omega_c t\right) \Leftrightarrow A_c \phi(t) \sin\left(\omega_c t\right). && \text{(B.126)} \end{aligned}$$

Taking the Fourier transform of $c(t)$, $C(\omega)$, we have

$$C(\omega) = \frac{A_c}{2}\left[\delta(\omega \Leftrightarrow \omega_c) + \delta(\Leftrightarrow \omega \Leftrightarrow \omega_c)\right] \Leftrightarrow j\frac{A_c}{2}\left[\Phi(\omega \Leftrightarrow \omega_c) \Leftrightarrow \Phi(\Leftrightarrow \omega \Leftrightarrow \omega_c)\right], \tag{B.127}$$

where $\Phi(\omega)$, the Fourier transform of $\phi(t)$, is the phase noise spectrum. Equation B.127 shows that, if the phase noise component is small enough, the phase noise spectrum of an oscillator overlays the main components.

The manufacturers of an oscillator will customarily publish the measured or theoretical power spectral density of its phase noise $\phi(t)$, $S_{\phi\phi}(\omega)$ [44]. The power spectral density is usually normalized with respect to $A_c^2/2$, and is plotted or tabulated in dBc/Hz versus offset from the carrier frequency in Hz. The power spectral density is usually provided for only one sideband, so the mean-square value of $\phi(t)$, $\left(\chi_{rms}^{phase}\right)^2$, can be found by integrating the power spectral density over one sideband and multiplying the result by a factor of 2.

Tabulated results on the phase noise distribution are usually provided at discrete points in frequency. The phase noise distribution, expressed in dBc/Hz, depends upon frequency according to the functional form $10a\log_{10}(f) + c$, where a and c are constants over a range of frequencies [28].

In general, suppose that N points of the distribution are provided between the frequencies $[f_i, f_{i+1}]$, $i = 0, 1, \ldots, N \Leftrightarrow 1$. Further suppose that the line segments fixed by these points have slopes a_i and intercepts c_i. Then, integrating over all segments, the mean-squared phase angle is

$$\left(\chi_{rms}^{phase}\right)^2 = 2\sum_{i=0}^{N}\int_{f_i}^{f_{i+1}} 10^{\frac{a_i \log f + c_i}{10}}\,df \tag{B.128}$$

$$= 2\sum_{i=0}^{N} 10^{c_i/10}\left[g(f, a_i)\right]_{f_i}^{f_{i+1}}, \tag{B.129}$$

where

$$g(f, a_i) = \begin{cases} \ln f \text{ for } a_i = \Leftrightarrow 1 \\ \frac{f^{(a_i+1)}}{a_i+1} \text{ otherwise} \end{cases}. \tag{B.130}$$

The factor of two is necessary to include the contributions of both sidebands, as discussed above.

As an application of Equation B.129, consider the Vectron CO-287W sinewave crystal oscillator, which has the phase noise distribution given in Table B.12 [45, p. 27]. From this table, the following slopes and intercepts are found:

$$(a_1, c_1) = (\Leftrightarrow 2.5, \Leftrightarrow 38), \tag{B.131}$$

$$(a_2, c_2) = (\Leftrightarrow 1.5, \Leftrightarrow 83), \tag{B.132}$$

and

$$(a_3, c_3) = (\Leftrightarrow 0.7, \Leftrightarrow 99). \tag{B.133}$$

In addition, it is assumed that the phase noise spectrum is constant below 100 Hz and above 50 kHz, where the white noise floor is assumed to dominate. The corresponding slopes and intercepts are

$$(a_0, c_0) \quad = \quad (0.0, \Leftrightarrow 88) \qquad \text{(B.134)}$$

and

$$(a_4, c_4) \quad = \quad (0.0, \Leftrightarrow 133). \qquad \text{(B.135)}$$

The phase noise spectrum is assumed to cut off at half the frequency of the oscillator. With these facts, by Equation B.129, it is found that $\left(\chi_{rms}^{phase}\right)^2 = 2 \times 10^{-6}$ rad^2 or $\chi_{rms}^{phase} = 0.05°$. This value is similar to that found in [44].

Table B.12: Phase noise values for the Vectron CO-287W sinewave crystal oscillator.

Offset from Carrier	**Phase Noise**
100 Hz	⇔88 dBc/Hz
1 kHz	⇔113 dBc/Hz
10 kHz	⇔128 dBc/Hz
50 kHz	⇔133 dBc/Hz

Linking the RMS Phase Error to the Spectrum

Numerical values of the phase noise $\phi(t)$ were randomly generated for simulations. The distribution of these numerical values needed to have (1) specific spectral properties and (2) a specified variance. The spectral properties were modeled by segregating the phase noise spectrum into two components: (1) a "close-in" component, and (2) a white noise floor component.

For modeling the "close-in" component, an approach was followed similar to what was alluded to in [46], in which the frequency response of a one-pole Butterworth filter was used to represent the spectrum of the phase noise, since the rolloff of the one-pole Butterworth filter has the same rolloff as the phase noise spectrum. Therefore, the desired spectral properties of the sequence of random phase noise numbers are produced by filtering a sequence of numbers having spectral properties of white noise with a one-pole Butterworth filter.

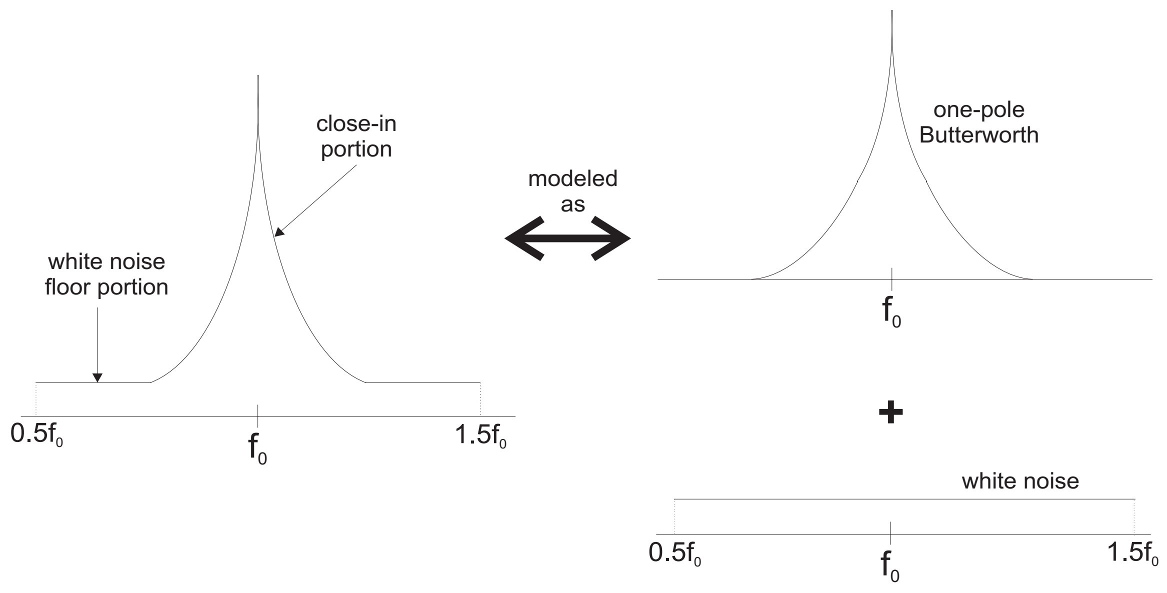

Figure B.51: The approach to modeling the phase noise spectrum was to segregate it into two components: a close-in component and a white noise floor component. The close-in component was modeled as a Butterworth spectrum. It is shown below that the bandwidth of the Butterworth can be related to the RMS phase deviation.

The variance of the phase noise sequence can be controlled by properly setting the bandwidth of the one-pole Butterworth filter. The expression relating the power spectral density (PSD) of the white noise sequence, $S_{xx}(\omega)$, to the PSD of the filtered sequence, $S_{yy}(\omega)$, is

$$S_{yy}(\omega) = |H(\omega)|^2 S_{xx}(\omega). \tag{B.136}$$

The PSD $S_{xx}(\omega)$ is denoted $S_{xx}(\omega) = N_0/2$, where $N_0/2$ is the two-sided PSD of white noise.

A one-pole Butterworth filter with a cutoff frequency Ω_c has the following transfer function

$$H(s) = \frac{1}{1 + j(\Omega/\Omega_c)}, \tag{B.137}$$

where Ω is the continuous-time frequency. Since the filtering is to be performed in the discrete-time domain, the continuous-time $H(s)$ must be transformed into its discrete-time counterpart. This is accomplished by the bilinear transform, in which the frequency terms are replaced by their pre-warped counterparts:

$$\Omega \rightarrow \frac{2}{T}\tan\left(\frac{\omega T_s}{2}\right), \tag{B.138}$$

$$\Omega_c \rightarrow \frac{2}{T}\tan\left(\frac{\omega_c T_s}{2}\right), \tag{B.139}$$

where $\Leftrightarrow\pi < \omega T_s \leq \pi$. After this transformation, it is found that the discrete-time frequency response of the single-pole Butterworth filter is

$$H(\omega) = \frac{1}{1 + j\left(\frac{\tan(\omega T_s/2)}{\tan(\omega_c T_s/2)}\right)}. \tag{B.140}$$

The relationship between the unfiltered and filtered variances can be found by integrating Equation B.136. The integration of the left hand side of this equation gives

$$\int_{-\omega_s/2}^{\omega_s/2} S_{yy}(\omega)\, d\omega = \sigma_y^2, \tag{B.141}$$

where ω_s is the sampling frequency. Integration of the right hand side of Equation B.136 gives

$$\begin{aligned}
\sigma_y^2 &= \frac{N_0}{2}\int_{-\omega_s/2}^{\omega_s/2} |H(\omega)|^2\, d\omega && \text{(B.142)}\\
&= \frac{N_0}{2}\int_{-\omega_s/2}^{\omega_s/2} \frac{1}{1 + \left(\frac{\tan(\omega T_s/2)}{\tan(\omega_c T_s/2)}\right)^2} d\omega && \text{(B.143)}\\
&= \frac{N_0}{2}\frac{2}{T_s}\int_{-\pi/2}^{\pi/2} \frac{\cos^2(x)}{1 + b\sin^2(x)} dx && \text{(B.144)}\\
&= \frac{N_0}{2}\frac{2}{T_s}\left[\frac{\sqrt{1+b}}{b}\tan^{-1}\left(\sqrt{1+b}\tan x\right) \Leftrightarrow \frac{x}{b}\right]_{-\pi/2}^{\pi/2} && \text{(B.145)}\\
&= \frac{N_0}{2}\frac{2\pi}{T_s}\left[\frac{\sqrt{1+b}\Leftrightarrow 1}{b}\right], && \text{(B.146)}
\end{aligned}$$

where the integration was performed with the aid of the tables in [47, p. 246], and

$$b = \frac{1}{\tan^2(\omega_c T/2)} \Leftrightarrow 1. \tag{B.147}$$

Since

$$\frac{N_0}{2}\frac{2\pi}{T_s} = N_0 \times \mathrm{BW} = \sigma_x^2, \tag{B.148}$$

where BW is the noise bandwidth, the following relation results:

$$\sigma_y^2 = \frac{\sqrt{1+b}\Leftrightarrow 1}{b}\sigma_x^2. \tag{B.149}$$

Equations B.147 and B.149 together show that the variance of the filtered sequence can be regulated by the cutoff frequency of the filter.

The simulation of the close-in phase noise proceeded by specifying a mean-squared phase noise angle, $\left(\chi_{rms}^{phase}\right)^2$, and equating this to σ_y^2. This specified a value of the parameter b in

Equation B.149, which, in turn, specified a value for the cutoff frequency of the single-pole Butterworth filter through the relation given in Equation B.147. When a white noise sequence having variance $\sigma_x^2 = \pi^2$ was filtered with the Butterworth filter, the resulting filtered sequence possessed both the spectral properties and the variance desired for simulating the phase noise, in radians, of a real-world oscillator.

The white noise floor portion of the oscillator spectrum was simulated by generating random samples of a Gaussian distribution with a specified variance, σ_w^2, in addition to the close-in portion. The white noise floor was assumed to contribute some fraction, ϵ, to the total variance σ_t^2, or $\sigma_w^2 = \epsilon\sigma_t^2$. The close-in portion of the spectrum was assumed to contribute the remainder of the total variance, so that $\sigma_y^2 = (1 \Leftrightarrow \epsilon)\sigma_t^2 = \sigma_x^2$. Figure B.52 shows the spectrum of simulated phase noise, with and without the white noise floor.

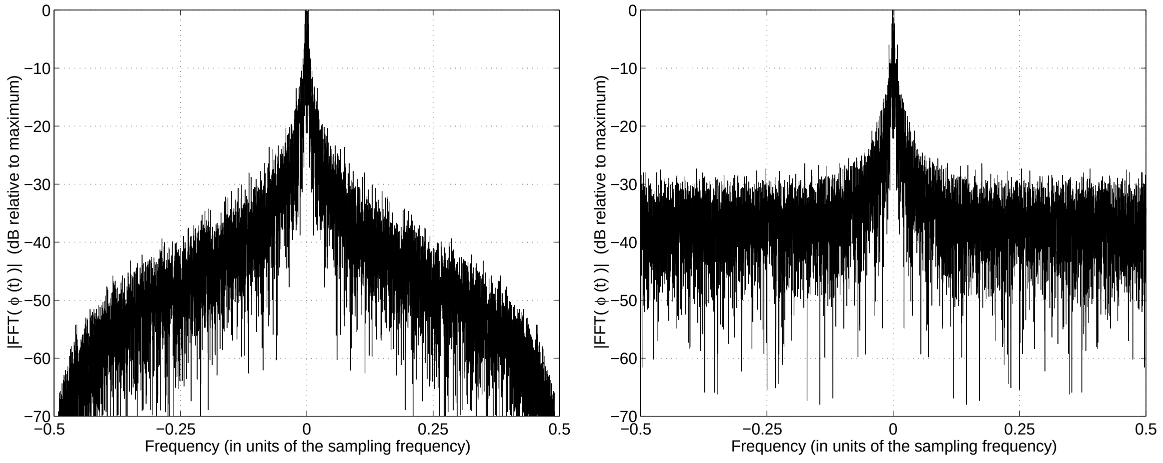

Figure B.52: The spectrum of simulated phase noise. On the left, the white noise floor has been omitted. On the right, the full simulation, with the white noise floor included, is shown.

B.5 Generation of Samples of Multiple Multipath Components from a Single Base Station

Section B.2 discussed how samples of a single multipath component from a single base station were generated. As discussed in Chapter 3, the channel can be expected to produce multipath components at the receiver.

As seen in Figure B.53, a sampling time of t' on one multipath component corresponds to sampling times of $t' \Leftrightarrow \tau_i$, $i = 1, \ldots, M$ on the other multipath components, where τ_i is

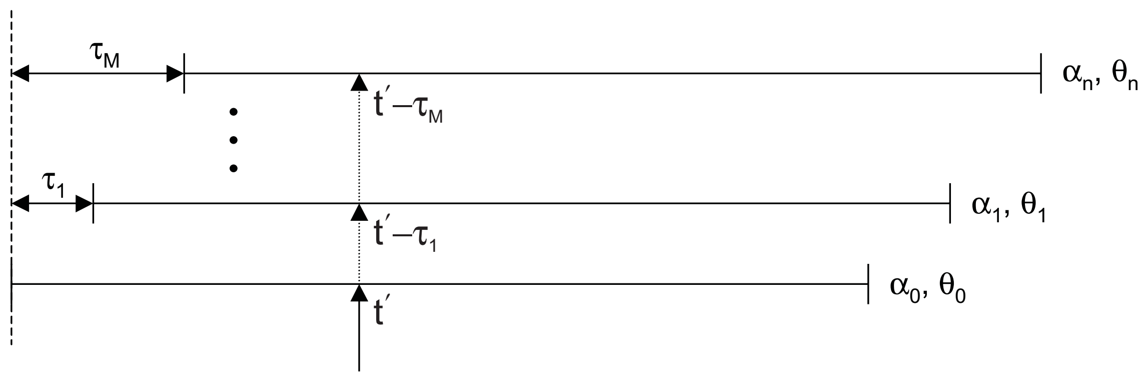

Figure B.53: The sample on one multipath component corresponds to samples on the other multipath components from the same base station.

the time in units of a chip period by which the i^{th} multipath component is delayed. The multipath components are assigned amplitudes and carrier phases of α_i $(i = 0, \ldots, M)$ and θ_i $(i = 0, \ldots, M)$, respectively. With these considerations, the sample at time t_0 of the k^{th} chip of the signal arriving at the receiver from a single base station, $r(k, t_0)$, can be written

$$r(k, t_0) = \sum_{i=0}^{M} \alpha_i s(k + \lfloor t_0 + 0.5 \Leftrightarrow \tau_i \rfloor, t_0 \Leftrightarrow (\tau_i \Leftrightarrow [\tau_i]), \theta_i), \tag{B.150}$$

where $s(k, t_0, \theta_i)$ is defined by Equation B.118, and $[\bullet]$ indicates that the argument $\bullet$ should be rounded to the nearest integer.

B.6 Generation of Samples of Multiple Multipath Components from Multiple Base Stations

Figure B.53 has been expanded to arrive at Figure B.54, which illustrates how sampling not only captures the signals of multiple multipath components from a given base station, but also captures the signal of multiple multipath components from multiple base stations. Equation B.150 is extended to represent this situation. The sample $q(k, t_0)$, which includes all multipath from all base stations, can be expressed

$$q(k, t_0) = \sum_{i=1}^{B} \beta_i \sum_{j=0}^{M} \alpha_{ij} s_i (k + \lfloor t_0 + 0.5 \Leftrightarrow \tau_{ij} \Leftrightarrow \tau_i \rfloor, t_0 \Leftrightarrow (\tau_{ij} + \tau_i \Leftrightarrow [\tau_{ij} + \tau_i]), \theta_{ij}), \tag{B.151}$$

where β_j is the relative amplitude of the signal contributed by the j^{th} base station and B is the number of base stations contributing to the signal.

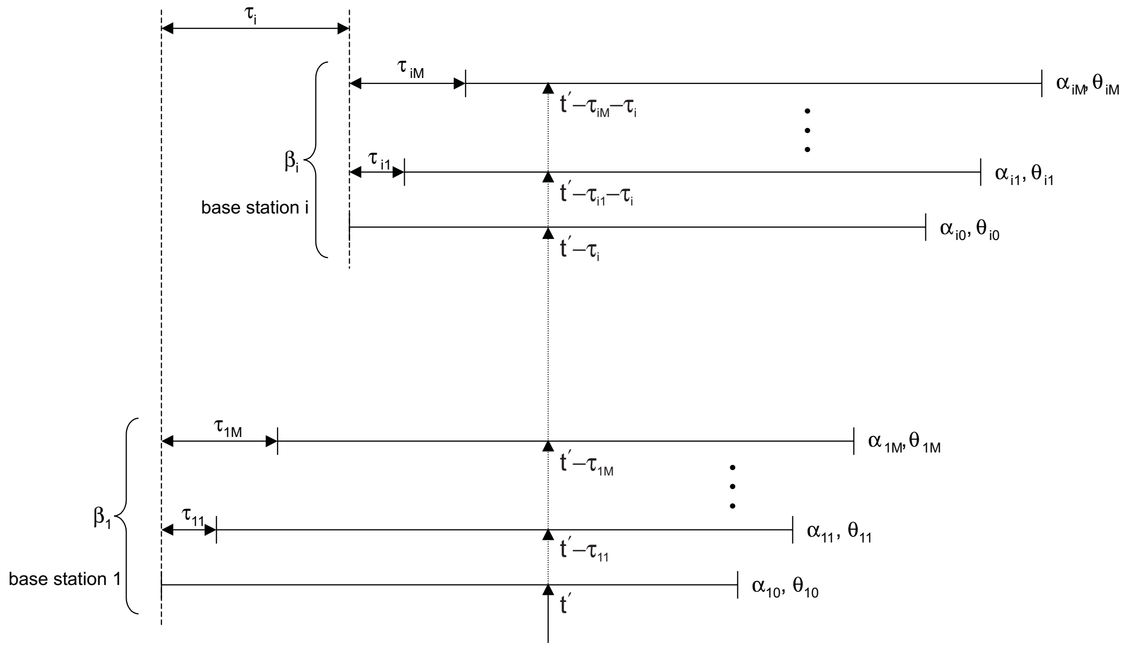

Figure B.54: The sample on one multipath component corresponds to samples on multiple multipath components from multiple base stations.

B.7 Computation of the Signal Power

Since all channels of the IS-95A forward link are orthogonal, the power levels of the channels (P_{pilot}, P_{sync}, P_{paging}, $P_{traffic}$) were specified such that

$$P_{pilot} + P_{sync} + P_{paging} + P_{traffic} = 1. \tag{B.152}$$

The number of pilot channels, sync channels, and paging channels was always assumed to be one apiece. Thus, the only other variable requiring specification was the number of active traffic channels, $n_{traffic}$. Then the amplitudes for each individual channel type (see Chapter 2) were determined as follows:

$$a_{pilot} = \sqrt{P_{pilot}} \tag{B.153}$$
$$a_{sync} = \sqrt{P_{sync}} \tag{B.154}$$
$$a_{paging} = \sqrt{P_{paging}} \tag{B.155}$$
$$a_{traffic} = \sqrt{\frac{P_{traffic}}{n_{traffic}}}. \tag{B.156}$$

In the simulation, a unity power signal was generated for each base station. As discussed above, the multipath phenomenon was modeled by adding together delayed versions of the signal with relative amplitudes α_{ij}, $i = 1, \ldots, M$, where M is the number of multipath components. All the multipath components from all base stations were added together (with the appropriate phase offsets) with relative weightings β_i, $i = 1, \ldots, B$, where B is the number of base stations, to arrive at the composite received signal.

It can be safely assumed that all components (i.e., all multipath components from all base stations) of the composite received signal are mutually orthogonal. Then the simulated signal power, S_{sim}, is

$$S_{sim} = \sum_{i=1}^{B} \sum_{j=1}^{M} \alpha_{ij}^2 \beta_i^2. \tag{B.157}$$

B.8 Simulation of AGC

The main AGC algorithm is described in Section 5.2.2. The AGC was modeled by multiplying the composite signal, including thermal noise, by a number, $\sqrt{G}$, which was continuously updated according to the following algorithm[1]. For every N_{agc} samples generated, the estimate of the signal variance, σ_{est}^2, was computed from the (quantized) values of the samples, $\{s_i\}$, according to

$$\sigma_{est}^2 = \frac{1}{N_{agc}} \sum_{i=1}^{N_{agc}} s_i^2. \tag{B.158}$$

Then the current gain value, G, was updated to a new gain value, G', according to the following equation:

$$G' = G \frac{\sigma_{target}^2}{\sigma_{est}^2}, \tag{B.159}$$

where σ_{target}^2 is a target variance. Eventually, G approached the value

$$G \rightarrow \frac{\sigma_{target}^2}{S_{sim}}. \tag{B.160}$$

B.9 Simulation of Quantization

In the simulation, the samples were initially generated using floating point arithmetic to emulate the continuous voltages at the input of the A/D converter. The A/D converter

[1]Note that the scaling of the signal occurs after the addition of noise, simulates an analog tuner architecture in which the gain control occurs on the final step before the A/D converter. Thus, it is assumed that most of the thermal noise is added by components prior to the gain control amplifier and that the gain control amplifier itself does not add a significant amount of noise.

maps signals between the voltages V_{lo} and V_{hi} to integer numbers between $[\Leftrightarrow 2^{n-1}, 2^{n-1} \Leftrightarrow 1]$, where n is the number of bits of resolution of the A/D converter.

The quantization was accomplished by dividing the region between the voltages $[V_{lo}, V_{hi}]$ into 2^n evenly-spaced bins. These bins were numbered between $\Leftrightarrow 2^{n-1}$ and $2^{n-1} \Leftrightarrow 1$, inclusive. The bin width was $\Delta = (V_{hi} \Leftrightarrow V_{lo})/2^n$, and the bin boundaries were at

$$\{ \Leftrightarrow\infty; V_{lo} + (2m \Leftrightarrow 1)\Delta/2, m = 1, \ldots, 2^n \Leftrightarrow 1; +\infty \} \tag{B.161}$$

The simulate the A/D conversion process, for each continuous voltage sample generated, a search is performed to identify the bin in which the sample lies, and the corresponding number is assigned to the sample.

B.10 Generation of Quantized Pilot Signals

All the computations for the receiver algorithms were performed using integer arithmetic. This was done because the digital signal processor which was selected for implementing the receiver, the Texas Instruments TMS320C541, is a fixed point machine.

To generate the quantized pilot signals, the pulse shaping filter coefficients in Table 2.3 were scaled by a factor, g, and converted to integers by rounding to the integer closest to zero. The factor g was chosen so that the most extreme negative and positive outputs of filtering the bipolar pilot PN sequences would lie at exactly $\Leftrightarrow 2^{15}$ and $2^{15} \Leftrightarrow 1$. If the pilot signal was desired to n_2-bit resolution, then the filter output was scaled to n_2 bits by dividing the filter output by 2^{16-n_2}.

B.11 Filtering in the Bandpass Detection Method

Section 5.1.2 discussed the presence of a digital bandpass filter for extracting a tone from the signal while rejecting the noise. It was also shown that the center frequency of this filter should be placed at $f_s/4$ to minimize the number of operations. This filtering was done using integer arithmetic to simulate the DSP. A specific example is given here to illustrate the method employed.

Assume the implementation of a filter of bandwidth 500 Hz is being attempted, the sampling rate is $f_s = 4.9152$ MHz, and the center frequency of the filter is at $f_s/4 = 1.2288$ MHz. From Section 5.1.2, along with Appendix D, it is found that $\omega_{low}T_s = 0.4999\pi$ and

$\omega_{high}T_s = 0.5001\pi$, making

$$\omega_0^2 = 1 \tag{B.162}$$
$$W = 6.3916 \times 10^{-4} \tag{B.163}$$

which produces the coefficients

$$B_0 = \frac{W}{1 + W + \omega_0^2} = 3.1948 \times 10^{-4} \tag{B.164}$$
$$A_1 = 0 \tag{B.165}$$
$$A_2 = 0.99936. \tag{B.166}$$

In order to implement the filtering in integer arithmetic, these coefficients were scaled by a constant, k, which was some power of two such that $\lfloor kB_0 \rfloor$ was a single-digit number. For the case illustrated here, $k = 2^{14}$ was used. As a result,

$$B_0' = \lfloor kB_0 \rfloor = 5 \tag{B.167}$$
$$A_2' = \lfloor kA_2 \rfloor = 16374. \tag{B.168}$$

In the bandpass correlation method, the signal was filtered using the coefficients B_0' and A_2', and the result was always right-shifted 14 digits to rescale the output. This procedure effected filtering with the coefficients B_0 and A_2 yet using intger arithmetic.

B.12 "Noise" in the Simulation

The term "noise" assumes several meanings in this thesis, based upon the context. Of interest is the noise at two locations in the receiver: (1) at the input of the A/D converter and (2) at the output of the correlation algorithms.

At the input of the A/D converter, the term "noise" refers to the thermal noise added by the analog components prior to the A/D converter. The effect of this noise was studied in Section 6.5.3. In addition, the CDMA signal contains not only the pilot signals which are of interest, but also the voltage summation of all other CDMA channels. If there are a large number of channels, as was illustrated in Section 5.2.1, this voltage summation appears in the receiver as Gaussian noise and certainly thwarts the pilot signal detection process. However, the SNR at the input of the A/D converter refers only to the ratio of the total power of all pilot signals (see Section B.7) to the power of the thermal noise.

At output of the correlation algorithms, noise refers mainly to the output which is observed when no pilot signal is in phase with the locally generated pilot signal. This phenomenon exists for a number of reasons. First, since partial correlation is employed, the probability of observing a similar (and therefore partially correlated) sequence to the local sequence has a non-trivial probability. However, this probability should decrease as the period of the partial correlation increases.

Second, any true pilot signal which arrives at the receiver (such as multipath components or signals from other base stations), although out of phase with the locally generated pilot signal, will nevertheless exhibit some small degree of correlation with the locally generated pilot signal. Third, the noise due to the use of all other CDMA channels (other than the pilot channel) in the system will induce non-trivial correlation outputs. This is also true of thermal noise at the input of the A/D converter. These two types of noise at the input of the A/D converter will add a random component to the pilot signal. When no pilot signal is in phase with the locally generated pilot signal, this random component induces another random component at the output of the correlation, much like noise which passes through a matched filter followed by envelope detection. Finally, there is quantization noise due to the A/D converter. Even if there were no other sources of noise in the system, the quantization of the signal would induce an error, which would appear as noise and hence would induce a random component at the output of the correlation algorithm.

B.13 Addition of Thermal Noise

Thermal noise was incorporated into the simulation by adding a random number from a zero-mean Gaussian distribution to each sample of the signal. The variance of the Gaussian distribution was determined by specifying a signal-to-noise ratio (SNR), which was held constant throughout the simulation. Assuming the simulated signal power to be S_{sim} (see Section B.7), and the AGC scaling factor to be $\sqrt{G}$, the variance of the Gaussian noise samples, σ_n^2, was taken to be

$$\sigma_n^2 = \frac{S_{sim} G}{\mathrm{SNR}}. \tag{B.169}$$

The Gaussian distribution was generated using the Box-Mueller method [43, pp. 175–176], and the uniform distribution required of this method was generated using the `ran2.c` routine from [41, pp. 280–282]. Note that this method of adding noise assumes that most of the thermal noise is contributed by components prior to the stage which provides the gain control.

Appendix C

Parameter Values for Simulations

C.1 Simulations for Analysis of P_{target}

Table C.13: Simulation parameters used to determine the effects of varying P_{target}.

Parameter	**Value**	**Parameter**	**Value**
N_{corr}	1024 chips	BW$_{bp}$	500 Hz
SNR$_{A/D}$	20 dB	P_{target}	$\in [⇔60, 60]$ dB
f_{IF}	20.8896 MHz	δ_f	0.0 ppm
χ_{rms}^{phase}	0.0°	ϵ	N/A
N_s	4	δ_s	0.0 ppm
χ_{rms}^{jitter}	0.0 ps	t_0^{offset}	0.0 chip
n_1	8	n_2	8(BB), 6(BP)
M	3	N_{ch}	16
$\{\tau_i\}$	$\{0, 3, 6\}$ chip	$\{\alpha_i\}$	$\{1.00, 0.71, 0.50\}$
B	3	$\{\beta_i\}$	$\{1.00, 0.71, 0.50\}$
$[V_{lim}^{lo}, V_{lim}^{hi}]$	$[⇔2.5, 2.5]$ V	–	–

C.2 Simulations for Analysis of Thermal Noise

Table C.14: Simulation parameters used to determine the effects of thermal noise.

Parameter	**Value**	**Parameter**	**Value**
N_{corr}	1024 chips	BW$_{bp}$	500 Hz
SNR$_{A/D}$	$\in [\Leftrightarrow 5, 20]$ dB	P_{target}	2 V^2
f_{IF}	20.8896 MHz	δ_f	0.0 ppm
χ_{rms}^{phase}	0.0°	ϵ	N/A
N_s	4	δ_s	0.0 ppm
χ_{rms}^{jitter}	0.0 ps	t_0^{offset}	0.0 chip
n_1	8	n_2	8(BB), 6(BP)
M	3	N_{ch}	16
$\{\tau_i\}$	$\{0, 3, 6\}$ chip	$\{\alpha_i\}$	$\{1.00, 0.71, 0.50\}$
B	3	$\{\beta_i\}$	$\{1.00, 0.71, 0.50\}$
$[V_{lim}^{lo}, V_{lim}^{hi}]$	$[\Leftrightarrow 2.5, 2.5]$ V	–	–

C.3 Simulations for Analysis of N_{corr}

Table C.15: Simulation parameters used to determine the effects of varying N_{corr}.

Parameter	**Value**	**Parameter**	**Value**
N_{corr}	$\in [512, 8192]$ chips	BW$_{bp}$	500 Hz
SNR$_{A/D}$	10 dB	P_{target}	2 V^2
f_{IF}	20.8896 MHz	δ_f	0.0 ppm
χ_{rms}^{phase}	0.0°	ϵ	N/A
N_s	4	δ_s	0.0 ppm
χ_{rms}^{jitter}	0.0 ps	t_0^{offset}	0.0 chip
n_1	8	n_2	8(BB), 6(BP)
M	3	N_{ch}	16
$\{\tau_i\}$	$\{0, 3, 6\}$ chip	$\{\alpha_i\}$	$\{1.00, 0.71, 0.50\}$
B	3	$\{\beta_i\}$	$\{1.00, 0.71, 0.50\}$
$[V_{lim}^{lo}, V_{lim}^{hi}]$	$[\Leftrightarrow 2.5, 2.5]$ V	–	–

C.4 Simulations for Analysis of t_0^{offset}

Table C.16: Simulation parameters used to determine the effects of sampling phase offset.

Parameter	**Value**	**Parameter**	**Value**
N_{corr}	1024 chips	BW_{bp}	500 Hz
$\text{SNR}_{A/D}$	20 dB	P_{target}	2 V^2
f_{IF}	20.8896 MHz	δ_f	0.0 ppm
χ_{rms}^{phase}	0.0°	ϵ	N/A
N_s	4	δ_s	0.0 ppm
χ_{rms}^{jitter}	0.0 ps	t_0^{offset}	$[\Leftrightarrow 0.125, 0.125]$ chip
n_1	8	n_2	8(BB), 6(BP)
M	1	N_{ch}	1
$\{\tau_i\}$	$\{0\}$ chip	$\{\alpha_i\}$	$\{1.00\}$
B	1	$\{\beta_i\}$	$\{1.00\}$
$[V_{lim}^{lo}, V_{lim}^{hi}]$	$[\Leftrightarrow 2.5, 2.5]$ V	–	–

C.5 Simulations for Analysis of Sampling Frequency Offset

Table C.17: Simulation parameters used to determine the effects of sampling frequency offset.

Parameter	**Value**	**Parameter**	**Value**
N_{corr}	1024 chips	BW_{bp}	500 Hz
$\text{SNR}_{A/D}$	10 dB	P_{target}	2 V^2
f_{IF}	20.8896 MHz	δ_f	0.0 ppm
χ_{rms}^{phase}	0.0°	ϵ	N/A
N_s	4	δ_s	$\in [\Leftrightarrow 50, 50]$ ppm
χ_{rms}^{jitter}	0.0 ps	t_0^{offset}	0.0 chip
n_1	8	n_2	8(BB), 6(BP)
M	3	N_{ch}	16
$\{\tau_i\}$	$\{0, 3, 6\}$ chip	$\{\alpha_i\}$	$\{1.00, 0.71, 0.50\}$
B	3	$\{\beta_i\}$	$\{1.00, 0.71, 0.50\}$
$[V_{lim}^{lo}, V_{lim}^{hi}]$	$[\Leftrightarrow 2.5, 2.5]$ V	–	–

C.6 Simulations for Analysis of Sampling Frequency Jitter

Table C.18: Simulation parameters used to determine the effects of sampling frequency jitter.

Parameter	Value	Parameter	Value
N_{corr}	1024 chips	BW_{bp}	500 Hz
$\text{SNR}_{A/D}$	10, 20 dB	P_{target}	2 V^2
f_{IF}	20.8896 MHz	δ_f	0.0 ppm
χ_{rms}^{phase}	0.0°	ϵ	N/A
N_s	4	δ_s	0.0 ppm
χ_{rms}^{jitter}	$\in [0, 100]$ ps	t_0^{offset}	0.0 chip
n_1	8	n_2	8(BB), 6(BP)
M	3	N_{ch}	16
$\{\tau_i\}$	$\{0, 3, 6\}$ chip	$\{\alpha_i\}$	$\{1.00, 0.71, 0.50\}$
B	3	$\{\beta_i\}$	$\{1.00, 0.71, 0.50\}$
$[V_{lim}^{lo}, V_{lim}^{hi}]$	$[\Leftrightarrow 2.5, 2.5]$ V	–	–

C.7 Simulations for Analysis of Fixed Point Effects

Table C.19: Simulation parameters used to determine the effects of a fixed point implementation.

Parameter	Value	Parameter	Value
N_{corr}	1024 chips	BW_{bp}	500 Hz
$\text{SNR}_{A/D}$	10 dB	P_{target}	2 V^2
f_{IF}	20.8896 MHz	δ_f	0.0 ppm
χ_{rms}^{phase}	0.0°	ϵ	N/A
N_s	4	δ_s	0.0 ppm
χ_{rms}^{jitter}	0.0 ps	t_0^{offset}	0.0 chip
n_1	$\{4, 8, 12\}$	n_2	$\{4, 8, 12\}$
M	3	N_{ch}	16
$\{\tau_i\}$	$\{0, 3, 6\}$ chip	$\{\alpha_i\}$	$\{1.00, 0.71, 0.50\}$
B	3	$\{\beta_i\}$	$\{1.00, 0.71, 0.50\}$
$[V_{lim}^{lo}, V_{lim}^{hi}]$	$[\Leftrightarrow 2.5, 2.5]$ V	–	–

C.8 Simulations for Analysis of Frequency Offset

Table C.20: Simulation parameters used to determine the effects of frequency offset.

Parameter	Value	Parameter	Value
N_{corr}	1024 chips	BW_{bp}	500 Hz
$\mathrm{SNR}_{A/D}$	20 dB	P_{target}	2 V^2
f_{IF}	20.8896 MHz	δ_f	$\in [\Leftrightarrow 120, 120]$ ppm
χ_{rms}^{phase}	0.0°	ϵ	N/A
N_s	4	δ_s	0.0 ppm
χ_{rms}^{jitter}	0.0 ps	t_0^{offset}	0.0 chip
n_1	8	n_2	8(BB), 6(BP)
M	3	N_{ch}	16
$\{\tau_i\}$	$\{0, 3, 6\}$ chip	$\{\alpha_i\}$	$\{1.00, 0.71, 0.50\}$
B	3	$\{\beta_i\}$	$\{1.00, 0.71, 0.50\}$
$[V_{lim}^{lo}, V_{lim}^{hi}]$	$[\Leftrightarrow 2.5, 2.5]$ V	–	–

C.9 Simulations for Analysis of LO Phase Noise

Table C.21: Simulation parameters used to determine the effects of LO phase noise.

Parameter	Value	Parameter	Value
N_{corr}	1024 chips	BW_{bp}	500 Hz
$\mathrm{SNR}_{A/D}$	20 dB	P_{target}	2 V^2
f_{IF}	20.8896 MHz	δ_f	0.0 ppm
χ_{rms}^{phase}	$\in [0.00°, 0.25°]$	ϵ	0.2
N_s	4	δ_s	0.0 ppm
χ_{rms}^{jitter}	0.0 ps	t_0^{offset}	0.0 chip
n_1	8	n_2	8(BB), 6(BP)
M	3	N_{ch}	16
$\{\tau_i\}$	$\{0, 3, 6\}$ chip	$\{\alpha_i\}$	$\{1.00, 0.71, 0.50\}$
B	3	$\{\beta_i\}$	$\{1.00, 0.71, 0.50\}$
$[V_{lim}^{lo}, V_{lim}^{hi}]$	$[\Leftrightarrow 2.5, 2.5]$ V	–	–

C.10 Simulations for Analysis of Multipath

Table C.22: Simulation parameters used to determine the effects of multipath.

Parameter	**Value**	**Parameter**	**Value**
N_{corr}	1024 chips	BW_{bp}	500 Hz
$\text{SNR}_{A/D}$	20 dB	P_{target}	2 V^2
f_{IF}	20.8896 MHz	δ_f	0.0 ppm
χ_{rms}^{phase}	0.0°	ϵ	N/A
N_s	4	δ_s	0.0 ppm
χ_{rms}^{jitter}	0.0 ps	t_0^{offset}	0.0 chip
n_1	8	n_2	8(BB), 6(BP)
M	$\in [1, 6]$	N_{ch}	16
$\{\tau_i\}$	see Figure 6.30	$\{\alpha_i\}$	see Figure 6.30
B	1	$\{\beta_i\}$	$\{1.00\}$
$[V_{lim}^{lo}, V_{lim}^{hi}]$	$[\Leftrightarrow 2.5, 2.5]$ V	–	–

C.11 Simulations for Analysis of Multiple Base Stations

Table C.23: Simulation parameters used to determine the effects of multiple base stations.

Parameter	**Value**	**Parameter**	**Value**
N_{corr}	1024 chips	BW_{bp}	500 Hz
$\text{SNR}_{A/D}$	20 dB	P_{target}	2 V^2
f_{IF}	20.8896 MHz	δ_f	0.0 ppm
χ_{rms}^{phase}	0.0°	ϵ	N/A
N_s	4	δ_s	0.0 ppm
χ_{rms}^{jitter}	0.0 ps	t_0^{offset}	0.0 chip
n_1	8	n_2	8(BB), 6(BP)
M	$\in [1, 3]$	N_{ch}	16
$\{\tau_i\}$	see Figure 6.30	$\{\alpha_i\}$	see Figure 6.30
B	$\in [1, 3]$	$\{\beta_i\}$	$\{1.00, 1.00, 1.00\}$
$[V_{lim}^{lo}, V_{lim}^{hi}]$	$[\Leftrightarrow 2.5, 2.5]$ V	–	–

Appendix D

Digital Bandpass Filter Design

A single-pole (IIR) filter was selected for the bandpass filtering required in the bandpass correlation algorithm. The design of this filter began by adopting the following lowpass filter response, $H(s)$:

$$H(s) = \frac{1}{1 + s/\Omega_c}, \tag{D.170}$$

where s is the continuous-time complex frequency parameter and Ω_c is the cutoff frequency. A lowpass to bandpass transformation was effected according to the following prescription:

$$\frac{s}{\Omega_c} \quad \Leftrightarrow\rightarrow \quad \frac{s^2 + {\omega_0'}^2}{sW'}, \tag{D.171}$$

where

$${\omega_0'}^2 \quad = \quad \omega'_{low}\omega'_{high} \tag{D.172}$$

and

$$W' \quad = \quad \omega'_{high} \Leftrightarrow \omega'_{low}. \tag{D.173}$$

The quantities ω'_{high} and ω'_{low}, which are specified in the s-domain, are the upper and lower cutoff frequencies of the filter, respectively, making W' the filter bandwidth. These cutoff frequencies are obtained by specifying the z-domain cutoff frequencies ω_{high} and ω_{low}, and then computing the s-domain cutoff frequencies through the following formulas:

$$\omega'_{low} \quad = \quad \frac{2}{T}\tan\left(\frac{\omega_{low}T_s}{2}\right), \tag{D.174}$$

and

$$\omega'_{high} \quad = \quad \frac{2}{T}\tan\left(\frac{\omega_{high}T_s}{2}\right), \tag{D.175}$$

where T is arbitrary (with units of time) and T_s is the sampling interval. The quantities $\omega_{low}T_s$ and $\omega_{high}T_s$ assume values between zero and π.

A bilinear transformation was used to map the s-domain filter into its z-domain counterpart. This was accomplished by making the following substitution in Equation D.170 (after the transformation of Equation D.170 by D.171):

$$s \Leftrightarrow\!\rightarrow \left(\frac{2}{T}\right)\frac{z \Leftrightarrow 1}{z+1}. \tag{D.176}$$

Some algebraic manipulation yielded the following discrete-time filter:

$$H(z) \quad = \quad \frac{\frac{W}{1+W+\omega_0^2}\left(1 \Leftrightarrow z^{-2}\right)}{1+\left[\frac{2\left(\omega_0^2-1\right)}{1+W+\omega_0^2}\right]z^{-1}+\left[\frac{1-W+\omega_0^2}{1+W+\omega_0^2}\right]z^{-2}}, \tag{D.177}$$

where

$$W = \frac{W'}{2/T} \tag{D.178}$$

and

$$\omega_0^2 = \frac{{\omega'_0}^2}{(2/T)^2}. \tag{D.179}$$

The implementation of this filter in fixed point arithmetic is discussed in Section B.11.

Appendix E

Recovery of Input Amplitude

A block diagram of the simulation scheme is given in Figure E.55. In the simulations, the signal $r(t)$, having power S_{sim}, is generated. The signal $r(t)$ is scaled by the AGC scaling factor $\sqrt{G}$. Gaussian noise, having a variance $\sigma_n^2 = S_{sim}G/\mathrm{SNR}$, is added to the signal next, where SNR is the simulated signal-to-noise ratio. Notice that this model does not assume that the noise power at the input of the A/D converter is constant. This would be the case if the noise contributed by the gain control amplifier was dominated by the noise contributed by all the components prior to the gain control amplifier.

The raw output of the correlation algorithms is indicative of the degree of correlation between the received signal and the locally generated pilot signal, and can be used by itself to determine whether a pilot signal is present. However, the raw output can also be used to produce an estimate, $\widetilde{a_0}$, the amplitude of the received pilot signal. The method by which this estimate can be obtained is the subject of this appendix.

E.1 $\widetilde{a_0}$ for the Baseband Detection Method

An expression for the output of the baseband detection method was provided in Equations 5.45 and 5.46 in Section 5.1.1. The algorithm relies on the desired part of the outputs, $a_0^2 R_{II_p}^2(n_o)/4$ and $a_0^2 R_{QQ_p}^2(n_o)/4$, being much larger than the interference part of the output, for $n_o = 0$. Therefore, only the terms

$$a_0^2 R_{II_p}^2(n_o)/4 \tag{E.180}$$

and

$$a_0^2 R_{QQ_p}^2(n_o)/4 \tag{E.181}$$

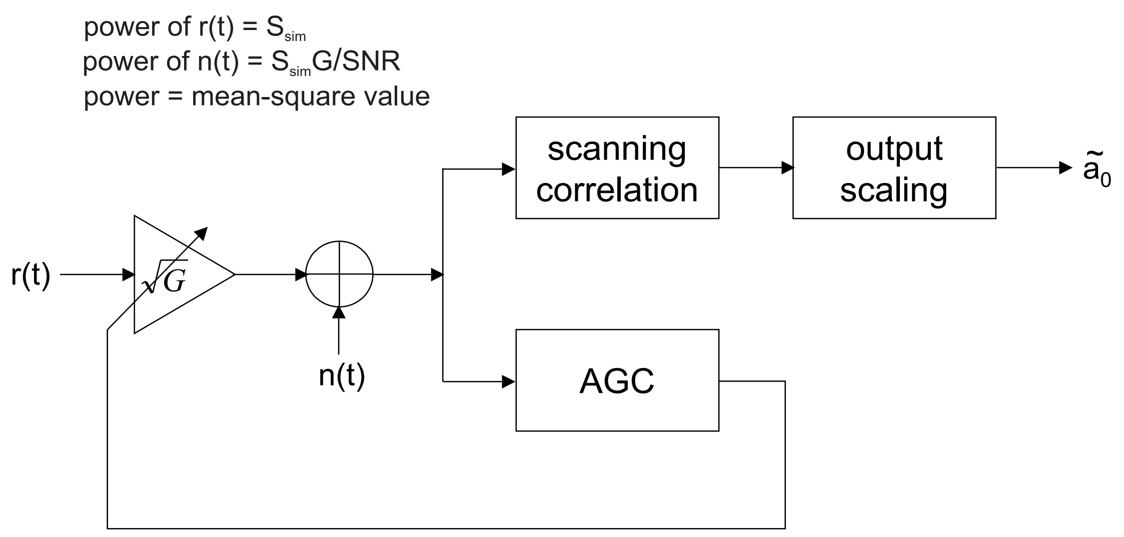
power of r(t) = S_{sim}
power of n(t) = $S_{sim}G/SNR$
power = mean-square value
r(t)
$\sqrt{G}$
n(t)
scanning correlation
output scaling
$\tilde{a}_0$
AGC

Figure E.55: Basic scheme and assumptions of the simulation program.

are treated in further analysis here. From Equations 5.43 and 5.44, we have

$$R_{II_p}(n_o) = \sum_{i=0}^{N_sN_{corr}-1} P_I(i)P_I(i+n_o) \tag{E.182}$$

$$R_{QQ_p}(n_o) = \sum_{i=0}^{N_sN_{corr}-1} P_Q(i)P_Q(i+n_o). \tag{E.183}$$

To normalize the output, the values of $R_{II_p}(0)$ and $R_{QQ_p}(0)$ must be determined. Here, the analysis is performed only for $R_{II_p}(0)$, but identical results hold for $R_{QQ_p}(0)$.

Let the sample at time t' of the I-channel pilot signal be

$$P_{Ij}(t') = \sum_{k=-\infty}^{\infty} g_j c_{Ik} p\,(t' \Leftrightarrow kT_c)\,, \tag{E.184}$$

where $c_{Ik} = c_{I(k+2^{15}m)}$ is the I-channel pilot PN sequence (m is any integer) and g_j ($j = 1, 2$) is a known scaling factor which depends upon whether the received pilot signal or the locally generated pilot signal is being considered. The function $p(t)$ is the IS-95A pulse and T_c is the chip period. Following the reasoning provided in Section B.2 and Figure B.50, the overlapping nature of the pulses can be accounted for by writing the i^{th} sample, ($i = 1, \ldots, 4$) on an arbitrary chip (labeled x) as follows:

$$s_{xi} = \sum_{l=-6}^{6} c_{I(x+l)} h_{21+i-4l}, \tag{E.185}$$

where h_k ($k = 0, \ldots, 47$) are the tap weights of the IS-95A pulse shaping filter in Table 2.3 and sampling phase offset has been ignored.

The quantity $R_{II_p}(0)$ results when the received signal is exactly in-phase with the locally generated signal in the receiver. Thus

$$\begin{aligned} R_{II_p}(0) &= g_1 g_2 \sum_{x=1}^{N_{corr}} \sum_{i=1}^{4} s_{xi}^2 && \text{(E.186)} \\ &= g_1 g_2 \sum_{x=1}^{N_{corr}} \sum_{i=1}^{4} \left(\sum_{l=-6}^{6} c_{I(x+l)} h_{21+i-4l} \right)^2 && \text{(E.187)} \\ &= g_1 g_2 \sum_{x=1}^{N_{corr}} \sum_{i=1}^{4} \sum_{l=-6}^{6} \sum_{k=-6}^{6} c_{I(x+l)} c_{I(x+k)} h_{21+i-4l} h_{21+i-4k} && \text{(E.188)} \end{aligned}$$

where N_{corr} is the number of chips over which the correlation is performed. The sequence $c_{I(x+l)}$, while produced deterministically in the transmitter, is safely assumed to be random in nature.

Since $R_{II_p}(0)$ is random in nature, its value can be estimated by taking its expectation. Taking the expectation of Equation E.188, we have

$$E\left\{R_{II_p}(0)\right\} = g_1 g_2 \sum_{x=1}^{N_{corr}} \sum_{i=1}^{4} \sum_{l=-6}^{6} \sum_{k=-6}^{6} E\left\{c_{I(x+l)} c_{I(x+k)}\right\} h_{21+i-4l} h_{21+i-4k}. \tag{E.189}$$

Because of the random nature of the pilot channel PN sequence, we can write

$$E\left\{c_{I(x+l)} c_{I(x+k)}\right\} = \delta_{lk}, \tag{E.190}$$

where δ_{lk} is the Kroneker delta. Thus, we have

$$E\left\{R_{II_p}(0)\right\} = g_1 g_2 \sum_{x=1}^{N_{corr}} \sum_{i=1}^{4} \sum_{k=-6}^{6} h_{21+i-4k}^2. \tag{E.191}$$

The summation $\sum_{k=-6}^{6} h_{21+i-4k}^2$ comprises 12 terms for each value of i. (The term $k = 6$ will be zero for $i = 1, 2$, and the term $k = \Leftrightarrow 6$ will be zero for $i = 3, 4$.) The sets of terms are mutually exclusive, and combined they make up the 48 coefficients of the IS-95A filter of Table 2.3. Thus,

$$\sum_{i=1}^{4} \sum_{k=-6}^{6} h_{21+i-4k}^2 = \sum_{m=0}^{47} h_m^2. \tag{E.192}$$

From Table 2.3, it is seen that

$$\sum_{m=0}^{47} h_m^2 = 3.9403. \tag{E.193}$$

Then the following can be written

$$E\left\{R_{II_p}(0)\right\} = 3.9403 g_1 g_2 N_{corr}, \tag{E.194}$$

where N_{corr} is the number of chips over which the correlation is performed.

Using the same notation for the correlator output as given in Equation 5.45, $\widetilde{a_0}$ can be determined by

$$\widetilde{a_0} = \frac{\sqrt{4 O_I^{bb}(0)}}{3.9403 g_1 g_2 N_{corr}}. \tag{E.195}$$

Let g_1 be the factor by which the received pilot signal is scaled. Then, if the full scale range of the A/D converter is V_{FS}, and the number of A/D bits is n_1, then we have

$$g_1 = \frac{2^{n_1}}{V_{FS}}. \tag{E.196}$$

Further, in Section B.10, the scaling of the pulse shaping filter coefficients so that the most extreme values of the locally generated pilot signals lie at -2^{n_2} and $2^{n_2}-1$ is discussed. This was found to be possible by using the scale factor

$$g_2 = \frac{15586}{2^{16-n_2}}. \tag{E.197}$$

E.2 $\widetilde{a_0}$ for the Bandpass Detection Method

From Equation 5.59, the output of the bandpass detection method, $O_I^{bp}(n_o)$, is

$$O_I^{bp}(n_o) \;=\; \frac{1}{2} \sum_{i=0}^{N_s N_{corr}-1} \left(P_I(i) P_I(i+n_o)\right)^2 + i_I^{bp}(n_o). \tag{E.198}$$

The normalization factor is determined by $O_I^{bp}(0)$, which entails finding $E\left\{O_I^{bp}(0)\right\}$. Computing the expectation, we have

$$E\left\{O_I^{bp}(0)\right\} \approx \frac{1}{2} R^2 g_1^2 g_2^2 a_0^2 \sum_{x=1}^{N_{corr}} \sum_{i=1}^{4} E\left\{\left(\sum_{l=-6}^{6} c_{I(x+l)} h_{21+i-4l}\right)^4\right\}. \tag{E.199}$$

R quantifies the apparent decrease in energy a sinusoid traversing the bandpass filter would experience due to the rise time of the filter:

$$R = \frac{E_{filt}}{E_{input}}, \tag{E.200}$$

where E_{filt} is the apparent energy of a filtered sinusoid and E_{input} is the energy of the sinusoid at the input of the filter. In general, $R < 1$, and $R \rightarrow 1$ as $N_{corr} \rightarrow \infty$.

With a reasoning similar to that above, it is concluded that

$$E\left\{c_{I(x+l)}c_{I(x+k)}c_{I(x+m)}c_{I(x+n)}\right\} = \delta_{lkmn}, \tag{E.201}$$

so that Equation E.199 reduces to

$$E\left\{O_I^{bp}(0)\right\} = \frac{1}{2}R^2 a_0^2 g_1^2 g_2^2 \sum_{i=1}^{N_{corr}} \sum_{m=0}^{47} h_m^4. \tag{E.202}$$

From Table 2.3, it is found that

$$\sum_{m=0}^{47} h_m^4 = 2.8457. \tag{E.203}$$

Thus,

$$E\left\{O_I^{bp}(0)\right\} = \frac{2.8457}{2} a_0^2 R^2 g_1^2 g_2^2 N_{corr}. \tag{E.204}$$

Therefore, if the output $O_I^{bp}(0)$ is measured, an estimate of the amplitude of the pilot signals, $\widetilde{a_0}$, is computed as

$$\widetilde{a_0} = \sqrt{\frac{2O_I^{bp}(0)}{2.8457 R^2 g_1^2 g_2^2 N_{corr}}}. \tag{E.205}$$

Appendix F

Third-Order Products and Spurious Free Dynamic Range

F.1 Third-Order Products

Assume two tones of frequencies f_1 and f_2 are present at the input of an RF device. Nonlinearities of the device tend to produce output components at frequencies $mf_1 \pm nf_2$, where m and n are non-negative integers. The most detrimental of these, because they often fall in the frequency band of interest, are the third-order products. These products occur at frequencies for which $(m, n) = (1, 2)$ or $(m, n) = (2, 1)$, and are illustrated in Figure F.56. It can be shown that for each 1 dB increase in the input power, the output power of the desired products increases by 1 dB, but the output power of the third-order products increases by 3 dB [30].

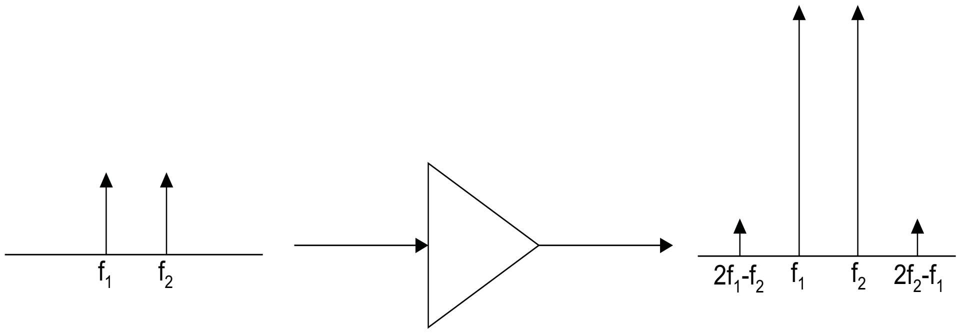

Figure F.56: The output spectrum of an RF device contains third-order products.

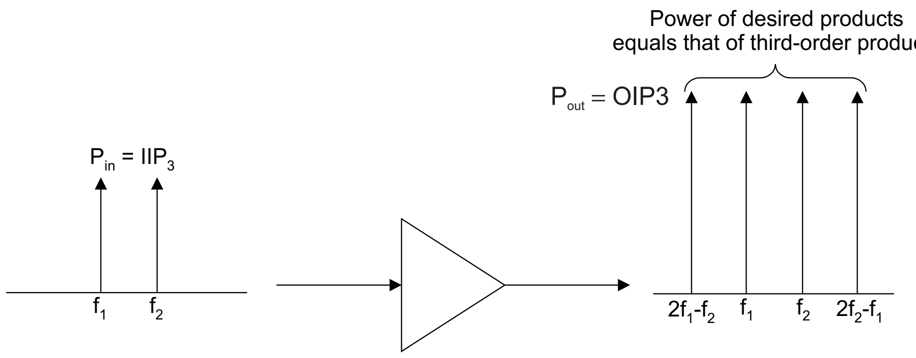
Power of desired products
equals that of third-order products
P_out = OIP3
P_in = IIP_3
f_1
f_2
2f_1-f_2
f_1
f_2
2f_2-f_1

Figure F.57: The concept of third-order intercept.

F.2 Third-Order Intercept Point

The input third-order intercept point, IIP3, is defined as the input power of the desired tones (at frequencies f_1 and f_2) which causes the power of the third-order products (at frequencies $2f_1 \Leftrightarrow f_2$ and $2f_2 \Leftrightarrow f_1$) to equal the output power of the desired tones. This is illustrated in Figure F.57.

The term "intercept point" is used because IIP3 is usually determined by plotting both the output power of the desired tones and the third-order tones versus the input power of the desired tones. The output third-order intercept, OIP3, is the output power at which the two straight lines cross. IIP3 is found by dividing OIP3 by the gain of the device.

As defined, IIP3 is actually fictitious. This is because practical devices saturate (i.e., exhibit a compressive behavior by possessing a maximum output power) when the input power is ten or 15 dB below IIP3. Thus, practical devices can never produce any products having output powers of OIP3. Nevertheless, IIP3 stands as a commonly-used measure of the ability of a device to handle signals with large input powers. In receiver design, a large IIP3 is desirable.

F.3 Cascaded Third-Order Intercept Point

Suppose two devices are cascaded, as shown in Figure F.58. The output power of the third-order products in dBm, $P_{3out}(\text{dBm})$, in terms of the input power of the desired products in dBm, $P_{in}(\text{dBm})$, and the gain of the device in dB, $G(\text{dB})$, can be expressed as follows [30]:

$$P_{3out}(\text{dBm}) = G(\text{dB}) + 3P_{in}(\text{dBm}) \Leftrightarrow 2\text{IIP3}(\text{dBm}) \qquad \text{(F.206)}$$

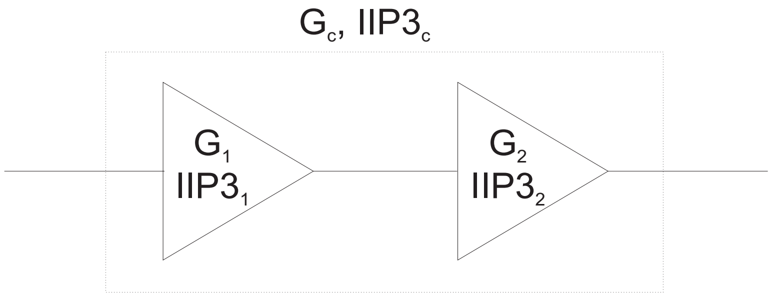

Figure F.58: A cascade of two RF devices. The gain and IIP3 of the first device are G_1 and IIP3$_1$, respectively, and the gain and IIP3 of the second device are G_2 and IIP3$_2$, respectively. The overall gain and IIP3 are G_c and IIP3$_c$, respectively.

where IIP3 is the input third-order intercept point. Written in absolute units

$$P_{3out} = G(\mathrm{IIP3})\left(\frac{P_{in}}{\mathrm{IIP3}}\right)^3. \tag{F.207}$$

Since the input and output third-order intercept points are related as OIP3 $= G(\mathrm{IIP3})$, Equation F.207 can be written

$$P_{3out} = \mathrm{OIP3}\left(\frac{GP_{in}}{\mathrm{OIP3}}\right)^3. \tag{F.208}$$

If the input powers of the desired tones are each P_i, at the output of the first device the powers of the desired components are each G_1P_i and the powers of the third-order products, P_{3A}, are each

$$P_{3A} = \mathrm{OIP3}_1\left(\frac{G_1P_i}{\mathrm{OIP3}_1}\right)^3. \tag{F.209}$$

The second device will (1) produce independent third-order products from the desired tones, each having powers of

$$P_{32} = \mathrm{OIP3}_2\left(\frac{G_1G_2P_i}{\mathrm{OIP3}_2}\right)^3, \tag{F.210}$$

and (2) amplify the third-order products from the output of the first device. Since these two contributions will add in phase at the output of the second device, the total output power of the third-order terms, P_{3B}, is obtained as the sum of the squares of the voltages. Thus, the total power in the third-order products at the output of the second device is

$$\begin{aligned} P_{3B} &= (\sqrt{G_2P_{3A}} + \sqrt{P_{32}})^2 && \text{(F.211)}\\ &= G_2P_{3A} + P_{32} + 2\sqrt{G_2P_{3A}P_{32}} && \text{(F.212)} \end{aligned}$$

$$= G_2\,(\mathrm{OIP3}_1)\left(\frac{G_1 P_i}{\mathrm{OIP3}_1}\right)^3 + (\mathrm{OIP3}_2)\left(\frac{G_1 G_2 P_i}{\mathrm{OIP3}_2}\right)^3 + \tag{F.213}$$

$$2\sqrt{G_2\,(\mathrm{OIP3}_1)\left(\frac{G_1 P_i}{\mathrm{OIP3}_1}\right)^3 (\mathrm{OIP3}_2)\left(\frac{G_1 G_2 P_i}{\mathrm{OIP3}_2}\right)^3} \tag{F.214}$$

$$= G_1^3 G_2^3 P_i^3 \left(\frac{1}{G_2\,(\mathrm{OIP3}_1)} + \frac{1}{\mathrm{OIP3}_2}\right)^2 . \tag{F.215}$$

The cascaded configuration can be replaced by an equivalent system having total gain G_1G_2 and output third-order intercept $\mathrm{OIP3}_c$. If the input power into this equivalent system is P_i, then the output power of the third-order products, $P_{3B\,c}$, is

$$P_{3B\,c} \;=\; \mathrm{OIP3}_c\left(\frac{G_1 G_2 P_i}{\mathrm{OIP3}_c}\right)^3 . \tag{F.216}$$

Equating P_{3B} and $P_{3B\,c}$ Equations F.215 and F.216, respectively, and simplifying, we have

$$\frac{1}{\mathrm{OIP3}_c} = \frac{1}{G_2\,(\mathrm{OIP3}_1)} + \frac{1}{\mathrm{OIP3}_2}. \tag{F.217}$$

To refer $\mathrm{OIP3}_c$ to the input of the equivalent system, we must divide by the gain G_1G_2. Defining $\mathrm{IIP3}_c = \mathrm{OIP3}_c/(G_1G_2)$, we obtain a relation for the third-order intercept of two cascaded devices:

$$\left.\frac{1}{\mathrm{IIP3}_c}\right|_{\text{two device}} = \frac{G_1}{\mathrm{OIP3}_1} + \frac{G_1 G_2}{\mathrm{OIP3}_2} \tag{F.218}$$

$$= \frac{1}{\mathrm{IIP3}_1} + \frac{G_1}{\mathrm{IIP3}_2}. \tag{F.219}$$

The most general formula for the cascaded third-order intercept, which follows from Equation F.219 by induction, is

$$\left.\frac{1}{\mathrm{IIP3}_c}\right|_{\text{general}} = \frac{1}{\mathrm{IIP3}_1} + \frac{G_1}{\mathrm{IIP3}_2} + \frac{G_1 G_2}{\mathrm{IIP3}_3} + \ldots . \tag{F.220}$$

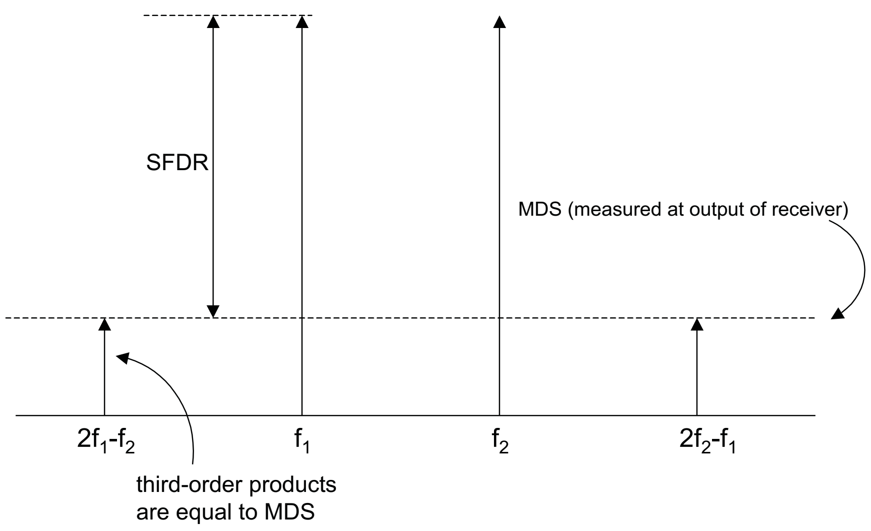
SFDR
MDS (measured at output of receiver)
$2f_1$-f_2
f_1
f_2
$2f_2$-f_1
third-order products are equal to MDS

Figure F.59: The spectrum at the output of a receiver, for two tones at the input. The SFDR is illustrated.

F.4 Spurious Free Dynamic Range

The minimum detectable signal power (MDS) was defined in Section 6.6.1 as

$$\text{MDS} = N_{th} + \text{NF} + \text{SNR}_{min}, \tag{F.221}$$

where N_{th} is the thermal noise floor of the receiver in dBm, NF is the noise figure of the receiver in dB, and SNR_{min} is the minimum desired input SNR in dB. Physically, the MDS is the minimum signal power at the receiver input for which adequate detection can be performed.

A useful and common figure-of-merit for radio receivers is the spurious free dynamic range (SFDR), a graphic illustration of which is provided in Figure F.59. The SFDR is defined as the difference in dB between the output power of the desired products and the MDS (as measured at the output of the receiver) when the third-order products are equal in power to the MDS (again, measured at the output of the receiver). Due to the fact that the third-order products increase three times as fast as the desired products, it can be shown [30, p. 81] that the SFDR can be written

$$\text{SFDR} = \frac{2}{3}\left(\text{IIP3} \Leftrightarrow \text{MDS}\right). \tag{F.222}$$

The receiver behaves in a linear fashion between the MDS and MDS + SFDR.

Appendix G

Simulation Source Code

The main executable cdma_sim, which the Makefile builds, accepts various parameters as input. These can be determined by typing cdma_sim as the Unix prompt. Some parameters also can be changed in the header file setup.h. The output is two files of correlation outputs – one for the baseband correlation algorithm and one for the bandpass correlation algorithm. These files have the extension *.dat. There are also two files which are print outs of the parameters used in the simulation. These files have the extension *.prm.

G.1 Makefile

Run make at the Unix prompt to compile and link the executable cdma_sim.

```
#
OBJFILES =  adv_phs.o adv_ts.o agc.o chck_bnd.o chckinpt.o filter.o genintr.o \
gennoise.o gensamp.o gensrop.o genwalsh.o prntdata.o quantize.o ran2.o rcvr1fps.o \
rcvr2fps.o splint.o
OBJFILES2 = timercv2.o
HDRFILES = chanmaps.h is95puls.h setup.h info_mp.h

cdma_sim:  cdma_sim.c $(OBJFILES) $(OBJFILES2) $(HDRFILES)
CC -g +w $(OBJFILES) $(OBJFILES2) cdma_sim.c -lm -o cdma_sim

agc.o: agc.c setup.h
CC -g +w -c agc.c

adv_phs.o: adv_phs.c setup.h
CC -g +w -c adv_phs.c

adv_ts.o: adv_ts.c setup.h
CC -g +w -c adv_ts.c

chck_bnd.o: chck_bnd.c setup.h
CC -g +w -c chck_bnd.c

chckinpt.o: chckinpt.c setup.h
CC -g +w -c chckinpt.c
```

```
filter.o: filter.c
CC -g +w -c filter.c

genintr.o: genintr.c ran2.o genwalsh.o chanmaps.h setup.h
CC -g +w -c genintr.c

gennoise.o: gennoise.c setup.h ran2.o
CC -g +w -c gennoise.c

gensamp.o: gensamp.c is95puls.h setup.h
CC -g +w -c gensamp.c

gensrop.o: gensrop.c setup.h
CC -g +w -c gensrop.c

genwalsh.o: genwalsh.c setup.h
CC -g +w -c genwalsh.c

prntdata.o: prntdata.c setup.h
CC -g +w -c prntdata.c

quantize.o: quantize.c setup.h
CC -g +w -c quantize.c

ran2.o: ran2.c
CC -g +w -c ran2.c

rcvr1fps.o: rcvr1fps.c is95puls.h setup.h
CC -g +w -c rcvr1fps.c

rcvr2fps.o: rcvr2fps.c filter.o is95puls.h setup.h
CC -g +w -c rcvr2fps.c

splint.o: splint.c
CC -g +w -c splint.c

timercv2.o: timercv2.c setup.h
CC -g +w -c timercv2.c

backup: $(OBJFILES) $(HDRFILES)
mv backup/*.c backup/*.h backup/backupprev
cp *.c *.h backup

clean:
\rm -f *.o
```

G.2 Source Code

G.2.1 adv_phs.c

```
#include <stdlib.h>
#include <math.h>
#include "setup.h"

int adv_phs(
             double f_if,
             double prev_samp,
             double curr_samp,
             double *phase,
 struct sim_parm *parm_ptr
           )
{
//
// THIS ROUTINE ADVANCES THE CARRIER PHASE.
// THE VARIABLES prev_samp AND curr_samp ARE GIVEN IN TERMS
// OF A CHIP PERIOD, WITH 0 AS THE CHIP CENTER.
//
    double dphi;
    double dt;
    double white_noise[2];
    static int first = TRUE;
    static double A_onepole;
    static double B_onepole[2];
    static double filt_ips[2];
    static double filt_ops = 0.0;
    static int oldest_ip = 0;
    double B_output;
    double A_output;
    double phase_noise;

    if ( first ) {
        double omega_c;
        double sqerr;
        first = FALSE;
        //
        // DETERMINE THE CUTOFF FREQUENCY
// NOTE THAT THE PHASE NOISE IS GENERATED IN DEGREES
        //
        sqerr = (1.0-WHITE_NOISE_FRAC) * pow( parm_ptr->rms_phs_err/180.0, 2.0 ) ;
        omega_c = 2.0 * atan( sqerr/(1.0-sqerr) );
        //
        // DETERMINE THE BUTTERWORTH FILTER COEFFICIENTS
        //
        B_onepole[0] = B_onepole[1] = tan( omega_c/2.0 ) / ( tan( omega_c/2.0 ) + 1.0 );
        A_onepole = ( tan( omega_c/2.0 ) - 1.0 ) / ( tan( omega_c/2.0 ) + 1.0 );
    }

//
    // DETERMINE THE SAMPLING INTERVAL IN CHIP PERIOD UNITS
//
    dt = curr_samp - prev_samp;
    if ( dt < 0.0 ) dt += 1;

//
// DETERMINE THE TOTAL NUMBER OF CYCLES THE CARRIER HAS ADVANCED
//
    dphi = f_if * dt / F_CHIP;
//
// DETERMINE THE FRACTIONAL PART OF A CYCLE THE CARRIER HAS ADVANCED
//
    dphi = 2.0 * PI * ( dphi - floor( dphi ) );
```

```
        //
        // GENERATE WHITE NOISE WITH UNITY VARIANCE
        //
        gennoise( 1.0, white_noise );
        //
        // THE FOLLOWING LINES OF CODE IMPLEMENT AN IIR FILTER
        //   USING TWO FIR FILTERS. THE FILTER IS A ONE-POLE
        //   BUTTERWORTH FILTER HAVING CUTOFF FREQUENCY omega_c.
        //
        filt_ips[oldest_ip] = PI * white_noise[0];
        oldest_ip = (++oldest_ip)%2;
        B_output = fir_filter( B_onepole, filt_ips, oldest_ip, 2 );
        A_output = A_onepole * filt_ops;
        filt_ops = B_output - A_output;
phase_noise = filt_ops +
  sqrt(WHITE_NOISE_FRAC) * parm_ptr->rms_phs_err * (PI/180.0) * white_noise[1];

    *phase += dphi + phase_noise;
    if ( *phase > 2.0 * PI ) *phase -= 2.0 * PI;

    return( 1 );
}
```

G.2.2 adv_ts.c

```
#include <stdlib.h>
#include "setup.h"

int adv_ts(
double *samp_chip,
int *curr_chip,
struct sim_parm *parms
      )
{
    double samp_chip_new;
    static int first = TRUE;
    static double rms_jitter;

    // ALL TIMES ARE NORMALIZED TO THE SAMPLING RATE IN THESE COMPUTATION

    if ( first ) {
        first = FALSE;
        rms_jitter = parms->t_jitter * (double)parms->Ns * F_CHIP;
    }

    //
    // ADVANCE THE SAMPLING TIME, ENSURING THAT NEW SAMPLING TIME
    // IS SUBSEQUENT TO CURRENT SAMPLING TIME
    //
    samp_chip_new = -1.0;
    while ( samp_chip_new < *samp_chip ) {
       double jitter;
       double noise[2];
       gennoise( 1.0, noise );
       jitter = rms_jitter * noise[0];
       samp_chip_new = *samp_chip +
   parms->delta_t*(1.0-parms->samp_offset*1.0e-6) + jitter;
    }
    //
    // DETERMINE IF WE ARE SAMPLING A NEW CHIP,
    // IF SO, ADVANCE CHIP COUNTER
    //
    if ( samp_chip_new > 0.5 ) {
        samp_chip_new -= 1.0;
        *curr_chip = (++*curr_chip)%PNLENGTH;
    }
    *samp_chip = samp_chip_new;

    return( 1 );

}
```

G.2.3 agc.c

```
#include <stdlib.h>
#include <math.h>
#include "setup.h"

int agc(
         int q_sample,
         double *agc_scale,
         struct sim_parm *parms
       )
{
    static double target;
    static long int previous = 0;
    static int rs;
    static int ncorr = 0;
    static int first = TRUE;

    if ( first ) {
        double scalfact;
        first = FALSE;
        *agc_scale = 1.0;
        scalfact = pow(
                       pow( 2.0, (double)parms->numadbits ) /  ( parms->vlimhi - parms->vlimlo ),
                       2.0
                      );
        target = SIGMA_2_TARGET*scalfact;
        rs = (int) rint( log( (double)AGC_CORR ) * 1.4426950489 );
        return( 1 );
    }

    previous += q_sample * q_sample;
    ncorr = (++ncorr)%AGC_CORR;

    if ( ncorr == 0 ) {
      previous >>= rs;
      *agc_scale = (*agc_scale) * sqrt( target / (double)previous );
      previous = 0;
    }

    return( 1 );
}
```

G.2.4 cdma_sim.c

```c
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <malloc.h>
#include "is95puls.h"
#include "setup.h"
#include "info_mp.h"
#include "info_bs.h"

int main( int argc, char *argv[] )
{
    int i, j;
    int *Ipilot;
    int *Qpilot;
    double *Uintr[NUM_BASE_STATION];
    int Inumprevzero = 0;
    int Qnumprevzero = 0;
    double samp_chip_prev;        // PREVIOUS SAMPLING TIME ON THE CURRENT CHIP OF PREVIOUS CHIP
    double sample_tmp;            // TEMPORARY VARIABLE
    double sample;                // VALUE OF CURRENT SAMPLE (ALL COMPONENTS INCLUDED)
    int q_sample;                 // VALUE OF THE QUANTIZED SAMPLE
    int Ioutput1;                 // VALUE OF THE I-CHANNEL CORRELATION PERFORMED IN RECEIVER 1
    int Qoutput1;                 // VALUE OF THE Q-CHANNEL CORRELATION PERFORMED IN RECEIVER 1
    int Ioutput2;                 // VALUE OF THE I-CHANNEL CORRELATION PERFORMED IN RECEIVER 2
    int Qoutput2;                 // VALUE OF THE Q-CHANNEL CORRELATION PERFORMED IN RECEIVER 2
    int cycles;                   // NUMBER OF CYCLES OF THE PN SEQUENCE GENERATED
    int Iregs = (0x4000);         // INITIALIZE THE I CHANNEL SHIFT REGISTER TO THE ALL 1'S STATE
    int Qregs = (0x4000);         // INITIALIZE THE Q CHANNEL SHIFT REGISTER TO THE ALL 1'S STATE
    struct sim_parm parms;        // HOLDS PARAMETERS OF THE SIMULATION
    int curr_chip_mp[NUM_MP][NUM_BASE_STATION];
                                  // CURRENT MULTIPATH CHIP BEING SAMPLED
    double samp_chip_mp[NUM_MP][NUM_BASE_STATION];
                                  // LOCATION ON CURRENT MULTIPATH CHIP BEING SAMPLED
    double phase;                 // CURRENT CARRIER PHASE IF MAIN COMPONENT
    double phase_diff_mp[NUM_MP][NUM_BASE_STATION];
                                  // CARRIER PHASE DIFFS BETWEEN MAIN COMP AND MULTIPATH COMPS
    double tmp_phase;             // TEMP VARIABLE TO COMPUTE ACTUAL PHASES OF MULTIPATH COMPS
    double input_snr_abs;         // SNR AT THE A/D INPUT AS A RATIO
    double noise[2];              // VECTOR FOR NOISE SAMPLES
    double agc_scale;             // SCALE FACTOR IMPOSED BY AGC
    int numbins;                  // NUMBER OF QUANTIZATION BINS
    int numbinbound;              // NUMBER OF QUANTIZATION BIN BOUNDARIES
    double *bins;                 // SIGNAL QUANTIZATION BINS
    long idum;                    // MEMORY LOCATION FOR RANDOM NUMBER GENERATOR
    long int cnt_samples;         // COUNTS THE NUMBER OF SAMPLES GENERATED
    int nskip;                    // NUMBER OF SAMPLES TO UPDATE WHEN UPDATING RECEIVER PN CODE PHASE
double S;                     // SIMULATED SIGNAL POWER

    //
    // CHECK INPUT
    //
    chckinpt( &argc, argv, &parms );

    //
    // ALLOCATE HEAP MEMORY FOR THE PILOT SEQUENCES AND INTERFERENCE ARRAYS
    //
    Ipilot = (int *)malloc( PNLENGTH*sizeof(int) );
    if ( Ipilot == NULL ) {
        printf( "Could not allocate memory for I pilot PN sequence.\n" );
        exit( 1 );
    }
    Qpilot = (int *)malloc( PNLENGTH*sizeof(int) );
    if ( Qpilot == NULL ) {
        printf( "Could not allocate memory for Q pilot PN sequence.\n" );
        exit( 1 );
    }
    for ( i=0; i<NUM_BASE_STATION; i++ ) {
        Uintr[i] = (double *)malloc( PNLENGTH*sizeof(double) );
        if ( Uintr[i] == NULL ) {
            printf( "Could not allocate memory for interference sequence.\n" );
            exit( 1 );
        }
    }

    //
    // INITIALIZE THE PILOT PN SPREADING SEQUENCES AND RETAIN IN MEMORY
    // ALSO INITIALIZE THE INTERFERENCE SEQUENCES AND RETAIN IN MEMORY
    //
    for( i=0; i<PNLENGTH; i++ ) {
        Ipilot[i] = gensrop( Imask, &Inumprevzero, &Iregs );
        Qpilot[i] = gensrop( Qmask, &Qnumprevzero, &Qregs );
    }
    //
    // GENERATE THE MULTIUSER INTERFERENCE FROM EACH BASE STATION
    //
    for ( i=0; i<NUM_BASE_STATION; i++ ) genintr( Uintr[i], PNLENGTH );

    //
    // INITIALIZE THE CARRIER PHASE OF MAIN COMPONENT
    //
    phase = (CARR_INIT_PHASE / 180.0) * PI;
    //
    // MAKE CARRIER PHASES OF MULTIPATH COMPONENTS RANDOM
    // HERE, THE PHASE DIFFERENCES BETWEEN THE MAIN COMPONENT
    // AND THE MULTIPATH COMPONENTS ARE COMPUTED
    //
    if ( 1 ) {
        //
        // INITIALIZE THE RANDOM NUMBER GENERATOR
        //
        time_t t1;
        struct tm *tmptr;
        t1 = time(( time_t * ) 0);
        tmptr = localtime( &t1 );
        idum = -(tmptr->tm_mon + tmptr->tm_mday + tmptr->tm_hour +
                 tmptr->tm_min + tmptr->tm_sec);
        ran2( &idum );
    }

    //
    // INITIALIZE THE QUANTIZATION BINS
    //
    numbins = (int)pow( 2.0, (double)parms.numadbits );
    numbinbound = numbins - 1;
    bins = (double *)malloc( numbinbound*sizeof(double) );
    if ( bins == NULL ) {
        printf( "Array of quantization bins could not be allocated.\n" );
        exit( 1 );
    }
    initbins( bins, numbinbound, parms.vlimlo, parms.vlimhi );

    //
    // MAKE RANDOM PHASE SHIFTS OF CARRIER PHASE BETWEEN MULTIPATH COMPONENTS
    //
    for ( i=0; i<NUM_BASE_STATION; i++ ) {
        for ( j=0; j<NUM_MP; j++ ) {
            phase_diff_mp[j][i] = 2.0 * PI * ran2( &idum );
        }
    }
    phase_diff_mp[0][0] = 0.0;
    //
    // COMPUTE THE INPUT SNR AS A RATIO
    //
    input_snr_abs = pow(10.0,parms.input_snr_db/10.0);

    //
    // INITIALIZE THE MAIN COMPONENT COUNTERS
    //
    curr_chip_mp[0][0] = CHIP_0 + displace_bs[0];
    chck_bnd( &curr_chip_mp[0][0] );
    samp_chip_mp[0][0] = parms.displace;
    //
    // INITIALIZE THE MULTIPATH COMPONENT COUNTERS
    //
    for ( i=0; i<NUM_BASE_STATION; i++ ) {
        for ( j=0; j<NUM_MP; j++ ) {
            if ( !( i==0 & j==0 ) ) {
                curr_chip_mp[j][i] =
                    (int) rint( curr_chip_mp[0][0] + samp_chip_mp[0][0] + displace_bs[0]
                            - displace_mp[j] - displace_bs[i] );
                samp_chip_mp[j][i] =
                    curr_chip_mp[0][0] + samp_chip_mp[0][0] + displace_bs[0]
                  - displace_mp[j] - displace_bs[i] - curr_chip_mp[j][i];
                chck_bnd( &curr_chip_mp[j][i] );
            }
        }
    }

    //
    // INITIALIZE THE COUNT OF THE NUMBER OF PN CYCLES GENERATED
    //
    cycles = 0;
    cnt_samples = 0;

//
// COMPUTE THE SIGNAL POWER
//
S = 0.0;
for ( i=0; i<NUM_BASE_STATION; i++ ) {
  for ( j=0; j<NUM_MP; j++ ) {
S += atten_bs[i]*atten_bs[i]*atten_mp[j]*atten_mp[j];
  }
}

    //
    // INITIALIZE THE AGC SCALE FACTOR
    //
    agc_scale = 1.0;

    while ( cycles < NUM_CYCL_GEN ) {

        sample = 0.0;
        //
        // COMPUTE THE CURRENT SAMPLE
        //
        for ( i=0; i<NUM_BASE_STATION; i++ ) {
            for ( j=0; j<NUM_MP; j++ ) {
                tmp_phase = phase + phase_diff_mp[j][i];
                if ( tmp_phase > 2.0*PI ) tmp_phase -= 2.0*PI;
                gensamp( curr_chip_mp[j][i], samp_chip_mp[j][i], Ipilot, Qpilot, Uintr[i],
                         tmp_phase, &sample_tmp );
                sample += atten_bs[i] * atten_mp[j] * sample_tmp;
            }
        }
        //
// SCALE THE SAMPLE BY THE AGC FACTOR
//
        sample *= agc_scale;

        //
        // ADD IN GAUSSIAN NOISE
        //
        gennoise( S * pow(agc_scale,2.0) / input_snr_abs, noise );
        sample += noise[0];

        //
        // QUANTIZE THE SAMPLE
        //
        q_sample = quantize( sample, bins, numbinbound );

        //
        // APPLY AUTOMATIC GAIN CONTROL
        //
        agc( q_sample, &agc_scale, &parms );

        //
        // PASS THE SAMPLE TO THE RECEIVER
        //
        rcvr1fps( q_sample, Ipilot, Qpilot, &Ioutput1, &Qoutput1, &parms, FALSE, &nskip );
        rcvr2fps( q_sample, Ipilot, Qpilot, &Ioutput2, &Qoutput2, &parms, FALSE, &nskip );

        //
        // PASS THE SAMPLE TO TIMING RECOVERY
        //
        timercv2( q_sample, Ipilot, Qpilot, &parms, FALSE, &nskip );

        //
        // ADVANCE THE SAMPLING TIME ON MAIN AND MULTIPATH COMPONENTS
        //
        samp_chip_prev = samp_chip_mp[0][0];
        adv_ts( &samp_chip_mp[0][0], &curr_chip_mp[0][0], &parms );
        for ( i=0; i<NUM_BASE_STATION; i++ ) {
            for ( j=0; j<NUM_MP; j++ ) {
                if ( !( i==0 & j==0 ) ) {
                  curr_chip_mp[j][i] =
                    (int) rint( curr_chip_mp[0][0] + samp_chip_mp[0][0] + displace_bs[0]
                             - displace_mp[j] - displace_bs[i] );
                  samp_chip_mp[j][i] =
                    curr_chip_mp[0][0] + samp_chip_mp[0][0] + displace_bs[0]
                  - displace_mp[j] - displace_bs[i] - curr_chip_mp[j][i];
                  chck_bnd( &curr_chip_mp[j][i] );
                }
            }
        }

        //
        // ADVANCE THE CARRIER PHASE
        //
        adv_phs( F_IF*(1.0+parms.f_offset*1.0e-6), samp_chip_prev, samp_chip_mp[0][0],
                 &phase, &parms );

        //
        // ADVANCE COUNTER OF CYCLES
        //
        cnt_samples++;
        if ( PNLENGTH*parms.Ns == cnt_samples ) {
            cycles++;
            cnt_samples = 0;
            //
            // GENERATE NEW MULTIUSER INTERFERENCE FROM EACH BASE STATION
            // THIS IS NOT 100% ACCURATE SINCE OVERLAP SHOULD BE TAKEN INTO ACCOUNT
            //
            for ( i=0; i<NUM_BASE_STATION; i++ ) genintr( Uintr[i], PNLENGTH );
        }
    }

    //
    // FREE HEAP MEMORY
    //
    free( Ipilot );
    free( Qpilot );
    free( bins );
    for ( i=0; i<NUM_BASE_STATION; i++ ) free( Uintr[i] );
    //
    // CLOSE ANY OPEN FILES
    //
    rcvr1fps( 0, Ipilot, Qpilot, &Ioutput1, &Qoutput1, &parms, TRUE, &nskip );
    rcvr2fps( 0, Ipilot, Qpilot, &Ioutput2, &Qoutput2, &parms, TRUE, &nskip );

    return ( 1 );
}
```

G.2.5 chck_bnd.c

```
#include <stdlib.h>
#include "setup.h"

int chck_bnd( int *chip_ind )
{
//
// THIS ROUTINE CHECKS IF ITS INPUT HAS OVERSTEPPED THE LIMITS OF THE
// LENGTH OF THE PN SEQUENCE. IF SO, THE VALUE IS CORRECTED SO THAT IT
// FALLS WITHIN THESE LIMITS.
//
    if ( *chip_ind < 0 ) *chip_ind += PNLENGTH;
    else if ( *chip_ind >= PNLENGTH ) *chip_ind -= PNLENGTH;

    return( 1 );
}
```

G.2.6 chckinpt.c

```
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "setup.h"

int chckinpt( int *argc, char *argv[], struct sim_parm *parm_ptr)
{
    double doub_temp;
    int    int_temp;
    char   str_temp[32];

    if ( *argc != 14 ) {
        printf( "Usage: cdma_sim Ns displacement numadbits vlimlo vlimhi\n" );
        printf( "       numcorr snr(dB) rms_phs_err(deg) f_offset(ppm) samp_offset(ppm)\n" );
        printf( "       rms_jitter(ps) r1data r2data\n" );
        exit( 1 );
    }

    parm_ptr->delta_t = 1.0 / atof( argv[1] );

    doub_temp = atof( argv[2] );
    if ( fabs(doub_temp) >= 0.5  ) {
        printf( "Displacement value out of range.\n" );
        exit( 1 );
    }
    else
        parm_ptr->displace = doub_temp;

    int_temp = atoi( argv[3] );
    if ( int_temp < 1 ) {
        printf( "Illegal number of A/D bits.\n" );
        exit( 1 );
    }
    else
        parm_ptr->numadbits = int_temp;

    parm_ptr->vlimlo = atof( argv[4] );

    doub_temp = atof( argv[5] );
    if ( doub_temp <= parm_ptr->vlimlo ) {
        printf( "Entered vlimlo > vlimhi.\n" );
        exit( 1 );
    }
    else {
        parm_ptr->vlimhi = doub_temp;
    }

    parm_ptr->numcorr = atoi( argv[6] );

    parm_ptr->input_snr_db = atoi( argv[7] );

    if ( ( parm_ptr->rms_phs_err = atof( argv[8] ) ) < 0.0 ) {
      printf( "Entered rms phase error less than zero.\n" );
      exit( 1 );
    }

    parm_ptr->f_offset = atof( argv[9] );

    parm_ptr->Ns = atoi( argv[1] );

    parm_ptr->samp_offset = atof( argv[10] );

parm_ptr->t_jitter = atof( argv[11] );
if ( parm_ptr->t_jitter < 0.0 ) {
```

```
printf( "Sampling jitter out of range.\n" );
exit( 1 );
}
parm_ptr->t_jitter *= 1.0e-12;

    strcpy( str_temp, "ls " );
    strcat( str_temp, argv[12] );
    strcat( str_temp, ".dat > /dev/null" );
    if ( system( str_temp ) == 0 ) {
      printf( "%s exists. Exiting.\n", argv[12] );
      exit( 1 );
    }
    else {
        strcpy( parm_ptr->rcv1data, argv[12] );
    }

    strcpy( str_temp, "ls " );
    strcat( str_temp, argv[13] );
    strcat( str_temp, ".dat > /dev/null" );
    if ( system( str_temp ) == 0 ) {
      printf( "%s exists. Exiting.\n", argv[13] );
      exit( 1 );
    }
    else {
        strcpy( parm_ptr->rcv2data, argv[13] );
    }


    return( 1 );
}
```

G.2.7 filter.c

```
double fir_filter( double *coeff, double *input, int ind_oldest, int num_coeff )
{
//
//  THIS ROUTINE IS A FIR FILTER
//  THE COEFFICIENTS ARE ASSUMED TO BE FIXED,
//  AND THE DATA TO BE FILTERED ARE ASSUMED TO BE IN A CIRCULAR BUFFER
//
//  VARIABLE DEFINITIONS:
//    coeffs       ARRAY CONTAINING THE FIR FILTER COEFFICIENTS
//    input        ARRAY (CIRCULAR BUFFER) CONTAINING THE INPUT SAMPLES
//    ind_oldest   INDEX TO THE OLDEST INPUT SAMPLE
//    num_coeff    VARIABLE CONTAINING THE NUMBER OF FIR FILTER COEFFICIENTS
//
//  KEITH BLANKENSHIP
//  MAY 10, 1997
//

    int i;
    int ind_data = ind_oldest;
    double sum;

    for ( i=0, sum=0.0; i<num_coeff; i++ ) {
      sum += coeff[i] * input[ind_data];
      ind_data = (++ind_data)%num_coeff;
    }

    return( sum );
}
```

G.2.8 genintr.c

```
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <sys/ddi.h>
#include "setup.h"
#include "chanmaps.h"

int genintr( double *sig_ptr, int length )
{
//
// THIS ROUTINE GENERATES CDMA SIGNALS AS THE SUM OF SIGNALS WHICH ARE
// RANDOM DATA BITS SPREAD BY WALSH FUNCTIONS.
//
//
    int i;
    int WalshFcn[WFLENGTH][WFLENGTH];
    int WF_cnt;
    int paging_bit[ max(PAGING_IN_USE,1) ];
    int sync_bit[ max(SYNC_IN_USE,1) ];
    int traffic_bit[ max(TRAFFIC_IN_USE,1) ];
    long idum;

    static int first = TRUE;

    if ( first ) {
      //
      // INITIALIZE THE RANDOM NUMBER GENERATOR
      //
      time_t t1;
      struct tm *tmptr;
      t1 = time(( time_t * ) 0);
      tmptr = localtime( &t1 );
      idum = -(tmptr->tm_mon + tmptr->tm_mday + tmptr->tm_hour + tmptr->tm_min + tmptr->tm_sec);
      ran2( &idum );
      //
      // GENERATE THE WALSH FUNCTIONS
      //
      genwalsh( WalshFcn );
      first = FALSE;
    }
    WF_cnt = 0;
    for ( i=0; i<length; i++ ) {
        int j;
        sig_ptr[i] = 0.0;
        for ( j=0; j<PAGING_IN_USE; j++ ) {
            if ( WF_cnt == 0 ) paging_bit[j] = ran2(&idum) > 0.5 ? 1 : -1 ;
            sig_ptr[i] += sqrt(PAGING_CHANNEL_PWR) *
                          WalshFcn[paging_map[j]][WF_cnt] * paging_bit[j];
        }
        for ( j=0; j<SYNC_IN_USE; j++ ) {
            if ( WF_cnt == 0 ) sync_bit[j] = ran2(&idum) > 0.5 ? 1 : -1 ;
            sig_ptr[i] += sqrt(SYNC_CHANNEL_PWR) *
                          WalshFcn[sync_map[j]][WF_cnt] * sync_bit[j];
        }
        for ( j=0; j<TRAFFIC_IN_USE; j++ ) {
            if ( WF_cnt == 0 ) traffic_bit[j] = ran2(&idum) > 0.5 ? 1 : -1 ;
            sig_ptr[i] += sqrt(TRAFFIC_CHANNEL_PWR) *
                          WalshFcn[traffic_map[j]][WF_cnt] * traffic_bit[j];
        }
        WF_cnt = (++WF_cnt)%WFLENGTH;
    }
    return( 1 );
}
```

G.2.9 gennoise.c

```
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <sys/types.h>
#include "setup.h"

int gennoise( double var, double *noise )
{
//
// THIS SUBROUTINE GENERATES TWO GAUSSIAN DISTRIBUTED RANDOM NUMBERS
// EACH HAVING ZERO MEAN AND VARIANCE var
//
// THE VECTOR OF NOISE SAMPLES IS ALLOCATED IN THE CALLING ROUTINE
//
// ALGORITHM TAKEN FROM "COMMUNICATION SYSTEM DESIGN USING DSP ALGORITHMS"
// BY STEVEN A. TRETTER, ALTHOUGH THE GENERAL METHOD IS WELL-KNOWN
//
// KEITH BLANKENSHIP
// MAY 14, 1997
//
    double R;
    static long idum;
    static int first = TRUE;

    //
    // INITIALIZE THE RANDOM NUMBER GENERATOR WITH THE LOCAL TIME
    // IDEA FOR THIS FROM NEIYER CORREAL
    // CODE NUGGET FROM "C BY DISCOVERY" BY L.S. FOSTER
    //
    if ( first ) {
      //
      time_t t1;
      struct tm *tmptr;
      //
      t1 = time(( time_t * ) 0);
      tmptr = localtime( &t1 );
      idum = -(tmptr->tm_mon + tmptr->tm_mday + tmptr->tm_hour + tmptr->tm_min + tmptr->tm_sec);
      ran2( &idum );
      first = FALSE;
    }

    R = sqrt( -2.0*var*log( 1.0-ran2(&idum) ) );

    noise[0] = R*cos( 2.0*M_PI*ran2(&idum) );
    noise[1] = R*sin( 2.0*M_PI*ran2(&idum) );

    return( 1 );
}
```

G.2.10 gensamp.c

```
#include <stdlib.h>
#include <math.h>
#include "setup.h"
#include "is95puls.h"

int gensamp( int curr_chip, double samp_chip, int Ipilot[], int Qpilot[],
             double Uintr[], double phase, double *sample )
{
    int i;
    double pulsval;        // VALUE OF THE PULSE AT SAMPLING INSTANT
    int chip_ind;          // INDEX OF CHIPS OF COMPUTING ISI
    double Isample;
    double Qsample;

    //
    // DETERMINE THE BASEBAND SAMPLE
    //
    Isample = 0.0;
    Qsample = 0.0;
    for ( i=ISIPREVCHIP; i>=-ISIFUTRCHIP; i-- ) {
        //
        // DETERMINE THE MAGNITUDE OF THE PULSE AT THE SAMPLING INSTANT
        // REMEMBER THAT BOTH PREVIOUS AND FUTURE PULSES ARE BEING SAMPLED
        //
        splint( is95puls_t, is95puls_h, is95puls_d, NUMBBFILTCOEFF,
                samp_chip+(double)i, &pulsval );
        //
        // DETERMINE THE CHIP NUMBER OF THE CHIP BEING SAMPLED
        //
        chip_ind = curr_chip - i;
        chck_bnd( &chip_ind );
        //
        // ACCUMULATE CONTRIBUTIONS OF PREVIOUS, CURRENT, AND FUTURE
        // PULSES TO THE CURRENT SAMPLE
        //
        Isample += Ipilot[chip_ind] *
                   ( sqrt(PILOT_CHANNEL_PWR) + Uintr[chip_ind] ) * pulsval;
        Qsample += Qpilot[chip_ind] *
                   ( sqrt(PILOT_CHANNEL_PWR) + Uintr[chip_ind] ) * pulsval;
    }

    //
    // USE BASEBAND SAMPLES TO MODULATE CARRIER
    //
    Isample = Isample * cos( phase );
    Qsample = Qsample * sin( phase );

    *sample = Isample + Qsample;

    return( 1 );
}
```

G.2.11 gensrop.c

```
#include <stdlib.h>
#include "setup.h"

int gensrop( int mask, int *numprevzero, int *regs )
{
//
// GENERATE SHIFT REGISTER OUTPUT GIVEN A MASK,
// THE NUMBER OF PREVIOUS ZERO OUTPUTS,
// AND A SET OF SHIFT REGISTERS
//
// KEITH BLANKENSHIP
// JULY 16, 1997
//
    int i;                                  // LOOP INDEX
    int op = 0;                             // FOR CALCULATING THE OUTPUT
    int n = mask & *regs;                   // EXTRACT THE CURRENT STATE OF THE SHIFT REGISTERS

    if ( *numprevzero == NTAP-1 ) {         // IF THE RIGHT NUMBER OF PREVIOUS OUTPUTS HAVE BEEN 0
      (*numprevzero)++;
      return( 1 );                          // THEN RETURN 0 (WHICH IS CONVERTED TO A 1)
    }
    else {                                  // OTHERWISE, CALCULATE THE NORMAL M-SEQUENCE TYPE OUTPUT
      for ( i=0; i<NTAP; i++  ) {           // EXAMINE THE CONTENTS OF THE SHIFT REGISTERS
        op ^= (1 & n);                      // EXCLUSIVE OR OF op WITH RIGHTMOST ELEMENT OF n
        n >>= 1;                            // RIGHT SHIFT n BY ONE BIT
      }
      if ( !op ) (*numprevzero)++;          // IF THE OUTPUT IS 0, INCREMENT COUNTER
      else (*numprevzero) = 0;              // OTHERWISE, RESET COUNTER
      *regs <<= 1;                          // LEFT SHIFT THE REGISTERS BY ONE
      if ( op ) (*regs)++;                  // IF THE OUTPUT IS 1, INTRODUCE A 1 IN RIGHTMOST REGISTER
      return( !op ? 1 : -1 );               // WHEN RETURNING, CONVERT 0 ==> 1, 1 ==> -1
    }
}
```

G.2.12 genwalsh.c

```
#include <stdlib.h>
#include <math.h>
#include "setup.h"

int genwalsh( int WalshFcn[WFLENGTH][WFLENGTH] )
{
//
// THIS SUBROUTINE GENERATES THE WALSH FUNCTIONS
//
// KEITH BLANKENSHIP
// MAY 12, 1997
//
    int i, j, k, l;

    WalshFcn[0][0] = 0;
    WalshFcn[0][1] = 0;
    WalshFcn[1][0] = 0;
    WalshFcn[1][1] = 1;

    //
    // MAKE PROVISIONS FOR RUNNING THE PROGRAM ON MY PC AT HOME
    //
    if ( WFLENGTH == 64 ) l = 6;
    else if ( WFLENGTH == 16 ) l = 4;
    else {
        printf( "Improper generation of the Walsh functions.\n" );
        exit( 1 );
    }

    //
    // THIS PART OF THE CODE GENERATES THE WALSH FUNCTIONS
    //
    for ( i=1; i<=6; i++ )
      for ( j=0; j<pow(2,i-1); j++ )
        for ( k=0; k<pow(2,i-1); k++ ) {
          WalshFcn[j                ][k+(int)pow(2,i-1)] =  WalshFcn[j][k];
          WalshFcn[j+(int)pow(2,i-1)][k                ] =  WalshFcn[j][k];
          WalshFcn[j+(int)pow(2,i-1)][k+(int)pow(2,i-1)] = !WalshFcn[j][k];
        }

    //
    // CONVERT 1 ==> -1, 0 ==> 1
    //
    for ( i=0; i<WFLENGTH; i++ )
      for ( j=0; j<WFLENGTH; j++ )
        if ( !WalshFcn[i][j] )
          WalshFcn[i][j] = 1;
        else
          WalshFcn[i][j] = -1;

    return( 1 );
}
```

G.2.13 prntdata.c

```
#include <stdlib.h>
#include "setup.h"

extern double displace_mp[NUM_MP];
extern double atten_mp[NUM_MP];
extern double displace_bs[NUM_MP];
extern double atten_bs[NUM_MP];

int prntdata( FILE *fp, struct sim_parm *parm_ptr )
{
    int i;

    fprintf( fp, "IF (Hz): %f\n", F_IF );
    fprintf( fp, "NumCorr: %d\n", parm_ptr->numcorr );
    fprintf( fp, "Number of samples/chip: %d\n", parm_ptr->Ns );
    fprintf( fp, "Delta_t (chip rate units): %f\n", parm_ptr->delta_t );
    fprintf( fp, "Initial displacement (chip rate units): %f\n", parm_ptr->displace );
    fprintf( fp, "A/D Bits: %d\n", parm_ptr->numadbits );
fprintf( fp, "[ Vlimlo, Vlimhi ] = [ %f, %f ]\n", parm_ptr->vlimlo, parm_ptr->vlimhi );
fprintf( fp, "Target signal variance: %f\n", SIGMA_2_TARGET );
fprintf( fp, "# bits rcvr pilot signal quantization rcvr 1: %d\n", NUM_BITS_PILOT_RCVR_1 );
fprintf( fp, "# bits rcvr pilot signal quantization rcvr 2: %d\n", NUM_BITS_PILOT_RCVR_2 );
    fprintf( fp, "Input SNR (dB): %f\n", parm_ptr->input_snr_db );
    fprintf( fp, "Initial Phase (degrees): %f\n", CARR_INIT_PHASE );
    fprintf( fp, "Local oscillator RMS phase error (degrees): %f\n", parm_ptr->rms_phs_err );
fprintf( fp, "RMS sampling jitter (ps): %f\n", parm_ptr->t_jitter/1.0e-12 );
    fprintf( fp, "LO frequency offset (ppm): %f\n", parm_ptr->f_offset );
    fprintf( fp, "Sampling frequency offset (ppm): %f\n", parm_ptr->samp_offset );
    fprintf( fp, "Pilot channel fractional power: %f\n", PILOT_PWR_FRAC );
    fprintf( fp, "Paging channels fractional power: %f\n", PAGING_PWR_FRAC );
    fprintf( fp, "Sync channel fractional power: %f\n", SYNC_PWR_FRAC );
    fprintf( fp, "Traffic channels fractional power: %f\n", TRAFFIC_PWR_FRAC );
    fprintf( fp, "Number paging channels in use: %d\n", PAGING_IN_USE );
    fprintf( fp, "Number traffic channels in use: %d\n", TRAFFIC_IN_USE );
    fprintf( fp, "Multipath components:\n" );
    fprintf( fp, "    disp   ampl\n" );
    fprintf( fp, "    ====   ====\n" );
    for ( i=0; i<NUM_MP; i++ )
        fprintf( fp, "    %4.2f   %4.2f\n", displace_mp[i], atten_mp[i]  );
    fprintf( fp, "Base stations:\n" );
    fprintf( fp, "    disp   ampl\n" );
    fprintf( fp, "    ====   ====\n" );
    for ( i=0; i<NUM_BASE_STATION; i++ )
        fprintf( fp, "    %4.2f   %4.2f\n", displace_bs[i], atten_bs[i]  );

fflush( fp );

    return( 1 );
}
```

G.2.14 quantize.c

```
#include <stdlib.h>
#include <malloc.h>
#include <math.h>
#include "setup.h"

int quantize(
              double sample,
              double *bins,
              int numbinbound
            )
{
//
// THIS ROUTINE QUANTIZES ITS INPUT TO THE CLOSEST QUANTIZATION LEVEL
// BETWEEN vlimlo AND vlimhi, INCLUSIVE. THE NUMBER OF QUANTIZATION BINS
// IS GIVEN BY 2^numadbits. A/D NONLINEARITIES SHOW UP IN THE WIDTH AND
// PLACEMENT OF THE QUANTIZATION BINS.
// THE OUTPUT IS AN INTEGER BETWEEN -2^(numadbits-1) AND 2^(numadbits-1)-1.
//
    int klo, khi, k;
    int offset;

    offset = (numbinbound+1) / 2;

    if ( sample < bins[0] ) return( -offset );
    else if ( sample > bins[numbinbound-1] ) return( offset-1 );
    //
    // THIS NUGGET OF CODE PERFORMS A SEARCH-BY-BISECTION TO FIND THE
    // QUANTIZATION BIN IN WHICH THE SAMPLE LIES.
    //
    klo = 0;
    khi = numbinbound-1;
    while ( khi-klo > 1 ) {
        k = (khi+klo) >> 1;
        if ( bins[k] > sample ) khi=k;
        else klo = k;
    }
    return( khi-offset );
}



int initbins(
              double *bins,
              int numbinbound,
              double vlimlo,
              double vlimhi
             )
{
//
// THIS ROUTINE INITIALIZES THE BINS OF THE A/D CONVERSION PROCESS.
//
    int i;
    double vstep;

    vstep = ( vlimhi - vlimlo ) / (double)(numbinbound+1);

    for ( i=0; i<=numbinbound-1; i++ ) {
        bins[i] = vlimlo + vstep/2.0 + (double)i*vstep;
    }

    return( 1 );

}
```

G.2.15 ran2.c

```
#define IM1 (2147483563)
#define IM2 (2147483399)
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 (40014)
#define IA2 (40692)
#define IQ1 (53668)
#define IQ2 (52774)
#define IR1 (12211)
#define IR2 (3791)
#define NTAB (32)
#define NDIV (1+IMM1/NTAB)
#define EPS (1.2e-7)
#define RNMX (1.0-EPS)

double ran2( long *idum )
{
//
// "BEST" RANDOM NUMBER GENERATOR TAKEN FROM "NUMERICAL RECIPES IN C"
// EXCEPT CHANGED ALL THE FLOATS TO DOUBLE
//
//
// KEITH BLANKENSHIP
// MAY 17, 1997
//
    int j;
    long k;
    static long idum2=123456789;
    static long iy=0;
    static long iv[NTAB];
    double temp;

    if ( *idum <= 0 ) {
      if ( -(*idum) < 1 ) *idum=1;
      else *idum = -(*idum);
      idum2 = (*idum);
      for ( j=NTAB+7; j>=0; j-- ) {
        k = (*idum)/IQ1;
        *idum = IA1*(*idum-k*IQ1) - k*IR1;
        if ( *idum < 0 ) *idum += IM1;
        if ( j < NTAB ) iv[j] = *idum;
      }
      iy = iv[0];
    }
    k = (*idum)/IQ1;
    *idum = IA1*(*idum-k*IQ1)-k*IR1;
    if (*idum < 0) *idum += IM1;
    k = idum2/IQ2;
    idum2 = IA2*(idum2-k*IQ2)-k*IR2;
    if (idum2 < 0) idum2 += IM2;
    j = (int)(iy/NDIV);
    iy = iv[j] - idum2;
    iv[j] = *idum;
    if ( iy < 1 ) iy += IMM1;
    if ( (temp=AM*iy) > RNMX ) return( RNMX );
    else return( temp );
}
```

G.2.16 rcvr1fps.c

```
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "is95puls.h"
#include "setup.h"

int rcvr1fps (
          int sample,
          int Ipilot[],
          int Qpilot[],
          int *Ioutput,
          int *Qoutput,
          struct sim_parm *parm_ptr,
          int last,
  int *nskip
             )
{
//
// IN THE REAL RECEIVER, THE MATCHED FILTER COMPONENTS WOULD BE PRECOMPUTED AND STORED IN MEMORY
// WE CAN'T DO THAT HERE, BECAUSE COMPILERS DON'T ALLOW THE USE OF SO MUCH MEMORY
//
    static double samp_chip;
    static double samp_chip_prev;
    static int curr_chip = 0;          // RCVR BEGINS CORRELATING WITH CHIP NUMBER 0
    static int IIcorr = 0.0;
    static int IQcorr = 0.0;
    static int QIcorr = 0.0;
    static int QQcorr = 0.0;
    static int samp_corr = 0;          // NUMBER OF SAMPLES CURRENTLY CORRELATED OVER
    static int first = TRUE;
    static FILE *fp_data;
    static FILE *fp_parm;
    static int carr_ptr = 0;
    // static int numbins;
    // static int numbinbound;
    static double *Ibins;
    static double *Qbins;

    int i;
    int chip_ind;
    double pulsval;
    int q_pulsval;
    int q_Ichipval;
    int q_Qchipval;
    int bfact;
    int dfact;


    if ( last ) {
        free( Ibins );
        free( Qbins );
        fclose( fp_data );
        return( 1 );
    }

    if ( first ) {
        //
        // OPEN FILE FOR PRINTING RECEIVER DATA
        //
        char datafile[13];
        char parmfile[13];
        first = FALSE;                              // NEVER DO THIS PART OF THE CODE AGAIN
        samp_chip = -0.5 + (parm_ptr->delta_t)/2.0; // RECEIVER SAMPLES CHIPS AT IDEAL LOCATIONS
        strcpy( datafile, parm_ptr->rcv1data );
        strcat( datafile, ".dat\0" );
        strcpy( parmfile, parm_ptr->rcv1data );
        strcat( parmfile, ".prm\0" );
        if ( (fp_data = fopen( datafile, "w" )) == NULL ) {
            printf( "Simulation data file could not be opened.\n" );
            exit( 1 );
        }
        if ( (fp_parm = fopen( parmfile, "w" )) == NULL ) {
            printf( "Simulation parameters file could not be opened.\n" );
            exit( 1 );
        }
        //
        // WRITE DESCRIPTIVE DATA TO FILE
        //
        fprintf( fp_parm, "Receiver 1 Data:\n" );
        prntdata( fp_parm, parm_ptr );
        fclose( fp_parm );
    }

    q_Ichipval = 0;
    q_Qchipval = 0;
    for ( i=ISIPREVCHIP; i>=-ISIFUTRCHIP; i-- ) {
        //
        // DETERMINE THE MAGNITUDE OF THE PULSE AT THE SAMPLING INSTANT
        // REMEMBER THAT BOTH PREVIOUS AND FUTURE PULSES ARE BEING SAMPLED
        //
        splint( is95puls_t, is95puls_h, is95puls_d, NUMBBFILTCOEFF,
                    samp_chip+(double)i, &pulsval );
        //
        // QUANTIZE COEFFICIENTS TO 16 BITS
        //
        q_pulsval = rint( BBFILTCOEFFSCALE * pulsval );
        //
        // DETERMINE THE CHIP NUMBER OF THE CHIP BEING SAMPLED
        //
        chip_ind = curr_chip - i;
        chck_bnd( &chip_ind );
        //
        // ACCUMULATE CONTRIBUTIONS OF PREVIOUS, CURRENT, AND FUTURE
        // PULSES TO THE CURRENT SAMPLE
        //
        q_Ichipval += Ipilot[chip_ind] * q_pulsval;
        q_Qchipval += Qpilot[chip_ind] * q_pulsval;
    }

    //
    // RIGHT SHIFT DOWN TO THE SPECIFIED NUMBER OF BITS
    //
    q_Ichipval >>= 16-NUM_BITS_PILOT_RCVR_1;
    q_Qchipval >>= 16-NUM_BITS_PILOT_RCVR_1;

    //
    // FACTORS DETERMINING CARRIER PHASE
    //
    carr_ptr = (++carr_ptr)%(4);
    if ( carr_ptr == 0 ) {
        bfact = 1;
        dfact = 1;
    }
    else if ( carr_ptr == 1 ) {
        bfact = 0;
        dfact = 1;
    }
    else if ( carr_ptr == 2 ) {
        bfact = 1;
        dfact = -1;
    }
    else if ( carr_ptr == 3 ) {
        bfact = 0;
        dfact = -1;
    }

    //
    // COMPUTE THE DOWNCONVERSION AND CORRELATION
    //
    IIcorr +=  bfact * dfact * q_Ichipval * sample;
    IQcorr += !bfact * dfact * q_Ichipval * sample;
    QIcorr +=  bfact * dfact * q_Qchipval * sample;
    QQcorr += !bfact * dfact * q_Qchipval * sample;
    samp_corr++;

    if ( samp_corr == parm_ptr->numcorr ) {
        //
        // COMPUTE THE OUTPUT
        //
        *Ioutput = IIcorr*IIcorr + IQcorr*IQcorr;
        *Qoutput = QIcorr*QIcorr + QQcorr*QQcorr;
        //
        // PRINT THE RESULTS
        //
        fprintf( fp_data, "%10.4e  %10.4e\n",
                 (double)IIcorr*(double)IIcorr + (double)IQcorr*(double)IQcorr,
                 (double)QIcorr*(double)QIcorr + (double)QQcorr*(double)QQcorr );
fflush( NULL );
        //
        // REINITIALIZE THE CORRELATORS
        //
        IIcorr = 0;
        IQcorr = 0;
        QIcorr = 0;
        QQcorr = 0;
        samp_corr = 0;
        carr_ptr = 0;
        //
        // CHANGE THE DELAY BETWEEN RECEIVED AND RECEIVER'S PN CODES
        //
        samp_chip += (double)(*nskip+1)*parm_ptr->delta_t;
        if ( samp_chip > 0.5 ) {
            samp_chip -= 1.0;
            curr_chip = (++curr_chip)%PNLENGTH;
        }
    }
    else {
        //
        // ADVANCE THE SAMPLING TIME
        //
        samp_chip_prev = samp_chip;
        samp_chip += parm_ptr->delta_t;

        //
        // DETERMINE IF WE ARE SAMPLING A NEW CHIP,
        // IF SO, ADVANCE CHIP COUNTER
        //
        if ( samp_chip > 0.5 ) {
            samp_chip -= 1.0;
            curr_chip = (++curr_chip)%PNLENGTH;
        }
    }
    return( 1 );
}
```

G.2.17 rcvr2fps.c

```
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "is95puls.h"
#include "setup.h"

int rcvr2fps (
          int sample,
          int Ipilot[],
          int Qpilot[],
          int *Ioutput,
          int *Qoutput,
          struct sim_parm *parm_ptr,
          int last,
  int *nskip
             )
{
//
// IN THE REAL RECEIVER, THE MATCHED FILTER COMPONENTS WOULD BE PRECOMPUTED AND STORED IN MEMORY
// WE CAN'T DO THAT HERE, BECAUSE COMPILERS DON'T ALLOW THE USE OF SO MUCH MEMORY
//
// NOTE: THE SIMULATION IS ARRANGED SO THAT THERE IS ALWAYS A COMPONENT AT ZERO DELAY
//
    static double samp_chip;
    static double samp_chip_prev;
    static int curr_chip = 0;
    static int Icorr = 0;
    static int Qcorr = 0;
    static int samp_corr = 0;          // NUMBER OF SAMPLES CURRENTLY CORRELATED OVER
    static int first = TRUE;
    static FILE *fp_data;
    static FILE *fp_parm;
    static int I_ips[3] = {0,0,0};
    static int I_ops[3] = {0,0,0};
    static int I_curr = 0;
    static int Q_ips[3] = {0,0,0};
    static int Q_ops[3] = {0,0,0};
    static int Q_curr = 0;
    static int B = 5;
    static int A = 16374;
    static int shift = 14;

    int i;
    int chip_ind;
    double pulsval;
    int Isample;
    int Qsample;
    int q_pulsval;
    int q_Ichipval;
    int q_Qchipval;

    if ( last ) {
        fclose( fp_data );
        fclose( fp_parm );
        return( 1 );
    }

    if ( first ) {
        char datafile[13];
        char parmfile[13];
        first = FALSE;
        samp_chip = -0.5 + (parm_ptr->delta_t)/2.0;
        strcpy( datafile, parm_ptr->rcv2data );
        strcat( datafile, ".dat\0" );
        strcpy( parmfile, parm_ptr->rcv2data );
        strcat( parmfile, ".prm\0" );
        if ( (fp_data = fopen( datafile, "w" )) == NULL ) {
            printf( "Simulation data file could not be opened.\n" );
            exit( 1 );
        }
        if ( (fp_parm = fopen( parmfile, "w" )) == NULL ) {
            printf( "Simulation parameters file could not be opened.\n" );
            exit( 1 );
        }
        fprintf( fp_parm, "Receiver 2 Data:\n" );
        prntdata( fp_parm, parm_ptr );
        fclose( fp_parm );
    }

    q_Ichipval = 0;
    q_Qchipval = 0;
    for ( i=ISIPREVCHIP; i>=-ISIFUTRCHIP; i-- ) {
        //
        // DETERMINE THE MAGNITUDE OF THE PULSE AT THE SAMPLING INSTANT
        // REMEMBER THAT BOTH PREVIOUS AND FUTURE PULSES ARE BEING SAMPLED
        //
        splint( is95puls_t, is95puls_h, is95puls_d, NUMBBFILTCOEFF,
                     samp_chip+(double)i, &pulsval );
        //
        // QUANTIZE COEFFICIENTS TO 16 BITS
        //
        q_pulsval = rint( BBFILTCOEFFSCALE * pulsval );
        //
        // DETERMINE THE CHIP NUMBER OF THE CHIP BEING SAMPLED
        //
        chip_ind = curr_chip - i;
        chck_bnd( &chip_ind );
        //
        // ACCUMULATE CONTRIBUTIONS OF PREVIOUS, CURRENT, AND FUTURE
        // PULSES TO THE CURRENT SAMPLE
        //
        q_Ichipval += Ipilot[chip_ind] * q_pulsval;
        q_Qchipval += Qpilot[chip_ind] * q_pulsval;
    }
    //
    // RIGHT SHIFT DOWN TO THE SPECIFIED NUMBER OF BITS
    //
    q_Ichipval >>= 16-NUM_BITS_PILOT_RCVR_2;
    q_Qchipval >>= 16-NUM_BITS_PILOT_RCVR_2;
    //
    // MULTIPLY BY PILOT SEQUENCES
    //
    Isample = sample * q_Ichipval;
    Qsample = sample * q_Qchipval;
    //
    // FILTER THE SIGNAL TO EXTRACT THE TONE, IF PRESENT.
    // THE FOLLOWING LINES OF CODE IMPLEMENT AN IIR FILTER
    //   USING TWO FIR FILTERS.
    //
    I_ips[I_curr] = Isample;
    I_ops[I_curr] = B * ( I_ips[I_curr] - I_ips[(I_curr+1)%3] ) - A * I_ops[(I_curr+1)%3];
    I_ops[I_curr] = I_ops[I_curr] >> shift;
    I_curr = (++I_curr)%3;
    Q_ips[Q_curr] = Qsample;
    Q_ops[Q_curr] = B * ( Q_ips[Q_curr] - Q_ips[(Q_curr+1)%3] ) - A * Q_ops[(Q_curr+1)%3];
    Q_ops[Q_curr] = Q_ops[Q_curr] >> shift;
    Q_curr = (++Q_curr)%3;
    //
    // SQUARE AND INTEGRATE
    //
    Icorr += I_ops[I_curr] * I_ops[I_curr];
    Qcorr += Q_ops[Q_curr] * Q_ops[Q_curr];
    samp_corr++;
    //
    // COMPUTE THE OUTPUT
    //
    *Ioutput = Icorr;
    *Qoutput = Qcorr;

    if ( samp_corr == parm_ptr->numcorr ) {
        //
        // PRINT THE RESULTS
        //
        fprintf( fp_data, "%d  %d\n", *Ioutput, *Qoutput);
        fflush( fp_data );
        //
        // REINITIALIZE THE CORRELATORS
        //
        Icorr = 0;
        Qcorr = 0;
        samp_corr = 0;
        //
        // REINITIALIZE THE CIRCULAR BUFFERS
        //
        for ( i=0; i<3; i++ ) {
          I_ips[i] = 0;
          Q_ips[i] = 0;
          I_ops[i] = 0;
          Q_ops[i] = 0;
        }
        //
        // REINITIALIZE THE CIRCULAR BUFFER POINTERS
        //
        I_curr = 0;
        Q_curr = 0;
        //
        // CHANGE THE DELAY BETWEEN RECEIVED AND RECEIVER'S PN CODES
        //
        samp_chip += (double)(*nskip+1)*parm_ptr->delta_t;
        if ( samp_chip > 0.5 ) {
            samp_chip -= 1.0;
            curr_chip = (++curr_chip)%PNLENGTH;
        }
    }
    else {
        //
        // ADVANCE THE SAMPLING TIME
        //
        samp_chip_prev = samp_chip;
        samp_chip += parm_ptr->delta_t;

        //
        // DETERMINE IF WE ARE SAMPLING A NEW CHIP,
        // IF SO, ADVANCE CHIP COUNTER
        //
        if ( samp_chip > 0.5 ) {
            samp_chip -= 1.0;
            curr_chip = (++curr_chip)%PNLENGTH;
        }
    }
    return( 1 );
}
```

G.2.18 splint.c

```
#include <stdlib.h>
#include <stdio.h>

void splint( double xa[], double ya[], double y2a[], int n, double x, double *y )
{
//
// SUBROUTINE FOR SPLINE INTERPOLATION.
// FROM NUMERICAL RECIPES IN C.
//
    int klo, khi, k;
    double h, b, a;

    if ( x < xa[0] ) {
      *y = 0.0;
      return;
    }
    else if ( x > xa[n-1] ) {
      *y = 0.0;
      return;
    }

    klo = 0;
    khi = n-1;

    while ( khi-klo > 1) {
        k = (khi+klo) >> 1;
        if (xa[k] > x) khi=k;
        else klo = k;
    }

    h = xa[khi] - xa[klo];
    if ( h == 0.0 ) {
        printf( "Bad xa input to splint routine.\n" );
        exit( 1 );
    }

    a = (xa[khi]-x) / h;
    b = (x-xa[klo]) / h;

    *y = a*ya[klo] + b*ya[khi] + ((a*a*a-a)*y2a[klo] + (b*b*b-b)*y2a[khi])*(h*h) / 6.0;
}
```

G.2.19 timercv2.c

```
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "is95puls.h"
#include "setup.h"

extern double displace_bs[NUM_BASE_STATION];

int timercv2 (
          int sample,
          int Ipilot[],
          int Qpilot[],
          struct sim_parm *parm_ptr,
          int last,
          int *nskip
             )
{
    static double samp_chip_hi;
    static int curr_chip_hi;
    static double samp_chip_lo;
    static int curr_chip_lo;
    static int IIcorr_hi = 0;
    static int IQcorr_hi = 0;
    static int QIcorr_hi = 0;
    static int QQcorr_hi = 0;
    static int IIcorr_lo = 0;
    static int IQcorr_lo = 0;
    static int QIcorr_lo = 0;
    static int QQcorr_lo = 0;
    static int samp_corr = 0;          // NUMBER OF SAMPLES CURRENTLY CORRELATED OVER
    static int first = TRUE;
    static int carr_ptr = 0;
    static int sIoutput_hi = 0;
    static int sQoutput_hi = 0;
    static int sIoutput_lo = 0;
    static int sQoutput_lo = 0;
    static int slope_sIoutput = 0;
    static int slope_sQoutput = 0;
    static int nsum = 0;
    static int tdfact = 0;
    static double *Ibins;
    static double *Qbins;

    int i;
    int chip_ind;
    double pulsval;
    int q_pulsval;
    int q_Ichipval_hi;
    int q_Qchipval_hi;
    int q_Ichipval_lo;
    int q_Qchipval_lo;
    int Ioutput_lo;
    int Qoutput_lo;
    int Ioutput_hi;
    int Qoutput_hi;
    int bfact;
    int dfact;


    if ( last ) {
        free( Ibins );
        free( Qbins );
        return( 1 );
    }
```

```
if ( first ) {
    first = FALSE;
    curr_chip_hi = CHIP_0 + displace_bs[0];
    samp_chip_hi = -0.375 + 2.0*parm_ptr->delta_t;
    curr_chip_lo = CHIP_0 + displace_bs[0];
    samp_chip_lo = -0.375 + 0.0*parm_ptr->delta_t;
}

q_Ichipval_hi = 0;
q_Qchipval_hi = 0;
for ( i=ISIPREVCHIP; i>=-ISIFUTRCHIP; i-- ) {
    //
    // DETERMINE THE MAGNITUDE OF THE PULSE AT THE SAMPLING INSTANT
    // REMEMBER THAT BOTH PREVIOUS AND FUTURE PULSES ARE BEING SAMPLED
    //
    splint( is95puls_t, is95puls_h, is95puls_d, NUMBBFILTCOEFF,
                samp_chip_hi+(double)i, &pulsval );
    //
    // QUANTIZE COEFFICIENTS TO 16 BITS
    //
    q_pulsval = rint( BBFILTCOEFFSCALE * pulsval );
    //
    // DETERMINE THE CHIP NUMBER OF THE CHIP BEING SAMPLED
    //
    chip_ind = curr_chip_hi - i;
    chck_bnd( &chip_ind );
    //
    // ACCUMULATE CONTRIBUTIONS OF PREVIOUS, CURRENT, AND FUTURE
    // PULSES TO THE CURRENT SAMPLE
    //
    q_Ichipval_hi += Ipilot[chip_ind] * q_pulsval;
    q_Qchipval_hi += Qpilot[chip_ind] * q_pulsval;
}

//
// RIGHT SHIFT DOWN TO THE SPECIFIED NUMBER OF BITS
//
q_Ichipval_hi >>= 16-NUM_BITS_PILOT_RCVR_1;
q_Qchipval_hi >>= 16-NUM_BITS_PILOT_RCVR_1;

q_Ichipval_lo = 0;
q_Qchipval_lo = 0;
for ( i=ISIPREVCHIP; i>=-ISIFUTRCHIP; i-- ) {
    //
    // DETERMINE THE MAGNITUDE OF THE PULSE AT THE SAMPLING INSTANT
    // REMEMBER THAT BOTH PREVIOUS AND FUTURE PULSES ARE BEING SAMPLED
    //
    splint( is95puls_t, is95puls_h, is95puls_d, NUMBBFILTCOEFF,
                samp_chip_lo+(double)i, &pulsval );
    //
    // QUANTIZE COEFFICIENTS TO 16 BITS
    //
    q_pulsval = rint( BBFILTCOEFFSCALE * pulsval );
    //
    // DETERMINE THE CHIP NUMBER OF THE CHIP BEING SAMPLED
    //
    chip_ind = curr_chip_lo - i;
    chck_bnd( &chip_ind );
    //
    // ACCUMULATE CONTRIBUTIONS OF PREVIOUS, CURRENT, AND FUTURE
    // PULSES TO THE CURRENT SAMPLE
    //
    q_Ichipval_lo += Ipilot[chip_ind] * q_pulsval;
    q_Qchipval_lo += Qpilot[chip_ind] * q_pulsval;
}
```

```
//
// RIGHT SHIFT DOWN TO THE SPECIFIED NUMBER OF BITS
//
q_Ichipval_lo >>= 16-NUM_BITS_PILOT_RCVR_1;
q_Qchipval_lo >>= 16-NUM_BITS_PILOT_RCVR_1;

//
// FACTORS DETERMINING CARRIER PHASE
//
carr_ptr = (++carr_ptr)%(4);
if ( carr_ptr == 0 ) {
    bfact = 1;
    dfact = 1;
}
else if ( carr_ptr == 1 ) {
    bfact = 0;
    dfact = 1;
}
else if ( carr_ptr == 2 ) {
    bfact = 1;
    dfact = -1;
}
else if ( carr_ptr == 3 ) {
    bfact = 0;
    dfact = -1;
}

//
// COMPUTE THE DOWNCONVERSION AND CORRELATION
//
IIcorr_hi +=  bfact * dfact * q_Ichipval_hi * sample *  tdfact;
IQcorr_hi += !bfact * dfact * q_Ichipval_hi * sample *  tdfact;
QIcorr_hi +=  bfact * dfact * q_Qchipval_hi * sample *  tdfact;
QQcorr_hi += !bfact * dfact * q_Qchipval_hi * sample *  tdfact;
IIcorr_lo +=  bfact * dfact * q_Ichipval_lo * sample * !tdfact;
IQcorr_lo += !bfact * dfact * q_Ichipval_lo * sample * !tdfact;
QIcorr_lo +=  bfact * dfact * q_Qchipval_lo * sample * !tdfact;
QQcorr_lo += !bfact * dfact * q_Qchipval_lo * sample * !tdfact;
samp_corr++;

*nskip = 1;
if ( N_TIME == samp_corr ) {
    //
    // UPDATE TAU-DITHER FACTOR
    //
    tdfact = !tdfact;
    //
    // COMPUTE THE OUTPUT
    //
    Ioutput_hi = IIcorr_hi*IIcorr_hi + IQcorr_hi*IQcorr_hi;
    Qoutput_hi = QIcorr_hi*QIcorr_hi + QQcorr_hi*QQcorr_hi;
    Ioutput_lo = IIcorr_lo*IIcorr_lo + IQcorr_lo*IQcorr_lo;
    Qoutput_lo = QIcorr_lo*QIcorr_lo + QQcorr_lo*QQcorr_lo;
    //
    // KEEP RUNNING SUMMATION OF THE DIFFERENCES IN THE OUTPUT
    //
    sIoutput_hi += Ioutput_hi >> 4;
    sQoutput_hi += Qoutput_hi >> 4;
    sIoutput_lo += Ioutput_lo >> 4;
    sQoutput_lo += Qoutput_lo >> 4;
    //
    // UPDATE THE SAMPLING COEFFICIENT
    //
    nsum = (++nsum) % (N_SUM_OUTPUT);
    if ( nsum == 0 ) {
            sIoutput_hi = sIoutput_hi >> N_SUM_OUTPUT_SHIFT;
            sQoutput_hi = sQoutput_hi >> N_SUM_OUTPUT_SHIFT;
            sIoutput_lo = sIoutput_lo >> N_SUM_OUTPUT_SHIFT;
            sQoutput_lo = sQoutput_lo >> N_SUM_OUTPUT_SHIFT;
            slope_sIoutput = sIoutput_hi - sIoutput_lo;
            slope_sQoutput = sQoutput_hi - sQoutput_lo;
            if ( slope_sIoutput > 7.0e6 ) {
            //  *nskip = 2;
            }
            else if ( slope_sIoutput < -7.0e6 ) {
            //  *nskip = 0;
            }
            // printf( "%d  %d\n", slope_sIoutput, slope_sQoutput );
            sIoutput_hi = 0;
            sQoutput_hi = 0;
            sIoutput_lo = 0;
            sQoutput_lo = 0;
        }
        fflush( NULL );
        //
        // REINITIALIZE THE CORRELATORS
        //
        IIcorr_hi = 0;
        IQcorr_hi = 0;
        QIcorr_hi = 0;
        QQcorr_hi = 0;
        IIcorr_lo = 0;
        IQcorr_lo = 0;
        QIcorr_lo = 0;
        QQcorr_lo = 0;
        samp_corr = 0;
        carr_ptr = 0;
    }
    //
    // ADVANCE THE SAMPLING TIME
    //
    samp_chip_hi += (*nskip) * parm_ptr->delta_t;
    if ( samp_chip_hi > 0.5 ) {
        samp_chip_hi -= 1.0;
        curr_chip_hi = (++curr_chip_hi)%PNLENGTH;
    }
    samp_chip_lo += (*nskip) * parm_ptr->delta_t;
    if ( samp_chip_lo > 0.5 ) {
        samp_chip_lo -= 1.0;
        curr_chip_lo = (++curr_chip_lo)%PNLENGTH;
    }

    return( 1 );
}
```

G.3 Header Files

G.3.1 chanmaps.h

```
int pilot_map[NUMPILOT] = {0};              // MAP WALSH FUNCTION NUMBER INTO PILOT CHANNEL NUMBER
int paging_map[NUMPAGING] = {1,2,3,4,5,6,7}; // MAP WALSH FUNCTION NUMBERS INTO PAGING CHANNEL NUMBERS
int sync_map[NUMSYNC] = {32};                // MAP WALSH FUNCTION NUMBER INTO SYNC CHANNEL NUMBER
int traffic_map[NUMTRAFFIC] = {8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,
33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63};
                                            // MAP WALSH FUNCTION NUMBERS INTO TRAFFIC CHANNEL NUMBERS
```

G.3.2 info_bs.h

```
//
// THE DELAYS OF EACH BASE STATION IN CHIP UNITS
//
double displace_bs[NUM_BASE_STATION] = {0};
//
// THE AMPLITUDES OF THE BASE STATIONS
//
double atten_bs[NUM_BASE_STATION] = {1.0};
```

G.3.3 info_mp.h

```
//
// HOW FAR EACH MULTIPATH IS DISPLACED FROM MAIN COMPONENT IN UNITS OF CHIPS
// I.E., THE MAIN COMPONENT IS THE FIRST ELEMENT, AND IS ALWAYS AT DISPLACEMENT ZERO
//
double displace_mp[NUM_MP] = {0};
//
// THE ATTENUATION FACTOR OF EACH MULTIPATH COMPONENT.
// THE FIRST ELEMENT OF THE ARRAY IS THE ATTENUATION FACTOR OF THE MAIN COMPONENT.
//
double atten_mp[NUM_MP] = {1.00};
```

G.3.4 is95puls.h

```
#define NUMBBFILTCOEFF (48)

//
// BASEBAND FILTER FOR IS-95A STANDARD
// FOUR SAMPLES PER CHIP
//
static double is95puls_h[NUMBBFILTCOEFF] = {
  -0.025288315,
  -0.034167931,
  -0.035752323,
  -0.016733702,
   0.021602514,
   0.064938487,
   0.091002137,
   0.081894974,
   0.037071157,
  -0.021998074,
  -0.060716277,
  -0.051178658,
   0.007874526,
   0.084368728,
    0.126869306,
    0.094528345,
   -0.012839661,
   -0.143477028,
   -0.211829088,
   -0.140513128,
    0.094601918,
    0.441387140,
    0.785875640,
    1.000000000,
    1.000000000,
    0.785875640,
    0.441387140,
    0.094601918,
   -0.140513128,
   -0.211829088,
   -0.143477028,
   -0.012839661,
    0.094528345,
    0.126869306,
    0.084368728,
    0.007874526,
   -0.051178658,
   -0.060716277,
   -0.021998074,
    0.037071157,
    0.081894974,
    0.091002137,
    0.064938487,
    0.021602514,
   -0.016733702,
   -0.035752323,
   -0.034167931,
   -0.025288315,
};


static double is95puls_t[NUMBBFILTCOEFF] = {
   -5.875,
   -5.625,
   -5.375,
   -5.125,
   -4.875,
   -4.625,
   -4.375,
   -4.125,
   -3.875,
   -3.625,
   -3.375,
   -3.125,
   -2.875,
   -2.625,
   -2.375,
   -2.125,
   -1.875,
   -1.625,
   -1.375,
   -1.125,
   -0.875,
   -0.625,
   -0.375,
   -0.125,
    0.125,
    0.375,
    0.625,
    0.875,
   1.125,
   1.375,
   1.625,
   1.875,
   2.125,
   2.375,
   2.625,
   2.875,
   3.125,
   3.375,
   3.625,
   3.875,
   4.125,
   4.375,
   4.625,
   4.875,
   5.125,
   5.375,
   5.625,
   5.875,
};


//
// THESE ARE THE SECOND DERIVATIVES OF THE IS-95A PULSE
// THEY ARE NEEDED FOR THE splint INTERPOLATION ROUTINE
// THEY WERE COMPUTED WITH THE spline ROUTINE FROM
// NUMERICAL RECIPES IN C
//
static double is95puls_d[NUMBBFILTCOEFF] = {
   0.0,
   0.07751501725632341,
   0.390281434974706,
   0.3392484908448535,
   0.1072137216458799,
  -0.2881267054283732,
  -0.6128499079323878,
  -0.6368717108420751,
  -0.2684620326993117,
   0.3431600976393214,
   0.849520330142026,
   0.8913174937925741,
   0.3387039346876777,
  -0.5717955045432839,
  -1.314909820514545,
  -1.353352957398532,
  -0.4742746698913268,
   1.01659298096384,
   2.38729221803597,
   2.842368066892283,
   1.967947770394898,
   0.006177747528124833,
  -2.2131440725074,
  -3.66855889749852,
  -3.668558897498521,
  -2.2131440725074,
   0.006177747528124722,
   1.967947770394898,
   2.842368066892283,
   2.38729221803597,
   1.01659298096384,
  -0.4742746698913268,
  -1.353352957398533,
  -1.314909820514545,
  -0.5717955045432838,
   0.338703934687677 6,
   0.8913174937925742,
   0.849520330142026,
   0.3431600976393215,
  -0.2684620326993118,
  -0.6368717108420749,
  -0.6128499079323878,
  -0.2881267054283732,
   0.10721372164588,
   0.3392484908448534,
   0.390281434974706,
   0.0775150172563234,
   0.0
};
```

G.3.5 setup.h

```
#include <stdio.h>

#define PI (3.14159265359)
#define TRUE (1)
#define FALSE (0)

//
// PARAMETERS RELATING TO THE SIMULATION OF ISI
//
#define ISIPREVCHIP (6)    // NUMBER OF PREVIOUS CHIPS CONTRIBUTING TO ISI
#define ISIFUTRCHIP (6)    // NUMBER OF FUTURE CHIPS CONTRIBUTING TO ISI

//
// CHANGE THE FOLLOWING IF WANT TO USE DIFFERENT PN SEQUENCE
//
#define PNLENGTH (32768)   // LENGTH OF THE PN SEQUENCES
#define Imask (0x42E2)     // MASK FOR EXCLUSIVE OR OF I CHANNEL SHIFT REGISTER
#define Qmask (0x4F1C)     // MASK FOR EXCLUSIVE OR OF Q CHANNEL SHIFT REGISTER
#define NTAP (15)          // NUMBER OF SHIFT REGISTER TAPS

//
// PARAMETERS RELATING TO THE SIMULATION SETUP
//
#define NUM_CYCL_GEN (13)   // TOTAL NUMBER OF PN SEQUENCE CYCLES TO GENERATE
#define CHIP_0 (50)         // CHIP ON WHICH RECEIVED PN SEQUENCE BEGINS

//
// PROTOTYPE OF THE STRUCTURE sim_parm
// WHICH CONTAINS THE PARAMETERS OF THE SIMULATION
//
struct sim_parm {
    int numcorr;         // NUMBER OF SAMPLES OVER WHICH TO CORRELATE
    double delta_t;      // NOMINAL SAMPLING INTERVAL IN UNITS OF A CHIP PERIOD
    double displace;     // DISPLACEMENT OF FIRST SAMPLING INSTANT FROM CENTER OF CHIP_0.
                         // THIS NUMBER RUNS BETWEEN -0.5 AND 0.5, AND, ALONG WITH CHIP_0, DETERMINES
                         // THE INITIAL LAG BETWEEN THE RECEIVER CORRELATOR AND THE RECEIVED SIGNAL
                         // NOTE: FIRST RECEIVER SAMPLE FOR CORRELATION IS AT -0.375
    int numadbits;       // NUMBER OF BITS USED BY THE A/D CONVERTER FOR QUANTIZATION
    double vlimlo;       // LOWEST VOLTAGE QUANTIZATION LEVEL
    double vlimhi;       // HIGHEST VOLTAGE QUANTIZATION LEVEL
    double input_snr_db; // SNR IN DB AT THE A/D INPUT
    double rms_phs_err;  // RMS PHASE ERROR IN DEGREES
    double f_offset;     // LOCAL OSCILLATOR FREQUENCY OFFSET IN PPM RELATIVE TO THE IF
    int Ns;              // NOMINAL NUMBER OF SAMPLES PER CHIP
    double samp_offset;  // SAMPLING FREQUENCY OFFSET IN PPM RELATIVE NOMINAL SAMPLING FREQUENCY
    double t_jitter;     // RMS SAMPLING JITTER IN ps
    char rcv1data[9];    // FILE NAME WHERE RECEIVER 1'S DATA GOES
    char rcv2data[9];    // FILE NAME WHERE RECEIVER 2'S DATA GOES
};

#define WFLENGTH (64)        // NUMBER OF CHIPS IN A WALSH FUNCTION

#define NUMPILOT (1)         // TOTAL NUMBER OF PILOT CHANNELS
#define NUMPAGING (7)        // TOTAL NUMBER OF PAGING CHANNELS
#define NUMSYNC (1)          // TOTAL NUMBER OF SYNC CHANNELS
#define NUMTRAFFIC (55)      // TOTAL NUMBER OF TRAFFIC CHANNELS

#define PILOT_IN_USE (1)     // NUMBER OF PILOT CHANNELS IN USE IN THE SIMULATION
#define PAGING_IN_USE (0)    // NUMBER OF PAGING CHANNELS IN USE IN THE SIMULATION
#define SYNC_IN_USE (0)      // NUMBER OF SYNC CHANNELS IN USE IN THE SIMULATION
#define TRAFFIC_IN_USE (0)   // NUMBER OF TRAFFIC CHANNELS IN USE IN THE SIMULATION

//
// THE FOLLOWING FOUR NUMBERS MUST ADD UP TO UNITY
//
#define PILOT_PWR_FRAC    (1.00) // FRACTION OF TOTAL POWER ALLOCATED TO PILOT CHANNEL B/S 1
#define PAGING_PWR_FRAC   (0.00) // FRACTION OF TOTAL POWER ALLOCATED TO PAGING CHANNELS B/S 1
#define SYNC_PWR_FRAC     (0.00) // FRACTION OF TOTAL POWER ALLOCATED TO SYNC CHANNEL B/S 1
#define TRAFFIC_PWR_FRAC  (0.00) // FRACTION OF TOTAL POWER ALLOCATED TO TRAFFIC CHANNELS B/S 1

#if PILOT_IN_USE>0
    #define PILOT_CHANNEL_PWR (PILOT_PWR_FRAC/PILOT_IN_USE)
#else
    #define PILOT_CHANNEL_PWR (0)
#endif

#if PAGING_IN_USE>0
    #define PAGING_CHANNEL_PWR (PAGING_PWR_FRAC/PAGING_IN_USE)
#else
    #define PAGING_CHANNEL_PWR (0)
#endif

#if SYNC_IN_USE>0
    #define SYNC_CHANNEL_PWR (SYNC_PWR_FRAC/SYNC_IN_USE)
#else
    #define SYNC_CHANNEL_PWR (0)
#endif

#if TRAFFIC_IN_USE>0
    #define TRAFFIC_CHANNEL_PWR (TRAFFIC_PWR_FRAC/TRAFFIC_IN_USE)
#else
    #define TRAFFIC_CHANNEL_PWR (0)
#endif

//
// THE INITIAL TRANSMITTER CARRIER PHASE IN DEGREES
//
#define CARR_INIT_PHASE (0.0)
//
// THE INTERMEDIATE FREQUENCY IN HZ
//
#define F_IF (20.8896e6)
//
// THE CHIP RATE IN HZ
//
#define F_CHIP (1.2288e6)
//
// NUMBER OF SAMPLES OVER WHICH THE AGC INTEGRATES
//
#define AGC_CORR (4096)
//
// NUMBER OF MULTIPATH COMPONENTS
//
#define NUM_MP (1)
```

```
//
// NUMBER OF BITS TO WHICH TO QUANTIZE PILOT SEQUENCE IN RCVRS
//
#define NUM_BITS_PILOT_RCVR_1 (8)
#define NUM_BITS_PILOT_RCVR_2 (6)
//
// EMPIRICAL # BY WHICH TO SCALE THE PULSE-SHAPING FILTER COEFFICIENTS
// SO THAT MAX OUTPUT LIES AT CORRECT VALUE
//
#define BBFILTCOEFFSCALE (15586)
//
// TARGET VARIANCE FOR AGC
// EXPRESSED AS A FRACTION OF THE SQUARE OF THE FULL SCALE RANGE
//
#define SIGMA_2_TARGET (2.0)
//
// NUMBER OF BASE STATIONS
//
#define NUM_BASE_STATION (1)
//
// THE INTEGRATION TIME FOR THE TIMING RECOVERY
//
#define N_TIME (4096)
//
// NUMBER OF ADVANCED AND DELAYED CORRELATION CYCLES TO SUM OVER
//
#define N_SUM_OUTPUT (4)
//
// HOW MANY BITS TO RIGHT SHIFT TO EFFECT THE DIVISION FOR AVERAGING THE CORRELATION OUTPUTS
//
#define N_SUM_OUTPUT_SHIFT (2)
//
// NEGATIVE THRESHOLD TO DETERMINE WHEN TO SKIP EXTRA SAMPLE
//
#define SKIP_THRESHOLD_NEG (-1.0e8)
//
// POSITIVE THRESHOLD TO DETERMINE WHEN TO SKIP EXTRA SAMPLE
//
#define SKIP_THRESHOLD_POS (1.0e8)
//
// FRACTION OF PHASE NOISE SPECTRUM IN WHITE NOISE FLOOR
//
#define WHITE_NOISE_FRAC (0.2)

void splint( double [], double [], double [], int, double, double * );
int gensrop( int, int *, int * );
int rcvr1fps( int, int [], int [], int *, int *, struct sim_parm *, int, int * );
int rcvr2fps( int, int [], int [], int *, int *, struct sim_parm *, int, int * );
int chckinpt( int *, char *[], struct sim_parm * );
int chck_bnd( int * );
int quantize( double, double *, int );
int initbins( double *, int, double, double );
int gensamp( int, double, int [], int [], double [], double, double * );
int adv_ts( double *, int *, struct sim_parm * );
int adv_phs( double, double, double, double *, struct sim_parm * );
int genintr( double *, int  );
int gennoise( double, double [] );
double ran2( long * );
int genwalsh( int [WFLENGTH][WFLENGTH] );
double fir_filter( double *, double *, int, int );
int prntdata( FILE *, struct sim_parm * );
int agc( int, double *, struct sim_parm * );
int timercv2( int, int [], int [], struct sim_parm *, int, int * );
```

Appendix H

TMS320C541 Assembly Language Source Code

H.1 TMS320C541 Assembly Files

These files can be assembled using the batch file ASEM.BAT below. After assembly, the file SCANCORR.OUT is created. This file can be run by starting the TI 'C54 debugger, then executing the command `load scancorr`. After successful loading, start the DSP running in free-running mode by issuing the command `runf`. Then type `quit` at the command prompt to exit the debugger. Back at the DOS prompt, type the command `retdata1`, which is the complilation of `retdata1.c` found below. The DSP will begin real-time correlations and the results will be transferred to the PC. The system can be halted by pressing any key on the PC.

H.1.1 SCANCORR.ASM

```
*********************************************************************
*  FILE NAME:  scancorr.asm                                         *
*                                                                   *
*  IMPLEMENTATION OF A SCANNING CORRELATOR                          *
*  Keith Blankenship                                                *
*  February 2, 1998                                                 *
*********************************************************************
        .title    "scanning correlator"

        .ref      ips

        .def      scancorr         ;
        .def      wlop             ;
        .def      corr             ;
        .def      Fenbpt           ;
        .def      Frstpt           ;
        .def      Frdpt             ;
        .def      STreg             ;
        .def      FIFOon            ;
        .def      FIFOoff           ;
        .def      p_i               ;
        .def      p_q               ;
        .def      seg_cnt           ;
        .def      NumSeg            ;

        .mmregs                     ; enter mem-map register into sym table

LAbsTab .equ    40
NumPhas .equ    0x4000
NumLoop .equ    0x0001
NumSeg  .equ    7
Wpt     .equ    0x0006          ; port number for AGC control word
Nlodpt  .equ    0x0007          ; port number for nload configuration
Fenbpt  .equ    0x0008          ; port number for enb/disable FIFO
Frstpt  .equ    0x0009          ; port number for FIFO reset
Frdpt   .equ    0x000A          ; port number for FIFO reads
onbFIFO .equ    0x0012          ; port number of onboard (EVM) FIFO
Areg    .equ    0x0010          ; port number of A register on EVM
STreg   .equ    0x0014          ; port number of status register
;
; THESE FILES CONTAIN THE QUANTIZED PULSE-SHAPED PILOT PN SEQUENCES.
;
p_i     .sect   "pn_i"
        .include  pn_i.dat
p_q     .sect   "pn_q"
        .include  pn_q.dat
AbsTab  .sect   "abs_tab"
        .include  abs_tab.dat
dBLuTab .sect   "dblu_tab"
        .include  dblu_tab.dat
AGCdat  .sect   "agc_dat"
; current value of Pest
Pest    .word   0
; current value of Gain
Gain    .word   0
; current value of Ptarget
Ptarget .word   20
; current control word
W       .word   156
; intermediate result
I       .word   0
Corrdat .sect   "corr_dat"
; current value of NumCorr
NumCorr .word   0
;
; THIS SECTION OF MEMORY CONTAINS THE PHASES OF THE LOCAL PILOT SIGNAL
;   FOR THE EIGHT SEGMENTS OF DATA CAPTURED FOR EVERY PERIOD OF THE
;   PILOT SIGNAL.
;
p_ptr   .word   0
        .word   0
        .word   0
        .word   0
        .word   0
        .word   0
        .word   0
        .word   0
Ahi     .word   0
Alo     .word   0
Bhi     .word   0
Blo     .word   0
seg_cnt .word   -1
```

```
;
; configuration parameters for the data acquisition board
;
BrdCon  .sect   "brd_con"
FIFOon  .word   0x0001
FIFOoff .word   0x0000
nload   .word   0x000C


        .text
scancorr:
;
; INITIALIZE INTERRUPTS
;
        RSBX    1, 11                   ; enable all unmasked interrupts
        STM     0x0001, IMR             ; enable INT0
;
; INITIALIZE GAIN
;
        STM     W, AR3                  ; AR3 points to control word
        PORTW   *AR3, Wpt               ; write out initial gain value
;
; INITIALIZE NUMBER OF ON/OFF CYCLES
;
        STM     nload, AR0              ; AR0 points to nload
        PORTW   *AR0, Nlodpt            ; configure data acquisition
;
; INITIALIZE COMMUNICATIONS BETWEEN PC AND EVM
;
init1   NOP                             ; watch for BIO to be pulsed
        BC      init1, BIO              ;
init2   NOP                             ;
        BC      init2, NBIO             ;
init3   NOP                             ;
        BC      init3, BIO              ;
        PORTW   AR3, Areg               ; write anything to A buffer
tlop    PORTR   STreg, *AR3             ; read the status register
        BIT     *AR3, 15                ; test the AXST bit
        BC      tlop, TC                ; branch until flag clear
;
; RESET TC
;
        RSBX    TC
;
; RESET THE EXTERNAL FIFO
;
        STM     FIFOoff, AR0            ; AR0 points to FIFOoff
        PORTW   *AR0, Fenbpt            ; disable FIFO
        PORTW   *AR0, Frstpt            ; reset the FIFO (doesn't matter what
                                        ; value is written, just have to pulse
                                        ; the address line)
        STM     FIFOon, AR0             ; AR0 points to FIFO on
        PORTW   *AR0, Fenbpt            ; enable the FIFO
;
; MAIN IDLE LOOP - WAIT UNTIL WE HAVE NEW DATA TO PROCESS
; THE PROGRAM IS ALERTED THAT NEW DATA IS PRESENT WHEN TC == 1
;
wlop    NOP                             ; idle loop
        BC      corr, TC                ; branch if TC == 1
        B       wlop                    ; return to top of wait loop
;
; MAIN DATA PROCESSING ROUTINE
;
corr    RSBX    TC                      ; reset TC bit
        SSBX    XF
;
```

```
; AGC ROUTINE
;
        STM     AbsTab, AR1             ; AR1 points to entry 1 of abs table
        STM     Pest, AR2               ; AR2 points to Pest in abs value
        STM     Gain, AR3               ; AR3 points to current gain
        STM     Ptarget, AR4            ; AR4 points to Ptarget in dB
        STM     ips, AR5                ; AR5 points to inputs
        STM     I, AR6                  ; AR6 points to intermediate result
        LD      #0x0000, A              ; clear A accumulator
        RPT     #4095                   ; repeat next 4096 times
        SQURA   *AR5+, A                ; running sum of squares
        SFTA    A, -12, A               ; divide result by 4096
        STL     A, *AR2                 ; store result in Pest
        STM     W, AR5                  ; AR5 points to control word
        SSBX    TC                      ; set the TC bit
        STM     LAbsTab-1, BRC          ; intialize BRC
        RPTB    end_lu                  ;
        LD      *AR1+, A                ; put value into accum
        SUB     *AR2, A                 ; compare Pest to current abs value
        XC      1, TC                   ; if TC is set, do next line
        SRCCD   *AR2, AGEQ              ; store BRC if A>=0
        XC      1, AGEQ                 ; if A>=0, do next line
end_lu  RSBX    TC                      ; reset TC
        XC      2, TC                   ; if Pest is greater than greatest entry
        ST      #0x0000, *AR2           ; store 0 in index of Pest in dB
        LD      *AR2, A                 ; put index of Pest in dB into A
        ADD     #dBLuTab, A             ; add address of entry 1 table of dB value
        NOP
        NOP
        STLM    A, AR2                  ; now AR2 points to Pest in dB
        NOP
        NOP
        LD      *AR3, A                 ; load current gain into A
        ADD     *AR4, A                 ; add Ptarget in dB to A
        SUB     *AR2, A                 ; subtract Pest in dB from A
        STL     A, *AR3                 ; store updated gain in memory
        ADD     #0x0078, A              ; add 120 to updated gain
        STL     A, *AR6                 ; store intermediate result
        MPY     *AR6, #0x0005, A        ; multiply result by 5
        SFTA    A, -2, A                ; divide by 4
        ADD     #0x0008, A              ; add 8
        STL     A, *AR5                 ; store result in W
        PORTW   *AR5, Wpt               ; write out word to control AGC
;
; CORRELATION LOOP.
;   THIS PROGRAM IMPLEMENTS THE SCANNING CORRELATION METHOD DESCRIBED
;     IN MY MASTER'S THESIS.
;   EIGHT SEGMENTS OF DATA, EACH CONTAINING 4096 SAMPLES, ARE CAPTURED
;     FOR EVERY PERIOD OF THE PILOT SIGNAL.
;   THERE ARE EIGHT "PHASE POINTERS," STARTING AT ADDRESS p_ptr.
;   A PHASE POINTER HOLDS THE SAMPLE NUMBER AT WHICH
;     THE CORRELATION BEGINS. THEREFORE, IT IS CONTINUALLY INCREMENTED,
;     IN A CIRCULAR FASHION BETWEEN 0X0000 AND 0X3FFF.
;   THE MEMORY-MAPPED REGISTER   AR4   HOLDS THE ADDRESS OF THE PHASE
;     POINTER FOR THE CURRENT SEGMENT. THEREFORE, IT IS CONTINUALLY
;     INCREMENTED, IN A CIRCULAR FASHION, BETWEEN p_ptr and p_ptr+7.
;   MORE THAN ONE CORRELATION CAN BE PERFORMED FOR EACH BLOCK
;     OF DATA CAPTURED. THE MEMORY-MAPPED REGISTER   AR7   KEEPS TRACK
;     OF THE NUMBER OF CORRELATIONS PERFORMED.
;
        STM     NumCorr, AR7            ; AR7 points to number of correlations
        ST      #0x0000, *AR7           ; zero out number of correlations
        STM     seg_cnt, AR1            ; AR1 points to segment counter
        LD      *AR1, A                 ; load segment counter into A
        ADD     #p_ptr, A               ; add in base address of phase pointer
        STLM    A, AR4                  ; now AR4 points to phase of current segment
        NOP
        NOP
ct      LD      *AR4, A                 ; load phase in A
        ADD     #p_i, A                 ; add I sequence base address to A
        STLM    A, AR2                  ; store phase in AR2
        LD      *AR4, A                 ; load phase in A
        ADD     #p_q, A                 ; add Q sequence base address to A
        STLM    A, AR3                  ; store phase in AR3
        LD      #0X0000, A              ; clear A accumulator
        LD      #0X0000, B              ; clear B accumulator
        ST      #0x03FF, BRC            ; repeat block 1024 times
        RPTBD   cyc_i
        STM     ips, AR5                ; store location of inputs
        MAC     *AR2+, *AR5+, A         ;
        MAC     *AR2+, *AR5+, B         ;
        MAS     *AR2+, *AR5+, A         ;
cyc_i   MAS     *AR2+, *AR5+, B         ;
        STM     Ahi, AR6                ; AR6 points to Ahi
        STH     A, *AR6+                ; store high A to mem
        STL     A, *AR6+                ; store low  A to mem
        STH     B, *AR6+                ; store high B to mem
        STL     B, *AR6+                ; store low  B to mem
        STM     Ahi, AR6                ; AR6 points to Ahi
        PORTW   *AR6+, onbFIFO          ; write high A to onboard FIFO
        PORTW   *AR6+, onbFIFO          ; write low  A to onboard FIFO
        PORTW   *AR6+, onbFIFO          ; write high B to onboard FIFO
        PORTW   *AR6+, onbFIFO          ; write low  B to onboard FIFO
        LD      #0X0000, A              ; clear A accumulator
        LD      #0X0000, B              ; clear B accumulator
        ST      #0x03FF, BRC            ; repeat block 1024 times
        RPTBD   cyc_q
        STM     ips, AR5                ; store location of inputs
        MAC     *AR3+, *AR5+, A         ;
        MAC     *AR3+, *AR5+, B         ;
        MAS     *AR3+, *AR5+, A         ;
cyc_q   MAS     *AR3+, *AR5+, B         ;
        ADDM    #0x0001, *AR4           ; increment phase pointer
        ADDM    #0x0001, *AR7           ; increment loop pointer
        STM     Ahi, AR6                ; AR6 points to Ahi
        STH     A, *AR6+                ; store low  A to mem
        STL     A, *AR6+                ; store high A to mem
        STH     B, *AR6+                ; store low  B to mem
        STL     B, *AR6+                ; store high B to mem
        STM     Ahi, AR6                ; AR6 points to Ahi
        PORTW   *AR6+, onbFIFO          ; write low  A to onboard FIFO
        PORTW   *AR6+, onbFIFO          ; write high A to onboard FIFO
        PORTW   *AR6+, onbFIFO          ; write low  B to onboard FIFO
        PORTW   *AR6+, onbFIFO          ; write high B to onboard FIFO
        PORTW   *AR6+, onbFIFO          ; write segment number to onboard FIFO
        PORTW   *AR4, onbFIFO           ; write phase to onboard FIFO
        CMPM    *AR4, NumPhas           ; check if all phases are exhausted
        NOP
        XC      1, TC                   ; if all phases exhausted, do next
        ST      #0x0000, *AR4           ; reset phase pointer
        CMPM    *AR7, NumLoop           ; check if all loops exhausted
        NOP
        BC      ct, NTC                 ; if not exhasted, branch to top
;
; SIGNAL PC TO READ OUR RESULTS
;
        STM     Gain, AR5               ; AR5 points to current gain
        PORTW   *AR5, Areg              ; signal PC by writing to A register
        STM     I, AR2                  ; AR2 points to temp mem loc
alop    PORTR   STreg, *AR2             ; read the status register
        BIT     *AR2, 15                ; test the AXST bit
        BC      alop, TC                ; branch until flag clear
        RSBX    XF
        RSBX    TC
        B       wlop                    ; return to top of wait loop
        .end
```

H.1.2 VECTORS.ASM

```
************************************************************************
* FILENAME: VECTORS.ASM
* This routine initializes the 54x vector table.
* Keith Blankenship
************************************************************************
.title  "54X Vector Table Initialization"

.ref      scancorr          ;
.ref      wlop              ;
.ref      corr              ;
.ref      Fenbpt            ;
.ref      Frstpt            ;
.ref      Frdpt             ;
.ref      STreg             ;
.ref      FIFOon            ;
.ref      FIFOoff           ;
.ref      p_i               ;
.ref      p_q               ;
.ref      seg_cnt           ; counts the segments
.ref      NumSeg            ; total number of segments minus one

.def    STACK
.def    ips

.mmregs
K_STACK_SIZE    .set    80

STACK           .usect  "stack",K_STACK_SIZE

ips             .usect  "inputs", 4096

.sect   "vectors"
reset:          BD      scancorr                ; RESET vector
STM     #K_STACK_SIZE+STACK, SP  ; initialize stack pointer to stack_start
nmi:            NOP                             ; ~NMI
NOP
NOP
RET

* software interrupts
sint17          .space  4*16
sint18          .space  4*16
sint19          .space  4*16
sint20          .space  4*16
sint21          .space  4*16
sint22          .space  4*16
sint23          .space  4*16
sint24          .space  4*16
sint25          .space  4*16
sint26          .space  4*16
sint27          .space  4*16
sint28          .space  4*16
sint29          .space  4*16
sint30          .space  4*16

int0:           STM     0x1000, AR0            ; data to set USRBOT0
PORTW   AR0, STreg              ; set USRBOT0
```

```
STM     ips, AR4                ; store location of inputs
STM     FIFOoff, AR0            ; AR0 points to FIFO off
PORTW   *AR0, Fenbpt            ; disable external FIFO
RPT     #0x0FFF                 ; repeat next 4096 times
PORTR   Frdpt, *AR4+            ; read from port 10
PORTW   *AR0, Frstpt            ; reset FIFO
STM     seg_cnt, AR1            ; AR1 points to segment counter
STM     FIFOon, AR0             ; AR0 points to FIFO on
PORTW   *AR0, Fenbpt            ; enable FIFO
CMPM    *AR1, NumSeg            ; check if segments exhausted
NOP
XC      2, TC                   ; if so, reset
ST      #0xFFFF, *AR1
ADDM    #0x0001, *AR1
STM     0x0000, AR0
PORTW   AR0, STreg              ; clear USRBOT0
SSBX    TC                      ; set TC bit to 1
RETE                            ; return
.end
```

H.1.3 SCANCORR.CMD

```
vectors.obj
scancorr.obj

-e scancorr
-o scancorr.out
-m scancorr.map

MEMORY
{
PAGE 0:                                           /* Pgm.space    */
PROG    : origin = 0x9000, length = 0x1000  /* Int.Pgm.area */
VECS    : origin = 0xff80, length = 0x007f  /* Vector       */
PAGE 1:                                           /* Data space   */
RAM0    : origin = 0x0060, length = 0x0020  /* scratch      */
RAM1    : origin = 0x1000, length = 0x1000  /* inputs       */
RAM2    : origin = 0x2000, length = 0x0050  /* stack        */
RAM3    : origin = 0x3000, length = 0x5000  /* i pn seq     */
RAM4    : origin = 0x8000, length = 0x5000  /* q pn seq     */
RAM5    : origin = 0xD000, length = 0x0010  /* Corr data    */
RAM6    : origin = 0xD010, length = 0x0028  /* Pest abs     */
RAM7    : origin = 0xD038, length = 0x0028  /* Pest dB      */
RAM8    : origin = 0xD060, length = 0x0005  /* AGC data     */
REGS    : origin = 0x0000, length = 0x0060  /* MMR's        */
PAGE 2:                                           /* IO space     */
IOPORT  : origin = 0x0000, length = 0x000f
}

SECTIONS
{
.text           : load = PROG PAGE 0   /* code */
vectors         : load = VECS PAGE 0   /* vector table */
brd_con         : load = RAM0 PAGE 1   /* board configuration data */
inputs          : load = RAM1 PAGE 1   /* input data */
stack           : load = RAM2 PAGE 1   /* stack */
pn_i            : load = RAM3 PAGE 1   /* i pn seq */
pn_q            : load = RAM4 PAGE 1   /* q pn seq */
corr_dat        : load = RAM5 PAGE 1   /* data for correlation */
abs_tab         : load = RAM6 PAGE 1   /* Pest abs */
dblu_tab        : load = RAM7 PAGE 1   /* Pest dB */
agc_dat         : load = RAM8 PAGE 1   /* data for AGC */
}
```

H.2 C Files

The file `retdata1.exe` must be created by compiling the program `RETDATA1.C`. When run along with the assembly program `SCANCORR.OUT`, `retdata1.exe` retrieves data from the DSP and writes it to the disk of the PC.

H.2.1 RETDATA1.C

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <math.h>
#include <graphics.h>

#define ARST           0x0002
#define HBIO           0x0800  /* mask for host input to EVM BIO line    */
#define CORR_PER_SEG   16384   /* total number of corrs per segment      */
#define CORR_PER_INT   1       /* number of corrs per interrupt          */
#define NUM_SEGMENT    8       /* tot number of segments                 */

#define PCBASE (0x0240)            /* PC base I/O address for hst-EVM comm. */
unsigned int  PC_CHA=PCBASE+0x0800; /* 0x0240 is the default value.         */
unsigned int  PC_CB1=PCBASE+0x0804;
unsigned int  PC_CB2=PCBASE+0x0806;
unsigned int  PC_ST1=PCBASE+0x0808;
unsigned int  PC_ST2=PCBASE+0x080A;

/* arst() checks the value of the channel A receive status bit in the
 * host control register. If ARST==0, it means the EVM has not written
 * a value into channel A. (EVM Tech. Ref. 3-8)
 */
int arst(void)
{
unsigned int tmp=inport(PC_ST1)&ARST;

if (tmp == ARST) return(1);

return(0);
}

void main( void )
{
   int j;
   FILE *fp_corr;

   /* open data file */
   fp_corr = fopen( "corr.dat", "w" );
   setbuf( fp_corr, NULL );

   /* clear out the fifo if needed */
   for ( j=0; j<64; j++ ) inport( PC_CB1 );

   outport(PC_ST2,inport(PC_ST1));

   while (!arst()) {
     /* pulse HBIO */
     outport(PC_ST1,inport(PC_ST1) |  HBIO);
     outport(PC_ST1,inport(PC_ST1) & ~HBIO);
     if (kbhit()) {
getch();
exit(-1);
```

```
      }
   }

   printf( "%x\n", inport(PC_CHA) );    /* receive acknowledgment */

   while ( 1 ) {

      while( !( inport(PC_ST1)&ARST ) );
      inport( PC_CHA );
      fprintf( fp_corr, "%d %d %d %d\n",
inport(PC_CB1), inport(PC_CB1), inport(PC_CB1), inport(PC_CB1) );
      fprintf( fp_corr, "%d %d %d %d\n",
inport(PC_CB1), inport(PC_CB1), inport(PC_CB1), inport(PC_CB1) );
      fprintf( fp_corr, "%d %d 0 0\n", inport(PC_CB1), inport(PC_CB1) );
/*      fflush( fp_corr ); */

      if ( kbhit() ) break;
   }

   fclose( fp_corr );

   return;
}
```

H.3 Batch Files

H.3.1 ASEM.BAT

```
asm500 -ls vectors
asm500 -ls scancorr
lnk500 scancorr.cmd
```

H.4 Data Files

These data files are needed in the assembly code `SCANCORR.ASM` above. They constitute the lookup table need to convert from absolute units into dB units. There are also two other data files which are not included here in the interest of space: `PN_I.DAT` and `PN_Q.DAT`. These data files contain a contiguous block of 20,480 8-bit samples of the pulse shaped I- and Q-channel PN sequences. They can be generated by the reader according the prescription given in Section B.10.

H.4.1 ABS_TAB.DAT

```
        .word   1
        .word   2
        .word   3
        .word   4
        .word   5
        .word   6
        .word   8
        .word   10
        .word   13
        .word   16
        .word   20
        .word   25
        .word   32
        .word   40
        .word   50
        .word   63
        .word   79
        .word   100
        .word   126
        .word   158
        .word   200
        .word   251
        .word   316
        .word   398
        .word   501
        .word   631
        .word   794
        .word   1000
        .word   1259
        .word   1585
        .word   1995
        .word   2512
        .word   3162
        .word   3981
        .word   5012
        .word   6310
        .word   7943
        .word   10000
        .word   12589
        .word   15849
```

H.4.2 DBLU_TAB.DAT

```
        .word   42
        .word   41
        .word   40
        .word   39
        .word   38
        .word   37
        .word   36
        .word   35
        .word   34
        .word   33
        .word   32
        .word   31
        .word   30
        .word   29
        .word   28
        .word   27
        .word   26
        .word   25
        .word   24
        .word   23
        .word   22
        .word   21
        .word   20
        .word   19
        .word   18
        .word   17
        .word   16
        .word   15
        .word   14
        .word   13
        .word   12
        .word   11
        .word   10
        .word   9
        .word   8
        .word   7
        .word   6
        .word   5
        .word   3
        .word   1
```

H.5 Analysis Files

After halting the receiver, the file `reconv.exe` should be run to convert the DSP output into actual correlation outputs. Then the file `plotoutp.m` can be run in MATLAB to plot the results.

H.5.1 reconv.c

```
#include <stdio.h>

int main( void )
{
FILE *fp_input;
FILE *fp_I1, *fp_I2, *fp_I3, *fp_I4;
FILE *fp_I5, *fp_I6, *fp_I7, *fp_I8;
FILE *fp_Q1, *fp_Q2, *fp_Q3, *fp_Q4;
FILE *fp_Q5, *fp_Q6, *fp_Q7, *fp_Q8;
int AIlower, AIupper, BIlower, BIupper;
int AQlower, AQupper, BQlower, BQupper;
int AI, BI;
int AQ, BQ;
int phase, seg, dummy1, dummy2;

fp_input = fopen( "c:\\users\\blanken\\testpilt\\CORR.DAT", "r" );
fp_I1 = fopen( "c:\\users\\blanken\\testpilt\\segI1.dat", "w" );
fp_I2 = fopen( "c:\\users\\blanken\\testpilt\\segI2.dat", "w" );
fp_I3 = fopen( "c:\\users\\blanken\\testpilt\\segI3.dat", "w" );
fp_I4 = fopen( "c:\\users\\blanken\\testpilt\\segI4.dat", "w" );
fp_I5 = fopen( "c:\\users\\blanken\\testpilt\\segI5.dat", "w" );
fp_I6 = fopen( "c:\\users\\blanken\\testpilt\\segI6.dat", "w" );
fp_I7 = fopen( "c:\\users\\blanken\\testpilt\\segI7.dat", "w" );
fp_I8 = fopen( "c:\\users\\blanken\\testpilt\\segI8.dat", "w" );
fp_Q1 = fopen( "c:\\users\\blanken\\testpilt\\segQ1.dat", "w" );
fp_Q2 = fopen( "c:\\users\\blanken\\testpilt\\segQ2.dat", "w" );
fp_Q3 = fopen( "c:\\users\\blanken\\testpilt\\segQ3.dat", "w" );
fp_Q4 = fopen( "c:\\users\\blanken\\testpilt\\segQ4.dat", "w" );
fp_Q5 = fopen( "c:\\users\\blanken\\testpilt\\segQ5.dat", "w" );
fp_Q6 = fopen( "c:\\users\\blanken\\testpilt\\segQ6.dat", "w" );
fp_Q7 = fopen( "c:\\users\\blanken\\testpilt\\segQ7.dat", "w" );
fp_Q8 = fopen( "c:\\users\\blanken\\testpilt\\segQ8.dat", "w" );


while ( fscanf( fp_input, "%d", &BIlower ) != EOF ) {
fscanf( fp_input, "%d", &BIupper );
fscanf( fp_input, "%d", &AIlower );
fscanf( fp_input, "%d", &AIupper );
AI = AIupper;
AI <<= 16;
AI += AIlower&0x0000FFFF;
BI = BIupper;
BI <<= 16;
BI += BIlower&0x0000FFFF;
fscanf( fp_input, "%d", &BQlower );
fscanf( fp_input, "%d", &BQupper );
fscanf( fp_input, "%d", &AQlower );
fscanf( fp_input, "%d", &AQupper );
AQ = AQupper;
AQ <<= 16;
AQ += AQlower&0x0000FFFF;
BQ = BQupper;
BQ <<= 16;
BQ += BQlower&0x0000FFFF;
fscanf( fp_input, "%d", &phase );
fscanf( fp_input, "%d", &seg );
fscanf( fp_input, "%d", &dummy1 );
fscanf( fp_input, "%d", &dummy2 );
if ( seg == 0 ) {
fprintf( fp_I1, "%d %d %d\n", AI, BI, phase );
fprintf( fp_Q1, "%d %d %d\n", AQ, BQ, phase );
}
else if ( seg == 1 ) {
fprintf( fp_I2, "%d %d %d\n", AI, BI, phase );
fprintf( fp_Q2, "%d %d %d\n", AQ, BQ, phase );
}
else if ( seg == 2 ) {
fprintf( fp_I3, "%d %d %d\n", AI, BI, phase );
fprintf( fp_Q3, "%d %d %d\n", AQ, BQ, phase );
}
else if ( seg == 3 ) {
fprintf( fp_I4, "%d %d %d\n", AI, BI, phase );
fprintf( fp_Q4, "%d %d %d\n", AQ, BQ, phase );
}
else if ( seg == 4 ) {
fprintf( fp_I5, "%d %d %d\n", AI, BI, phase );
fprintf( fp_Q5, "%d %d %d\n", AQ, BQ, phase );
}
else if ( seg == 5 ) {
fprintf( fp_I6, "%d %d %d\n", AI, BI, phase );
fprintf( fp_Q6, "%d %d %d\n", AQ, BQ,phase );
}
else if ( seg == 6 ) {
fprintf( fp_I7, "%d %d %d\n", AI, BI, phase );
fprintf( fp_Q7, "%d %d %d\n", AQ, BQ, phase );
}
else if ( seg == 7 ) {
fprintf( fp_I8, "%d %d %d\n", AI, BI, phase );
fprintf( fp_Q8, "%d %d %d\n", AQ, BQ, phase );
}
}

fclose( fp_input );
fclose( fp_I1 );
fclose( fp_I2 );
fclose( fp_I3 );
fclose( fp_I4 );
fclose( fp_I5 );
fclose( fp_I6 );
fclose( fp_I7 );
fclose( fp_I8 );
fclose( fp_Q1 );
fclose( fp_Q2 );
fclose( fp_Q3 );
fclose( fp_Q4 );
fclose( fp_Q5 );
fclose( fp_Q6 );
fclose( fp_Q7 );
fclose( fp_Q8 );

return( 1 );
}
```

H.5.2 plotoutp.m

```
clear all
close all

load segi1.dat
load segi2.dat
load segi3.dat
load segi4.dat
load segi5.dat
load segi6.dat
load segi7.dat
load segi8.dat
load segq1.dat
load segq2.dat
load segq3.dat
load segq4.dat
load segq5.dat
load segq6.dat
load segq7.dat
load segq8.dat

segi1n = segi1(:,1).^2 + segi1(:,2).^2;
segi2n = segi2(:,1).^2 + segi2(:,2).^2;
segi3n = segi3(:,1).^2 + segi3(:,2).^2;
segi4n = segi4(:,1).^2 + segi4(:,2).^2;
segi5n = segi5(:,1).^2 + segi5(:,2).^2;
segi6n = segi6(:,1).^2 + segi6(:,2).^2;
segi7n = segi7(:,1).^2 + segi7(:,2).^2;
segi8n = segi8(:,1).^2 + segi8(:,2).^2;
segq1n = segq1(:,1).^2 + segq1(:,2).^2;
segq2n = segq2(:,1).^2 + segq2(:,2).^2;
segq3n = segq3(:,1).^2 + segq3(:,2).^2;
segq4n = segq4(:,1).^2 + segq4(:,2).^2;
segq5n = segq5(:,1).^2 + segq5(:,2).^2;
segq6n = segq6(:,1).^2 + segq6(:,2).^2;
segq7n = segq7(:,1).^2 + segq7(:,2).^2;
segq8n = segq8(:,1).^2 + segq8(:,2).^2;

part = 1:16384;
segi = [segi1n(part);segi2n(part);segi3n(part);segi4n(part);...
      segi5n(part);segi6n(part);segi7n(part);segi8n(part)];
segq = [segq1n(part);segq2n(part);segq3n(part);segq4n(part);...
      segq5n(part);segq6n(part);segq7n(part);segq8n(part)];

figure
subplot(211)
plot(segi)
subplot(212)
plot(segq)

figure
plot(0.5*(segi+segq))
```

Bibliography

[1] T. S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall PTR, Upper Saddle River, New Jersey, 1996.

[2] United States Federal Communications Commission, "Wireless Telecommunications Bureau – Personal Communications Service," *published at the WWW site http://www.fcc.gov/wtb/pcs/pcssrv.html*, 1997.

[3] H.V. Poor, "Signal Detection in Multiple Access Channels," Tech. Rep., U.S. Army Research Office, 1980.

[4] S. Verdu, "Minimum Probability of Error for Asynchronous Gaussian Multiple Access Channels," *IEEE Transactions on Information Theory*, vol. IT–32, no. 1, pp. 85–96, January 1986.

[5] S. Berruto et al., "Research Activities on UMTS Radio Interface, Network Architectures, and Planning," *IEEE Communications Magazine*, vol. 36, no. 2, pp. 82–95, February 1998.

[6] P. Clarke, "ETSI Pushes for World Mobile Telecom Standard," *EE Times*, February 1998.

[7] P. Clarke, "Europeans Divided on Cellular Standards," *EE Times*, January 1998.

[8] P. Clarke, "A Wideband CDMA Finds Support as Europe's Interface," *EE Times*, December 1997.

[9] Telecommunications Industry Association, "TIA/EIA Interim Standard: Mobile Station-Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular System," Tech. Rep., IS-95A, May 1995.

[10] S. Seidel, *Site-Specific Propagation Prediction for Wireless In-Building Personal Communication System Design*, Ph.D. Dissertation, Virginia Tech, Department of Electrical and Computer Engineering, 1993.

[11] G. Durgin, "Advanced Site-Specific Propagation Prediction Techniques," Master's Thesis, Virginia Tech, Dept. of Electrical and Computer Engineering, 1998.

[12] American National Standards Institute, "Personal Station-Base Station Compatibility Requirements for 1.8–2.0 GHz Code Division Multiple Access (CDMA) Personal Communication Systems," Tech. Rep., ANSI J-STD-008, March 1995.

[13] J.R. Deller, J.G. Proakis, and J.H.L. Hansen, *Discrete Time Processing of Speech Signals*, Macmillan Publishing Company, New York, 1993.

[14] Telecommunications Industry Association, "Recommended Minimum Performance Standards for Base Stations Supporting Dual-Mode Wideband Spread Spectrum Cellular Mobile Stations," Tech. Rep., IS-97A, July 1996.

[15] Telecommunications Industry Association, "Recommended Minimum Performance Standards for Dual-Mode Wideband Spread Spectrum Cellular Mobile Stations," Tech. Rep., IS-98A, July 1996.

[16] R.C. Dixon, *Spread Spectrum Systems*, John Wiley and Sons, New York, 2^{nd} edition, 1984.

[17] Federal Communications Commission, *Code of Federal Regulations, Title 47, Volume 2, Part 22, Subpart H – Cellular Radiotelephone Service*, U.S. Government Printing Office, October 1996.

[18] D.M.J. Devasirvatham, R.R. Murray, H.W. Arnold, and D.C. Cox, "Four-Frequency CW Measurements in Residential Environments for Personal Communications," in *Proceedings of the 1994 Third Annual Conference on Universal Personal Communications*, 1994, pp. 140–144.

[19] D.M.J. Devasirvatham and R.R. Murray, "Time Delay Spread Measurements at Two Frequencies in a Small City," in *Proceedings of the 1995 MILCOM Conference*, 1995, vol. 3, pp. 942–946.

[20] D.M.J. Devasirvatham, R.R. Murray, and D.R. Wolter, "Time Delay Spread Measurements in a Wireless Local Loop Test Bed," in *Proceedings of the 1995 IEEE* 45^{th} *Vehicular Technology Conference*, 1995, vol. 1, pp. 241–245.

[21] M. J. Feuerstein, K. L. Blackard, T. S. Rappaport, S. Y. Seidel, and H. H. Xia, "Path Loss, Delay Spread, and Outage Models as Functions of Antenna Height for Microcellular System Design," *IEEE Transactions on Vehicular Technology*, vol. 43, no. 3, pp. 487–498, August 1994.

[22] J.D. Parsons, *The Mobile Radio Propagation Channel*, John Wiley & Sons, New York, 1994.

[23] L. Couch, *Digital and Analog Communication Systems*, Prentice Hall, Upper Saddle River, 4th edition, 1993.

[24] B. Sklar, "Rayleigh Fading Channels in Mobile Digital Communication Systems Part 1: Characterization," *IEEE Communications Magazine*, vol. 35, no. 9, pp. 136–146, September 1997.

[25] B. Razavi, *RF Microelectronics*, Prentice Hall PTR, Upper Saddle River NJ, 1998.

[26] D. W. Rice and K. H. Wu, "Quadrature Sampling with High Dynamic Range," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-18, no. 4, pp. 736–739, November 1982.

[27] V. Considine, "Digital Complex Sampling," *Electronics Letters*, vol. 19, no. 16, pp. 608–609, August 1983.

[28] R.E. Ziemer and R.L. Peterson, *Introduction to Digital Communications*, Macmillan Publishing Company, New York, 1992.

[29] N. Patwari, "The Analog Tuner of the MPRG Pilot Signal Scanning Receiver," Tech. Rep., Mobile and Portable Radio Research Group, Virginia Tech, Blacksburg, VA, 1998.

[30] W.B. Kuhn, *Design of Integrated, Low Power, Radio Receivers in BiCMOS Technologies*, Ph.D. Dissertation, Virginia Tech, Blacksburg, VA, December 1995.

[31] M.J. Demler, *High-Speed Analog-to-Digital Conversion*, Academic Press, San Diego, 1991.

[32] J. A. Wepman and J. R. Hofman, "RF and IF Digitization in Radio Receivers: Theory, Concepts, and Examples," Tech. Rep. NTIA Report 96-328, National Telecommunications and Information Administration, U.S. Department of Commerce, March 1996.

[33] B.L. Fox, "Analysis and Dynamic Range Enhancement of the Analog-to-Digital Interface in Multimode Radio Receivers," Master's Thesis and MPRG Technical Report MPRG-TR-97-02, Virginia Polytechnic Institute and State University, Dept. of Electrical Engineering, February 1997.

[34] Analog Devices, "Complete 12-Bit 1.25/3.0/10.0 MSPS Monolithic A/D Converters: AD9221/AD9223/AD9220," Tech. Rep., Analog Devices, Norwood, MA, 1996.

[35] Texas Instruments, "TMS320C54x DSP: CPU and Peripherals," Tech. Rep. 1, Texas Instruments, 1996.

[36] Texas Instruments, "TMS320C54x Evaluation Module: Technical Reference," Tech. Rep., Texas Instruments, 1995.

[37] B.W. Parkinson and eds. J.J. Spilker, *Global Positioning System: Theory and Applications, Volume 1*, American Institute of Aeronautics and Astronautics, Washington, D.C., 1996.

[38] B.W. Parkinson and eds. J.J. Spilker, *Global Positioning System: Theory and Applications, Volume 2*, American Institute of Aeronautics and Astronautics, Washington, D.C., 1996.

[39] Rand McNally, *Rand McNally 1992 Commercial Atlas & Marketing Guide,* 123rd *ed.*, 1992.

[40] Federal Communications Commission, *Code of Federal Regulations, Title 47, Volume 2, Part 24, Subpart E – Broadband PCS*, U.S. Government Printing Office, October 1996.

[41] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 2nd edition, 1992.

[42] P. L'Ecuyer, "Efficient and Portable Combined Random Number Generators," *Communications of the ACM*, vol. 31, no. 6, pp. 742–749, 774, June 1988.

[43] S.A. Tretter, *Communication System Design Using DSP Algorithms*, Plenum Press, New York, 1995.

[44] K. Stewart, "Effect of Sample Clock Jitter on IF-Sampling IS-95 Receivers," in *Proceedings of the 1997 Personal, Indoor, and Mobile Radio Communications Conference*, 1997, pp. 266–269.

[45] Vectron Laboratories, "Vectron Crystal Oscillators Handbook and Catalog," Tech. Rep., Vectron Laboratories, 1995.

[46] Z. Kostic and N.A. Abbasi, "Effects of Transmitter, Receiver and Channel Impairments on the Performance of the Enhanced IS-136 Digital Cellular System for Transmission of High-Quality Speech," in *Proceedings of the 1997 Vehicular Technology Conference*, 1997.

[47] ed. W.H. Beyer, *CRC Standard Mathematical Tables and Formulae*, CRC Press, Boca Raton, 29th edition, 1991.

Vita

Keith Blankenship was born on June 6, 1966 in Arlington, VA, and was reared in Manassas, VA. He was graduated from Virginia Commonwealth University in Richmond, VA in 1988 with Bachelor degrees in Mathematics and Music. Subsequently, he was employed at the Institute for Defense Analyses in Alexandria, VA, where he worked on cost-benefit studies and statistical cost modeling for the Office of the Secretary of Defense. In 1990, he entered the graduate school of Virginia Tech for post-baccalaureate studies in physics, winning the Philip Morris fellowship in academic year 1991. He was graduated with the Ph.D. degree in Physics in May 1995, after which he began studies in Electrical Engineering. Keith is a member of the IEEE Communications Society, the IEEE Antennas and Propagation Society, and the Eta Kappa Nu Electrical Engineering honor society. Following receipt of his M.S.E.E., he will be employed by Motorola in Austin, TX working on system design and digital signal processing for advanced and third generation wireless networks.