NAG5-1491

THE KLIPSCH SCHOOL OF

ELECTRICAL AND COMPUTER ENGINEERING

TECHNICAL REPORT SERIES

CARRIER ESTIMATION USING CLASSIC SPECTRAL ESTIMATION TECHNIQUES FOR THE PROPOSED DEMAND ASSIGNMENT MULTIPLE ACCESS SERVICE

Bradley James Scaife, B.S.

NMSU-ECE-99-007 August 1999

CARRIER ESTIMATION USING CLASSIC SPECTRAL ESTIMATION TECHNIQUES FOR THE PROPOSED DEMAND ASSIGNMENT MULTIPLE ACCESS SERVICE

BY

BRADLEY JAMES SCAIFE, B.S.

A Thesis submitted to the Graduate School

in partial fulfillment of the requirements

for the Degree

Master of Science in Electrical Engineering

Major Subject: Electrical Engineering

New Mexico State University

Las Cruces, New Mexico

August 1999

"Carrier Estimation Using Classic Spectral Estimation Techniques for the Proposed Demand Assignment Multiple Access Service," a thesis prepared by Bradley James Scaife in partial fulfillment of the requirements for the degree, Master of Science, has been approved and accepted by the following:

Timothy J. Pettibone
Dean of the Graduate School

Phillip L. De León
Chair of the Examining Committee

July 26, 1999
Date

Committee in charge:

Dr. Phillip L. De León, Chair

Dr. Gerald J. Dunn

Dr. Stephen Horan

Dr. James P. Le Blanc

ACKNOWLEDGMENTS

The author would like to take the opportunity to thank Frank Hartman, Cliff Baxtor, and the entire White Sands Complex Staff for their support and for the opportunity to gather test data in a very busy environment. Additionally, the author wishes to express gratitude to Lawrence Alvarez for invaluable assistance on numerous occasions in setting up hardware and test beds.

The author would like to thank Dr. Paul Klipsch for his incredible service to New Mexico State University, the field of electrical engineering, and to the author personally. It was an honor to be a graduate scholarship recipient but a greater honor to meet Dr. Klipsch personally.

The author would like to express his appreciation and admiration for the professors of the Klipsch School of Electrical Engineering and in particular to Dr. James Le Blanc for excellent instruction and invaluable assistance with this work.

The author would like to take the opportunity to express a lifetime of gratitude for the many lessons taught him by his advisor Dr. Phillip L. De León. His work on this project and support of it as well as his willingness to direct, correct, and instruct with seemingly endless patience has made him more friend than advisor.

A final word of gratitude to my wife Nancy for ever demonstrating faith, hope, and love, but the greatest of these....

VITA

[redacted]—Born in [redacted]

1987—Graduated from Ashland High School, Ashland Oregon

1994—Graduated from Oregon Institute of Technology, Klamath Falls, Oregon

1994-1997—Intel Corporation, Portland, Oregon

1998-1999—Research Assistant, New Mexico State University Manuel Lujan Center for Space Telemetering

1999—Teaching Assistant, Klipsch School of Electrical and Computer Engineering, New Mexico State University

PROFESSIONAL SOCIETIES

Institute of Electrical and Electrical Engineers

PUBLICATIONS

Scaife, Bradley J. 1998. Doppler Shifted Spread Spectrum Carrier Recovery Using Real-Time DSP Techniques. *International Telemetering Conference.*

FIELD OF STUDY

Major Field: Signal Processing and Computer Architecture

ABSTRACT

CARRIER ESTIMATION USING CLASSIC SPECTRAL ESTIMATION TECHNIQUES FOR THE PROPOSED DEMAND ASSIGNMENT MULTIPLE ACCESS SERVICE

BY

BRADLEY JAMES SCAIFE, B.S.

Master of Science in Electrical Engineering

New Mexico State University

Las Cruces, New Mexico, 1999

Dr. Phillip L. De León, Chair

In any satellite communication, the Doppler shift associated with the satellite's position and velocity must be calculated in order to determine the carrier frequency. If the satellite state vector is unknown then some estimate must be formed of the Doppler-shifted carrier frequency. One elementary technique is to examine the signal spectrum and base the estimate on the dominant spectral component. If, however, the carrier is spread (as in most satellite communications) this technique may fail unless the chip rate-to-data rate ratio (processing gain) associated with the carrier is small.

In this case, there may be enough spectral energy to allow peak detection against a noise background.

In this thesis, we present a method to estimate the frequency (without knowledge of the Doppler shift) of a spread-spectrum carrier assuming a small processing gain and binary-phase shift keying (BPSK) modulation. Our method relies on an averaged discrete Fourier transform along with peak detection on spectral match filtered data. We provide theory and simulation results indicating the accuracy of this method. In addition, we will describe an all-digital hardware design based around a Motorola DSP56303 and high-speed A/D which implements this technique in real-time. The hardware design is to be used in NMSU's implementation of NASA's demand assignment, multiple access (DAMA) service.

Contents

List of Figures ix

List of Tables xi

Frequently Used Terminology xii

1 Overview 1

1.1 Introduction . . . 1

1.2 Comparison of Goddard and NMSU's DAMA Proposals . . . 2

1.3 Proposed Solution . . . 3

1.4 Simulations, Test Data, and Theory . . . 4

1.5 DAMA Hardware . . . 6

2 DAMA Project Description 8

2.1 Current WSC Operations . . . 8

2.2 NMSU's Proposal . . . 10

2.3 Spread Spectrum Fundamentals . . . 12

2.4 Operating Parameters Description . . . 16

2.5 Carrier Estimation Problem . . . 18

2.6 Proposed Solution . . . 19

3 Theory and Simulation 24

3.1 Signal Description . . . 24

3.2 Estimation Accuracy Theory . . . 25

3.3 Comparison of Theory to Simulation . . . 29

3.4 Simulation Model Description 31

3.5 Simulation Results 32

3.6 Theoretical/Simulation Data Summary 38

4 WSC Data Collection Experiment **39**

4.1 Motivation of WSC Data Collection Experiment 39

4.2 Data Collection Setup 40

4.3 WSC Collected Data Processing 43

4.4 WSC Data Results Compared to Simulation 44

4.5 Conclusions of WSC Data Collection 46

5 Carrier Estimation Hardware and Software **48**

5.1 Motorola DSP56303EVM Description 48

5.2 Burr Brown 800kHz A/D 51

5.3 A/D Interface Board 51

5.4 Additional Hardware 52

5.5 Software Description for Real-Time Carrier Estimation 53

6 Conclusions and Future Work **56**

A Matlab Simulation Code **57**

B Motorola DSP 56303EVM Code **94**

References **134**

List of Figures

2.1 Illustration of Doppler Shift to DAMA Carrier 9

2.2 DAMA Carrier Placement Against MA Spectrum 12

2.3 TDRSS MA Spectrum at IF from Actual Data 13

2.4 2047 PN Code Implementation 14

2.5 Spreading Effect of Processing Gains 16

2.6 Flowchart for Carrier Estimation Algorithm 23

3.1 Signal Space Diagram for BPSK 25

3.2 Comparison of Theoretical Curve Against Simulation 30

3.3 Family of Curves for Non-SMF Case 33

3.4 Non-SMF Frequency Estimation for PG = 10, SNR = 2 dB 34

3.5 Non-SMF Frequency Estimation for PG = 20, SNR = 2 dB 34

3.6 Non-SMF Frequency Estimation for PG = 100, SNR = 2 dB . . . 35

3.7 Family of Curves for SMF Case 35

3.8 SMF Frequency Estimation for PG = 10, SNR = 2 dB 36

3.9 SMF Frequency Estimation for PG = 40, SNR = 2 dB 37

3.10 SMF Frequency Estimation for PG = 100, SNR = 2 dB 37

4.1 Experiment Setup 40

4.2 Placement of Test Frequencies Near TDRSS Null 42

4.3 Comparison of Simulated Results vs. WSC Captured Data 45

4.4 Carrier Estimation Summary 47

5.1 DSP56300 Core System Block Diagram 49

5.2 Locking Tone Generation 53

List of Tables

1 Test Sets Captured 43

Frequently Used Terminology

- AWGN: Additive White Gaussian Noise
- BPSK: Binary Phase Shift Keying(ed)
- BW: Bandwidth
- CDMA: Code Division Multiple Access
- DAMA: Demand Access Multiple Assignment
- DSO: Digital Storage Oscilloscope
- DSP: Digital Signal Processing/Processor
- MA: Multiple Access
- PG: Processing Gain
- PN: Psuedo-Noise
- PSD: Power Spectral Density
- Rx: Receive
- SMF: Spectral Matched Filter
- SN: Space Network
- SNR: Signal to Noise Ratio

- SS: Spread Spectrum
- TDMA: Time Division Multiple Access
- TDRS: Tracking and Data Relay Satellite
- TDRSS: Tracking and Data Relay Satellite System
- Tx: Transmit
- WSC: NASA's White Sands Complex

1 Overview

1.1 Introduction

In an effort to provide increased access to NASA's Space Network (SN), a Demand Access Multiple Assignment communication scheme has been proposed. Under this scheme, users would have the option to communicate short information packets at low data rates *on demand.* This scheme is driven by the increased ability of modern satellites (spacecraft) to detect error conditions on board the satellite [1]. Under current SN operations, communication services are pre-scheduled and the schedules often have significant delay and are not easily modified. The DAMA service is to be designed such that it operates independently of the current scheduled Multiple Access (MA) service.

The SN consists of geostationary Tracking and Data Relay Satellites (TDRS) that operate as a virtual "radio frequency (RF) mirror" for data transmissions from/to low Earth orbiting (LEO) spacecraft communicating to/from ground stations. These LEO spacecraft are in orbit about the earth and due to their relative motion to a receiving TDRS, a Doppler shift is induced in transmissions. Since the MA service is pre-scheduled, these Doppler shifts may be accounted for allowing the ground station to detect and receive the LEO signal. In the case of the DAMA system, where data transmissions are to be scheduled on demand, the Doppler shift information may be unknown and thus a system must be designed to estimate (to within ± 3 kHz) the carrier of the LEO signals.

1.2 Comparison of Goddard and NMSU's DAMA Proposals

Two independent proposals have been offered to implement the DAMA service. The first has been proposed by NASA/Goddard Space Flight Center (GSF) and the second by New Mexico State University. The Goddard proposal intends to provide continuous tracking of all LEO satellites equipped with DAMA capability. Thus with the state vectors of all of these LEO satellites known, the Doppler shift of the carrier can be computed much like for the MA service. Whereas the Goddard proposal must maintain these state vectors so that the ground station receiver may demodulate, the NMSU proposal forgoes LEO satellite state vector knowledge to simplify the required ground station hardware. In the NMSU proposal only a single element of the TDRS antenna array is used as a global beacon [1]. With a global beacon configuration, even a satellite that is experiencing alignment problems may transmit an emergency message to ground station users utilizing the DAMA communication system. As the state vector of the communicating spacecraft has been given up, the ground station will not know the position of transmitting LEO space vehicle and thus the Doppler shift cannot be accounted for [2]. Previous work has shown that the Doppler shift of a LEO spacecraft and a TDRSS can vary by as much as ± 50 kHz which is outside of the ground station receiver (GSR) tolerance of ± 3 kHz [3].

The fundamental problem to NMSU's proposal lies with estimating the carrier of the transmitted signal to within the tolerance of the GSR. Hardware must be developed that will provide a locking tone, accurate to within the ground station tolerance, in order for the ground station to demodulate. The problem is exacerbated by the nature of the DAMA signal. The DAMA carrier is to employ a Spread Spectrum (SS) scheme. SS signals tend to suppress spectral peaks of carriers and have responses that are spread out over a wider bandwidth. As we shall show below, this makes the proposed solution to carrier estimation more difficult.

1.3 Proposed Solution

The proposed solution to the carrier estimation problem described above is to employ classical Digital Signal Processing (DSP) spectral estimation techniques to estimate the carrier frequency. We shall employ a Discrete Fourier Transform (DFT) to generate magnitude squared spectral data from the received signal. The resolution of the DFT will be set to provide accuracy to within the ground station tolerance. A single iteration of this process will not be sufficient to protect the carrier estimation from noise so we will average the magnitude squared data to limit the effects of noise. This process results in a *periodogram.* Having obtained the periodogram of the received signal, we shall employ a frequency domain matched filter to maximize the Spectrum-to-Noise ratio (SPNR). The frequency domain matched filter will be predetermined based primarily upon the predefined SS frequency characteristics, such as processing gain (PG) and power, of the received signal. The results of the application of the matched filter, like those of the time domain equivalent, provide an optimal solution by enhancing the spectrum prior to searching. We then search the enhanced periodogram for a peak with which we base our carrier estimation. The accuracy of this technique will be shown to depend primarily on the PG of the DAMA carrier and the Signal-to-Noise ratio(SNR).

1.4 Simulations, Test Data, and Theory

At the core of the proof of concept for the proposed solution is a simulation model designed in Matlab. The simulation models the DAMA carrier against additive white Gaussian noise (AWGN), where we use AWGN to effectively model the white noise like spectrum of the MA service [2]. Having approximated the TDRSS channel by its most critical feature(presence of the MA service) we add the DAMA carrier and simulate carrier estimation based upon the approach described in section 1.3. The simulation model provides estimation accuracy as a function of the SNR and the PG of the DAMA carrier. We will show that accurate estimation, to within the ground station tolerance, is achievable 80%-90% of the time for the given DAMA data rates and corresponding spreading rates.

To verify the accuracy of the simulation model, we include the results of an experiment with test data gathered at NASA's White Sands Complex (WSC). In this experiment, actual data vectors were streamed to a ground station transmitter for transmission to a TDRS (in orbit) and sent back to the ground station receiver. The parameters of the experiment were set such that we could observe several key issues with carrier estimation. The data gathered was processed with the proposed algorithm and compared to simulation results. The most significant conclusion of the experiment was that carrier estimation with the collected data was nearly equivalent to results obtained through simulation. The simulation is then recognized to accurately model the actual TDRSS channel.

We have developed a theoretical analysis that leads to a rough approximation for carrier estimation accuracy. The analysis of carrier frequency estimation is based upon use of the DFT, and the description of carrier estimation accuracy as a random variable [4]. From this description and the use of various approximations, we obtain an expression that describes the root mean square error (RMSE) between the actual carrier frequency and the estimate. The result is expressed as a function of SNR, data (chip) rate, and window type and length. Though the approximations break down in low SNR cases, in the higher SNR cases theory agrees with simulation results.

1.5 DAMA Hardware

As described above, we seek to provide the ground station with an accurate carrier frequency based upon our estimation. We perform this estimation through the use of specifically designed hardware. The base of the hardware utilizes Motorola's DSP56303EVM (EVM). The EVM utilizes Motorola's DSP56300 core which is capable of 80 million instructions per second at 80 MHz and has enough available on-chip memory to implement the algorithm described above. As we will be required to sample the incoming signals at rates greater than that allowed by the EVM, we have integrated an 800 kHz 12 bit Burr-Brown ADS7810/19 analog to digital converter (A/D) into the design. To interface the EVM and the A/D requires some additional logic and level translators that are implemented on an additional interface card. The card allows the EVM to control the A/D while allowing samples from the A/D to be passed directly into the memory of the EVM for processing. These three components, excluding some additional analog pre-processing and post-processing equipment, make up the core of the carrier estimation hardware. The hardware is designed to receive signals that have been filtered and frequency shifted to baseband, estimate the carrier, and then provide a locking tone to the GSR. The GSR will use this carrier estimate to demodulate the DAMA carrier.

The hardware has been tested with synthesized waveforms as well as actual waveforms captured during the WSC experiment and performs as designed/required.

2 DAMA Project Description

2.1 Current WSC Operations

TDRSS was originally devised by NASA as an efficient means to control costs associated with providing a ground station for each satellite [5]. The concept of a space network was formed where users could transmit and receive all communications through a common ground station. NASA operates TDRSS as a space network (SN) using it to provide customers with communication access to their Low Earth Orbiting (LEO) spacecraft. The SN consists of six geostationary Tracking and Data Relay Satellites (TDRS) located 22,250 miles in orbit and a ground station located at the WSC (other operational ground stations exist as well) [5]. The function of a TDRS is to act as a virtual "RF mirror" through which communication signals are relayed between user spacecraft and the ground station. An antenna array, located on each TDRS, is tuned by weighting antenna elements to provide a spot beacon to the spacecraft. This requires a unique weighting vector and associated signal processing equipment for each user spacecraft [1]. Two communication schemes are used by the SN to fulfill various communication needs [5] [3]:

- multiple access (MA) at low data rates of 100 bps to 50 kbps operating in the S-band (2.1031 GHz - 2.1097 GHz forward service, 2.2845 GHz - 2.2905 GHz return) and using CDMA spread spectrum with a chip rate of 3 Mchips/s.

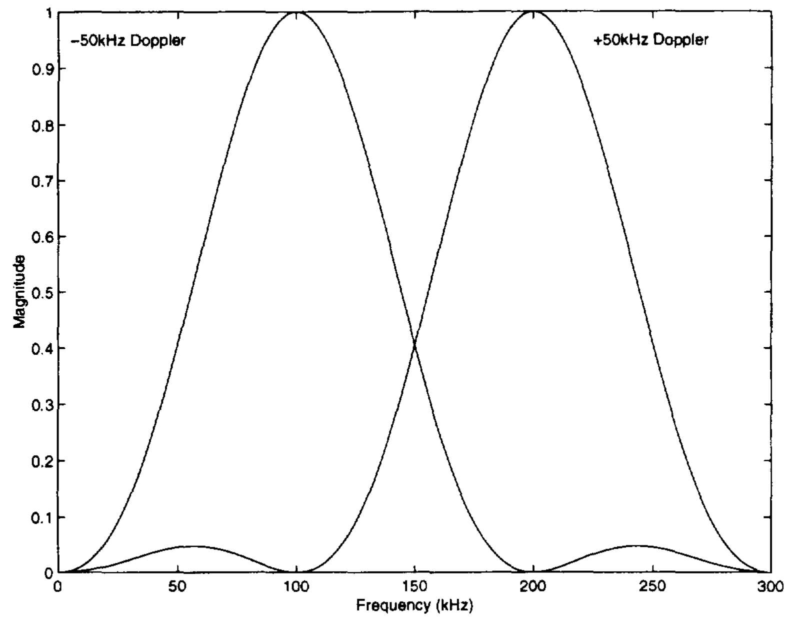

Figure 2.1: Illustration of Doppler Shift to DAMA Carrier

- single access (SA) at up to 300 kbps operating in either the S-band or the K-band (2.0204 GHz - 2.1233 GHz forward service, 2.2 GHz - 2.3GHz return) using TDMA.

The SN is able to provide 80% - 100% coverage for LEO spacecraft and is capable of simultaneously supporting 26 user spacecraft.

TDRSS consists of geostationary satellites but the LEO spacecraft that use this system are not necessarily geostationary. We know that signals originating from a source moving relative to a TDRS will experience a Doppler shift [6]. It has been shown that the Doppler shift of these signals can be as much as ± 50 kHz [3]. This is illustrated in Figure 2.1. The GSR normally maintains the state vector of the satellite it is intended to communicate with and hence can simply calculate an estimate of the Doppler shift. Provided that the estimated Doppler shift is within ± 3 kHz of the actual Doppler shift, the ground station can demodulate the received signal. In any system where one would forgo knowledge of the state vector of these satellites, the result would be that in general the GSR could not synchronize to the Doppler shifted carrier and thus the carrier need be estimated.

The SN currently works under a scheduling process whereby a request for service must be made in advance (prescheduling) to utilize the SN. The scheduling delay often takes as much as 21 days for the request to be processed [3]. While the request can be serviced quicker in emergencies, the delay does not allow customers to react in near real-time to emergency situations that may arise with a user spacecraft. DAMA is a proposal that seeks to provide on demand communications between a user and their spacecraft without the need for prescheduling.

2.2 NMSU's Proposal

The initial scope of the NMSU proposal is towards implementing a "911" service where satellites that have an gone into an error state may communicate this to the user when it detects such a condition. The eventual scope is to provide this service as a standard service to all DAMA capable spacecraft that require only low data rates with small data packets. Additionally it must be expanded to allow *multiple access*—or use by multiple users. For this thesis we assume a single DAMA user at a time. The algorithm to be developed below is scalable to allow for multiple DAMA users at some point in the future.

The NASA GSFC proposal seeks to implement the DAMA service by maintaining the state vector information for each user spacecraft that is DAMA capable by continuously tracking each of these spacecraft. As in the MA service, with the state vector of the spacecraft known, it is routine to estimate the Doppler-shifted carrier and provide this estimate to the GSR. In contrast, NMSU's proposal gives up this state vector knowledge so that the DAMA ground station equipment is simplified. This leads to a problem with the Doppler estimation as it now must be estimated and supplied by means other than from the state vector. We propose a solution to this problem with the algorithm to be developed below that will execute on the hardware that was also developed to provide the GSR with this Doppler estimate. DAMA is to be implemented with a SS BPSK modulated communication scheme like the MA service described above. However, there are certain restrictions that determine the parameters of the scheme. The DAMA carrier is to be placed just inside the first upper TDRSS null as observed in Figure 2.2. The carrier will be placed such that a maximal Doppler shift of +50 kHz will not place the carrier too near the null so that the rolloff of the TDRSS channel and other associated GSR equipment, which bandpass filters on the mainlobe, will not adversely affect carrier estimation. The signal in Figure 2.3 consists of the MA service and the DAMA signal and demonstrates the overall response of

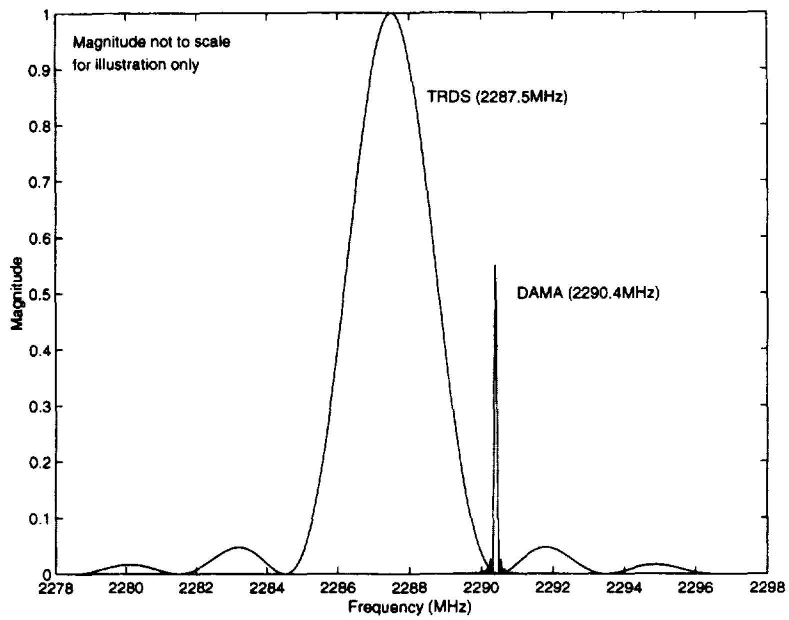

Figure 2.2: DAMA Carrier Placement Against MA Spectrum

the TDRSS system with the addition of the DAMA signal. We furthermore see the effects of sidelobe rejection of the TDRSS system.

2.3 Spread Spectrum Fundamentals

To discuss the operational parameters of the proposed DAMA carrier estimation, it will first be necessary to provide some fundamentals of SS communication schemes and definitions of important parameters. These parameters directly affect carrier estimation performance.

We begin with a basic and widely used definition for SS systems: SS systems are distinguished by the characteristic that their signals consume a bandwidth greater than the information rate [7]. Though there are several different tech-

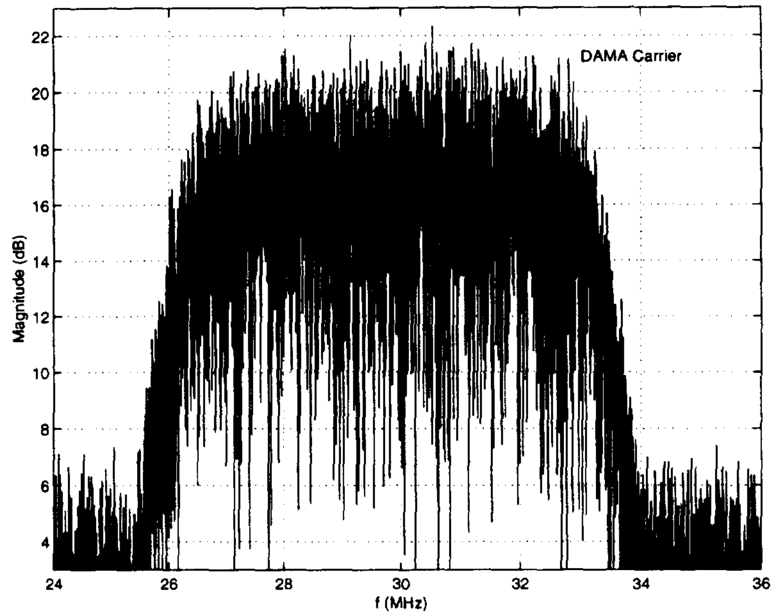

Figure 2.3: TDRSS MA Spectrum at IF from Actual Data

niques for implementing the spreading, we focus on the technique known as spread spectrum by direct sequence (DS). DS spread systems implement a scheme where the information data is acted upon by pseudo-noise (PN) data, whose elements are referred to as chips, to produce a spread spectrum bandwidth (BW). The ratio of chips to bits is typically an integer and the chip rate is often much higher than the data rate. This ratio is defined as the processing gain (PG) where

$$PG = \frac{R_c}{R_b} \tag{2.1}$$

and R_c is the chip rate in chips/s and R_b is the data rate in bits per second (bps). The PG also describes the ratio of chips/bit from which we see that each bit will be acted upon by PG chips through the use of *modulo-2 addition*. The PN code

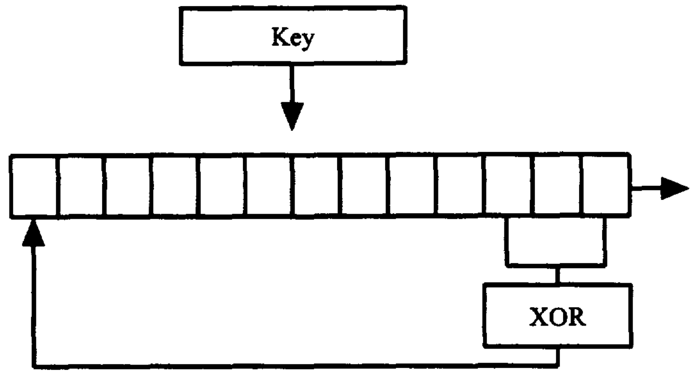

Figure 2.4: 2047 PN Code Implementation

sequence, $\mathbf{C}_i$, has the property that it approximates a white noise sequence and is periodic. $\mathbf{C}_i$ is designed purposely such that

$$\mathbf{C}_i^T \mathbf{C}_j = \delta(i-j) \tag{2.2}$$

indicating orthogonality between PN codes of equal length but different "keys". In practice, PN codes are only approximately orthogonal. The number of 1's and 0's, with $\mathbf{C}_i \in \{0,1\}$, differ by at most one. Many different techniques exist for the generation of these codes and we provide the 2047 PN code as an example. Though the 2047 PN code exhibits the qualities of white noise it is in fact periodic with period 2047. This PN code may be viewed as a primitive polynomial and implemented with a shift register as seen in Figure 2.4.

In general the initial state of the shift register is a "key" and each key represents a different PN code that is orthogonal to other PN codes as described in (2.2). In this manner each PN code operates as an orthogonal basis function for each vector of data. Multiple users are allowed in the same bandwidth precisely because each message is orthogonal to the other.

For the MA service, $R_c = 3$ Mchips/s. From theory it is known that the BW consumed by this modulation scheme follows the relation:

$$BW \propto 2R_c \tag{2.3}$$

where BW is the bandwidth, and R_c is the chip or spreading rate [8]. From (2.3) we observe that the MA service will occupy approximately 6 MHz of BW as is illustrated in Figure 2.2 and Figure 2.3.

We will show that PG plays a large role in carrier estimation but first it is useful to see how PG will affect the BW of a signal. As we "spread" a carrier more and more (increase R_c relative to R_b), the spectrum of the carrier will tend to spread out and flatten. This can be observed in Figure 2.5 below where we have spread a BPSK at various rates. Both PG's in Figure 2.5 are relatively low however, it will be shown that low PG's are required for accurate carrier estimation. Estimation of the carrier becomes more difficult at higher PGs since the carrier power is not concentrated over a small band of frequencies, which would result in a sharp

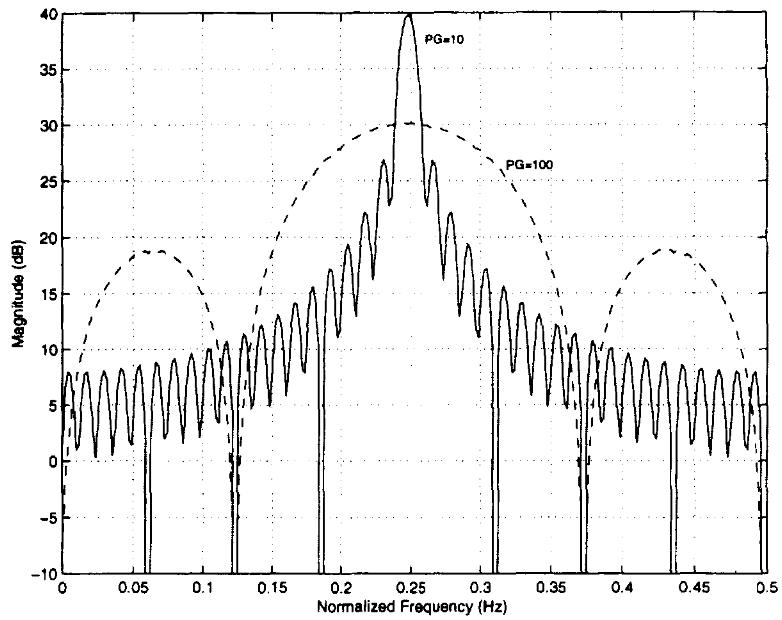

Figure 2.5: Spreading Effect of Processing Gains

spectral peak, but rather distributed over a wider range. The peak in the SS case has less power and therefore noise may bury it (as intended for SS systems).

2.4 Operating Parameters Description

We now describe some of the communication parameters of the DAMA proposal. We are particularly interested in three parameters when dealing with digital communication systems: power, bit rate, and probability of bit error. For DAMA carrier estimation, we are not required to demodulate the signal and thus do not look at the probability of bit error. The first two, however, will affect our ability to to perform carrier estimation.

For the purposes of this thesis, we will describe DAMA power in two different ways. The first will be SNR in dB of the DAMA carrier-to-MA spectrum. We use this SNR definition when observing simulated work since the MA spectrum was modeled as AWGN. We can gain insight into this definition by observing Figure 2.3. The DAMA carrier will exhibit a peak against the *passband* of the MA service. Thus, we are interested in the DAMA power to that flat passband. The second way of describing power will be as the ratio C_b/N_o which is used in WSC operations. This is a measure of DAMA carrier power to the noise floor. Through observations of spectra (under typical conditions) collected at WCS, the mapping between the two power ratios was seen to be approximately

$$C_b/N_o = 45\text{dB} \approx \text{SNR} = 2\text{dB}. \tag{2.4}$$

In this thesis, we shall use C_b/N_o to describe results of actual signals and SNR when describing simulation results. To avoid confusion when comparing the two, we shall map SNR to C_b/N_o.

Since we intend DAMA to be a SS BPSK system, we must not only describe data rates, but also chip rates and therefore PGs. Due to the nature of the DAMA service, it has been proposed that $R_b = 1$ kbps [3]. The PG, as defined in (2.1), will be another parameter that is of primary concern since it determines the BW of the DAMA carrier and impacts the sampling rate. DAMA carrier estimation accuracy will be shown to depend on PG to a large extent and therefore the chip rate will be a matter of investigation.

Sampling rate is another important parameter. It was initially proposed that the PG would be set to $PG = 100$ [3]. With the data rate set as above, this implies a chip rate, $R_c = 100$ kbps. This will exhibit a mainlobe width of 200 kHz by (2.3). From Figure 2.2 we can observe that we will need to account for another 100 kHz due to Doppler shift. The total possible BW of the DAMA carrier is then 300 kHz. We recall that to avoid aliasing while sampling the DAMA carrier, the DAMA signal must be bandlimited and be sampled at

$$f_s \geq 600\text{kHz}. \tag{2.5}$$

A commonly available, inexpensive 800 kHz A/D was found that matched the requirements and we thus chose $f_s = 800$ kHz. This has implementation ramifications that will be discussed below.

2.5 Carrier Estimation Problem

The problem with demodulating the DAMA carrier is the same as that for the MA carrier (though for the MA case the estimate is derived from the state vector of the satellite): the ground station is not capable of demodulating a signal if the error of the estimate is greater than ±3 kHz from the actual. As NMSU's proposal will not keep track of state vector information for each user spacecraft, the carrier must be estimated reliably and efficiently and then passed to the ground station receiver. NMSU's DAMA proposal hinges upon accurate carrier estimation. We have developed an algorithm and hardware that performs this task and we shall describe the algorithm as well as the parameters for operation that will provide accurate carrier estimation. The hardware will provide a locking tone to the ground station receiver.

2.6 Proposed Solution

The proposed solution relies on classical spectral estimation theory with some modifications to improve performance. We begin by assuming that external analog hardware required to bandlimit and frequency shift to baseband the TDRSS and DAMA signals is available. Contained in the 400 kHz band (assuming $f_s = 800$ kHz) will be a portion of the TDRSS signal along with the entire DAMA signal.

We next employ an averaged DFT (implemented with an FFT) which, if mag-squared values are computed, is also known as a periodogram. Since we assume AWGN with zero mean, the averaging has the effect of reducing estimation error variance of the carrier based on the DFT estimate of the DAMA spectrum. We express this as

$$X(k) = \frac{1}{P}\left\{\sum_{p=1}^{P}\left|\sum_{n=0}^{N-1} x[n+pN]\, e^{-\frac{j2\pi kn}{N}}\right|^2\right\} \tag{2.6}$$

where P is the number of blocks in the average, N is the number of points in the block, n is the sample index, and k is the frequency index [9]. To obtain a frequency resolution that will enable us to estimate within the accuracy of the

GSR, we choose

$$N = \frac{f_s}{\Delta f} = \frac{800kHz}{3kHz} \approx 267 \tag{2.7}$$

but this would not allow us to use the *radix-2* based FFT. We alternatively round to $N = 512$ for use with the FFT which yields

$$\Delta f = \frac{f_s}{N} = \frac{800kHz}{512} = 1562.5Hz \tag{2.8}$$

This increases our increases our physical resolution beyond what is actually required.

From the result of the periodogram, we are left with an estimate of the spectrum of the received signal. We estimate the carrier frequency by choosing the maxima of the periodogram. We assume that we are operating the DAMA service such that a peak will be observed in the average. We can improve the *spectrum-to-noise ratio* by utilizing a method from communication theory. It is known that the optimal solution for a receiver corrupted by AWGN is obtained by implementing a matched filter [7]. We will employ the frequency domain equivalent, spectral matched filter (SMF), which like its time domain analog is optimal and will maximize the SNR. In the time domain, a matched filter can be described by its impulse response

$$h(t) = s(T - t) \tag{2.9}$$

with

$$0 \leq t < T$$

where $s(t)$ is the time reversed equivalent of the received signal. The matched filter is then convolved with the received signal yielding

$$y(t) = \int_0^t s(\tau)s(T - t + \tau)d\tau. \tag{2.10}$$

The same approach may be applied in the frequency domain where we have a SMF that is matched to the expected, frequency-reversed Power Spectral Density (PSD) of the DAMA carrier. Through a discrete convolution between the SMF and estimated DAMA spectrum, we may arrive at a desired optimal solution that maximizes the spectrum-to-noise ratio. The SMF is described as

$$H(k) = X(N - k) \quad , 0 \leq k < N \tag{2.11}$$

where $X(N - k)$ is the frequency-reversed equivalent of the PSD of the DAMA carrier. The SMF is then convolved with the spectrum of the received signal as

$$\mathbf{X}_{smf} = \mathbf{X} * \mathbf{H} \tag{2.12}$$

where $*$ indicates the convolution operator. The results of applying the SMF to the process will be described in detail in section three. We then form our carrier estimate with

$$\hat{f} = \frac{\arg\max(X_{smf})}{N} f_s \tag{2.13}$$

where $\hat{f}$ is the estimated frequency of the carrier, N is the number of points of the FFT and f_s is the sampling frequency. The discrete convolution in (2.12) smears the number of points by

$$L_{X_{smf}} = L_H + M \tag{2.14}$$

where $L_{X_{smf}}$ is the translated length of X_{smf}, L_H is the length of the SMF ($N = 512$), and M is the length of the periodogram ($N = 512$). We must subtract the maxima of the SMF, purposely located at $N/2$ - or 256, from the index obtained by $\arg\max(X_{smf})$. The SMF is generated in practice by generating a simulated DAMA carrier (without noise) with random data and averaging 1000 periodograms based on the simulated signal. The simulated signal is set with the PG that we wish to test. A flowchart that describes the algorithm is shown in Figure 2.6.

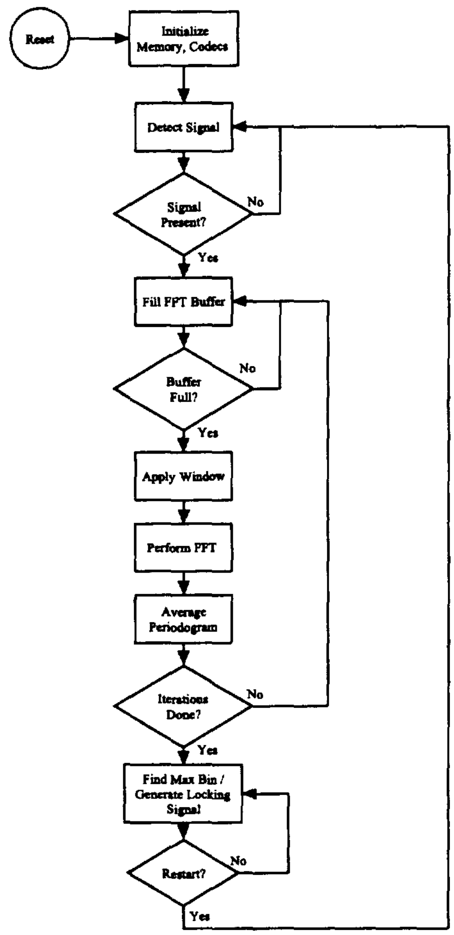

Figure 2.6: Flowchart for Carrier Estimation Algorithm

3 Theory and Simulation

3.1 Signal Description

We have previously stated that the DAMA carrier will use SS BPSK and we begin with a description of the communication scheme. BPSK is a communication scheme where data is encoded into the phase of a sinusoid (carrier). The general equation for *M-ary* PSK is shown in (3.1)

$$s(t) = A\cos(\omega_c t + \psi_m(t)) \tag{3.1}$$

where A is the amplitude of the carrier, ω_c is the carrier frequency and $\psi_m(t)$ is the phase component. In general we may add more power to the sinusoid by increasing A but we assume $A = 1$ for the purposes of this thesis. $\psi_m(t)$ may take on m distinct phases based upon the mapping of data to phase. For the binary case we use two distinct phases. Though we could choose any two distinct phases, in order to minimize error we typically choose the antipodal signaling as seen in Figure 3.1 as it provides the optimal decision boundary [8]. With this configuration, the phase term $\psi(t)$ may take on values corresponding to $\psi(t) \in \{0, \pi\}$. This is equivalent to modulating the sign of the sinusoid such that we may revise (3.1) to

$$s(t) = A\,d(t)\cos(\omega_c t) \tag{3.2}$$

where $d(t) \in \{-1, 1\}$ and is dependent on the mapping of data. As explained in Section 2.3, the data is DS spread by a PN code. For purposes of the analysis that

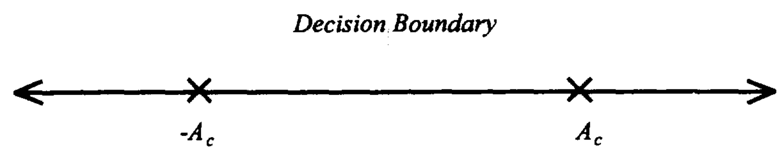

Figure 3.1: Signal Space Diagram for BPSK

follows, we may look at SS BPSK as just BPSK at a higher data rate, namely the chip rate. We make this assumption since we are interested only in searching for a spectral peak of magnitude-squared data and not in demodulating the signal.

3.2 Estimation Accuracy Theory

The following arguments follow closely the work of Boaz Porat in his book, A Course in Digital Signal Processing. We have only extended his arguments to frequency estimation of a BPSK signal [4].

To begin, we make the following assumptions: $s(n)$ is a BPSK signal as described in (3.2), $S(e^{j\omega n})$ is the DTFT of $s(n)$, $w(n)$ is the rectangular windowing function, and $v(n)$ is AWGN of zero mean. The received signal would then be

$$x(n) = s(n)w(n) + v(n)w(n)$$

taking the Fourier Transform of this yields

$$X(e^{j\omega n}) = \sum_{n=0}^{N-1} s(n)w(n)e^{-j\omega n} + \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega n}$$

$$
\begin{aligned}
&= \sum_{n=0}^{N-1} s(n)e^{-j\omega n} + \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega n} \\
&= S(e^{j\omega n}) + \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega n}
\end{aligned}
$$

The PSD of $s(n)$ is given by [8]

$$
P_s(e^{j\omega n}) = E\left[S^2(e^{j\omega n})\right] = \frac{A^2}{4R}\left[sinc\left(\frac{(\omega-\omega_c)\text{osf}}{2\pi}\right)\right]^2 \tag{3.3}
$$

where we have transformed (3.3) from continuous time to discrete time, R is the data rate in bits/sec, and osf is the oversample factor in samples/bit. The maxima of the sinc function as well as the maxima of the BPSK periodogram will occur at $\omega = \omega_c$ hence

$$
X(e^{j\omega n})|_{\omega=\omega_c} = S(e^{j\omega_c}) + \sum_{n=0}^{N-1} v(n)w(n)e^{j\omega_c n} \tag{3.4}
$$

and taking the magnitude-squared of both sides,

$$
\begin{aligned}
|X(\omega_c)|^2 &= \left(S(e^{j\omega_c}) + \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega_c n}\right)\left(S(e^{j\omega_c}) + \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega_c n}\right)^* \\
&= \left(S(e^{j\omega_c}) + \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega_c n}\right)\left(S^*(e^{j\omega_c}) + \sum_{n=0}^{N-1} v(n)w(n)e^{j\omega_c n}\right) \\
&= |S(e^{j\omega_c})|^2 \\
&\quad +2\Re\left[S(e^{j\omega_c})\sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega_c n}\right] \\
&\quad +\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} w(n)v(n)w(m)v(m)e^{-j\omega_c(n-m)}
\end{aligned}
$$

Taking the expectation of the magnitude-squared value yields,

$$
E[|X(\omega_c)|^2] = E\left[|S(e^{j\omega_c})|^2\right]
$$

$$
\begin{aligned}
& +E\left[2\Re\left[S(e^{j\omega_c})\sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega_c n}\right]\right] \\
& +E\left[\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} w(n)v(n)w(m)v(m)e^{-j\omega_c(n-m)}\right] \\
= \; & P_s(e^{j\omega n}) \\
& +2\Re\left[S(e^{j\omega_c})\sum_{n=0}^{N-1} E[v(n)]w(n)e^{-j\omega_c n}\right] \\
& +\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} w(n)w(m)E[v(n)v(m)]e^{-j\omega_c(n-m)}
\end{aligned}
$$

The first term is by definition as provided in (3.3) and is evaluated as

$$
P_s(e^{j\omega})|_{\omega=\omega_c} = \frac{A^2}{4R}
$$

The middle term evaluates to 0 since we have assumed $E[v(n)] = 0$. We further assume $E[v(n)v(m)] = \gamma_\nu\delta(n-m)$, then

$$
\begin{aligned}
E[|X(\omega_c)|^2] &= \frac{A^2}{4R} + \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} w(n)w(m)\gamma_\nu\delta(n-m)e^{-j\omega_c(n-m)} \\
&= \frac{A^2}{4R} + \gamma_\nu\sum_{n=0}^{N-1} w^2(n) \\
&= \frac{A^2}{4R} + \gamma_\nu N
\end{aligned}
$$

We next define an output SNR using (3.7) as the ratio between

$$
SNR_O = \frac{\frac{A^2}{4R}}{\gamma_\nu N} \tag{3.5}
$$

We define the input SNR as

$$
SNR_I = \frac{\frac{A^2}{2}}{\gamma_\nu} \tag{3.6}
$$

where $A^2/2$ is the average power of a BPSK signal [8]. We delineate between output and input SNR to account for the application of the window function. Finally we define a window processing gain as

$$\begin{aligned} W_g &= \frac{SNR_O}{SNR_I \frac{1}{2} N} && (3.7) \\ &= \frac{2}{N}\left[\frac{A^2}{4R\gamma_\nu N}\frac{\gamma_\nu}{\frac{A^2}{2}}\right] && (3.8) \\ &= \frac{1}{RN^2} && (3.9) \end{aligned}$$

We next employ a rule of thumb which states

$$\frac{NA^2W_g}{\gamma_\nu} \geq 100 \tag{3.10}$$

which is given in Porat as a requirement for the reliable detection and frequency estimation of a real sinusoid in noise [4]. We modify (3.10) for the BPSK case in terms of (3.5), (3.6), and (3.9) which yields

$$\frac{A^2}{RN\gamma_\nu} \geq 100 \tag{3.11}$$

In the development provided by Porat, the mean square error is then given as

$$E[\hat{f}_c - f_c]^2 \approx \frac{24N_oJ_w}{(2\pi)^2A^2D^3} \tag{3.12}$$

where N_o is the power density of the noise, J_w is a window parameter [4], D is the measurement interval ($D = NT_s$, where T_s is the sampling period), and A is the amplitude of the sinusoid. We further make the assumption that $E[\hat{f}_c - f_c] = 0$.

With this assumption and provided that (3.10) is true, the approximation in (3.12) is valid [4]. The approximation given in (3.12) can now be modified to yield

$$E[\hat{f}_c - f_c]^2 \approx \frac{24N_oJ_w}{(2\pi)^2\frac{A^2}{4R}D^3} \tag{3.13}$$

where we have applied (3.3) evaluated at $\omega = \omega_c$. The result of (3.13) is an approximation of the mean square error (MSE) of the BPSK carrier frequency estimate. We note that like the sinusoid in noise case, the approximation in (3.13) is only valid when we ensure that (3.11) is true.

Equation (3.13) provides an approximation for the expected value of the MSE frequency estimate. We can use it to provide an approximation of the error in estimated frequency $\hat{f}_c$ from a true frequency f_c from using a DFT approach to estimation. It is recognized that with the many assumptions and the inclusion of a rule of thumb that (3.13) can only give a rough approximation to the actual MSE of the frequency estimation [4].

3.3 Comparison of Theory to Simulation

Having obtained the MSE, we now show how (3.13) compares to simulation data. In the simulation we modeled a BPSK signal with a carrier frequency, f_c, of 178 kHz (typical of DAMA) and the sampling frequency, f_s, set to 800 kHz (as in DAMA). The simulation performed an $N = 512$ point DFT on the BPSK signal and measured the MSE of the estimated frequency, $\hat{f}_c$, to the actual frequency f_c. The results of the simulation are shown in Figure 3.2 where we have shown the

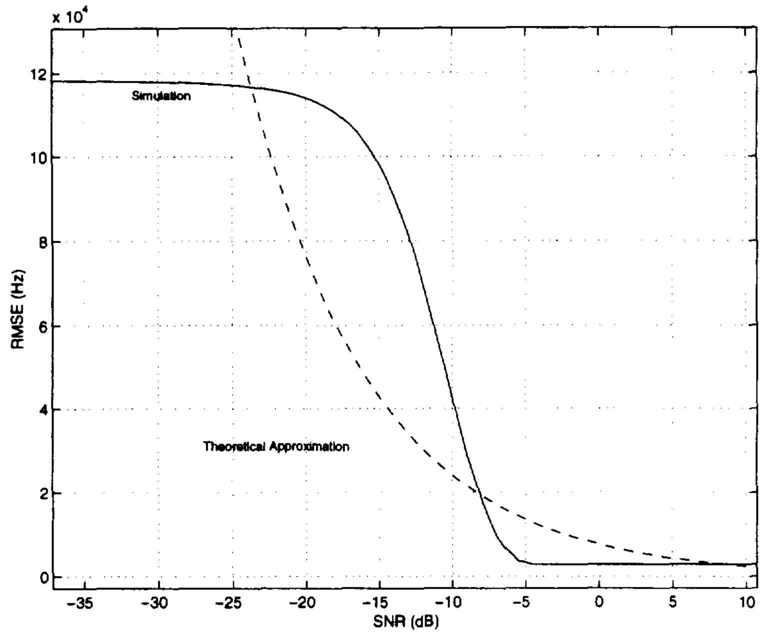

Figure 3.2: Comparison of Theoretical Curve Against Simulation

simulation results versus the approximation developed in (3.13). The root mean square error (RMSE) is simply the square root of (3.13) and it is the expected value in Hz that $\hat{f}_c$ is from f_c as a function of SNR. This relates directly to the accuracy of estimating f_c with $\hat{f}_c$ to within the GSR tolerance and provides a means by which we can approximately determine the accuracy for a given SNR.

We can conclude from the simulation that at high SNR's (< 5 dB), the theoretical curve and the simulation curve match, while at low SNR's the theoretical curve becomes invalid (due to assumptions made during calculation) [4]. We further conclude that at high SNR's (3.13) will provide a good estimate of the mean of the estimation accuracy and allow us to predict performance. At low

SNR's the simulation curve approaches a condition where the noise is dominating the spectrum and the maximum peak is uniformly distributed over the 400 kHz band, yielding near random estimates. The theoretical curve is asymptotic to zero RMSE whereas the simulation curve will actually converge only to the difference between the true frequency and the selected DFT frequency—the difference is unlikely to be zero. This illustrates an important point regarding accuracy of carrier estimation: since we quantize to the DFT frequency points, there will most likely be an error irrespective of what SNR we are operating at. The maximum amount of this error assuming large SNR is given by

$$\max(f_c - \hat{f}_c) < \frac{f_s}{N} \tag{3.14}$$

This error can only be reduced by increasing N, the number of points of the DFT.

3.4 Simulation Model Description

Though we have developed a theoretical analysis culminating in (3.13), the approximation is not tight enough to provide proof of concept. We turn to a Matlab simulation model to provide a core proof of concept. We shall use this simulation model to explore the operational parameters of the algorithm. The simulation models the DAMA carrier against other MA users and AWGN. The simulation builds up a SS BPSK digital waveform based on the PG that we wish to test and then adds appropriate AWGN. It then estimates the doppler-shifted carrier frequency based upon the algorithm developed in section two. A record is kept of carrier frequency estimations and those that fall within the tolerance of the ground station at WSC are counted as an accurate estimation. Likewise any estimation that is outside of the ground station tolerance is considered inaccurate. By performing the simulation 10,000 times for each SNR and desired PG, we obtain a plot that describes estimation accuracy.

The simulation was originally written without the SMF and then rewritten to include the SMF process. The SMF improves estimation accuracy and allows for higher PG's. The Matlab code for the simulation is given in Appendix A.

3.5 Simulation Results

We now provide results of simulation both in the SMF and non-SMF cases. The simulation results for the non-SMF are shown in Figure 3.3 and illustrate estimation accuracy as a ratio of accurate estimations to total estimations versus SNR. In Figure 3.3, we plot three curves for PG = 10, 20, and 100. The curves are generated through 10000 estimates per SNR and we perform 8 DFT blocks of $N = 512$ points. At an SNR = 2 dB, the results of Figure 3.3 demonstrate that estimation accuracy would be approximately 14%, 60%, and 88% for PG = 10, 20, and 100 respectively. NASA has placed the requirement that any implementation

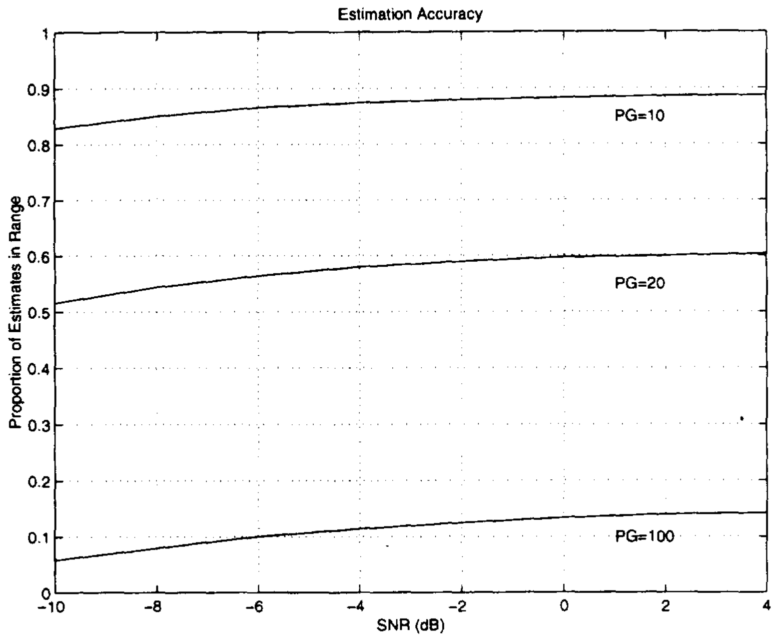

Figure 3.3: Family of Curves for Non-SMF Case

should have as high a PG as possible. From Figure 3.3 it is clear that only the PG = 10 case is practical.

Carrier frequency estimation may also be observed through the use of histograms which also relate information regarding the variance of the estimations. In Figure 3.4, Figure 3.5, and Figure 3.6 we show histograms for PG = 10, 20, and 100 respectively. Each is shown for SNR = 2. In each figure the actual carrier frequency is denoted by the center dashed line while the GSR tolerance (± 3 kHz) is shown by the outer dashed lines.

The results with the SMF enhancement, as described in section two, are much better (Figure 3.7). The results shown in Figure 3.7 are generated as in the

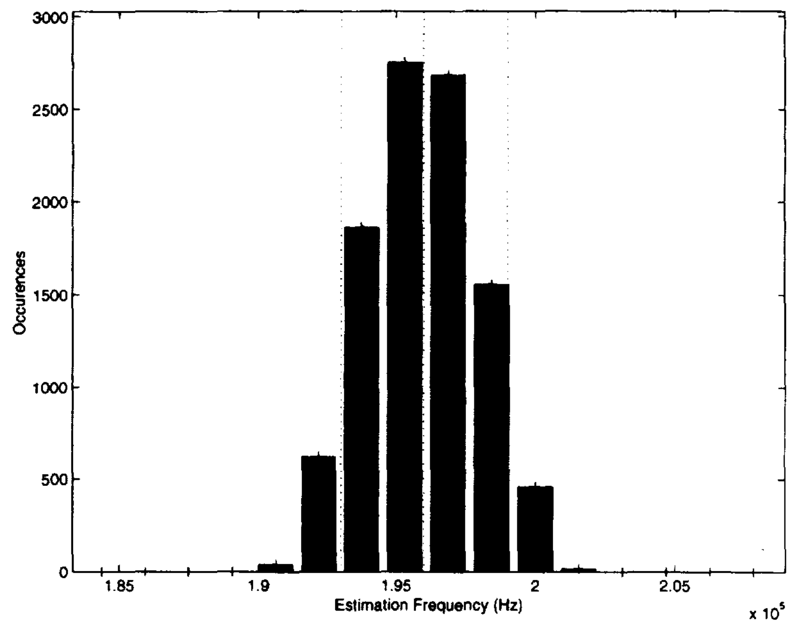

Figure 3.4: Non-SMF Frequency Estimation for PG = 10, SNR = 2 dB

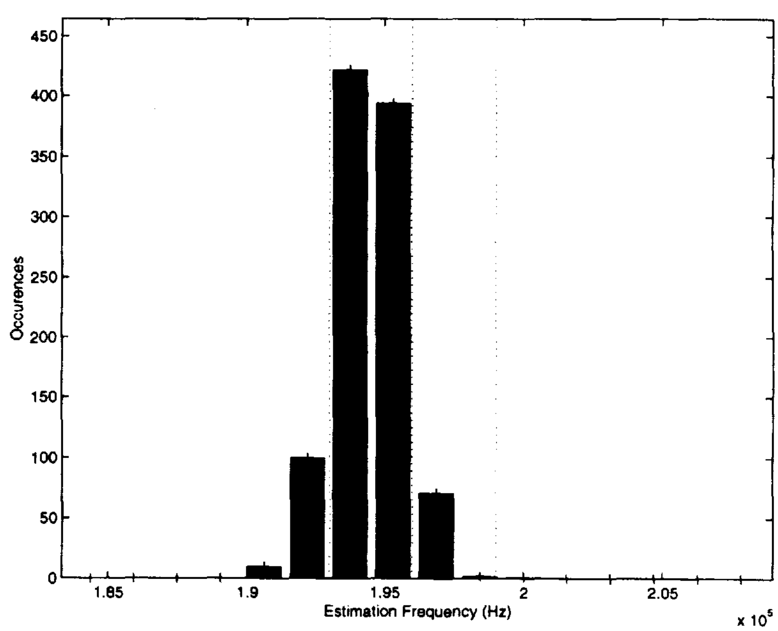

Figure 3.5: Non-SMF Frequency Estimation for PG = 20, SNR = 2 dB

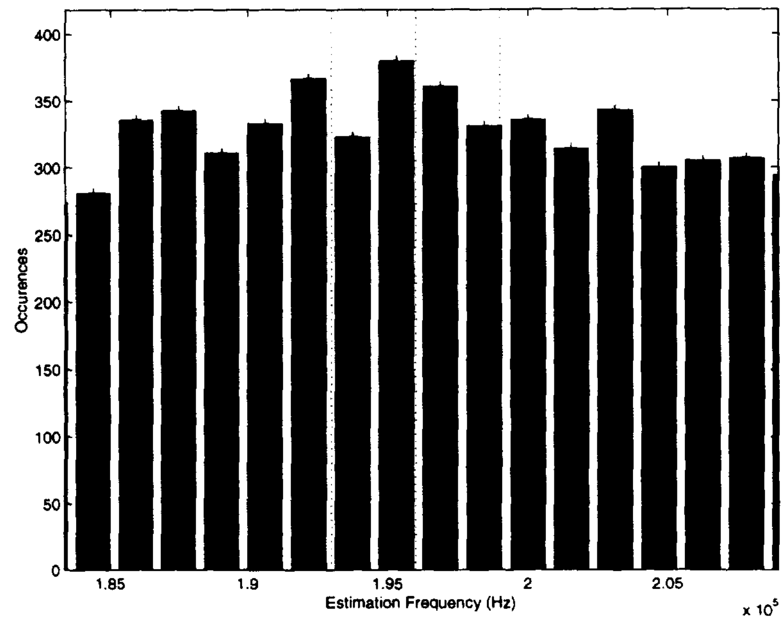

Figure 3.6: Non-SMF Frequency Estimation for PG = 100, SNR = 2 dB

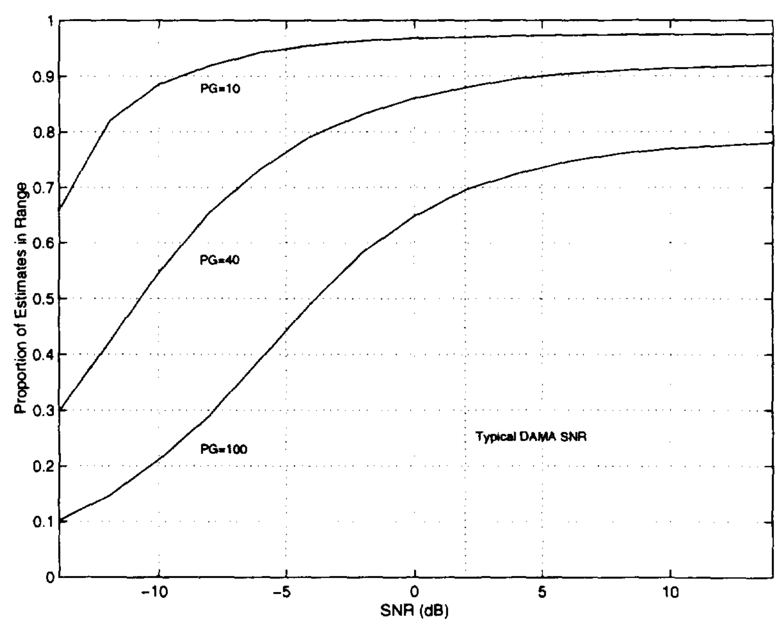

Figure 3.7: Family of Curves for SMF Case

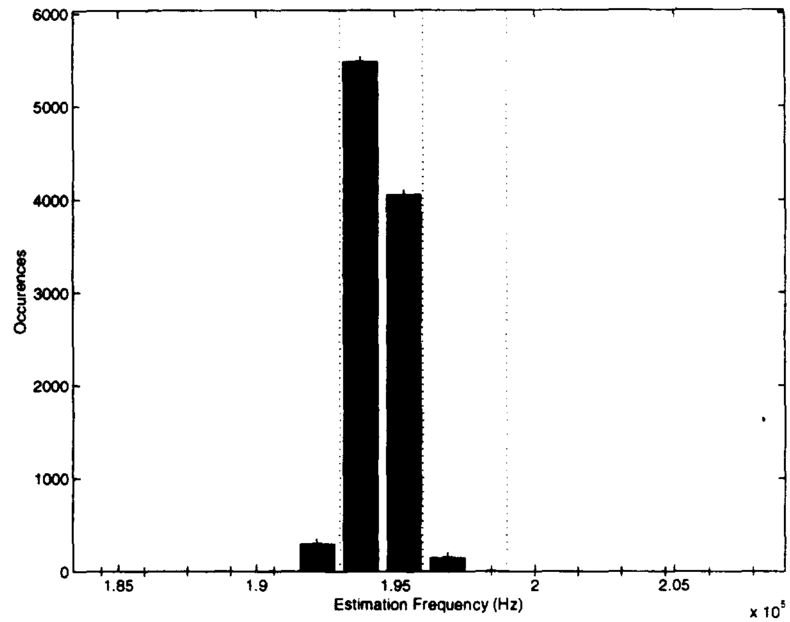

Figure 3.8: SMF Frequency Estimation for PG = 10, SNR = 2 dB

non-SMF case with the exception that we have now included SMF processing. In Figure 3.7, we show estimation accuracy curves for PG = 10, 40, and 100. Here we see that estimation accuracy of the $PG = 100$ case has improved seven-fold. The case of $PG = 40$, a likely DAMA operating parameter, demonstrates accuracy at nearly 90% in a typical operating SNR region. As in the non-SMF case, we observe the histograms of Figure 3.8, Figure 3.9, and Figure 3.10 which are shown for PG = 10, 40, and 100 respectively. Each figure is shown at SNR = 2. From these figures it is clear that there is less variance and therefore more estimations within the GSR tolerance.

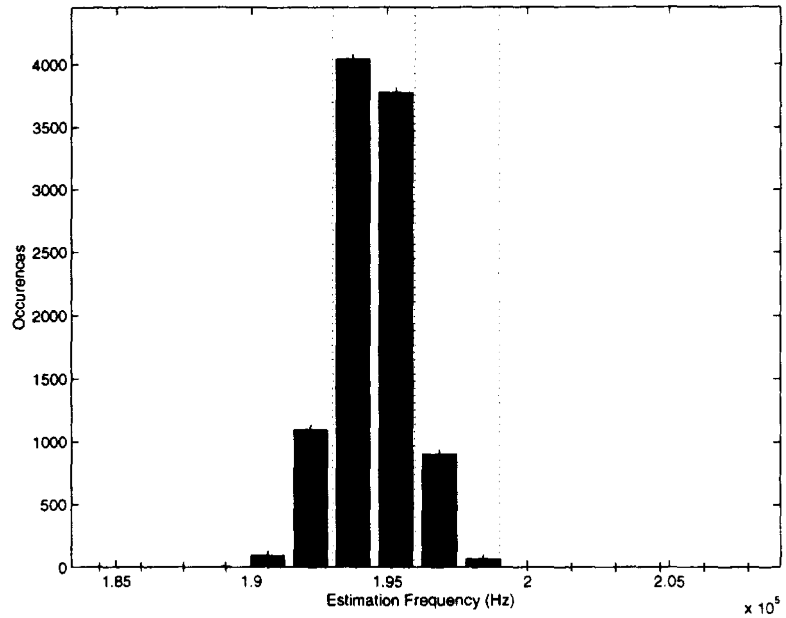

Figure 3.9: SMF Frequency Estimation for PG = 40, SNR = 2 dB

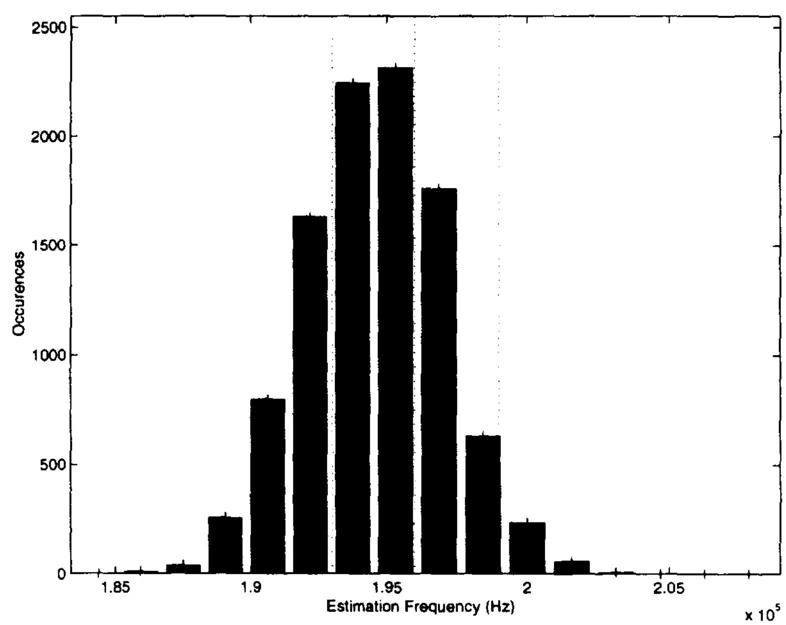

Figure 3.10: SMF Frequency Estimation for PG = 100, SNR = 2 dB

3.6 Theoretical/Simulation Data Summary

We have demonstrated results for a model that attempts to simulate DAMA carrier estimation in the presence of the TDRSS spectrum which is modeled with AWGN. We provide results indicating that accurate carrier estimation is possible with the algorithm developed in section two. Additionally, we have determined some operating points for the PG parameter. We have shown that we can accurately estimate the DAMA carrier 90% of the time with a $PG = 40$ at the typical SNR range. The simulation has provided a demonstration and we offer a loose theoretical approximation (3.13) to the estimation accuracy. In section four, we shall use data collected at NASA's White Sands Complex (WSC) to further validate the simulation results.

Appendix A includes all of the developed code for the simulation model including instructions for its use. Additional code is provided to perform data visualization.

4 WSC Data Collection Experiment

4.1 Motivation of WSC Data Collection Experiment

An experiment was devised to perform validation of the simulation model by collecting actual signals transmitted through TDRSS and collected at WSC. We subsequently processed them offline using the estimation algorithm. The fundamental idea of this experiment is that we can observe the performance of the algorithm with actual DAMA signals. If the algorithm using actual signal data performs similar to the simulation under various PGs and at various SNRs, we may state with a degree of certainty that the simulation model is indeed accurate enough to predict carrier estimation. This also allows for the prediction of the performance of the algorithm if any of the operating parameters need to be changed. A side benefit is that we may also perform *in-house* testing using simulated signals without the expense of interrupting critical TDRSS operations for testing.

This section discusses the actual experiment performed at WSC, including details regarding setup, the processing of the data, and the conclusions drawn. We verify the simulation model's results with actual TDRSS data. Additionally, the experiment points out the realistic operating boundaries which are an important part of the DAMA design.

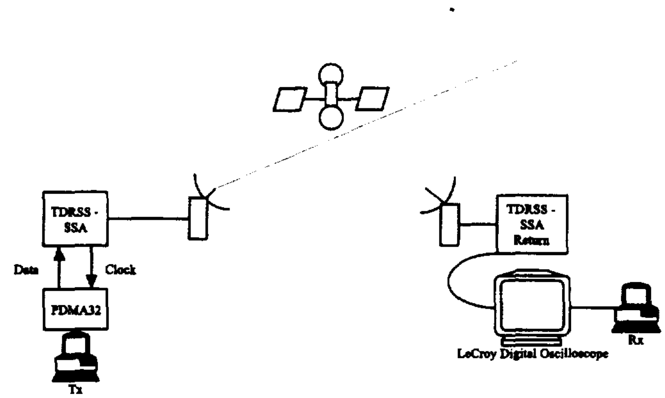

Figure 4.1: Experiment Setup

4.2 Data Collection Setup

Two sets of equipment were used to collect test signals that were sent from the ground station and relayed off of a TDRS back to the ground station. The setup was as shown in Figure 4.1. The Transmit (Tx) equipment consisted of a computer equipped with a high speed PDMA32 Data Transfer Card (PDMA32). The purpose of the Tx equipment was to send a data vector to ground station equipment where it was BPSK modulated and transmitted to a TDRS. The data vector sent consisted of underlying data bits of $\{\pm 1\}$ with $R_b = 1$ kbps which was then spread by a spreading vector at a rate of $R_c = PG \cdot R_b$ chips/s. Several of these data vectors, consisting of raw binary data and DS spread with a PN code (see section two), were generated in advance of the experiment and then used as a data source. The source code (wsands.m, wsands2.m, wsands3.m, and wsands4.m) is given in Appendix A.

The Receive (Rx) equipment consisted mainly of a LeCroy Digital Storage Oscilloscope (DSO) to capture IF signals and another computer to store captured signals. As the TDRSS channel is bandlimited to approximately 40 MHz, the DSO was set to sample at a rate of 100 MHz. This was the closest value to the Nyquist rate that the DSO was capable of sampling at. The DSO was capable of storing the captured data waveforms with either 50,000 samples or 100,000 samples depending on the storage medium that was used and both sizes were collected.

With the hardware setup described, we now turn out attention to the test set. The test set was established to test a variety of key parameters and determine the operational bounds of each. The key parameters are

- Processing Gain
- DAMA Carrier power-to-noise ratio (C_b/N_o)
- Placement of IF Carrier Frequency against TDRSS spectrum

The PG, defined in (2.1), is the most critical of the three. Initial work with the PG indicated a PG of 100 could be used [3]. Subsequent simulations indicated that at this PG value the estimation accuracy was not reliable and lower values were investigated (see section three). It was included in the test set for completeness. PG's of 10, 20, 40, and 100 *chips per bit* were the focus of the test set.

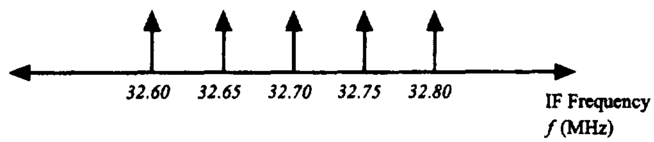

Figure 4.2: Placement of Test Frequencies Near TDRSS Null

Three different carrier power-to-noise ratios were investigated. These ratios were 40, 45, 50 dB. These values were chosen as typical values based upon the experience of WSC staff [10]. It should be noted again that these values are described as a ratio of the carrier power to the noise floor in dB and not carrier power to MA spectrum as described in section three.

The last test parameter is that of the carrier placement. The carrier frequencies were chosen such that a range of frequencies near the TDRSS null could be examined. Recall from section two that we intend to place the DAMA carrier at 2.29 GHz which at IF is 2.9 MHz above the TDRSS center frequency. The range of test frequencies are observed in Figure 4.2. By examining a range of IF frequencies, we are allowed to effectively simulate a Doppler shift as well as examine the effects of the rolloff of the TDRSS channel. Comparing this to the TDRSS spectrum shown in Figure 2.2, we see that the upper range will be affected by the TDRSS rolloff. The full test set is shown in Table 1. The full test set had to be reduced for logistical reasons to lessen impact on TDRSS operations, however, the reduced test set provides enough insight for simulation verification.

Table 1: Test Sets Captured

		C_b/N_o		
PG	F_c *MHz*	40 *dB*	45 *dB*	50 *dB*
10	32.60		*	
	32.65	*	*	*
	32.70	*	*	
	32.80		*	
20	32.60			
	32.65	*	*	*
	32.70		*	
	32.80			
100	32.60			
	32.65		*	
	32.70			
	32.80			
* indicates data collected				

4.3 WSC Collected Data Processing

We noted above that the captured waveforms were sampled at 100 MHz and stored as either 50,000 or 100,000 length vectors. In section two it was also noted that the bandwidth of the DAMA carrier plus maximum possible Doppler shift yielded a 300 kHz frequency search space and that the FFT resolution will be 1562.5 *Hz*. Downsampling the captured waveforms to $f_s = 800\ kHz$ was impractical due to the problems of designing a narrow band decimation filter sharp enough for the rate conversion. Instead we seek a solution that will allow us to perform a frequency search with a nearly equivalent FFT resolution. We see then that

$$\Delta f_{experiment} = \frac{f_s}{N} = \frac{100MHz}{65536} = 1525.8Hz \tag{4.1}$$

provides a nearly equivalent FFT resolution to (2.8). In order to provide the resolution in (4.1) and maximize the use of the short data sets, the vectors were either *overlapped* in the 100,000 sample case or *zero-padded* in the 50,000 length case. Without zero-padding the 50,000 length vector to 65536, the DFT does not meet the required resolution. The net result of this operation is a collection of 65536 length vectors with which to work with. These vectors were then used in a periodogram of eight data blocks chosen at random and processed with the simulation code that was used with a synthetic waveform (see Appendix A). This process was iterated 500 times, due to maximum data support, to form simulation bounds using WSC data.

4.4 WSC Data Results Compared to Simulation

The results of the processed data in comparison to simulation results are shown in Figure 4.3 below. The plot shown is for $PG = 10$ with the simulation represented by the curve and the individual data points being the collected data across the various frequencies indicated. The plot indicates successful carrier estimation to within ± 3 kHz as the ratio of accurate estimates to total estimates versus C_b/N_o. We can make several observations from the plot.

The first observation we can make is in regards to carrier power C_b/N_o. We observe that at a $C_b/N_o = 40\ dB$, the estimation accuracy is much less than the simulation exhibits indicating a practical limit to C_b/N_o. As C_b/N_o increases to 45 and 50 C_b/N_o, estimates performed with actual WSC signal data match

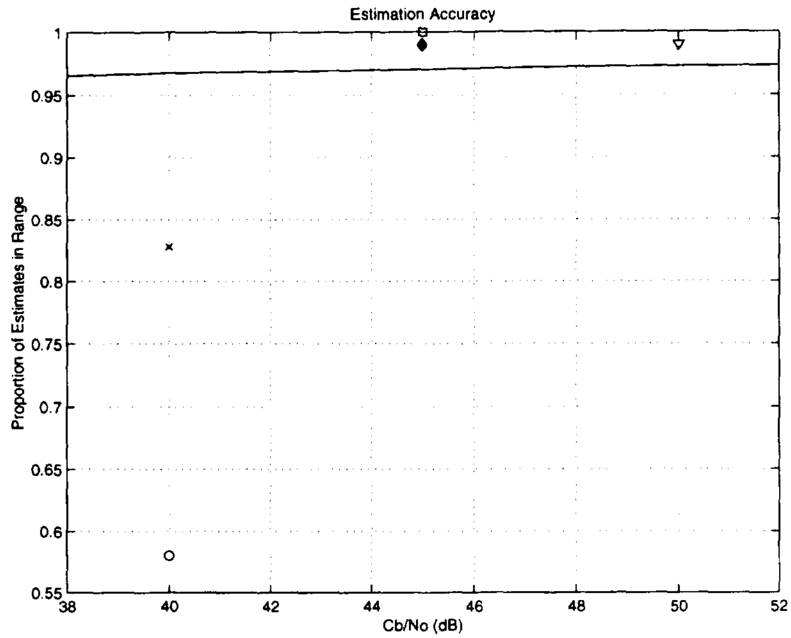

Figure 4.3: Comparison of Simulated Results vs. WSC Captured Data

those using synthetic data. This result indicates that we can estimate Doppler accurately provided that the carrier power is high enough for the range of carrier frequencies specified in Figure 4.2. This is an intuitive result as well since we would fully expect that a carrier with higher power will exhibit a more defined peak against the TDRSS spectrum. We also observe that the nominal carrier power provided by WSC will be sufficient as an operational bound. Previous work indicated that the DAMA carrier power and the MA carrier power can be relatively the same [3]. For the algorithm developed however we must have enough power so that the DAMA carrier exhibits a peak against the MA spectrum. Based on collected data for the $C_b/N_o = 40$ case, the DAMA carrier is lost in the MA spectrum and in general cannot meet reliability requirements.

4.5 Conclusions of WSC Data Collection

We have shown some important results with the WSC Data Collection analysis. The algorithm using actual TDRSS signals performs nearly the same as when using synthesized signals (see Figure 4.4). Therefore the synthesis of modeled signals is accurate to actual TDRSS signals. We rely heavily on the simulation to provide proof of concept and so the justification of the simulation is vital: the simulation is used to narrow in on the operating parameters that we should like to run at. Though we only have significant data for the $PG = 10$ case, we extend the results described above to higher PG's with the assumption that the effects of spreading the signal even further in the frequency domain will cause neglible disparities between the simulation and actual operation. We can use simulated waveforms to implement additional, and important, real-time hardware tests.

There remains one outstanding issue with regards to accurate carrier estimation. We have seen previously that the TDRSS channel begins to rolloff sharply near the location that we would like to place the DAMA carrier. Though the data did not support the investigation—at least in the spectral matched filter case—it is nevertheless true that if the DAMA carrier is nominally placed too close to the

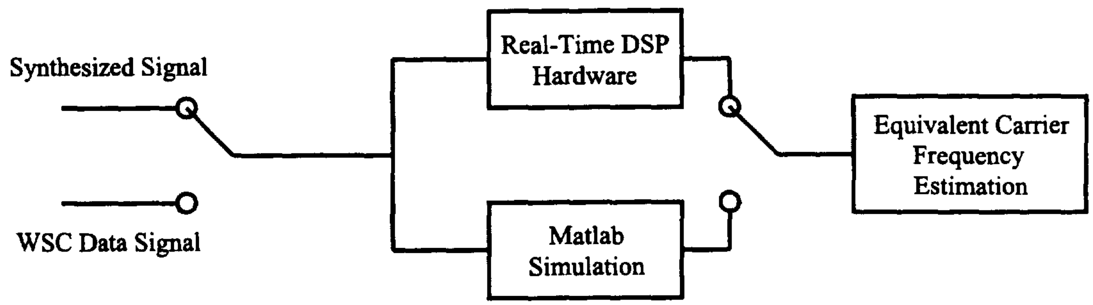

Figure 4.4: Carrier Estimation Summary

null, then the rolloff will adversely affect accuracy. We must also avoid placing the DAMA carrier in the MA passband.

Overall the successful comparison of captured data to simulation data indicates that provided the DAMA carrier has sufficient power, we can explore additional operation bounds with a high degree of confidence.

5 Carrier Estimation Hardware and Software

5.1 Motorola DSP56303EVM Description

The *DSP56303EVM*(EVM) is the core hardware component of the carrier estimation hardware developed. It executes the DSP specific real-time assembly language version of the carrier estimation algorithm (see Appendix B). The EVM is an evaluation module which is designed to be used in prototyping applications. As such, it offers a variety of interface options and configuration settings that make it adaptable to many different types of development projects. The board contains a DSP56303 24-bit digital signal processor that executes the assembly code routines. It also features a DSP56002 specifically for use in I/O functions through a JTAG/OnCE port. JTAG is a protocol that was developed to allow hardware and software developers to observe and manipulate hardware for troubleshooting purposes. The JTAG port for the EVM is primarily used with a PC and debugger software that allows a developer to load code into the EVM, single step through sections of code, and observe the contents of memory and registers. The onboard DSP56002 is not available for coprocessing code, but rather is used to control the I/O functions to the host PC. The EVM also contains 32K x 24-bit Fast Static Ram (SRAM) built with three banks of 32k x 8 bit SRAMS with 15 ns access times. An additional 64k x 8 bit Flash Programmable Erasable Read-Only Memory (PEROM) is provided for stand alone operation. The EVM also contains

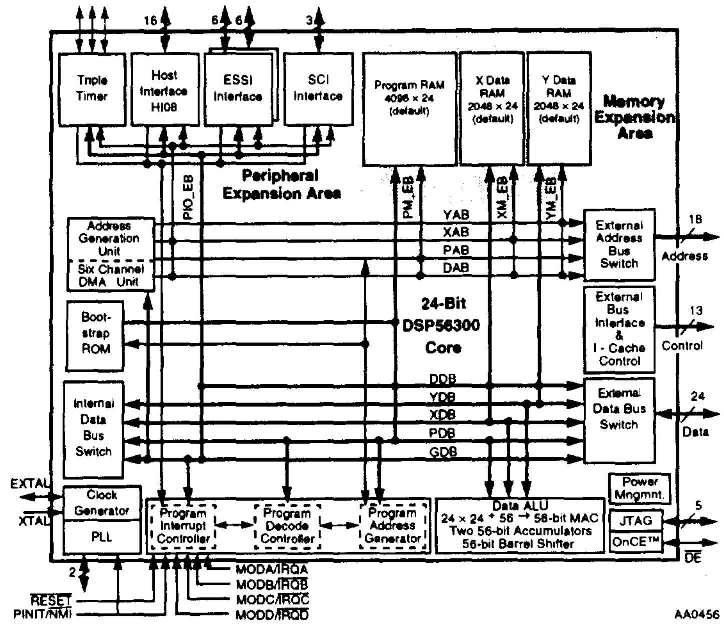

Figure 5.1: DSP56300 Core System Block Diagram

a 16 bit CODEC for sampling incoming analog waveforms and producing analog waveforms out from digital data [11].

The EVM is based upon the DSP56300 core which is shown in Figure 5.1. The EVM specifically uses the DSP56303 processor. The processor is capable of 80 MIPS with an 80 MHz clock. It provides for backwards compatibility with 56k core code so that code written for earlier processors should function equally well on the newer cores. Due to a seven level instruction pipeline architecture, the DSP56303 is capable of effectively an instruction every clock cycle. It is based on the *Harvard* architecture so that it works with a dual memory structure. This architecture is particularly suited for parallel move instructions. This allows the programmer to access two different memory structures at a time. For more on the DSP56303 please see [12].

The DSP56303 not only provides the processing power required to perform carrier estimation efficiently, but it is highly configurable and can interface to a large number of peripheral products. Though higher level languages (C) may be used to program the processor, it is most efficient when programmed in assembly. The assembly language that the DSP56303 uses is specialized to perform DSP tasks and includes features for that purpose. The Real-Time version of the carrier estimation algorithm is developed completely in assembly (see Appendix B).

To create a program for the DSP56303, code is written in assembly language, and then assembled through the use of Motorola software tools to machine usable object code (.cld files). We may then use a software tool like Domain Technologies Debug-EVM to load the code into program memory of the DSP56303. With the Debugger software, we can utilize the JTAG port to single step through code and observe the results on calculations.

For the DAMA project, as was described in section two, we are required to sample at a higher rate than the EVM can perform with the on board CODEC. We will use the on board CODEC in the generation of the locking tone to the ground station receiver. To sample the DAMA spectrum, we need to interface a high speed A/D.

5.2 Burr Brown $800kHz$ A/D

Earlier we noted we must search over a 300 kHz BW (200 kHz for the DAMA mainlobe, assuming no more than a $PG = 100$, and with ± 50 kHz for maximum Doppler shift) which was revised from an earlier estimate of 364 kHz. The Burr Brown 12 bit 800 kHz ADS7810/19 (BB A/D) was chosen since it has a sampling frequency of 800 kHz which gives a Nyquist interval of 400 kHz and because it is available in a convenient evaluation package for easy interfacing to the DSP56303EVM. The 12-bit samples of the BB A/D have a dynamic range of -72 dB which is sufficient for the TDRSS system. It was necessary to build an interface board to allow samples collected with the BB A/D to be passed to the 303.

5.3 A/D Interface Board

The A/D Interface Board (ADIB) was developed as a Masters project by Tim Baggett [13]. Its purpose is to provide an interface to allow the EVM to communicate with the BB A/D. The board was developed to allow samples taken with the BB A/D to be passed into a peripheral (upper) memory location, which corresponds to the memory mapped I/O portion of memory. Though it is not a requirement, we may access the data samples with an efficient fast interrupt routine which is preferable. It is also possible with the interface board to use direct memory transfers (DMA) though this was not implemented in the Real-Time version of the carrier estimation algorithm. The board allows for user configuration of the specific memory location the samples will be written in to. The ADIB was further complicated by voltage level discrepancies between the BB A/D and the 303. This was overcome through the use of *zero wait-state* level translators that adjust the output levels of the BB A/D to levels acceptable by the 303.

5.4 Additional Hardware

In an actual implementation of the DAMA carrier estimation hardware, there will be additional hardware required. Recall from the discussion of the DAMA project implementation, it was stated that WSC could provide the DAMA carrier at an IF of 32.65 MHz. The range of frequency values that a DAMA carrier can take, identified as 300 kHz, must be pre-filtered to bandlimit the signal containing the DAMA carrier from the MA waveform or aliasing will result. This is to be accomplished using an analog pre-filter set to a passband over the range if interest.

With the signal now appropriately pre-filtered, the signal must be *frequency shifted* to baseband utilizing additional commonly available analog hardware. With this pre-processing accomplished, the DAMA carrier can be sampled and the digital signal processed to detect and estimate the carrier frequency utilizing the algorithm described in section two. Upon the algorithm's determination of the DAMA carrier frequency, the CODEC on the EVM is utilized to provide a locking tone. By choosing a sample rate of 32 kHz for the synthesis of the locking tone, we

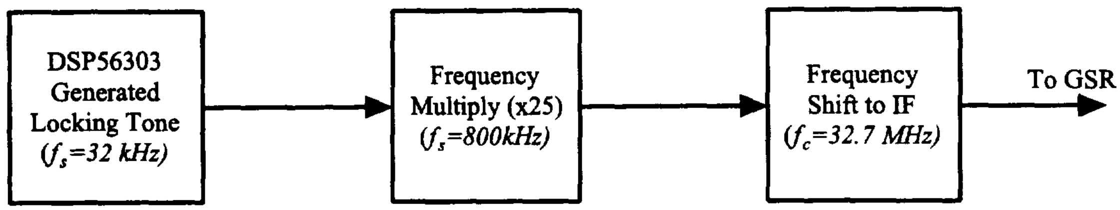

Figure 5.2: Locking Tone Generation

need only frequency multiply by 25 to get the appropriate frequency value (< 400 kHz). Afterward we frequency shift this tone back to IF for the GSR. This is illustrated in Figure 5.2. The analog hardware required to implement the frequency multiplication and shifting is commonly available and an essential component in implementation.

5.5 Software Description for Real-Time Carrier Estimation

The carrier estimation code is designed to work with the EVM and the ADIB to sample the incoming signal, process the signal, and then produce a locking tone to the ground station. The core of the program is based upon the code supplied with the EVM named **pass.asm**. The purpose of the **pass.asm** is to initialize the EVM (CODEC and processor), then transfer samples to/from the CODEC from/to the processor. All signal processing is performed between transfers. This code is important because in most cases of programming a Motorola DSP56xEVM, the algorithm to be implemented will use **pass.asm** as the starting point. The carrier estimation code is slightly modified in that while it uses **pass.asm** as the core, it must set the EVM to properly receive samples from the BB A/D and instead of reading sample values from the on-board CODEC, it will instead read sample values from an upper memory location as described above. Upon completing the initialization of the EVM (CODEC and processor) to work with the BB A/D, the carrier estimation algorithm begins. We begin by looking at the flowchart in Figure 2.6 which describes the algorithm.

The initial step from the reset condition is to initialize memory and on-board CODEC and prepare the EVM for communication with the ADIB. The algorithm then begins by filling the sample buffer with values. Once it has achieved a full block of 512 samples, it applies a windowing function of the user's choice. Typically either Rectangular or Hamming window coefficients are used. Included in this window function is an iteration scaling factor of 1/P, where P is the block number, which is nominally set to 0.125. This is included to scale for averaging. This has the added advantage of saving computation time since both windowing and average scaling are accomplished at once.

From the flowchart we now see that the actual FFT is performed. The FFT routine is supplied by Motorola (see Appendix B) and produces a normally ordered Fourier transform, as opposed to a bit-reversed, complex result from a normally ordered input. The result is stored such that the real component of the result is in X memory and the imaginary in Y memory. We compute the *magnitude-squared* of the FFT and repeat this step averaging magnitude-squared data P = 8 times to form the periodogram. If we assume that $x(n) \in \Re$ where $x(n)$

is the received, sampled signal, we need only search the first $N/2$ points due to Hermitian symmetry

$$X(k) = X^*(N-k), \quad 0 \leq k \leq N-1. \tag{5.1}$$

This reduces the computational time and memory usage.

Continuing with the flowchart in Figure 2.6, once the periodogram is computed, we convolve the result with the SMF and begin a search for the maxima of the enhanced spectrum. The result of this search is an index corresponding to the dominant spectral component. From this index a frequency may be found according to the relation

$$\hat{f} = \frac{k - \frac{L}{2}}{N} f_s \tag{5.2}$$

where $\hat{f}$ is the estimated frequency [14], k is the index found from the search, $L = N$ is the length of the SMF, N is the number of points of the FFT, and f_s is the sampling frequency. A sine wave table is then used to provide a locking tone at 1/25 frequency that the ground station expects. As described above, this locking tone will be frequency multiplied up IF. The DSP56303 assembly code is given in Appendix B.

6 Conclusions and Future Work

In this thesis we have described an algorithm to perform carrier estimation for the DAMA project. We have shown that the estimation accuracy depends primarily on the processing gain of the SS BPSK signal though some tertiary parameters also affect accuracy. We have developed a simulation model that provides proof of concept and provides DAMA operational parameters. Through the use of data collected at the WSC, we have verified the simulation model and established an operational point of PG = 40 with a corresponding accuracy of 90%.

In addition to the development above, we have developed Real-Time code based on the algorithm developed utilizing a Motorola DSP56303EVM. The entire project is implemented in hardware and functions equivalently to the simulation model.

Future work might involve the formulation of a tighter theoretical bound as well the development of the analog hardware required for implementation. This work could be applied in other applications relating to communications between user spacecraft and commercial telecommunication satellites. In this scheme, additional carrier pre-shifting would be required.

A Matlab Simulation Code

dama.m

```
%*************************************************************************
% DAMA dama
%
% Description: This function allows simulation of the estimation of a DAMA
%  user's Doppler-shifted carrier frequency.  We assume a spread spectrum
%  system and the presence of other SS systems (modeled by AWGN).
%
% Programmer: Phillip De Leon
%
% Creation Date: December 31, 1995
% Last Revision: Sept. 24, 1998
%
% Version History:
% 1.0 - frequency estimate of sinusoid in noise
% 2.0 - frequency estimate of DAMA carrier in noise (spread spectrum)
% 3.0 - better memory management, more efficient
% 4.0 - STFTs or average FFTs or MA filter smoothing of FFT bins
% 5.0 - 800kHz sampling rate, 512-point FFT
% 6.0 - frequency estimate now over 512 blocks (of a longer signal).
% 7.0 - added frequency-domain matched filter to search
%
% Required subroutines: Use PLOT_DAMA_DATA.M to visualize data
%
% Notes:
% 1) For n MA users DAMA-to-MA power ratio is
%    SNR = (Rb_DAMA/Rc_DAMA)/(n*Rb_MA/Rc_MA)
%        = (1000/10000)/(n*10000/3000000) = 30/n.
%    For n = 5, SNR = 7.78dB;
%
% 2) This code is compute intensive!!!
%
% References: Digital and Analog Communication Systems by Couch p.359
%
% Copyright (c) 1998 by Phillip De Leon, All Rights Reserved
%*************************************************************************

clear;

% ----------------------
% DAMA User's Parameters
% ----------------------
Fc_DAMA = 164e3;      % DAMA user's carrier frequency (after shifting) in Hz
main_lobe_BW = 200e3; % 200kHz main lobe BW centered on F_DAMA
max_Doppler = 64e3;   % maximum Doppler shift of +/-64kHz
est_tolerance = 3e3;  % must estimate carrier to within +/- est_tolerance

% ------------------------------
% Communication System Parameters
% ------------------------------
Fs = 800000;  % output signal sampling freq. (samples/s)
Fc = Fc_DAMA + 0.5*max_Doppler;  % assume DAMA carrier is +50% max
                                 % Doppler shifted (for testing purposes)

Rb_DAMA = 1000;    % data rate for DAMA (bits/s) Fs/Rb_DAMA must be integer
Rc_DAMA = 100000;  % chip rate for DAMA (chips/s) 100K?
samples_per_bit = Fs/Rb_DAMA;   % must be integer
samples_per_chip = Fs/Rc_DAMA;  % must be integer
Rb_MA = 10*Rb_DAMA;             % data rate for MA (bits/s)
```

```
Rc_MA = 3000000;               % chip rate for MA (chips/s) 3000K

% -------------------------------
% Frequency Estimation Parameters
% -------------------------------
% Estimation of Fc occurs over a bandwidth of...
% Fc - max_Doppler - main_lobe_BW/2 < F_DAMA < Fc + max_Doppler + main_lobe_BW/2
% 0 <= f <= 328
N = 512;     % length data block to take FFT over
window = 0;  % if 0 assume rectangular window, 1 Hamming Window
if (window)
   w = hamming(N);  % Hamming window used in FFT
end;

%----------------------
% Simulation Parameters
%----------------------
number_of_estimates = 30;  % number of frequency estimates to perform
                           % for each SNR typically 10000
num_FFTs = 8;  % average 8 FFTs per estimate;
snr = [-5]';   % vector of SNRs (in dB) to simulate typically from -2.2 to 4.7dB

%-----------
% Simulation
%-----------
est_data = zeros(length(snr),N/2);  % malloc storage for estimation data

for m = 1:length(snr);  % m is index for SNRs
   disp(sprintf('*** SNR = %d ***',snr(m)));
   max_pos_est_error = 0;  % reset variables
   max_neg_est_error = 0;
   rand('seed',0);    % reset uniform generator seed
   randn('seed',0);   % reset normal generator seed

   for n = 1:number_of_estimates;  % n is index for estimate #
      n;
      sum_mag_X_sqrd = zeros(N/2,1);  % malloc storage for accumulation of FFT data
      sum_mag_S_sqrd = zeros(N/2,1);  % malloc storage for accumulation of FFT data

      % --------------------
      % Synthesize SS signal
      % --------------------
      % Generate digital message
        data = round(rand(ceil(num_FFTs*N/samples_per_bit),1));  % generate data (bits) to modulate

      % Digital BPSK Modulation
        start = 0;
        stop = samples_per_bit - 1;
        msg = [];
        for i = 1:length(data)
           n = [start:stop];
           if (~data(i))  % if 0 multiply by -1
              msg = [msg' (-1)*sin(2*pi*n*Fc/Fs)]';  % build up message signal
           else           % if 1 multiply by 1
              msg = [msg' sin(2*pi*n*Fc/Fs)]';  % build up message signal
           end;
           start = stop + 1;
           stop = min((start + samples_per_bit - 1),num_FFTs*N);
        end;

      % Direct Sequence Spreading
        PN_code = round(rand(ceil(num_FFTs*N/samples_per_chip),1));
```

```
        start = 1;
        stop = samples_per_chip;
        s = [];
        for i = 1:length(PN_code)
           n = [start:stop];
           if (~PN_code(i))  % if 0 multiply by -1
              s = [s' (-1)*msg(n)']';  % build up spread signal
           else              % if 1 multiply by 1
              s = [s' msg(n)']';  % build up spread signal
           end;
           start = stop + 1;
           stop = min((start + samples_per_chip - 1),num_FFTs*N);
        end;

   % Compute FFT of BPSK-SS for Matched Filter Use
       S = fft(s(1:N));
       mag_S_sqrd = real(S(1:N/2).*conj(S(1:N/2)));
       sum_mag_S_sqrd = sum_mag_S_sqrd + mag_S_sqrd;
       avg_mag_S_sqrd = sum_mag_S_sqrd/num_FFTs;
       MF = flipud(avg_mag_S_sqrd);

        for k = 1:num_FFTs;  % k is index on FFT number

         % Add AWGN noise (MA users + channel noise) according to desired SNR to SS signal
         noise = randn(N,1) .* sqrt(cov(s)/(10^(snr(m)/10)));  % ./ ?
         s_prime = s(1+N*(k-1):N*(k-1)+512);  % Take length N buffer of samples from total.
         r = s_prime + noise;  % carrier + noise
         r = r ./ sqrt(cov(r));  % scale to unit variance

         % -------------
         % Window signal
         % -------------
         if (~window)
            x = r;  % no window => rectangular window
         else
            x = w .* r; % window with preset window
         end;

         X = fft(x);
         mag_X_sqrd = real(X(1:N/2).*conj(X(1:N/2)));  % due to symmetry need only scan lower half
         sum_mag_X_sqrd = sum_mag_X_sqrd + mag_X_sqrd;

      end;  % FFT loop

      % --------------------
      % Frequency estimation
      % --------------------
      avg_mag_X_sqrd = sum_mag_X_sqrd/num_FFTs;

   % Frequency-Domain Matched Filtering
   MF_mag_X = conv(MF,avg_mag_X_sqrd);

if (0)
      k = find(avg_mag_X_sqrd == max(avg_mag_X_sqrd));
      if (length(k) > 1)
        k = median(k);  % choose the median k if there are several
      else
        k = k;
   end;
end;
if (1)
   k = find(MF_mag_X == max(MF_mag_X));
```

```
      if (length(k) > 1)
        k = median(k);  % choose the median k if there are several
      else
        k = k;
      end;
   k = k - N/2 - 1; % MF spreads range by 2
end;

   est_data(m,k) = est_data(m,k) + 1;  % tally estimate

   end;  % estimates loop
   save dama_data est_data
end;  % snr loop

% ------------------------------
% Save Data in MATLAB 4.x format
% ------------------------------
x = version;
if (x(1)==53)  % running on MATLAB 5.x
   save -v4 dama_data Rc_DAMA Fs Fc snr est_tolerance N window number_of_estimates est_data
else           % running on MATLAB 4.x
   save dama_data Rc_DAMA Fs Fc snr est_tolerance N window number_of_estimates est_data avg_mag_X_sqrd
end;
```

mf_bpsk_ss.m

```
%***************************************************************************
% MF_BPSK
%
% Description: This function computes the matched filter for the PSD of the
%   BPSK-SS frequency-domain waveform. The matched filter coefficients are
%   stored in a file for use in the DAMA.M code. We may be able to use the
%   theoretical PSD.
%
% Programmer: Phillip De Leon
%
% Creation Date: Sept. 25, 1998
% Last Revision: Oct. 24, 1998
%
% Version History:
% 1.0 - MF for the BPSK-SS
% 1.1 - New Vectorization Methods - will reduce run time.
% 1.2 - Added Crash Recovery - ALL data is saved to temp.mat prior to
%       next major algorithm step.
%
% Required subroutines:
%
% Notes:
% 1) N must be the same as in DAMA.M
%
% Approximate Runtime:
%   P200 win98  : ~22 minutes
%   Mac         : Not performed
%   Pxx Linux   : Not performed
%
% References:
%
% Copyright (c) 1998 by Phillip De Leon, All Rights Reserved
%***************************************************************************
ON = 1;          % Used to turn options on or off.
OFF = 0;
ERR = 0;         % Set ERR to level achieved before crash - results
                 % are saved in temp.mat

if ERR == 0 disp('Operation Level One:')
% ----------------------
% Code Parameters
% ----------------------
spreading_opt = ON; % ON to spread - OFF to just BPSK modulate.
window = OFF;       % if OFF assume rectangular window, ON Hamming Window

% ----------------------
% DAMA User's Parameters
% ----------------------
Fc_DAMA = 200e3;    % DAMA user's carrier frequency (after shifting) in Hz
                    % Set to middle of DAMA frequency range.

% ------------------------------
% Communication System Parameters
% ------------------------------
Fs = 800000;   % output signal sampling freq. (samples/s)
Fc = Fc_DAMA;  % assume unshifted DAMA carrier

Rb_DAMA = 1000;   % data rate for DAMA (bits/s) Fs/Rb_DAMA must be integer
Rc_DAMA = 50000;  % chip rate for DAMA (chips/s) 100K?
if rem(Fs,Rb_DAMA) ~= 0
```

```
   error('samples_per_bit MUST be an integer!!') % Check to make sure that the
                                                 % samples per bit is integer.
end
if rem(Fs,Rc_DAMA) ~= 0
   error('samples_per_chip MUST be an integer!!') % Check to make sure that the
                                                  % samples per chip is integer.
end
samples_per_bit = Fs/Rb_DAMA;   % must be integer
samples_per_chip = Fs/Rc_DAMA;  % must be integer

% -------------------------------
% Frequency Estimation Parameters
% -------------------------------
% Estimation of Fc occurs over a bandwidth of...
% Fc - max_Doppler - main_lobe_BW/2 < F_DAMA < Fc + max_Doppler + main_lobe_BW/2
% 0 <= f <= 328
N = 512;  % length data block to take FFT over

if (window)
   w = hamming(N);  % Hamming window used in FFT
else
   w = ones(N,1);
end

% ---------------------
% Averaging Parameters
% ---------------------
num_FFTs = 5000;    % Average iterations - higher -> better resolution

save temp           % End of level One - save work so far to temp.
end

if ERR == 1 | ERR == 0  % Check Error Level.
disp('Operation Level Two:')
% -------------------------
% Generate digital message
% -------------------------
M = num_FFTs*N*Rb_DAMA/Fs; % number of bits to take PSD over

disp('Preallocating...')
msg = zeros(M*samples_per_bit,1);  % Preallocate msg signal
data = msg;             % Preallocate data signal (-1,+1)
s = msg;                % Preallocate s - see variable description.
disp('Done Preallocating...')

msg = sin(2*pi*(0:M*samples_per_bit-1)*Fc/Fs);  % BIG vector - not for the weak!!
msg = msg'; data =
filter(ones(samples_per_bit,1),1,upsamp((-1).^(round(rand(M,1))),samples_per_bit));
                                 % Previous line generates a data vector with each
                                 % bit represented by samples_per_bit number of
                                 % samples.
msg = msg.*data;                 % data vector is multiplied by the signal vector
                                 % to generate the message vector - BPSK modulated.

save temp               % End of Level Two - save work so far to temp.
end

if ERR == 2 | ERR == 1 | ERR == 0; disp('Operation Level Three:')

% -------------------
% Spread BPSK Signal
% -------------------
```

```
if spreading_opt        % If spreading option is on then spread signal.
PN_code = filter(ones(samples_per_chip,1),1, ...
   upsamp((-1).^(round(rand(ceil(num_FFTs*N/samples_per_chip),1))),...
   samples_per_chip));
s = msg.*PN_code;   % s is the BPSK modulated spread spectrum signal.
end

% Compute FFT of BPSK-SS for Matched Filter Use from averages.

% h = waitbar2(0,'Generating Matched Filter Frequency Response...');
start = 1;
stop = N;

save temp       % End of Level Three - save work so far to temp.
end

if ERR == 3 | ERR == 2 | ERR == 1 | ERR == 0

disp('Operation Level Four:')

sum_mag_S_sqrd = zeros(N/2,1);
for l = 1:num_FFTs              % Generate averaged mag-squared
   % waitbar2(l/num_FFTs);      % data from NN block FFTs.
   l
   n = [start:stop];
   temp = w.*s(n);
   S = fft(temp);
   mag_S_sqrd = real(S(1:N/2).*conj(S(1:N/2)));
   sum_mag_S_sqrd = sum_mag_S_sqrd + mag_S_sqrd;
   start = stop + 1;
   stop = stop + N;
end

% close(h)
avg_mag_S_sqrd = sum_mag_S_sqrd/num_FFTs;   % average.
MF = flipud(avg_mag_S_sqrd);            % "time reverse" data to prepare
N1 = Fc*N/Fs;                   % for dama_mf code.
                                % Record index of carrier frequency.

end

if window
   wtype = ['ha'];
else
   wtype = ['re'];
end fname = ['pg' int2str(Rc_DAMA/Rb_DAMA) wtype];

mver = version if (mver(1) == '5')
   eval(['save -v4 ' fname ' MF N1 N'])
else
   eval(['save ' fname ' MF N1 N'])
end


% -------------------------------------------------------------------------
%
% Variable Description
%
% Name          Description            Purpose
%--------------------------------------------------------------------------
% ON            Program Control     Used to turn ON/OFF Program Options
% OFF           Program Control     Used to turn ON/OFF Program Options
```

```
% ERR            Program Control     Used in disaster recovery
% spreading_opt  Program Control     Used to turn ON/OFF spreading
% window         Program Control     Used to turn ON/OFF windowing
% Fc_DAMA        Program Parameter   DAMA Carrier Frequency - for the
%                                    purposes of this program it is used
%                                    to set the reference index.
% Fs             Program Parameter   Sampling Frequency
% Fc             Program Parameter   Carrier Frequency - in this program
%                                    it is set to the value of Fc_DAMA.
% Rb_DAMA        Program Parameter   Bit Rate
% Rc_DAMA        Program Parameter   Chip Rate
% samples_per_bit
%                Calculated Parameter Each bit is represented by # of samples
% samples_per_chip
%                Calculated Parameter Each chip is represented by # of
%                                     samples.
% N              Program Parameter   Length of FFT block
% w              Calculated Parameter window coefficients
% num_FFTs       Program Parameter   Number of Iterations
% M              Calculated Parameter The number of bits required in the data
%                                     vector.
% msg            Calculated Vector   The actual sinusoidal signal - later it is
%                                    BPSK modulated.
% data           Calculated Vector   Binary data stored as (-1,1)
% s              Calculated Vector   BPSK SS carrier
% PN_code        Calculated Vector   The psuedo-random noise vector to cause
%                                    spreading.
% start          Program Parameter   Used in indexing
% stop           Program Parameter   Used in indexing
% l              Program Control     Used in FOR loop
% h              Program Control     Handle to figure
% sum_mag_S_sqrd
%                Calculated Vector   Holds sum of magnitude squared data from FFT.
% S              Calculated Vector   Holds FFT data for current iteration
% mag_S_sqrd     Calculated Vector   Holds magnitude squared data from FFT for
%                                    current iteration.
% avg_mag_S_sqrd
%                Calculated Vector   Holds the average mag_S_sqrd vector.
% MF             Calculated Vector   Frequency reversed - think of it as time
%                                    reversed data.
% wtype          Program Control     Type of window employed for inclusion into
%                                    file name.
% fname          Program Control     Filename
% mver           Program Control     mver(1) holds matlab version.
%
%-----------------------------------------------------------------------------------
```

plot_dama_data.m

```
%***********************************************************************
% PLOT_DAMA_DATA plot_dama_data
%
% Description: This function provides a variety of tools for DAMA data
%  visualization using the file saved in DAMA.M
%
% Call Syntax: plot_dama_data
%
% Programmer: Phillip De Leon
%
% Creation Date: December 31, 1995
% Last Revision: July 1, 1997
%
% Required subroutines:
%
% Notes: Assume workspace (DAMA_DATA.MAT) has already been loaded.
%
% References:
%
% Copyright (c) 1998 by Phillip De Leon, All Rights Reserved
%***********************************************************************

%clg;

 selection = menu('Select a Plot',...
    'Estimation Ranges',...     % 1)
    'Estimation Histograms',... % 2)
    'Estimation Accuracy',...   % 3)
    'Exit');

% ---------------------
% Estimation range
% ---------------------
if (selection == 1)
   i = find(est_data(1,:));   % find non-zero elements
   max_neg_est_error = ((min(i)-1) * Fs / N) - Fc;  % 1 <= i <= N but need 0 <= i <= N-1
   max_pos_est_error = ((max(i)-1) * Fs / N) - Fc;
   plot([snr(1) snr(1)], [max_neg_est_error max_pos_est_error]);
   hold on
   for m = 2:length(snr)
      i = find(est_data(m,:));   % find non-zero elements
      max_neg_est_error = ((min(i) - 1) * Fs / N) - Fc;
      max_pos_est_error = ((max(i) - 1) * Fs / N) - Fc;
      plot([snr(m) snr(m)], [max_neg_est_error max_pos_est_error]);
   end;
   plot([-100 100],[est_tolerance est_tolerance],':')
   plot([-100 100],[-est_tolerance -est_tolerance],':')
   hold off;
   ylabel('Estimation error (Hz)');
   xlabel('SNR (dB)');
   if (window)
      title(['Frequency Estimation Ranges (Rc=',sprintf('%d',Rc_DAMA), ...
             ', N=',sprintf('%d', N),', Hamming window)']);
   else
      title(['Frequency Estimation Ranges (Rc=',sprintf('%d',Rc_DAMA), ...
             ', N=',sprintf('%d', N),', Rectangular window)']);
   end;
   [i,j] = find(est_data);
   Ymin = min(j);
   Ymax = max(j);
```

```
   axis([(snr(1)-10) (snr(length(snr))+10) 1.1*min(Ymin,-est_tolerance) ...
         1.1*max(Ymax,est_tolerance)]);

% --------------------
% Estimation histogram
% --------------------
elseif (selection == 2)
   for m = 1:length(snr)
      bar([0:N/2-1].*(Fs/N),est_data(m,1:N/2));
          hold on;
      plot([0:N/2-1].*(Fs/N),est_data(m,1:N/2),'+');
          plot([Fc Fc],[0 1.1*max(est_data(m,1:N/2))],'b:');
          plot([Fc+est_tolerance Fc+est_tolerance],[0 1.1*max(est_data(m,1:N/2))],'r:');
          plot([Fc-est_tolerance Fc-est_tolerance],[0 1.1*max(est_data(m,1:N/2))],'r:');
          hold off;
      ylabel('Occurences')
      xlabel('Estimation Frequency (Hz)')
      if (window)
         title(['Frequency Estimation for SNR of ',sprintf('%g', snr(m)), ...
                'dB (Rc=',sprintf('%d',Rc_DAMA),', N=',sprintf('%d',N),', Hamming window)']);
      else
        % title(['Frequency Estimation for SNR of ',sprintf('%g', snr(m)), ...
        %        'dB (Rc=',sprintf('%d',Rc_DAMA),', N=',sprintf('%d',N),', Rectangle window)']);
         title(['Frequency Estimation for SNR of ',sprintf('%g', snr(m)), ...
                'dB']);
      end;
      axis([0.95*(Fc-est_tolerance) 1.05*(Fc+est_tolerance) 0 1.1*max(est_data(m,1:N/2))]);
      disp('Hit any key to plot next histogram...');
      pause;
   end;

% --------------------
% Statistical Accuracy
% --------------------
elseif (selection == 3)
   lower_index_bound = ceil(N*(Fc-est_tolerance)/Fs)+1 % 1 <= lower_index_bound <= N
%    lower_index_bound = 1;
   upper_index_bound = floor(N*(Fc+est_tolerance)/Fs)+1 % 1 <= upper_index_bound <= N
%    upper_index_bound = 211;

   pause
   accuracy = zeros(length(snr),1);
   for m = 1:length(snr)
      number_accurate_estimates=sum(est_data(m,lower_index_bound:upper_index_bound));
      accuracy(m) = number_accurate_estimates / number_of_estimates;
   end;
   plot(snr,accuracy,'k')
   xlabel('SNR (dB)');
   ylabel('Proportion of Estimates in Range');
   title('Estimation Accuracy');
   grid;
else
   return;
end;
% plot_dama_data
```

wsands4.m

```
% White Sands Data Collection Code ver 4.0
% Started on 6/26/98
% Fill in comments. ---> Still need to comment
%
% Signal Vector Descriptions:
%   1: Rb = 1000 bits/sec   Rc = 100000 chips/sec
%   2: Rb = 1000 bits/sec   Rc = 20000 chips/sec
%   3: Rb = 1000 bits/sec   Rc = 10000 chips/sec
%   4: Rb = 1000 bits/sec   Rc = 40000 chips/sec
%
% Revision History:
%
%   Ver 1.0:    Baseline, one file generated per user input
%   Ver 2.0:    Four "useful" files generated at once, no user
%               input. Some code cleaned up (vectorized).
%   Ver 3.0:    More vectorization, utilizes MATLAB's filter
%               function. Vectors are pre-allocated.
%   Ver 3.01:   Changed code so that # of bits may be individually
%               selected.
%   Ver 4.0:    To allow for larger data file generation, a menu
%               system is implemented so that only one data file
%               is generated.  This compensates for memory
%               problems. Edit file for different spread/data
%               rates.
%   Ver 4.01:   "2047" NASA PN sequence took to long to generate
%               so the sequence is generated offline for faster
%               performance. This restricts resulting binary file
%               length to 2meg.
%
% Last Update: 7/13/98
% Written by: Bradley James Scaife
%*************************************************************************
timer = cputime
preamble_length = 5000;

choice = menu('Please Select File to be
Generated:','1k_100','1k_20','1k_10','1k_40');

if choice == 1
% Generate file one
%********************

% Message One Parameters
%************************
%msg_len1 = input('Enter Message Length One (bits): ');
msg_len1 = 20e3; %Restricted value - only change if spread vector code is altered.
Rb1 = 1000; Rc1 = 100000;

% Calculated Parameters: Message One
%************************************
spread_factor1 = ceil(Rc1/Rb1);   % Set as an integer.

% Preallocate MSG1 vector
%**************************
msg1_init = zeros(msg_len1,1); msg1 =
zeros(msg_len1*spread_factor1,1);
%spreading_vector1 = zeros(msg_len1*spread_factor1,1);

% MSG 1 and Spreading Vector Generation
%****************************************
```

```
rand('seed',0);

msg1_init = round(rand(msg_len1,1));

msg1 = upsamp(msg1_init,spread_factor1)';
msg1 = filter(ones(spread_factor1,1),1,msg1);
clear msg1_init

load svec

% Generate s1
%***************
disp('Generating s1...')

s1 = 255*xor(msg1,spreading_vector)';

% Binary File Generation
%*************************
disp('Generating binary file...')
disp('Inserting preamble...')
disp('Inserting signal vector...')

data_out1 = [zeros(1,preamble_length) s1];
fid =fopen('1k_100.bin','wb');
fwrite(fid,data_out1,'int8');
fclose(fid);

elseif choice == 2

% Generate File 2: Per definition in header above.
%****************************************************
%msg_len2 = input('Enter Message Length Two (bits): ');
msg_len2 = 100e3; %Restricted value - only change if spread vector code is altered.
Rb2 = 1000; Rc2 = 20000;

% Calculated Parameters
%************************
spread_factor2 = ceil(Rc2/Rb2);

% Preallocate MSG2 vectors
%**************************
msg2_init = zeros(msg_len2,1);
msg2 = zeros(msg_len2*spread_factor2,1);
%spreading_vector2 = zeros(msg_len2*spread_factor2,1);

% Message and Spreading Vector Generation
%*****************************************
msg2_init = round(rand(msg_len2,1));
msg2 = upsamp(msg2_init,spread_factor2)';
msg2 = filter(ones(spread_factor2,1),1,msg2);
clear msg2_init

load svec

% Generate s2
%***************
disp('Generating s2...')

s2 = 255*xor(msg2,spreading_vector)';

% Binary File Generation
%*************************
```

```
disp('Generating binary file...')
disp('Inserting preamble...')
disp('Inserting signal vector...')

data_out2 = [zeros(1,preamble_length) s2];

fid = fopen('1k_20.bin','wb');
fwrite(fid,data_out2,'int8');
fclose(fid);


elseif choice == 3
% Generate File 3: Per definition in header above.
%****************************************************
%msg_len3 = input('Enter Message Length Three (bits): ');
msg_len3 = 200e3; %Restricted value - only change if spread vector code is altered.
Rb3 = 1000; Rc3 = 10000;

% Calculated Parameters
%************************
spread_factor3 = ceil(Rc3/Rb3);

% Preallocate MSG3 vectors
%**************************
msg3_init = zeros(msg_len3,1);
msg3 = zeros(msg_len3*spread_factor3,1);
spreading_vector3 = zeros(msg_len3*spread_factor3,1);

% Message and Spreading Vector Generation
%*****************************************
msg3_init = round(rand(msg_len3,1));
msg3 =upsamp(msg3_init,spread_factor3)';
msg3 = filter(ones(spread_factor3,1),1,msg3);
clear msg3_init

load svec

% Generate s3
%***************
disp('Generating s3...')

s3 = 255*xor(msg3,spreading_vector)';

% Binary File Generation
%*************************
disp('Generating binary file...')
disp('Inserting preamble...')
disp('Inserting signal vector...')

data_out3 = [zeros(1,preamble_length) s3];

fid = fopen('1k_10.bin','wb');
fwrite(fid,data_out3,'int8');
fclose(fid);

else
% Generate File 4: Per definition in header above.
%****************************************************
%msg_len4 = input('Enter Message Length Four (bits): ');
msg_len4 = 50e3;       %Restricted value - only change if spread vector code is altered.
Rb4 = 1000; Rc4 = 40000;

```

```
% Calculated Parameters
%*************************
spread_factor4 = ceil(Rc4/Rb4);

% Preallocate MSG4 vectors
%***************************
msg4_init = zeros(msg_len4,1);

msg4 = zeros(msg_len4*spread_factor4,1);
% spreading_vector4 = zeros(msg_len4*spread_factor4,1);

% Message and Spreading Vector Generation
%*******************************************
msg4_init = round(rand(msg_len4,1));

msg4 = upsamp(msg4_init,spread_factor4)';

msg4 = filter(ones(spread_factor4,1),1,msg4);
clear msg4_init

load svec

% Generate s4
%***************
disp('Generating s4...')

s4 = 255*xor(msg4,spreading_vector)';

% Binary File Generation
%**************************
disp('Generating binary file...')
disp('Inserting preamble...')
disp('Inserting signal vector...')

data_out4 = [zeros(1,preamble_length) s4];

fid = fopen('1k_40.bin','wb');
fwrite(fid,data_out4,'int8');
fclose(fid);

end

disp('All Done :) ')
run_time = cputime - timer
```

cap_mf.m

```
% Title:
%   cap_mf.m
%
% Purpose:
%   The purpose of this code is to generate estimation data similar to
%   dama_mf with the exception that this code is tailored to use data
%   captured during the White Sands Complex (WSC) experiment.
%
% Revision:
%   1.0 -- 11/20/98
%
% Revision History:
%   none => baseline
%
% Author:
%   Brad Scaife
%
% Notes:
%   This code requires as input data that has been prepared by wsco_d2.m which
%   prepares the captured data and generates a matrice of usable vectors (see
%   wsco_d2 comments for more information).  This code also relies on an index
%   matice that is generated by in_prp.m.  In both cases these files have already been
%   prepared and may simply be loaded following prescribed naming conventions.  In
%   the event that these data files have been lost though it was thought helpful to
%   comment on how to regenerate them.
%
% Average runtime:
%    ~ 8 minutes
%
%*******************************************************************************************
clear all;

ON = 1; OFF = 0;

% Code Options
%::::::::::::::
window = OFF;               % Use to toggle use of hamming window
matched_filter = ON;    % Use to toggle use of matched filter
insight = OFF;              % Use to gain instantaneous frequency estimation

% Code Parameters
%:::::::::::::::::
Fs = 100e6;         % DO NOT CHANGE - this was the sample rate used.
%snr = 1.5          % As per dama_mf definition - not wsc definition.
est_tolerance = 3e3;% +/- range of acceptable error in Hz.
N = 65536;          % FFT block length.
number_of_estimates = 1000; % # of estimation attempts.  Don't modify!!!
fft_avg = 8;                % Number of FFT's to average over - changing this will
                            % require changes to code mentioned in documentation above
                            % as well as the rerunning of this code.

alpha = 2;      % Constant used to obtain consistent data points - does not
                % add to statistical meaning nor detract from it.
f_lo = 32.50e6;
f_hi = 32.90e6;

lower_bound_index = ceil(f_lo*N/Fs)+1;

starting_bound = lower_bound_index;

```

```
upper_bound_index = floor(f_hi*N/Fs)+1;
Rb_DAMA = 1e3;

% Program Flow
%::::::::::::::
if window
   wtype = ['h'];
else
   wtype = ['r'];
end

menu_sel = menu('Select Data to Process:',...
   'SF=10, SNR1, F2',...         % Selection 1
   'SF=10, SNR1, F3',...         % Selection 2
   'SF=10, SNR2, F1',...         % Selection 3
   'SF=10, SNR2, F2',...         % Selection 4
   'SF=10, SNR2, F3',...         % Selection 5
   'SF=10, SNR2, F5',...         % Selection 6
   'SF=10, SNR3, F2',...         % Selection 7
   'SF=20, SNR2, F3')            % Selection 8

if menu_sel == 1
   data_filename = ['c:\research\code\matlab\whites~2\wscmat~1\snr1_f2' wtype];
   eval(['load ' data_filename]);
   num_col = size(x,2);
   index_filename = ['c:\research\code\matlab\whites~2\cmb8_' int2str(num_col)];
   eval(['load ' index_filename]);
   iterations = choose(num_col,8);
   %snr = -5;
   snr = 40;
   Rc_DAMA = 10e3;
   Fc = 32.65e6;

elseif menu_sel == 2
   data_filename = ['c:\research\code\matlab\whites~2\wscmat~1\snr1_f3' wtype];
   eval(['load ' data_filename]);
   num_col = size(x,2);
   index_filename = ['c:\research\code\matlab\whites~2\cmb8_' int2str(num_col)];
   eval(['load ' index_filename]);
   iterations = choose(num_col,8);
   %snr = -5;
   snr = 40;
   Rc_DAMA = 10e3;
   Fc = 32.70e6;


elseif menu_sel == 3
   data_filename = ['c:\research\code\matlab\whites~2\wscmat~1\snr2_f1' wtype];
   eval(['load ' data_filename]);
   num_col = size(x,2);
   index_filename = ['c:\research\code\matlab\whites~2\cmb8_' int2str(num_col)];
   eval(['load ' index_filename]);
   iterations = choose(num_col,8);
   %snr = 1.5;
   snr = 45;
   Rc_DAMA = 10e3;
   Fc = 32.6e6;


elseif menu_sel == 4
   data_filename = ['c:\research\code\matlab\whites~2\wscmat~1\snr2_f2' wtype];
   eval(['load ' data_filename]);
```

```
    num_col = size(x,2);
    index_filename = ['c:\research\code\matlab\whites~2\cmb8_' int2str(num_col)];
    eval(['load ' index_filename]);
    iterations = choose(num_col,8);
    % snr = 1.5;
    snr = 45;
    Rc_DAMA = 10e3;
    Fc = 32.65e6;


elseif menu_sel == 5
    data_filename = ['c:\research\code\matlab\whites~2\wscmat~1\snr2_f3' wtype];
    eval(['load ' data_filename]);
    num_col = size(x,2);
    index_filename = ['c:\research\code\matlab\whites~2\cmb8_' int2str(num_col)];
    eval(['load ' index_filename]);
    iterations = choose(num_col,8);
    % snr = 1.5;
    snr = 45;
    Rc_DAMA = 10e3;
    Fc = 32.70e6;


elseif menu_sel == 6
    data_filename = ['c:\research\code\matlab\whites~2\wscmat~1\snr2_f5' wtype];
    eval(['load ' data_filename]);
    num_col = size(x,2);
    index_filename = ['c:\research\code\matlab\whites~2\cmb8_' int2str(num_col)];
    eval(['load ' index_filename]);
    iterations = choose(num_col,8);
    %snr = 1.5;
    snr = 45;
    Rc_DAMA = 10e3;
    Fc = 32.80e6;


elseif menu_sel == 7
    data_filename = ['c:\research\code\matlab\whites~2\wscmat~1\snr3_f2' wtype];
    eval(['load ' data_filename]);
    num_col = size(x,2);
    index_filename = ['c:\research\code\matlab\whites~2\cmb8_' int2str(num_col)];
    eval(['load ' index_filename]);
    iterations = choose(num_col,8);
    %snr = 2.2;
    snr = 50;
    Rc_DAMA = 10e3;
    Fc = 32.65e6;


else
    data_filename = ['c:\research\code\matlab\whites~2\wscmat~2\snr2_f3' wtype];
    eval(['load ' data_filename]);
    num_col = size(x,2);
    index_filename = ['c:\research\code\matlab\whites~2\cmb8_' int2str(num_col)];
    eval(['load ' index_filename]);
    iterations = choose(num_col,8);
    snr = 1.5;
    Rc_DAMA = 10e3;
    Fc = 32.70e6;

end

```

```
% Some Corrections for code operation
%::::::::::::::::::::::::::::::::::::::
if iterations > 1000
   iterations = 1000;
end

if iterations == 1000
   alpha = 1;
else
   alpha = 2;
end

% Load Matched Filter
%::::::::::::::::::::::
if matched_filter
   if window
      wtype2 = ['ha'];
   else
      wtype2 = ['re'];
   end

   fname = ['pg' int2str(Rc_DAMA/Rb_DAMA) wtype2];
   eval(['load c:\research\code\matlab\damane~1\' fname])
end

% Process Begin
%:::::::::::::::
N = 65536;  % Reset N after loading matched filter - required.
% Preallocation
%:::::::::::::::
X = zeros(N,fft_avg);
%X = zeros(N/2,fft_avg);
X_mag_squared = zeros(length(lower_bound_index:upper_bound_index),fft_avg);
%X_mag_squared = zeros(N/2,fft_avg);
X_avg = zeros(length(lower_bound_index:upper_bound_index),fft_avg);
%X_avg = zeros(N/2,fft_avg);
est_data = zeros(length(snr),length(lower_bound_index:upper_bound_index));
%est_data = zeros(length(snr),N/2);

% Begin FFT Estimation Process
%:::::::::::::::::::::::::::::
for n = 1:iterations
   n
   temp = x(:,index(n,:));                         % Select current index pattern.
   X = fft(temp);                                  % Perform 8 FFT's
   X = X(lower_bound_index:upper_bound_index,:);   % Restrict to search bound
   %X = Xt(1:N/2,:);
   X_mag_squared = (X.*conj(X)).';
   X_avg = (sum(X_mag_squared) ./ fft_avg).';      % Sum and average.

   if matched_filter
      MF_mag_X = conv(MF,X_avg);
   end

   k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg))));
   if insight
    disp('Frequency found to be:')
    (k+20791)*100e6/64e3
      pause(10)
   end
   if matched_filter
      k = find(MF_mag_X(1:length(MF_mag_X)) == max(MF_mag_X(1:length(MF_mag_X))));
```

```

      if (length(k) > 1)
         k = median(k);
      end

      k = k - N1;
   else
      k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg))));

      if (length(k) > 1)
         k = median(k);
      end

   end

    est_data(1,k) = est_data(1,k) + 1;  % Increment bin

end

est_data(1,:) = est_data(1,:) .* alpha; % Sponge data to look like a larger estimate

% ------------------------------
% Save Data in MATLAB 4.x format
% ------------------------------
x = version;
if (x(1)=='5')  % running on MATLAB 5.x
   save -v4 cap_dat Fc Fs N N1 est_data est_tolerance number_of_estimates ...
            starting_bound snr window wtype2
else           % running on MATLAB 4.x
   save cap_dat Fc Fs N N1 est_data est_tolerance number_of_estimates ...
        starting_bound snr window wtype2
end;
```

wsco_d2.m

```
% DAMA Test Signal Analysis
%
% Purpose:
%   This code is to be used as driver code for wsco.m. This code
%   converts captured whites sands files and converts them into
%   usable length N blocks.  Not entirely automated. Please read
%   info in wsco.m for details on project purpose.
%
% Input:
%   This code requires the DAMA test files named as s1,s2,...sn.
%
% Output:
%   Undecided at this point.
%
% Revision History:
%
%   ver 1.0:    baseline
%   ver 2.0:    Changed scope of code.  This program is now to
%               to be used to drive other code only.
%   ver 2.5:    Restructed the format data is to be saved in.
%
%
% Current Version - Date:
%   2.5 - 8/4/98
%
% Author:
%   Brad Scaife
%
% Date:
%   7/23/98
%
% Notes:
%   Remember to change the save filename at the end of the code.  Sorry
%       got to do a little work to run this one.
%*****************************************************************************
clear all
ON = 1;
OFF = 0;
window = OFF;

% Change to working directory
%******************************
cd c:\research\data\damaus~1\1k_10\snr2\f3

% Code Options
%******************************
% capture_option = 0;       % 0 for 50ksample data else 1 (for 100k)

% Code Parameters
%******************************
num_files = 6;      % 6- for snr1_f3; 9- for snr2_f3; 12- for snr3_f2
half = 65536;
short_cap = 50002;
long_cap = 100002;

load s1;
if length(s1) == long_cap
   capture_option = 1;
else
   capture_option = 0;
```

```
end
clear s1;
pack
% Load raw signals
%******************************
k = 0;
h = waitbar2(0,'Loading files...');
while 1
   k = k+1;
   waitbar2(k/num_files);
   sk = ['s' int2str(k)];
   filename = sk;
   if ~exist(filename), break, end
      eval(['load ' filename])
   end
close(h)
% Preallocation
%******************************
x = zeros(half,(1+capture_option)*num_files);
cd c:\research\code\matlab\whites~2
if capture_option
   h = waitbar2(0,'Parsing 100 kSample Files...');
   index = 1;
   if window
      W = hamm(half);
   else
      W = ones(half,1);
   end
   for k = 1:2:2*num_files
      waitbar2(index/num_files);
      varname = ['s' int2str(index)];
      temp = eval([varname]);
      index = index +1;
      x(:,k) = temp(1:half).*W;     % For others
      x(:,k+1) = temp(length(temp)-half+1:length(temp)).*W; % For others
   end
   close(h);
else
   h = waitbar2(0,'Parsing 50 kSample Files...');
   if window
      W = hamm(length(s1));
   else
      W = ones(length(s1),1);
   end
   for k = 1:num_files
```

```
        waitbar2(k/num_files);
        varname = ['s' int2str(k)];
        temp = eval([varname]);
        temp = temp.*W;
        temp = [temp;zeros(half - length(temp),1)];
        x(:,k) = temp;
    end
    close(h);
end

cd c:\research\code\matlab\whites~2\wscmat~2

mlver = version;
if (mlver(1) == '5')          % Running on MATLAB 5.x
    save -v4 snr2_f3r x
else
    save snr2_f3r x
end

cd c:\research\code\matlab\whites~2
%clear all
```

prepcap2.m

```
% Title:
%   prepcap.m
%
% Purpose:
%   The purpose of this code is to prepare DAMA captured signals for
%   comparison against simulated results.
%
% Revision:
%       1.0 -- 7/26/98
%       1.5 -- 8/2/98
%       2.0 -- 11/2/98
%
% Revision History:
%       none
%       1.5 --  Altered code to accept new signal matrix generated
%               in wsco_d.m
%       2.0 --  Employing Spectral Matched Filter
%
% Author:
%   Brad Scaife
%
% Date:
%   7/26/98
%
% Notes:
%   This code requires input signals matrices that have been prepared
%   by wsco_d.m and vector sequence matrices prepared by goob.m. Due to
%   limited test set, the code makes some approximations that may or may
%   not be valid.  Therefore, this code should not be considered in any
%   way a proof but rather a "ballpark" type of justification: Could it
%   work in the real world?
%
%******************************************************************************
clear all
ON = 1;
OFF = 0;

% Code Parameters
%*****************
snr = 1:7;                % SNR Ranges from So/N estimate.
Fc = 32.7e6;              % Measured DAMA carrier - from WSC.
Fs = 100e6;               % Burr-Brown Sampling Rate.
est_tolerance = 3e3;      % +/- acceptable error.
N = 65536;                % FFT block length.
number_of_estimates = 1000; % # of estimation attempts
fft_avg = 8;              % Number of FFT's to perform estimation over.
alpha = 2;                % Correction to number of estimates.
insight = 0;
lower_bound_index = 21291;  % Index of lower search bound.
upper_bound_index = 21504;  % Index of upper search bound.
starting_bound = lower_bound_index;
window = OFF;
matched_filter = ON;

% Preallocation
%****************
X = zeros(N,fft_avg);

X_mag_squared = zeros(length(lower_bound_index:upper_bound_index),fft_avg);

```

```
X_avg = zeros(length(lower_bound_index:upper_bound_index),fft_avg);
est_data = zeros(length(snr),length(lower_bound_index:upper_bound_index));
if window
   load c:\research\code\matlab\damane~1\pg10ha
else
   load c:\research\code\matlab\damane~1\pg10re
end
% Load SNR1_F2 Data Matrix (65536x12) and Combination Matrix (495x8)
%*******************************************************************************
load c:\research\code\matlab\whites~2\snr1_f2 load
c:\research\code\matlab\whites~2\cmb8_12 h =
waitbar2(0,'Formulating SNR1 Estimation for f2...');
for n = 1:495
    index = cmbo3;      % Load unique index pattern matrix.
    waitbar2(n/495);
    temp = x(:,index(n,:)); % Select current index pattern.
    X = fft(temp);      % Perform 8 FFT's
    X = X(lower_bound_index:upper_bound_index,:);   % Restrict to search bound
    X_mag_squared = (X.*conj(X)).';
    X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.
    k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg))));
    if insight
       disp('Frequency found to be:')
       (k+20791)*100e6/64e3
       pause(10)
    end
    if (length(k) > 1)
       k = median(k);
       disp('ooops')
    end
    est_data(1,k) = est_data(1,k) + 1;  % Increment bin
end
est_data(1,:) = est_data(1,:) .* alpha; % Sponge data to look like a larger estimate
clear x
clear index;
close(h);
% Load SNR1_F3 Data Matrix (65536x12) and Combination Matrix (495x8)
%*******************************************************************************
load c:\research\code\matlab\whites~2\snr1_f3
load c:\research\code\matlab\whites~2\cmb8_12
h = waitbar2(0,'Formulating SNR1 Estimation for f3...');
for n = 1:495
    index = cmbo3;      % Load unique index pattern matrix.
    waitbar2(n/495);
    temp = x(:,index(n,:)); % Select current index pattern.
```

```
    X = fft(temp);      % Perform 8 FFT's
    X = X(lower_bound_index:upper_bound_index,:);   % Restrict to search bound
    X_mag_squared = (X.*conj(X)).';
    X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.

    k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg))));
    if insight
        disp('Frequency found to be:')
        (k+20791)*100e6/64e3
        pause(10)
    end
    if (length(k) > 1)
        k = median(k);
        disp('ooops')
    end

    est_data(2,k) = est_data(2,k) + 1;  % Increment bin

end

est_data(2,:) = est_data(2,:) .* alpha; % Sponge data to look like a larger estimate

clear x
clear index;
close(h);


% Load SNR2_F1 Data Matrix (65536x12) and Combination Matrix (495x8)
%********************************************************************************
load c:\research\code\matlab\whites~2\snr2_f1
load c:\research\code\matlab\whites~2\cmb8_12
h = waitbar2(0,'Formulating SNR2 Estimation for f1...');

for n = 1:495

    index = cmbo3;      % Load unique index pattern matrix.
    waitbar2(n/495);

    temp = x(:,index(n,:)); % Select current index pattern.
    X = fft(temp);      % Perform 8 FFT's
    X = X(lower_bound_index:upper_bound_index,:);   % Restrict to search bound
    X_mag_squared = (X.*conj(X)).';
    X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.

    k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg))));
    if insight
        disp('Frequency found to be:')
        (k+20791)*100e6/64e3
        pause(10)
    end
    if (length(k) > 1)
        k = median(k);
        disp('ooops')
    end

    est_data(3,k) = est_data(3,k) + 1;  % Increment bin

end

est_data(3,:) = est_data(3,:) .* alpha; % Sponge data to look like a larger estimate

clear x
```

```
clear index;
close(h);



% Load SNR2_F2 Data Matrix (65536x12) and Combination Matrix (495x8)
%*********************************************************************************
load c:\research\code\matlab\whites~2\snr2_f2
load c:\research\code\matlab\whites~2\cmb8_12
h = waitbar2(0,'Formulating SNR2 Estimation for f2...');

for n = 1:495

    index = cmbo3;       % Load unique index pattern matrix.
    waitbar2(n/495);

    temp = x(:,index(n,:)); % Select current index pattern.
    X = fft(temp);       % Perform 8 FFT's
    X = X(lower_bound_index:upper_bound_index,:);    % Restrict to search bound
    X_mag_squared = (X.*conj(X)).';
    X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.

    k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg))));
    if insight
        disp('Frequency found to be:')
        (k+20791)*100e6/64e3
        pause(10)
    end
    if (length(k) > 1)
        k = median(k);
        disp('ooops')
    end

    est_data(4,k) = est_data(4,k) + 1;  % Increment bin

end

est_data(4,:) = est_data(4,:) .* alpha; % Sponge data to look like a larger estimate

clear x
clear index;
close(h);


% Load SNR2_F3 Data Matrix (65536x18) and Combination Matrix (1000x8)
%*********************************************************************************
load c:\research\code\matlab\whites~2\snr2_f3
load c:\research\code\matlab\whites~2\mc_cmb2
h = waitbar2(0,'Formulating SNR2 Estimation for f3...');

for n = 1:1000

    waitbar2(n/1000);

    temp = x(:,index(n,:));
    X = fft(temp);       % Perform 8 FFT's
    X = X(lower_bound_index:upper_bound_index,:);
    X_mag_squared = (X.*conj(X)).';
    X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.

    k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg))));
    if (length(k) > 1)
```

```
        k = median(k);
        disp('ooops')
    end

    est_data(5,k) = est_data(5,k) + 1;  % Increment bin

end

clear x;
close(h);

% Load SNR2_F5 Data Matrix (65536x12) and Combination Matrix (495x8)
%*******************************************************************************
load c:\research\code\matlab\whites~2\snr2_f5
load c:\research\code\matlab\whites~2\cmb8_12
h = waitbar2(0,'Formulating SNR2 Estimation for f5...');

for n = 1:495

    index = cmbo3;      % Load unique index pattern matrix.
    waitbar2(n/495);

    temp = x(:,index(n,:)); % Select current index pattern.
    X = fft(temp);      % Perform 8 FFT's
    X = X(lower_bound_index:upper_bound_index,:);   % Restrict to search bound
    X_mag_squared = (X.*conj(X)).';
    X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.

    k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg))));
    if insight
        disp('Frequency found to be:')
        (k+20791)*100e6/64e3
        pause(10)
    end
    if (length(k) > 1)
        k = median(k);
        disp('ooops')
    end

    est_data(6,k) = est_data(6,k) + 1;  % Increment bin

end

est_data(6,:) = est_data(6,:) .* alpha; % Sponge data to look like a larger estimate

clear x
clear index;
close(h);

% Load SNR3_F2 Data Matrix (65536x12) and Combination Matrix (495x8)
%*******************************************************************************
load c:\research\code\matlab\whites~2\snr3_f2
load c:\research\code\matlab\whites~2\cmb8_12
h = waitbar2(0,'Formulating SNR3 Estimation for f2...');

for n = 1:495

    index = cmbo3;      % Load unique index pattern matrix.
    waitbar2(n/495);

    temp = x(:,index(n,:)); % Select current index pattern.
    X = fft(temp);      % Perform 8 FFT's
```

```
    X = X(lower_bound_index:upper_bound_index,:);   % Restrict to search bound
    X_mag_squared = (X.*conj(X)).';
    X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.

    k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg))));
    if insight
        disp('Frequency found to be:')
        (k+20791)*100e6/64e3
        pause(10)
    end
    if (length(k) > 1)
        k = median(k);
        disp('ooops')
    end

    est_data(7,k) = est_data(7,k) + 1;  % Increment bin

end

est_data(7,:) = est_data(7,:) .* alpha; % Sponge data to look like a larger estimate

clear x
clear index;
close(h);


mlver = version;
if (mlver(1) == 5)      % Running on MATLAB 5.x
    save -v4 dama_cap Fs Fc snr est_tolerance N number_of_estimates ...
             est_data lower_bound_index upper_bound_index
else                 % Running on MATLAB 4.x
    save dama_cap Fs Fc snr est_tolerance N number_of_estimates ...
         est_data lower_bound_index upper_bound_index
end
```

ex1.m

```
%************************************************************************
%
% Experiment One: Finding Probabilities and MSE estimates of estimating
%                 a complex sinusoid in noise.
%
% Purpose: The purpose of this code is to determine and prove the
%          relationship between the DAMA curves and actual MSE of
%          estimation.  This baseline test will provide insight into the
%          simpler case of a complex sinusoid in noise which will then
%          be extended to the more complex DAMA carrier case.
%
% Programmer:       Brad Scaife
% Date:             2/14/99
% Revision Date:    4/20/99
% Current Revision: 1.10
% Revision History:
%                   1.0   - Baseline
%                   1.01  - Corrected Noise Power and theoretical curve
%                   1.02  - Fixed indexing problem and clearing of mse
%                           through each iteration.
%                   1.03  - Added support for estimation range compared
%                           to some baseline DFT resolution for comparison.
%                   1.10  - Increasing the computational resolution of the
%                           FFT so that the the sim curve can more closely
%                           match the Porat Curve.
%
% Notes:   See Porat.  As per indicated in Porat, the results of
%      estimation are valid only when the "rule of thumb" are
%      satisfied.  Thus any processing of snr's below the ROT are
%      not valid with the theoretical curve.
%************************************************************************
clear all
clc

% Program Parameters:
%*********************
N = 512;                      % - Normally set to 512.
L = 8*N;                      % - Computational Resolution.
Nbase = N;                    % - Basic DFT resolution.
fs = 800e3;                   % - Sampling frequency.
Ts = 1/fs;                    % - Sampling interval.
f = 200e3 + fs/(8*N);         % - Sinusoid frequency.
stime = 0; etime = (N-1)/fs;
phi = 0;                      % - Phase offset.
A = 1;                        % - Amplitude of sinusoid.
Jw = 1;                       % - Set as window function - see Porat.
D = Nbase*Ts;                 % - As per Porat.


snr = [-20:.5:30]';
number_estimates = 1000;

% Calculated Program Parameters
%*******************************
est_range = floor(fs/L)/2;    % - Frequency estimation range as a function of
                              %   basic DFT resolution.
f_lo = f - est_range;         % - Lower "accurate" estimation bound.
f_hi = f + est_range;         % - Upper "accurate" estimation bound.
k_lo = ceil(f_lo*N/fs);       % - Lower index bound.
k_hi = floor(f_hi*N/fs);      % - Upper index bound.
```

```
if N == Nbase
   k_hi = k_lo;
end

res_factor = N/L;           % - Resolution Factor

% Preallocate
%*************
success = zeros(1,length(snr));
mse_calc = zeros(1,length(snr));
mse = zeros(1,length(snr));
rmse = zeros(1,length(snr));
rmse_calc = zeros(1,length(snr));
mse_sum = 0;
%err_sum = 0;

% Signal Generation, Finding of true frequency
%**********************************************
s = csin_gen(f,phi,A,fs,stime,etime);    % Verified power = 1.
signal_power = cov(s);
S = fft(s);
S_mag = abs(S);
%true_ind = find(S_mag == max(S_mag))-1
%f_true = true_ind/N*fs
%pause
f_true = f;
% true_ind = floor(f_true*N/fs)

% Display Parameters
%*********************
disp(sprintf('True Frequency: %10.5f Hz',f_true))
disp(sprintf('Base Points: %d', N))
disp(sprintf('DFT Points: %d',L))
disp(sprintf('Base DFT Resolution: %10.5f Hz', fs/N))
disp(sprintf('Calculation DFT Resolution: %10.5f Hz', fs/L))
disp(sprintf('Resolution Factor: %d', res_factor))
disp(sprintf('Estimation Range: %10.5f Hz', est_range))
disp(sprintf('Lower Index Bound: %d', k_lo))
disp(sprintf('Upper Index Bound: %d', k_hi))
disp('Press a key to continue...') pause

for n = 1:length(snr)
   snr(n)                     % Current SNR to be tested.
   snr_mod = sqrt(cov(s)/(10^(snr(n)/10))); % Standard Deviation of noise
   % No = Ts * sqrt(2) * (snr_mod)^2;       % Noise Power in W/Hz.
   % No = Ts * (2 * snr_mod^2)^2;
   No = Ts * snr_mod^2;
   mse_calc(n) = (6*No*Jw)/((2*pi)^2 * A^2 * D^3);
   % rmse_calc(n) = sqrt( 6*Jw*sqrt( cov(s)/(10^(snr(n)/10) ) )/(100*pi^2))/D;
   % rmse_calc(n) = (1/D)*sqrt(6*Jw*PG/(100*pi^2));
   randn('seed',0);

   for k = 1:number_estimates
      v = [randn(N,1) + j*randn(N,1)]*snr_mod/sqrt(2);

      %cov(s)              % Remove me
      %cov(v)                      % Remove me
      %10*log10(cov(s) / cov(v))   % Remove me
      %pause                       % Remove me
      y = s + v;

```

```
        Y = fft(y,L);
        Y_mag = abs(Y);
        %Y_mag = Y_mag(1:N/2 + 1);

        max_ind = min(find(Y_mag == max(Y_mag))) - 1;
        f_found = max_ind/L*fs;

        mse_sum = mse_sum + (f_true - f_found)^2;
        % err_sum = err_sum + abs(f_true - f_found);

        if (max_ind >= k_lo & max_ind <= k_hi)
           success(n) = success(n) +1;
        end



    end

    % mse(n) = (err_sum / number_estimates)^2;
    mse(n) = mse_sum / number_estimates;
    rmse(n) = sqrt(mse(n));
    % err_sum = 0;
    mse_sum = 0;

    success(n) = success(n)/number_estimates;
end

rmse_calc = sqrt(mse_calc);
```

ex2.m

```
%**************************************************************************
%
% Experiment Two: Finding Probabilities and MSE estimates of estimating
%                 a real sinusoid in real noise.
%
% Purpose: The purpose of this code is to determine and prove the
%          relationship between the DAMA curves and actual MSE of
%          estimation.  This baseline test will provide insight into the
%          simpler case of a complex sinusoid in noise which will then
%          be extended to the more complex DAMA carrier case.
%
% Programmer:       Brad Scaife
% Date:             3/18/99
% Revision Date:    3/18/99
% Current Revision: 1.0
% Revision History:
%                   1.0   - Baseline
%
% Notes:   See Porat.  As per indicated in Porat, the results of
%      estimation are valid only when the "rule of thumb" are
%      satisfied.  Thus any processing of snr's below the ROT are
%      not valid with the theoretical curve.
%**************************************************************************
clear all clc

% Program Parameters:
%*********************
N = 65536;                  % - Normally set to 512.
Nbase = 65536;              % - Basic DFT resolution.
fs = 800e3;                 % - Sampling frequency.
Ts = 1/fs;                  % - Sampling interval.
f = 200e3 + fs/(4*N);       % - Sinusoid frequency.
stime = 0; etime = (N-1)/fs;
phi = 0;                    % - Phase offset.
A = 1;                      % - Amplitude of sinusoid.
Jw = 1;                     % - Set as window function - see Porat.
D = Nbase*Ts;               % - As per Porat.


snr = [-20:.2:10]';
number_estimates = 100;

% Calculated Program Parameters
%*******************************
est_range = floor(fs/Nbase)/2;     % - Frequency estimation range as a function of
                                   %   basic DFT resolution.
f_lo = f - est_range;              % - Lower "accurate" estimation bound.
f_hi = f + est_range;              % - Upper "accurate" estimation bound.
k_lo = ceil(f_lo*N/fs);            % - Lower index bound.
k_hi = floor(f_hi*N/fs);           % - Upper index bound.

if N == Nbase
   k_hi = k_lo;
end

res_factor = N/Nbase;                % - Resolution Factor

% Preallocate
%*************
success = zeros(1,length(snr));
```

```
mse_calc = zeros(1,length(snr));
mse = zeros(1,length(snr));
rmse = zeros(1,length(snr));
rmse_calc = zeros(1,length(snr));
mse_sum = 0;
%err_sum = 0;

% Signal Generation, Finding of true frequency
%*********************************************
s = sinu_gen(f,phi,A,fs,stime,etime);           % Verified power = 1.
signal_power = cov(s);
S = fft(s);
S_mag = abs(S);
%true_ind = find(S_mag == max(S_mag))-1
%f_true = true_ind/N*fs
%pause
f_true = f;
% true_ind = floor(f_true*N/fs)

% Display Parameters
%*******************
disp(sprintf('True Frequency: %10.5f Hz',f_true))
disp(sprintf('Base Points: %d', Nbase))
disp(sprintf('DFT Points: %d',N))
disp(sprintf('Base DFT Resolution: %10.5f Hz', fs/Nbase))
disp(sprintf('Calculation DFT Resolution: %10.5f Hz', fs/N))
disp(sprintf('Resolution Factor: %d', res_factor))
disp(sprintf('Estimation Range: %10.5f Hz', est_range))
disp(sprintf('Lower Index Bound: %d', k_lo))
disp(sprintf('Upper Index Bound: %d', k_hi))
disp('Press a key to continue...') pause

for n = 1:length(snr)
   snr(n)                                    % Current SNR to be tested.
   snr_mod = sqrt(cov(s)/(10^(snr(n)/10))); % Standard Deviation of noise
   % No = Ts * sqrt(2) * (snr_mod)^2;       % Noise Power in W/Hz.
   % No = Ts * (2 * snr_mod^2)^2;
   No = Ts * snr_mod^2;
   mse_calc(n) = (24*No*Jw)/((2*pi)^2 * A^2 * D^3);
   % rmse_calc(n) = sqrt( 6*Jw*sqrt( cov(s)/(10^(snr(n)/10) ) )/(100*pi^2))/D;
   % rmse_calc(n) = (1/D)*sqrt(6*Jw*PG/(100*pi^2));

   for k = 1:number_estimates
      v = randn(N,1) * snr_mod;

      % cov(s)                      % Remove me
      % cov(v)                      % Remove me
      % 10*log10(cov(s) / cov(v))   % Remove me
      % pause                       % Remove me
      y = s + v;

      Y = fft(y);
      Y_mag = abs(Y);
      Y_mag_p = Y_mag(1:N/2 + 1);

      max_ind = min(find(Y_mag_p == max(Y_mag_p))) - 1;
      f_found = max_ind/N*fs;

      mse_sum = mse_sum + (f_true - f_found)^2;
      % err_sum = err_sum + abs(f_true - f_found);

      if (max_ind >= k_lo & max_ind <= k_hi)
```

```
            success(n) = success(n) +1;
         end



      end

      % mse(n) = (err_sum / number_estimates)^2;
      mse(n) = mse_sum / number_estimates;
      rmse(n) = sqrt(mse(n));
      % err_sum = 0;
      mse_sum = 0;

      success(n) = success(n)/number_estimates;
   end

   rmse_calc = sqrt(mse_calc);
```

ex3.m

```
%***************************************************************************
%
% Experiment Three: Finding Probabilities and MSE estimates of estimating
%                   a BPSK carrier in real noise.
%
% Purpose: The purpose of this code is to determine and prove the
%          relationship between the DAMA curves and actual MSE of
%          estimation.  This test is an extension of experiments one
%          and two where a BPSK carrier is under test.
%
% Programmer:       Brad Scaife
% Date:             3/30/99
% Revision Date:    3/30/99
% Current Revision: 1.0
% Revision History:
%                   1.0   - Baseline
%
% Notes:   See Porat.  As per indicated in Porat, the results of
%      estimation are valid only when the "rule of thumb" are
%      satisfied.  Thus any processing of snr's below the ROT are
%      not valid with the theoretical curve.
%***************************************************************************
clear all clc
%--------------------
% Program Parameters
%--------------------
N = 512;
L = 2^0 * N;
A = 1;
Jw = 1;
fft_avg = 25;
% ------------------------------
% Communication System Parameters
% ------------------------------
Fs = 800e3;          % output signal sampling freq. (samples/s)
Fc = 178e3;          % BPSK carrier frequency in Hz. (cycles/sec)
kc = floor(Fc*L/Fs);
Rb = 10e3;           % data rate (bits/s) Fs/Rb must be integer
samples_per_bit = Fs/Rb;  % must be integer
D = N/Fs;
%----------------------
% Simulation Parameters
%----------------------
number_of_estimates = 1000;% # of frequency estimates to perform for each SNR typically 10000
snr = [-12:2:14]';
msg = zeros(N,1);
s = zeros(N,1);
r = zeros(N,1);
%-------------
% Display Info
%-------------
disp(sprintf('Sampling Frequency Fs: %5.15f',Fs));
disp(sprintf('Carrier Frequency Fc: %5.15f',Fc));
disp(sprintf('FFT Resolution (Data Supported): %5.15f',Fs/N));
disp(sprintf('FFT Computational Resolution: %5.15f',Fs/L));
```

```
pause

%--------------------------
% Begin Iterative SNR Loop
%--------------------------

for k = 1:length(snr)

   rand('seed',1000);
   randn('seed',0);
   mse_sum = 0;

   noise_power = (A^2/2)/(10^(snr(k)/10));
   No = (noise_power) / Fs;
   help_factor = 1;
   mse_calc(k) = (24*No*Jw)/(help_factor*(2*pi)^2 * (A^2/(4*Rb)) * D^3);

   for l = 1:number_of_estimates

      R_mag_sum = zeros(L/2,1);

      for n = 1:fft_avg
         %----------------------
         % Generate BPSK Signal
         %----------------------
         msg = A*[cos(2*pi*([0:N-1])*Fc/Fs)].';
         data = filter(ones(samples_per_bit,1),1,upsamp ...
         ((-1).^(round(rand(ceil(N/samples_per_bit),1))),...
         samples_per_bit));

         data = data(1:length(msg));
         s = msg .* data;


         noise = randn(length(msg),1) .* sqrt(cov(s)/(10^(snr(k)/10)));
         r = s + noise;
         r = r ./ sqrt(cov(r));



         R = fft(r,L);
         R_mag_sum = R_mag_sum + R(1:L/2).*conj(R(1:L/2));
      end

      R_mag = R_mag_sum ./ fft_avg;

      kmax = min(find(R_mag == max(R_mag)));
      f_est = (kmax-1)/L*Fs;

      mse_sum = mse_sum + (Fc - f_est)^2;
      %[Fc - f_est,kmax]

   end

   mse(k) = mse_sum/number_of_estimates;
   rmse(k) = sqrt(mse(k));
   rmse_calc(k) = sqrt(mse_calc(k));
end

clf
```

```
plot(snr,rmse_calc,'--')
hold on
plot(snr,rmse)
hold off
grid
```

B Motorola DSP 56303EVM Code

rev30.asm

```
; REV 3.0
; Just for convenience - delete later!!!  Turns on/off D/A codec
;   bclr    #19,x:M_CRB0    ;Disable Rx on A-codec
;   bclr    #18,x:M_CRB0    ;Used to disable Tx interrupt
    opt now
;   nolist
    include 'ioequ.asm'
    include 'intequ.asm'
    include 'ada_equ.asm'
    include 'vectors.asm'
    include '7819equ.asm'
;list
;*****************************************************************************
; Initial Layouts:  This section of code sets up the D/A memory resources,
;  the program memory resources and defines the FFT macro.
;*****************************************************************************
    include 'CS4215.asm'    ;D/A Memory Resources
    include 'fftr2cn.asm'   ;FFT Macro
    include 'convm.asm'     ;Convolution Macro
    include 'mlayout.dat'   ;Memory Layout
;*****************************************************************************
; Fast Interrupt - IRQB
;*****************************************************************************
    org pli:I_IRQB
    movep   y:BB7819_DR,x:(r0)+
    org pli:I_IRQB+1
    bset    #0,x:FLAGS
    org     p:$100
START
main
;*****************************************************************************
; Set Operating Frequency
;*****************************************************************************
        movep   #CLK_RATE,x:M_PCTL ;Set PLL and Chip Operating Frequency
;*****************************************************************************
; Set Operating Parameters of DSP56303
;*****************************************************************************
        move    #OP_MODE,omr       ;Set Operating Mode of 303
;*****************************************************************************
; Setup Stack
;*****************************************************************************
        movec   #0,sp              ;clear hardware stack pointer
        move    #STACK,r6          ;initialise stack pointer
        move    #-1,m6             ;linear addressing
;*****************************************************************************
```

```
; Set AAR Wait States for External Memory(32k) and A/D Codec
;*****************************************************************************
        include 'ws_set.asm'       ;Set Wait States
;*****************************************************************************
; Set Up IRQB Interrupt Parameters.  IRQB is the interrupt designated to
; the A/D Codec.
;*****************************************************************************
        ori     #$03,mr        ; Mask all interrupts until needed.
        include 'core_ipl.asm' ; Set IRQB Interrupt Parameters
AAR2    equ     $fffc21        ; Compare Upper 12 bits to fffxxx
        movep   #AAR2,x:M_AAR2 ; Setup AAR2
        jsr     INIT           ; Register Initialization Routine
        andi    #$fc,mr        ; Re-enable all interrupts
main_loop
        jclr    #0,x:FLAGS,*   ;Wait for Sample In
        bclr    #0,x:FLAGS
        jsr     process_sample
        jmp     main_loop
process_sample
        clr     a
        move    x:SAMPLE_COUNTER,a0
        dec     a
        move    a0,x:SAMPLE_COUNTER
        tst     a
        jne     GET_NEXT_SAMPLE
        ori     #$03,mr        ;Disable Interrupts
        jsr     WIN_N_SCALE
        jsr     COMPUTE_FFT
        jsr     AVG_FFT
        andi    #$fe,mr         ;Enables only 800k A/D
        movep   #IPRC,x:M_IPRC      ;Re-enable A/D Codec
GET_NEXT_SAMPLE
        rts
; Subroutines:
;**************
        include 'comp_fft.asm'
        include 'wsc.asm'
        include 'avg_fft.asm'
        include 'init.asm'
        include 'get_bin.asm'
        include 'sinwgid.asm'
        include 'ada_init.asm'
echo
end
```

mlayout.dat

```
;******************************************************************************
;mlayout.DAT: This data file is used with rev1.ASM to lay things out in memory
;
; Notes: For use with rev 2.1 code.
;******************************************************************************
; References:
;       DSP56300 Family Manual  (300FM)
;       DSP56303 User's Manual  (303UM)
;******************************************************************************
; Equates:
;**********

;******************************************************************************
; 56303 Processor Operating Parameters Control
;******************************************************************************
CLK_RATE        equ     $040004 ;Chip Operating Clock - See Section 9.3 300FM
OP_MODE         equ     $389    ;Chip Operating Mode
                                ;Please use either 389 or 3C9 for proper
                                ;operation.  Please see DAMA Programming Notes
                                ;and 303UM:3-13 for details.
FS              equ     32000   ;Please set the same as D/A sample rate.
;******************************************************************************
; DAMA Project Memory Settings. PLEASE DO NOT CHANGE!!!  CODE WILL LIKELY NOT
;  FUNCTION.  THE MEMORY HAS BEEN SPECIFICALLY SETUP UTILIZING ALL ON-CHIP
;  MEMORY.
;******************************************************************************
POINTS          equ     512     ;Number of Points (samples)
TABLE_SIZE      equ     512     ;Sine Wave Lookup Table Size (Will adjust output)
ITERS           equ     8       ; FFT Iterations
OFFSET          equ     128     ;Correction from Spectral Smearing due to Convolution.
;OUTPUT_SEC     equ     2       ;Please enter duration of output in seconds.
;OUT_LENGTH     equ     @CVI(FS*OUTPUT_SEC)

; Long Memory:
;**************
    org l:$000a

SAMPLE_DATA     dsm     POINTS  ;Signal buffer (0200 - 03ff)
FFT_DATA        dsm     POINTS  ;FFT Output buffer (0400 - 05ff)
;FFT_RESULT     dsm     POINTS  ;Result FFT Data  (0600 - 07ff)
COEFF           dsm     POINTS  ;Sine-Cosine "Twiddle" Factor Lookup (0800 - 09ff)

; X Memory:
;***********
    org x:$000a ;see ADA_INIT.ASM for why we start at x:$000a

SA_DATA_PTR     ds      1   ;SAMPLE_DATA Pointer Storage
FT_DATA_PTR     ds      1   ;FFT_DATA Pointer Storage
IFFT_PTR        ds      1   ;Imaginary FFT Data Pointer Storage
IFFT_MOD        ds      1   ;Imaginary FFT Data mod Storage
;FT_RES_PTR     ds      1   ;FFT_RESULT Pointer Storage
MAG_PTR         ds      1   ;Magnitude Squared Data Pointer Storage
COEFF_PTR       ds      1   ;Coeff Pointer Storage
WAV_PTR         ds      1   ;Sine Wave Table Pointer Storage
WIN_PTR         ds      1   ;Window Pointer Storage
WIN_MOD         ds      1   ;Window Modulo Storage
SMF_PTR         ds      1   ;SMF Pointer Storage
CNVO_PTR        ds      1   ;Convolution Result Buffer Pointer Storage
FFT_COUNTER     ds      1   ;FFT Counter
SAMPLE_COUNTER  ds      1   ;Sample Counter
```

```
MAX_VAL         ds      1   ;Maximum value storage
MAX_LOCATION    ds      1   ;Holds address of max location
INT_DELTA       ds      1   ;Delta for Carrier Reconstruction.
FRAC_DELTA      ds      1   ;
R0_STORE        ds      1   ;r0 storage
R1_STORE        ds      1   ;r1 storage
R2_STORE        ds      1   ;r2 storage
R3_STORE        ds      1   ;r3 storage
R4_STORE        ds      1   ;r4 storage
R5_STORE        ds      1   ;r5 storage
R7_STORE        ds      1   ;r7 storage
M0_STORE        ds      1   ;m0 storage
M1_STORE        ds      1   ;m1 storage
M2_STORE        ds      1   ;m2 storage
M3_STORE        ds      1   ;m3 storage
M4_STORE        ds      1   ;m4 storage
M5_STORE        ds      1   ;m5 storage
M7_STORE        ds      1   ;m7 storage
N5_STORE        ds      1   ;n5 storage
N7_STORE        ds      1   ;n7 storage
FLAGS           ds      1   ;User Defined Flag Register
OUT_COUNTER     ds      1   ;Output Sample Counter
CNV_MEM         ds      1   ;For use in convolutional code.
STACK           equ     *   ;Beginning of Stack

    org x:$800
;*********************************
; Magnitude Squared Data
;*********************************
MAG_SQ_DATA     dsm     POINTS/2

    org x:$A00
;*********************************
; Generate Sine Wave Lookup Table
;*********************************
TAB             dsm     TABLE_SIZE
    include 'sintab.asm'
    sintab  TABLE_SIZE,TAB

; Y Memory:
;***********
    org y:$0
;*************************************
; Generate Hamming Window w/ Prescale
;*************************************
HAMM            dsm     POINTS      ;Hamming Window table.
    include 'hamming.asm'
    hamming POINTS,HAMM


; Build Twiddle factor lookup tables for FFT Routine
;****************************************************
    include 'sincos.asm'    ;Twiddle factor macro - builds lookup tables
    sincos  POINTS,COEFF    ;Build lookup tables.


; Spectral Matched Filter
;*************************
SMF             dsm     POINTS/2
    org y:SMF
    include 'smf20.dat'

```

```

; Convolution Output
;********************
CNV_OUT dsm 2*POINTS-1
```

wsc.asm

```
;*****************************************************************************
; win_n_scale Subroutine
;
; Purpose: The purpose of this subroutine is to scale the input data to avoid
;  overflow problems.  This will have the effect of lowering the overall values
;  of the spectrum but will not alter the shape of the spectrum.
;
; In:       x:(r0) - Sample Buffer
;           y:(r4) - Hamming buffer
;
; OUT:      x:(r0) - Sample Buffer w/ Window, scale, and iteration
;           adjustments
;
; Alters:   a,b,r0,r3,r4,x0,x1,y0,y1
;
; Written By:   Brad Scaife
; Date:         2/20/98
; Platform:     Motorola DSP56303
; Calls:        None
;
; This code is verified for use with version three code. See rev30.asm.
;*****************************************************************************
WIN_N_SCALE


    move    x:(r0)+,x0 y:(r4)+,y0       ;Preload values.
    do      #POINTS/2-1,ND_SCALE        ;
    mpyr    x0,y0,a x:(r0)+,x1 y:(r4)+,y1   ;x'(x)*(w(n)*scale/# iterations
    mpyr    x1,y1,b x:(r0)+,x0 y:(r4)+,y0   ;Second Iteration
    move    a,x:(r3)+           ;Store a into sample buffer
    move    b,x:(r3)+           ;Store b into sample buffer
ND_SCALE
    mpyr    x0,y0,a x:(r0)+,x1 y:(r4)+,y1   ;Loop clean up: Two mults and
    mpyr    x1,y1,b             ;corresponding writes to memory
    move    a,x:(r3)+           ;Counters back to top of
    move    b,x:(r3)+           ;buffer

    rts
```

comp_fft.asm

```
;****************************************************************************
; COMPUTE FFT Subroutine
;
; Purpose: The purpose of this subroutine is to compute the FFT of the input
;  signal and store it in memory.
;
; In:       r0,r1,r2,r3,r4,r5,m0,m1,m2,m3,m4,m5
; Out:      x:(r1),r0,r1,r2,r3,r4,r5,m0,m1,m2,m3,m4,m5
; Alters:   Everything
;****************************************************************************
COMPUTE_FFT
    move    r0,x:R0_STORE
    move    r1,x:R1_STORE
    move    r2,x:R2_STORE
    move    r3,x:R3_STORE
    move    r4,x:R4_STORE
    move    r5,x:R5_STORE
    move    r7,x:R7_STORE
    move    m0,x:M0_STORE
    move    m1,x:M1_STORE
    move    m2,x:M2_STORE
    move    m3,x:M3_STORE
    move    m4,x:M4_STORE
    move    m5,x:M5_STORE
    move    m7,x:M7_STORE
    move    n5,x:N5_STORE
    move    n7,x:N7_STORE

    fftr2cn POINTS,SAMPLE_DATA,FFT_DATA,COEFF

    move    x:N7_STORE,n7
    move    x:N5_STORE,n5
    move    x:M7_STORE,m7
    move    x:M5_STORE,m5
    move    x:M4_STORE,m4
    move    x:M3_STORE,m3
    move    x:M2_STORE,m2
    move    x:M1_STORE,m1
    move    x:M0_STORE,m0
    move    x:R7_STORE,r7
    move    x:R5_STORE,r5
    move    x:R4_STORE,r4
    move    x:R3_STORE,r3
    move    x:R2_STORE,r2
    move    x:R1_STORE,r1
    move    x:R0_STORE,r0
    rts
```

avg_fft.asm

```
; for use with rev 2.1 code only AVG_FFT
        move   x:(r1)+,x0 y:(r7)+,y0
        do     #(POINTS/2),END_TLOOP
        mpy    x0,x0,a x:(r2),y1
        macr   y0,y0,a
        add    y1,a
        move   x:(r1)+,x0 y:(r7)+,y0
        move   a,x:(r2)+
END_TLOOP
        clr    b
        move   x:FFT_COUNTER,b0
        dec    b
        move   b0,x:FFT_COUNTER
        tst    b
        jseq   GET_MAX_BIN
;****************************************
; Prepare to perform next FFT iteration
;****************************************
        move   #POINTS,x1      ;Reload sample counter for next sample
        move   x1,x:SAMPLE_COUNTER ;buffering.
;       move   #FFT_RESULT,r2      ;re-Setup FFT Result ptr
        move   #MAG_SQ_DATA,r2     ;re-Setup Mag Squared data ptr
        move   #FFT_DATA,r1        ;Setup FFT Data ptr
        move   r1,r7               ;Imag. Pointer to FFT Buffer
        move   #$0,x0
        do     #POINTS,CLR_DAT     ;Clear FFT Data buffer
        move   x0,x:(r1)           ;Real
        move   x0,y:(r1)+          ;Imaginary
CLR_DAT
        do     #POINTS,CLR_SMP
        move   x0,x:(r0)
        move   x0,y:(r0)+
CLR_SMP
        rts
```

get_bin.asm

```
;******************************************************************************
; GET_MAX_BIN Subroutine
;
; Purpose: The purpose of this subroutine is to determine the frequency bin
;  that has the largest component and then to determine the delta for the
;  sine wave generation routine.
;
; In:       x:(r2)
; Out:      b
; Alters:   b,x1,y1,r2
;
; Notes:  For use with rev 2.1 code only!!
;******************************************************************************
GET_MAX_BIN

;*****************************************
; Clean Up From GET_BIN Subroutine
;*****************************************
        move    #POINTS,x1       ;Reload sample counter for next sample
        move    x1,x:SAMPLE_COUNTER ;buffering.
        move    #FFT_DATA,r1        ;re-Setup FFT Data ptr
        move    r1,r7            ;Imag. Pointer to FFT Buffer
;       move    #FFT_RESULT,r2      ;re-Setup FFT Result ptr
        move    #MAG_SQ_DATA,r2     ;re-Setup Mag Sq Data ptr
;******************************
; Perform SMF Convolution
;******************************
        move    r0,x:R0_STORE
        move    r1,x:R1_STORE
        move    r4,x:R4_STORE
        move    m0,x:M0_STORE
        move    m1,x:M1_STORE
        move    m4,x:M4_STORE

        convm   POINTS/2-1,MAG_SQ_DATA,SMF,CNV_OUT,CNV_MEM

        move    x:R0_STORE,r0
        move    x:R1_STORE,r1
        move    x:R4_STORE,r4
        move    x:M0_STORE,m0
        move    x:M1_STORE,m1
        move    x:M4_STORE,m4

        move    #CNV_OUT,r7
        move    #POINTS/2-2,m7
        clr     b
        do      #POINTS-1,ND_MAX
        move    x:(r7),x1        ;Bin Comparison
        cmp     x1,b             ;b-x1
        jlt     NEW_MAX          ;b will always hold max
        jmp     DUMMY

NEW_MAX
        move    x1,b                ;Store New Max Location
        move    r7,x:MAX_LOCATION
DUMMY
        move    (r7)+
        nop
ND_MAX
        move    x:MAX_LOCATION,b    ;Subtract max location from base
```

```
;
        NOP                                  ;- Reserved
        NOP
;
        jmp     *
        NOP                     ;- SCI Receive Data
;
        jmp     *
        NOP                     ;- SCI Receive Data w/ Exception Status
;
        jmp     *
        NOP                     ;- SCI Transmit Data
;
        jmp     *
        NOP                     ;- SCI Idle Line
;
        jmp     *
        NOP                     ;- SCI Timer
;
        NOP                                  ;- Reserved
        NOP
;
        NOP                                  ;- Reserved
        NOP
;
        NOP                                  ;- Reserved
        NOP
;
;
        jmp     *
        NOP                     ; Host receive data full
;
;
        jmp     *
        NOP                     ;- Host transmit data empty
;
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
```

```
;        move    #FFT_RESULT-1,y1    ;location to get the actual index
         move    #MAG_SQ_DATA-1,y1   ;location to get the actual index
         sub     y1,b                ;Equals index imax
         move    #OFFSET,y0          ;
         sub     y0,b                ;
         move    b1,n5               ;
         jsr     SINWGID             ;end program

         move    #>ITERS,x1          ;Init FFT Counter
         move    x1,x:FFT_COUNTER    ;Reset FFt counter for next iteration.
         rts
```

sinwgid.asm

```
;*****************************************************************************
; SINWGID Subroutine
;
; Purpose:  The purpose of this subroutine is to generate a tone at a frequency
;  based upon the delta value passed in from GET_BIN.ASM.
;
; In:        b
; Out:       n/a
; Alters:    a,x1,r5,n5,y0,
;
; Notes:
;   For use with rev 2.1 code.
;*****************************************************************************
SINWGID

        movep   #$0,x:M_IPRC        ;Disable A/D
        andi    #$fc,mr         ;Enable all interrupts

;*****************************************************************************
; Initialization: D/A Codec and Setup Control Words.  Only initialize the
; first time.
;*****************************************************************************
;       jset    #1,x:FLAGS,send_loop    ;Skip after initial pass
        jsr     ada_init                ; initialize codec

        move    #TONE_OUTPUT,y0         ;set up control words
        move    y0,x:TX_BUFF_BASE+2
        move    #TONE_INPUT,y0
        move    y0,x:TX_BUFF_BASE+3

;       bset    #1,x:FLAGS          ;Set after initialization.


send_loop
        jset    #2,x:M_SSISR0,* ;wait for frame sync to pass
        jclr    #2,x:M_SSISR0,* ;wait for frame sync

        move    x:(r5)+n5,y0

;       clr     a                   ;Test for end of duration of
;       move    x:OUT_COUNTER,a0    ;samples out phase.
;       dec     a
;       move    a0,x:OUT_COUNTER
;       tst     a
;       jeq     restart

        move    y0,x:TX_BUFF_BASE   ;transmit left
        move    y0,x:TX_BUFF_BASE+1 ;transmit right
        jmp     send_loop

restart
;       move    #OUT_LENGTH,x1      ;ReSet output duration counter.
;       move    x1,x:OUT_COUNTER    ;

        rts
```

ws_set.asm

```
;*****************************************************************************
; Wait State Parameter Settings Routine
;
; Purpose: The purpose of this routine is to set the Bus Control Register (BCR)
;  to the proper number of wait states required by each AAR device.  In the
;  DAMA Project, the AAR devices are:
;
;       32k SRAM
;       Burr-Brown Codec (Operating @ 800kHz for DAMA Project)
;
;*****************************************************************************
;Wait State Settings:
;********************
DEFAULT_WS      equu    $0f     ;default are wait states
SRAM_WS         equ     $0f     ;32KW SRAM
FLASH_WS        equ     $00     ;FLASH
PERIPH_WS       equ     $0f     ;A/D Peripheral board


AREA0           equ     SRAM_WS
AREA1           equ     FLASH_WS
AREA2           equ     PERIPH_WS
AREA3           equ     SRAM_WS

BBS             equ     $0      ;Bus State
BLH             equ     $0      ;Bus Lock Hold
BRH             equ     $0      ;Bus Request Hold


BCR             equ     (BBS<<21)+(BLH<<22)+(BRH<<23)+(DEFAULT_WS<<16&M_BDFW)\
                        +(AREA3<<13&M_BA3W)+(AREA2<<10&M_BA2W)+(AREA1<<5&M_BA1W)+(AREA0&M_BA0W)

                movep   #BCR,x:M_BCR    ;Initialize Bus Control Register
```

convm.asm

```
convm       macro   length,xcoefs,hcoefs,result,cnv_mem
convm       ident   1,0

;*************************************************************************
;
; Macro Name: CONVM.ASM
;-------------------------------------------------------------------------
; Purpose:    The purpose of this macro is to provide the convolution
;         of two sequences stored in memory.  The algorithm does
;         a nested structure to minimize the memory required.
;
; Programmer:       Brad Scaife
; Initial Date:     2/20/99
; Current Rev:      1.0
; Curr. Rev. Date:  2/20/99
; Revision History:
;
;           1.0     - Baseline
;
;-------------------------------------------------------------------------
; Legal Statement:
; This DSP56xxx macro may be freely used with out the permission
; of the author.  The author provides the code with the intent that
; it is not to be used where such use may endanger life and property.
; Use of this macro code releases the author from ANY litagation both
; past, present, and future from ANY and ALL such liability claims.
; Use of this code is expressly permitted at your own risk.
;-------------------------------------------------------------------------
; Resources Used:
;
; Registers Used:
;       a,b,r0,r1,r4,n0,n4,m0,m1,m4,x0,x1,y1
;
; Notes:
;    Please note that this revision of the code requires the two
; input sequences to be of equal length.
;*************************************************************************

K           equ     length
K_ALL       equ     2*K-1


            move    #xcoefs,r0
            move    #K-1,m0
            move    #hcoefs,r4
            move    #K-1,m4
            move    #result,r1
            move    #K_ALL,m1

; Begin Calculation

            move    #0,x0
            clr     b x0,x:CNV_MEM
            clr     a

            move    x:(r0),x0 y:(r4),y1
            do      #K_ALL/2+1,FIRST
            move    b0,x:CNV_MEM
            do      x:CNV_MEM,END_F
```

```
            mac     x0,y1,a x:(r0)-,x0 y:(r4)+,y1

END_F
            inc     b
            move    #hcoefs,r4
            macr    x0,y1,a
            move    b0,n0
            move    a,x:(r1)+
            move    #xcoefs,r0
            nop
            nop
            clr     a
            move    (r0)+n0
            move    x:(r0)-,x0 y:(r4)+,y1

FIRST

            dec     b
            dec     b
            move    #>1,x1
            move    x:(r0)-,x0 y:(r4)+,y1
            do      #K_ALL/2,LAST
            move    b0,x:CNV_MEM
            do      x:CNV_MEM,END_L
            mac     x0,y1,a x:(r0)-,x0 y:(r4)+,y1
END_L

            move    #hcoefs+1,r4
            move    x1,n4
            move    b0,x:(r6)+
            move    x1,b0
            inc     b
            macr    x0,y1,a
            move    b0,x1
            move    #xcoefs+K-1,r0
            move    x:-(r6),b0
            move    a,x:(r1)+
            dec     b
            clr     a
            move    (r4)+n4
            move    x:(r0)-,x0 y:(r4)+,y1


LAST

            endm
```

cs4215.asm

```
;---Buffer for talking to the CS4215

        org     x:0
RX_BUFF_BASE    equu    *
RX_data_1_2     ds      1   ;data time slot 1/2 for RX ISR
RX_data_3_4     ds      1   ;data time slot 3/4 for RX ISR
RX_data_5_6     ds      1   ;data time slot 5/6 for RX ISR
RX_data_7_8     ds      1   ;data time slot 7/8 for RX ISR

TX_BUFF_BASE    equ     *
TX_data_1_2     ds      1   ;data time slot 1/2 for TX ISR
TX_data_3_4     ds      1   ;data time slot 3/4 for TX ISR
TX_data_5_6     ds      1   ;data time slot 5/6 for TX ISR
TX_data_7_8     ds      1   ;data time slot 7/8 for TX ISR

RX_PTR          ds      1   ; Pointer for rx buffer
TX_PTR          ds      1   ; Pointer for tx buffer

TONE_OUTPUT     EQU     HEADPHONE_EN+LINEOUT_EN+(4*LEFT_ATTN)+(4*RIGHT_ATTN)
TONE_INPUT      EQU     MIC_IN_SELECT+(15*MONITOR_ATTN)
CTRL_WD_12      equ     NO_PREAMP+HI_PASS_FILT+SAMP_RATE_32+STEREO+DATA_16    ;CLB=0
CTRL_WD_34      equ     IMMED_3STATE+XTAL1_SELECT+BITS_64+CODEC_MASTER
CTRL_WD_56      equ     $000000
CTRL_WD_78      equ     $000000
```

init.asm

```
;******************************************************************************
;  INIT Subroutine
;
; Purpose: The purpose of this subroutine is to initialize pointers to memory
;  and clear out buffers.
;
; In:        none
; OUT:       none
; Alters:    r0,m0,r1,m1,r2,m2,r3,m3,r4,m4
;
; Notes: For use with rev 3.0 code.
;******************************************************************************
INIT
        move    #SAMPLE_DATA,r0     ;Setup Sample buffer ptr
        move    r0,r3               ;Alternate Sample buffer ptr
        move    #POINTS-1,m0        ;Setup Sample buffer mod
        move    m0,m3               ;Alternate Sample Buffer mod

        move    #$0,x0
        do      #POINTS,CLEAR_SAMPLE    ;Clear Sample Buffer
        move    x0,x:(r0)       ;Real
        move    x0,y:(r0)+      ;Imaginary
CLEAR_SAMPLE

        move    #FFT_DATA,r1        ;Setup FFT Data ptr
        move    r1,r7               ;Imag. Pointer to FFT Buffer
        move    #POINTS-1,m1        ;Setup FFT Data mod
        move    m1,m7               ;Imag. FFT Buffer mod
        do      #POINTS,CLEAR_DATA  ;Clear FFT Data buffer
        move    x0,x:(r1)           ;Real
        move    x0,y:(r1)+          ;Imaginary
CLEAR_DATA

;       move    #FFT_RESULT,r2      ;Setup FFT Result ptr
        move    #MAG_SQ_DATA,r2     ;
        move    #POINTS/2-1,m2      ;Setup FFT Result mod
        do      #POINTS/2,CLEAR_MAGSQ   ;Clear FFT Result buffer
        move    x0,x:(r2)           ;Real
        move    x0,y:(r2)+          ;Imaginary
CLEAR_MAGSQ

        move    r7,x:IFFT_PTR       ;Store IFFT for r7 reuse.
        move    m7,x:IFFT_MOD       ;
        move    #CNV_OUT,r7         ;Setup Convolution Output ptr
        move    #2*POINTS-2,m7      ;Setup Conv. Output mod.
        do      #2*POINTS-1,CLEAR_CNV   ;Clear Conv. Output
        move    x0,y:(r7)+
CLEAR_CNV
        move    x:IFFT_PTR,r7
        move    x:IFFT_MOD,m7

        move    #HAMM,r4            ;Setup Hamming ptr
        move    #POINTS-1,m4        ;Setup Hamming mod
        move    #TAB,r5             ;Sine Table
        move    #TABLE_SIZE-1,m5    ;Sine Table mod

        move    #POINTS,x1          ;Initialize Sample Counter
        move    x1,x:SAMPLE_COUNTER
        move    #>ITERS,x1          ;Init FFT Counter
        move    x1,x:FFT_COUNTER
```

```
        move    x0,x:FLAGS          ;Clear User Defined Flag Register

;       move    #OUT_LENGTH,x1      ;Set output duration counter.
;       move    x1,x:OUT_COUNTER    ;

        rts
```

hamming.asm

```
hamming  macro   points,hamm_loc
hamming  ident   1,2

py              equ     3.141592654
FREQ_INC        equ     2.0*py/@cvf(points-1)     ;frequency increment
SCALE_SHIFT     equ     @CVI(@log(@cvf(points))/@log(2.0)) ;Shifts to Produce 1/POINTS
;ITERATIONS     equ     2   ;Number of FFT Iterations to perform. There
                            ;exists a limit before overflow.
;SCALE_FAC      equ     @cvf(points)*@cvf(ITERATIONS) ;Scale Factor

    org y:hamm_loc
N               set     0
                dup     points
                dc      (0.54-0.46*@cos(FREQ_INC*@cvf(N)))/@cvf(points/2)
N               set     N+1
                endm
                endm    ;end of hamming macro
```

core_ipl.asm

```
;******************************************************************************
; Core Interrupt Priority Configuration Routine:
;
; Purpose:  The purpose of this routine is to set the core Interrupt
;  priorities.  For the DAMA Project, only IRQB need concern us presently.
;  Thus, only bits 5 to 3 are relevant. The following table suggests the
;  proper settings:
;
;   IBL2: 0 for level triggering, 1 for edge triggering (DAMA uses edge)
;   IBL1-0:
;                   00  01  10  11
;       -------------------------------------------
;       Enabled     No  Yes Yes Yes
;       Priority    -   0   1   2
;
; For details see 303UM:D-17.
;
; Current Settings:
;   Currently IRQB is the only interrupt enabled and it has been set to
;   priority level 2 (highest) and negative edge triggering.
;
; Written By:   Tim Bagget
; Adapted By:   Brad Scaife
; Date:     3/22/98
;
; Notes: For use with rev 2.1 code.
;******************************************************************************
; CORE Interrupt Priority and Configuration
IBM         equ     $1      ;IRQB trigger (0 level, 1 neg edge)
IBP         equ     $1      ;IRQB priority level 0, 1, or 2;

IBL         equ     (IBM<<2)+(IBP+1)&3

;IPRC       equ     IBL<<M_IBL0&M_IBL  ;Disabled for the time being.
IPRC        equ     $000038
            movep   #IPRC,x:M_IPRC     ;Initialize Interrupt Priority/Config
```

sintab.asm

```
sintab      macro   tasiz,tab_loc
sintab      ident   1,2

pie         equu    3.141592654
TAB_INC     equ     2.0*pie/@cvf(tasiz)

    org     x:tab_loc
N           set     0
            dup     tasiz
            dc      @sin(TAB_INC*@cvf(N))/2.0
N           set     N+1
            endm
            endm
                    ;end sine table generation macro
```

7819equ.asm

```
BB_ADR        equu    $0                ;DIP Switch Address SW1 ($0 - $3f)
BB7819_DR     equ     $FFFF80+BB_ADR ; ADS7819 Data Register
```

fftr2cn.asm

```
;
; This program originally available on the Motorola DSP bulletin board.
; It is provided under a DISCLAMER OF WARRANTY available from
; Motorola DSP Operation, 6501 Wm. Cannon Drive W., Austin, Tx., 78735.
;
; Radix 2, In-Place, Decimation-In-Time FFT (fast).
;
; Last Update 18 Aug 88   Version 1.0
;
fftr2cn  macro   points,data,odata,coef
fftr2cn  ident   1,0
;
; Radix 2 Decimation in Time In-Place Fast Fourier Transform Routine
;
;    Complex input and output data
;        Real data in X memory
;        Imaginary data in Y memory
;    Normally ordered input data
;    Normally ordered output data
;       Coefficient lookup table
;        -Cosine values in X memory
;        -Sine values in Y memory
;
; Macro Call - fftr2cn   points,data,odata,coef
;
;       points     number of points (16-32768, power of 2)
;       data       start of data buffer
;       odata      start of output data buffer
;       coef       start of sine/cosine table
;
; Alters Data ALU Registers
;       x1      x0      y1      y0
;       a2      a1      a0      a
;       b2      b1      b0      b
;
; Alters Address Registers
;       r0      n0      m0
;       r1      n1      m1
;               n2
;
;       r4      n4      m4
;       r5      n5      m5
;       r7      n7      m7
;
; Alters Program Control Registers
;       pc      sr
;
; Uses 6 locations on System Stack
;
; Latest Revision - 18 Aug-88
;
     move #data,r0             ;initialize input pointer
     move #points/4,n0         ;initialize input and output pointers offset
     move n0,n4                ;
     move n0,n7                ;initialize coefficient offset
     move #points-1,m0         ;initialize address modifiers
     move m0,m1                ;for modulo addressing
     move m0,m4
     move m0,m5
;
```

```
; Do first and second Radix 2 FFT passes, combined as 4-point butterflies
;
     move           x:(r0)+n0,x0
     tfr  x0,a      x:(r0)+n0,y1

     do   n0,_twopass
     tfr  y1,b      x:(r0)+n0,y0
     add  y0,a      x:(r0),x1                      ;ar+cr
     add  x1,b      r0,r4                          ;br+dr
     add  a,b       (r0)+n0                        ;ar'=(ar+cr)+(br+dr)
     subl b,a       b,x:(r0)+n0                    ;br'=(ar+cr)-(br+dr)
     tfr  x0,a      a,x0           y:(r0),b
     sub  y0,a                     y:(r4)+n4,y0    ;ar-cr
     sub  y0,b      x0,x:(r0)                      ;bi-di
     add  a,b                      y:(r0)+n0,x0    ;cr'=(ar-cr)+(bi-di)
     subl b,a       b,x:(r0)                       ;dr'=(ar-cr)-(bi-di)
     tfr  x0,a      a,x0           y:(r4),b
     add  y0,a                     y:(r0)+n0,y0    ;bi+di
     add  y0,b      x0,x:(r0)+n0                   ;ai+ci
     add  b,a                      y:(r0)+,x0      ;ai'=(ai+ci)+(bi+di)
     subl a,b                      a,y:(r4)+n4     ;bi'=(ai+ci)-(bi+di)
     tfr  x0,a                     b,y:(r4)+n4
     sub  y0,a      x1,b                           ;ai-ci
     sub  y1,b      x:(r0)+n0,x0                   ;dr-br
     add  a,b       x:(r0)+n0,y1                   ;ci'=(ai-ci)+(dr-br)
     subl b,a                      b,y:(r4)+n4     ;di'=(ai-ci)-(dr-br)
     tfr  x0,a                     a,y:(r4)+
_twopass
;
; Perform all next FFT passes except last pass with triple nested DO loop
;
     move #points/8,n1          ;initialize butterflies per group
     move #4,n2                 ;initialize groups per pass
     move #-1,m2                ;linear addressing for r2
     move #0,m7                 ;initialize C address modifier for
                                ;reverse carry (bit-reversed) addressing

     do   #@cvi(@log(points)/@log(2)-2.5),_end_pass  ;-1 ??? example: 7 passes for 1024 pt. FFT
     move #data,r0                                     ;initialize A input pointer
     move r0,r1
     move n1,r2
     move r0,r4                                        ;initialize A output pointer
     move (r1)+n1                                      ;initialize B input pointer
     move r1,r5                                        ;initialize B output pointer
     move #coef,r7                                     ;initialize C input pointer
     lua  (r2)+,n0                                     ;initialize pointer offsets
     move n0,n4
     move n0,n5
     move (r2)-                                        ;butterfly loop count
     move           x:(r1),x1      y:(r7),y0           ;x1=br,y0=wi,lookup -sine and -cosine values
     move           x:(r7)+n7,x0   y:(r0),b            ; b=ai, x0=wr,update C pointer, preload data
     mac  x1,y0,b                  y:(r1)+,y1           ;y1=bi,b=ai+br*wi,
     macr -x0,y1,b                 y:(r0),a            ; a=ai again,b=ai+br*wi-bi*wr=bi'

     do   n2,_end_grp
     do   r2,_end_bfy                                  ;loop BF-1 times and start Radix 2 DIT BF kernel
     subl b,a       x:(r0),b       b,y:(r4)            ;b=ar,a=ai-br*wi+bi*wr=ai',         PUT bi'
     mac  -x1,x0,b  x:(r0)+,a      a,y:(r5)            ;a=br,b=ar-br*wr,                   PUT ai'
     macr -y1,y0,b  x:(r1),x1                          ;b=ar-(br*wr+bi*wi)=br',x1=next br
     subl b,a       b,x:(r4)+      y:(r0),b           ;b=nai,a=2*ar-ar+br*wr+bi*wi=ar',   PUT br'
     mac  x1,y0,b                  y:(r1)+,y1          ;y1=nbi,b=nai+nbr*wi
     macr -x0,y1,b  a,x:(r5)+      y:(r0),a            ;a=nai,b=nai+nbr*wi-nbi*wr=nbi'     PUT ar'
```

```
_end_bfy
      move (r1)+n1                                         ;points to first B in next group
      subl b,a        x:(r0),b        b,y:(r4)             ; PUT last bi' in a group
      mac  -x1,x0,b   x:(r0)+n0,a     a,y:(r5)             ; PUT last ai' in a group and update A pointer
      macr -y1,y0,b   x:(r1),x1       y:(r7),y0
      subl b,a        b,x:(r4)+n4     y:(r0),b             ; PUT last br' in a group and update A' pointer
      mac  x1,y0,b    x:(r7)+n7,x0    y:(r1)+,y1           ; update W pointer
      macr -x0,y1,b   a,x:(r5)+n5     y:(r0),a             ; PUT last ar' in a group and update B' pointer
_end_grp
      move n1,b1
      lsr  b     n2,a1       ;divide butterflies per group by two
      lsl  a     b1,n1       ;multiply groups per pass by two
      move a1,n2
_end_pass
;
; Do last FFT pass
;
      move #2,n0             ;initialize pointer offsets
      move n0,n1
      move #points/4,n4      ;output pointer A offset
      move n4,n5             ;output pointer B offset
      move #data,r0          ;initialize A input pointer
      move #odata,r4         ;initialize A output pointer
      move r4,r2             ;save A output pointer
      lua  (r0)+,r1          ;initialize B input pointer
      lua  (r2)+n2,r5        ;initialize B output pointer
      move #0,m4             ;bit-reversed addressing for output ptr. A
      move m4,m5             ;bit-reversed addressing for output ptr. B
      move #coef,r7          ;initialize C input pointer
      move (r5)-n5           ;predecrement output pointer
      move            x:(r1),x1       y:(r7),y0 ;x1=br,y0=wi
      move            x:(r5),a        y:(r0),b ;a=?,b=ai

      do   n2,_lastpass      ;Radix 2 DIT butterfly kernel with one butterfly per group
      mac  x1,y0,b    x:(r7)+n7,x0    y:(r1)+n1,y1   ;b=ai+br*wi,x0=wr, y1=bi
      macr -x0,y1,b   a,x:(r5)+n5     y:(r0),a       ;b=ai+br*wi-bi*wr=bi',a=ai, PUT previous ar'
      subl b,a        x:(r0),b        b,y:(r4)       ;a=ai',b=ar,                PUT bi'
      mac  -x1,x0,b   x:(r0)+n0,a     a,y:(r5)       ;b=ar-br*wr,a=ar,           PUT ai'
      macr -y1,y0,b   x:(r1),x1       y:(r7),y0         ;b=br',x1=nbr,y0=nwi
      subl b,a        b,x:(r4)+n4     y:(r0),b       ;a=ar',b=nai,               PUT br'
_lastpass
      move            a,x:(r5)+n5                    ;                           PUT ar'
      endm
```

sincos.asm

```
;
; This program originally available on the Motorola DSP bulletin board.
; It is provided under a DISCLAMER OF WARRANTY available from
; Motorola DSP Operation, 6501 Wm. Cannon Drive W., Austin, Tx., 78735.
;
; Sine-Cosine Table Generator for FFTs.
;
; Last Update 25 Nov 86   Version 1.2
;
sincos  macro   points,coef
sincos  ident   1,2
;
;       sincos  -       macro to generate sine and cosine coefficient
;                       lookup tables for Decimation in Time FFT
;                       twiddle factors.
;
;       points  -       number of points (2 - 32768, power of 2)
;       coef    -       base address of sine/cosine table
;                       negative cosine value in X memory
;                       negative sine value in Y memory
;
; Latest revision - 25-Nov-86
;

pi              equ     3.141592654
freq            equ     2.0*pi/@cvf(points)

                org     x:coef
count           set     0
                dup     points/2
                dc      -@cos(@cvf(count)*freq)
count           set     count+1
                endm

                org     y:coef
count           set     0
                dup     points/2
                dc      -@sin(@cvf(count)*freq)
count           set     count+1
                endm

                endm    ;end of sincos macro
```

ada_equ.asm

```
        page    132,60
;*************************************************************************
;   ADA_EQU.ASM
;   Initialization constants to facilitate initialization of the CS4215
;
;   Copyright (c) MOTOROLA 1996
;               Semiconductor Products Sector
;               Digital Signal Processing Division
;
;*************************************************************************
;
NO_PREAMP       equu    $100000
LO_OUT_DRV      equ     $080000
HI_PASS_FILT    equ     $008000
SAMP_RATE_9     equ     $003800         ;  9.6 kHz sample rate
SAMP_RATE_48    equ     $003000         ; 48   kHz sample rate
SAMP_RATE_32    equ     $001800         ; 32   kHz sample rate
SAMP_RATE_27    equ     $001000
SAMP_RATE_16    equ     $000800
SAMP_RATE_8     equ     $000000
STEREO          equ     $000400
DATA_8LIN       equ     $200300
DATA_8A         equ     $200200
DATA_8U         equ     $200100
DATA_16         equ     $200000
IMMED_3STATE    equ     $800000
XTAL1_SELECT    equ     $100000         ; 24.576 MHz
XTAL2_SELECT    equ     $200000         ; 16.9344 MHz
BITS_64         equ     $000000
```

```
BITS_128        equ     $040000
BITS_256        equ     $080000
CODEC_MASTER    equ     $020000
CODEC_TX_OFF    equ     $010000
;CTRL_WD_12     equ NO_PREAMP+HI_PASS_FILT+SAMP_RATE_48+STEREO+DATA_16    ;CLB=0
;CTRL_WD_34     equ IMMED_3STATE+XTAL1_SELECT+BITS_64+CODEC_MASTER
;CTRL_WD_56     equ     $000000
;CTRL_WD_78     equ     $000000
HEADPHONE_EN    equ     $800000
LINEOUT_EN      equ     $400000
SPEAKER_EN      equ     $004000
MIC_IN_SELECT   equ     $100000
LEFT_ATTN       equ     $010000 ;63*LEFT_ATTN   = -94.5 dB, 1.5 dB steps
RIGHT_ATTN      equ     $000100 ;63*RIGHT_ATTN  = -94.5 dB, 1.5 dB steps
LEFT_GAIN       equ     $010000 ;15*LEFT_GAIN    = 22.5 dB, 1.5 dB steps
RIGHT_GAIN      equ     $000100 ;15*RIGHT_GAIN   = 22.5 dB, 1.5 dB steps
MONITOR_ATTN    equ     $001000 ;15*MONITOR_ATTN = mute,    6   dB steps
;OUTPUT_SET     equ HEADPHONE_EN+LINEOUT_EN+(LEFT_ATTN*4)
;INPUT_SET      equ MIC_IN_SELECT+(15*MONITOR_ATTN)+(RIGHT_ATTN*4)
```

vectors.asm

```
;
        page    132,60
;*****************************************************************************
;   VECTORS.ASM
;   Vector table for the 56303
;
;   Copyright (c) MOTOROLA 1996
;           Semiconductor Products Sector
;           Digital Signal Processing Division
;
;*****************************************************************************
;
        ORG     P:0
;
vectors JMP     START                           ; Hardware RESET
;
        jmp     *
        NOP                     ;  Stack Error

        jmp     *
        NOP                     ;- Debug Request Interrupt
;
        jmp     *
        NOP                     ;- Debug Request Interrupt
;
        jmp     *
        NOP                     ;- Trap
;
        jmp     *
        NOP                     ;- NMI
;
        NOP                             ;- Reserved
        NOP
;
        NOP                             ;- Reserved
        NOP
;
        jsr     main                    ;- IRQA
;
        jmp     *
        NOP                     ;- IRQB
;
        jmp     *
        NOP                     ;- IRQC
;
        jsr     echo                    ;- IRQD
;
        jmp     *
        NOP                     ;- DMA Channel 0
;
        jmp     *
        NOP                     ;- DMA Channel 1
;
        jmp     *
        NOP                     ;- DMA Channel 2
;
        jmp     *
        NOP                     ;- DMA Channel 3
;
        jmp     *
```

```
        NOP                     ;- DMA Channel 4
;
        jmp     *
        NOP                     ;- DMA Channel 5
;
        jmp     *
        NOP                     ;- Timer 0 Compare
;
        jmp     *
        NOP                     ;- Timer 0 Overflow
;
        jmp     *
        NOP                     ;- Timer 1 Compare
;
        jmp     *
        NOP                     ;- Timer 1 Overflow
;
        jmp     *
        NOP                     ;- Timer 2 Compare
;
        jmp     *
        NOP                     ;- Timer 2 Overflow
;
        jsr     ssi_rx_isr              ;- ESSI0 Receive Data
;
        jsr     ssi_rxe_isr             ;- ESSI0 Receive Data w/ Exception Status
;
        jsr     ssi_rxls_isr            ;- ESSI0 Receive last slot
;
        jsr     ssi_tx_isr              ;- ESSI0 Transmit Data
;
        jsr     ssi_txe_isr             ;- ESSI0 Transmit Data w/ Exception Status
;
        jsr     ssi_txls_isr            ;- ESSI0 Transmit last slot
;
        NOP                             ;- Reserved
        NOP
;
        NOP                             ;- Reserved
        NOP
;
        jmp     *
        NOP                     ;- ESSI1 Receive Data
;
        jmp     *
        NOP                     ;- ESSI1 Receive Data w/ Exception Status
;
        jmp     *
        NOP                     ;- ESSI1 Receive last slot
;
        jmp     *
        NOP                     ;- ESSI1 Transmit Data
;
        jmp     *
        NOP                     ;- ESSI1 Transmit Data w/ Exception Status
;
        jmp     *
        NOP                     ;- ESSI1 Transmit last slot
;

        NOP                             ;- Reserved
        NOP
```

```
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *

        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
```

```
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
NOP                     ; Available for Host Command
jmp     *
```

```
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
        jmp     *
        NOP                     ; Available for Host Command
;
;
```

intequ.asm

```
;******************************************************************************
;
;     EQUATES for ONYXE 56302 interrupts
;
;     Last update: June 11 1995
;
;******************************************************************************

    page    132,55,0,0,0
    opt mex
intequ  ident   1,0

    if  @DEF(I_VEC)
    ;leave user definition as is.
    else
I_VEC   equ $0
    endif

;------------------------------------------------------------------------
; Non-Maskable interrupts
;------------------------------------------------------------------------
I_RESET  EQU  I_VEC+$00   ; Hardware RESET
I_STACK  EQU  I_VEC+$02   ; Stack Error
I_ILL    EQU  I_VEC+$04   ; Illegal Instruction
I_DBG    EQU  I_VEC+$06   ; Debug Request
I_TRAP   EQU  I_VEC+$08   ; Trap
I_NMI    EQU  I_VEC+$0A   ; Non Maskable Interrupt

;------------------------------------------------------------------------
; Interrupt Request Pins
;------------------------------------------------------------------------
I_IRQA   EQU  I_VEC+$10   ; IRQA
I_IRQB   EQU  I_VEC+$12   ; IRQB
I_IRQC   EQU  I_VEC+$14   ; IRQC
I_IRQD   EQU  I_VEC+$16   ; IRQD

;------------------------------------------------------------------------
; DMA Interrupts
;------------------------------------------------------------------------
I_DMA0   EQU  I_VEC+$18   ; DMA Channel 0
I_DMA1   EQU  I_VEC+$1A   ; DMA Channel 1
I_DMA2   EQU  I_VEC+$1C   ; DMA Channel 2
I_DMA3   EQU  I_VEC+$1E   ; DMA Channel 3
I_DMA4   EQU  I_VEC+$20   ; DMA Channel 4
I_DMA5   EQU  I_VEC+$22   ; DMA Channel 5

;------------------------------------------------------------------------
; Timer Interrupts
;------------------------------------------------------------------------
I_TIM0C  EQU  I_VEC+$24   ; TIMER 0 compare
I_TIM0OF EQU  I_VEC+$26   ; TIMER 0 overflow
I_TIM1C  EQU  I_VEC+$28   ; TIMER 1 compare
I_TIM1OF EQU  I_VEC+$2A   ; TIMER 1 overflow
I_TIM2C  EQU  I_VEC+$2C   ; TIMER 2 compare
I_TIM2OF EQU  I_VEC+$2E   ; TIMER 2 overflow

;------------------------------------------------------------------------
; ESSI Interrupts
;------------------------------------------------------------------------
I_SI0RD  EQU  I_VEC+$30   ; ESSI0 Receive Data
```

```
I_SIORDE EQU  I_VEC+$32   ; ESSIO Receive Data With Exception Status
I_SIORLS EQU  I_VEC+$34   ; ESSIO Receive last slot
I_SIOTD  EQU  I_VEC+$36   ; ESSIO Transmit data
I_SIOTDE EQU  I_VEC+$38   ; ESSIO Transmit Data With Exception Status
I_SIOTLS EQU  I_VEC+$3A   ; ESSIO Transmit last slot
I_SI1RD  EQU  I_VEC+$40   ; ESSI1 Receive Data
I_SI1RDE EQU  I_VEC+$42   ; ESSI1 Receive Data With Exception Status
I_SI1RLS EQU  I_VEC+$44   ; ESSI1 Receive last slot
I_SI1TD  EQU  I_VEC+$46   ; ESSI1 Transmit data
I_SI1TDE EQU  I_VEC+$48   ; ESSI1 Transmit Data With Exception Status
I_SI1TLS EQU  I_VEC+$4A   ; ESSI1 Transmit last slot
;------------------------------------------------------------------------
; SCI Interrupts
;------------------------------------------------------------------------
I_SCIRD  EQU  I_VEC+$50   ; SCI Receive Data
I_SCIRDE EQU  I_VEC+$52   ; SCI Receive Data With Exception Status
I_SCITD  EQU  I_VEC+$54   ; SCI Transmit Data
I_SCIIL  EQU  I_VEC+$56   ; SCI Idle Line
I_SCITM  EQU  I_VEC+$58   ; SCI Timer
;------------------------------------------------------------------------
; HOST Interrupts
;------------------------------------------------------------------------
I_HRDF   EQU     I_VEC+$60   ; Host Receive Data Full
I_HTDE   EQU     I_VEC+$62   ; Host Transmit Data Empty
I_HC     EQU     I_VEC+$64   ; Default Host Command
;------------------------------------------------------------------------
; INTERRUPT ENDING ADDRESS
;------------------------------------------------------------------------
I_INTEND EQU  I_VEC+$FF   ; last address of interrupt vector space
         LIST
```

sincos.asm

```
;
; This program originally available on the Motorola DSP bulletin board.
; It is provided under a DISCLAMER OF WARRANTY available from
; Motorola DSP Operation, 6501 Wm. Cannon Drive W., Austin, Tx., 78735.
;
; Sine-Cosine Table Generator for FFTs.
;
; Last Update 25 Nov 86   Version 1.2
;
sincos  macro   points,coef
sincos  ident   1,2
;
;       sincos  -       macro to generate sine and cosine coefficient
;                       lookup tables for Decimation in Time FFT
;                       twiddle factors.
;
;       points  -       number of points (2 - 32768, power of 2)
;       coef    -       base address of sine/cosine table
;                       negative cosine value in X memory
;                       negative sine value in Y memory
;
; Latest revision - 25-Nov-86
;

pi      equ     3.141592654
freq    equ     2.0*pi/@cvf(points)

        org     x:coef
count   set     0
        dup     points/2
        dc      -@cos(@cvf(count)*freq)
count   set     count+1
        endm

        org     y:coef
count   set     0
        dup     points/2
        dc      -@sin(@cvf(count)*freq)
count   set     count+1
        endm

        endm    ;end of sincos macro
```

ada_init.asm

```
	page	132,60
;***************************************************************************
;	ADA_INIT.ASM	Ver.2.0
;	Example program to initialize the CS4215
;
;	Copyright (c) MOTOROLA 1995, 1996
;			Semiconductor Products Sector
;			Digital Signal Processing Division
;
;		History:
;			14 June 1996:  RLR/LJD - ver.1.0
;***************************************************************************

; PLEASE NOTE: For use with rev 2.1 code.
;***************************************************************************
;
;	portc usage:
;	bit8: SSI TX (from DSP to Codec)
;	bit7:
;	bit6:
;	bit5:
;	bit4: codec reset (from DSP to Codec)
;	bit3:
;		bit2: data/control bar
;			0=control
;			1=data
;
;***************************************************************************
;*****		initialize the CS4215 codec			*****
;***************************************************************************
;
;
;  PROGRAM OUTLINE:
;
;1 program fsync and sclk == output
;2 write pc0 = 0 (control mode)
;3 send 64 bit frame x times, with dcb bit = 0, keep doing until read back as 0
;4 send 64 bit frame x times, with dcb bit = 1, keep doing until read back as 1
;5 re-program fsync and sclk == input
;6 write pc0 = 1 (data mode)
;7 receive/send data (echo slots 1,2,3,4; slots 5,6,7,8 == constants)
;
;***************************************************************************
;
;	initialize ssi -- fsync and sclk ==> outputs
;
	org	p:
ada_init
	movep	#$0000,x:M_PCRC		; turn off ESSI0 port (for now)
	movep	#$103807,x:M_CRA0	; 40MHz/16 = 2.5MHz SCLK, WL=16 bits, 4W/F
	movep	#$ff313C,x:M_CRB0	; RIE,TIE,RLIE,TLIE,RE,TE,sc2/sck outputs
	movep	#$0003,x:M_PRRC		; setup pd0 and pd1 as gpio output
	movep	#$0,x:M_PDRC		; send out a 0 on DC~ and RST_CODEC~

	  ;----reset delay for codec ----
	do	#1000,_delay_loop
	rep	#2000			; 100 us delay (assuming 40MHz VCO)
	nop
_delay_loop
```

```
        bset    #0,x:M_PDRC             ; sends out a 1 on pd0 (rst_codec=1)
        movep   #$0008,x:M_IPRP         ; set interrupt priority level for ESSI0 to 1
        andi    #$FC,mr                 ; enable interrupts

;******************************************************************************
; The following data sets up the CS4215 control mode data:
;    (CTS = Control Time Slot, U/LN = upper/lower Nibble)
;
;       +------ CTS1-UN:  0      0      1      MLB      0 0 0 0
;       |+----- CTS1-LN:  OLB    CLB    X      X        0 0 0 0
;       ||+---- CTS2-UN:  HPF    X      DFR2   DFR1     0 0 1 0
;       |||+--- CTS2-LN:  DFR0   ST     DF1    DF0      1 1 0 0
; x0 = $002Cxx
;
;       +------ CTS3-UN:  ITS    MCK2   MCK1   MCK0     1 0 0 0
;       |+----- CTS3-LN:  BSEL1  BSEL0  XCLK   XEN      1 0 0 0
;       ||+---- CTS4-UN:  TEST   TEST   TEST   TEST     (TEST MUST BE 0)
;       |||+--- CTS4-LN:  TEST   TEST   ENL    DAD      0 0 0 0
; x0 = $8800xx
;******************************************************************************

;--- set up buffer with control mode data
        move    #CTRL_WD_12,x0
        move    x0,x:TX_BUFF_BASE
        move    #CTRL_WD_34,x0
        move    x0,x:TX_BUFF_BASE+1
        move    #CTRL_WD_56,x0
        move    x0,x:TX_BUFF_BASE+2
        move    #CTRL_WD_78,x0
        move    x0,x:TX_BUFF_BASE+3

        movep   #$003C,x:M_PCRC         ;turn on ESSI0 except for sc0 and sc2

;
; CLB == 0
;
        jclr    #3,x:M_SSISR0,*         ; wait until rx frame bit==1
        jset    #3,x:M_SSISR0,*         ; wait until rx frame bit==0
        jclr    #3,x:M_SSISR0,*         ; wait until rx frame bit==1
        jset    #18,x:RX_BUFF_BASE,*    ; loop until CLB set

;
; CLB == 1
;
        bset    #18,x:TX_BUFF_BASE      ;set CLB
        do      #4,_init_loopB
        jclr    #2,x:M_SSISR0,*         ; wait until tx frame bit==1
        jset    #2,x:M_SSISR0,*         ; wait until tx frame bit==0
_init_loopB
        movep   #$0000,x:M_PCRC         ; disable ESSI0

;******************************************************************************
;    now CLB should be 1 -- re-program fsync and sclk direction (i/p) -- also,
;  circular buffer pointers for echoing data r0=current, r1=old data to send
;  1 frame later
;
        movep   #$103807,x:M_CRA0       ; 40MHz/16 = 2.5MHz SCLK, WL=16 bits, 4W/F
        movep   #$FF310C,x:M_CRB0       ; sckd and fsync (sc02) as inputs
        movep   #$0003,x:M_PDRC         ; D/C~ pin = 1 ==> data mode
        movep   #$003C,x:M_PCRC         ; turn on ESSI0 except for sc0 and sc2
        rts

```

```
;******************************************************************************
;   SSIO_ISR.ASM    Ver.2.0
;   Example program to handle interrupts through
;       the 56303 SSIO to move audio through the CS4215
;
;   Copyright (c) MOTOROLA 1995, 1996
;             Semiconductor Products Sector
;             Digital Signal Processing Division
;
;       upon entry:
;               R6 must be the stack pointer
;       corrupts:
;               R6
;
;       History:
;               14 June 1996: RLR/LJD - ver 1.0
;******************************************************************************


;----the actual interrupt service routines (ISRs)  follow:

;************************ SSI TRANSMIT ISR *********************************
ssi_txe_isr
        bclr    #4,x:M_SSISR0           ; Read SSISR to clear exception flag
                                        ; explicitly clears underrun flag
ssi_tx_isr
        move    r0,x:(r6)+              ; Save r0 to the stack.
        move    m0,x:(r6)+              ; Save m0 to the stack.
        move    #3,m0                   ; Modulus 4 buffer.
        move    x:TX_PTR,r0             ; Load the pointer to the tx buffer.
        nop
        movep   x:(r0)+,x:M_TX00        ; SSI transfer data register.
        move    r0,x:TX_PTR             ; Update tx buffer pointer.
        move    x:-(r6),m0              ; Restore m0.
        move    x:-(r6),r0              ; Restore r0.
        rti

;******************** SSI TRANSMIT LAST SLOT ISR *************************
ssi_txls_isr
        move    r0,x:(r6)+              ; Save r0 to the stack.
        move    #TX_BUFF_BASE,r0        ; Reset pointer.
        move    r0,x:TX_PTR             ; Reset tx buffer pointer just in
                                        ; case it was corrupted.
        move    x:-(r6),r0              ; Restore r0.
        rti

;************************** SSI receive ISR ********************************
ssi_rxe_isr
        bclr    #5,x:M_SSISR0           ; Read SSISR to clear exception flag
                                        ; explicitly clears overrun flag
ssi_rx_isr
        move    r0,x:(r6)+              ; Save r0 to the stack.
        move    m0,x:(r6)+              ; Save m0 to the stack.
        move    #3,m0                   ; Modulo 4 buffer.
        move    x:RX_PTR,r0             ; Load the pointer to the rx buffer.
        nop
        movep   x:M_RX0,x:(r0)+         ; Read out received data to buffer.
        move    r0,x:RX_PTR             ; Update rx buffer pointer.
        move    x:-(r6),m0              ; Restore m0.
        move    x:-(r6),r0              ; Restore r0.
        rti

```

```
;********************** SSI receive last slot ISR **************************
ssi_rxls_isr
        move    r0,x:(r6)+              ; Save r0 to the stack.
        move    #RX_BUFF_BASE,r0        ; Reset rx buffer pointer just in
                                        ; case it was corrupted.
        move    r0,x:RX_PTR             ; Update rx buffer pointer.
        move    x:-(r6),r0              ; Restore r0.
        rti
```

References

[1] Stephen Horan. An operational concept for a demand assignment multiple access system for the space network. Technical report, New Mexico State University, 1996.

[2] Phillip L. De Leon. Real-time dsp-based carrier recovery with unknown doppler shift. Technical report, New Mexico State University, 1998.

[3] Monica M. Sanchez. Doppler extraction for a demand assignment multiple access service for NASA's space network. Technical report, New Mexico State University, August 1996.

[4] Boaz Porat. *A Course in Digital Signal Processing.* John Wiley and Sons, Inc., first edition, 1997.

[5] NASA. Tdrss system description. world wide web, 1998.

[6] Franklin Miller. *College Physics.* Harcourt Brace Jovanovich, Inc., fourth edition, 1977.

[7] John G. Proakis. *Digital Communications.* McGraw-Hill, Inc., third edition, 1995.

[8] Leon W. Couch II. *Digital and Analog Communication Systems.* Prentice-Hall, Inc., fifth edition, 1997.

[9] Simon Haykin. *Adaptive Filter Theory.* Prentice-Hall, Inc., third edition, 1996.

[10] Frank Hartman and Cliff Baxter. Private communication, July 1998.

[11] Motorola. *DSP 36303 User's Manual.* Motorola, Inc., first edition, 1995.

[12] Motorola. *DSP 56300 Family Manual.* Motorola, Inc., second edition, 1995.

[13] Tim Bagget. High speed A/D interface for carrier doppler tracking. Master's thesis, New Mexico State University, 1998.

[14] Sophocles J. Orfanidis. *Introduction to Signal Processing.* Prentice-Hall, Inc., first edition, 1996.