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

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

Born in  
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


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
## 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 AgainstSimulation ................. 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: Pseudo-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 approxima-
tions, 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.
• 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
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-
Techniques 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

\[ PG = \frac{R_c}{R_b} \]  

(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
sequence, $C_i$, has the property that it approximates a white noise sequence and is periodic. $C_i$ is designed purposely such that

$$C_i^T C_j = \delta(i - j)$$

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 $C_i \in \{0, 1\}$, 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:

$$BW \propto 2R_c$$

(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 perform carrier estimation.
For the purposes of this thesis, we will describe DAMA power in two different ways. The first will be SNR in $\text{dB}$ 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 $C_b/N_0$ 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

$$C_b/N_0 = 45\text{dB} \approx \text{SNR} = 2\text{dB}.$$  \hspace{1cm} (2.4)

In this thesis, we shall use $C_b/N_0$ to describe results of actual signals and SNR when describing simulation results. To avoid confusion when comparing the two, we shall map SNR to $C_b/N_0$.

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, \( R_c = 100 \text{ kbps} \). 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

\[
 f_s \geq 600\text{kHz}. \quad (2.5)
\]

A commonly available, inexpensive 800 kHz A/D was found that matched the requirements and we thus chose \( f_s = 800 \text{ kHz} \). 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 \( \pm 3 \text{ kHz} \) 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

$$X(k) = \frac{1}{P} \left\{ \sum_{p=1}^{P} \left[ \sum_{n=0}^{N-1} x[n + pN] e^{-j2\pi kn/N} \right]^2 \right\}$$  \hspace{1cm} (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
\[
N = \frac{f_s}{\Delta f} = \frac{800kHz}{3kHz} \approx 267
\] (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
\[
\Delta f = \frac{f_s}{N} = \frac{800kHz}{512} = 1562.5Hz
\] (2.8)
This 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
\[
h(t) = s(T - t)
\] (2.9)
with
\[
0 \leq t < T
\]
20
where \( s(t) \) is the time reversed equivalent of the received signal. The matched filter is then convolved with the received signal yielding

\[
y(t) = \int_0^t s(\tau)s(T - t + \tau)\,d\tau.
\]

(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

\[
H(k) = X(N - k), \quad 0 \leq k < N
\]

(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

\[
X_{smf} = X \ast H
\]

(2.12)

where \( \ast \) 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

\[
\hat{f} = \frac{\arg \max(X_{smf})}{N} f_s
\]

(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

\[ L_{X_{smf}} = L_H + M \]  \hspace{1cm} (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 \( \text{arg max}(X_{smf}) \). 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, \( \omega_c \) 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
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), $S(e^{j\omega})$ 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

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

Figure 3.1: Signal Space Diagram for BPSK
\[
N-1 \sum_{n=0}^{N-1} s(n)e^{-j\omega n} + \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega n} = S(e^{j\omega n}) + \sum_{n=0}^{N-1} v(n)w(n)e^{-j\omega n}
\]

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

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

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

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

and taking the magnitude-squared of both sides,

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

Taking the expectation of the magnitude-squared value yields,

\[E[|X(\omega_c)|^2] = E \left[ |S(e^{j\omega_c})|^2 \right] \]
The first term is by definition as provided in (3.3) and is evaluated as

\[ P_s(e^{j\omega}) = \frac{A^2}{4R} \]

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

\[ E[|X(\omega_c)|^2] = \frac{A^2}{4R} + \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} w(n)w(m)\gamma_v \delta(n - m)e^{-j\omega_c(n-m)} \]
\[ = \frac{A^2}{4R} + \gamma_v \sum_{n=0}^{N-1} w^2(n) \]
\[ = \frac{A^2}{4R} + \gamma_v N \]

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

\[ SNR_O = \frac{A^2}{4R \gamma_v N} \] (3.5)

We define the input SNR as

\[ SNR_I = \frac{A^2}{2 \gamma_v} \] (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

\[
W_g = \frac{SNR_O}{SNR_{1/2}N} \tag{3.7}
\]

\[
= \frac{2}{N} \left[ \frac{A^2}{4R\gamma_N A^2 \gamma_N} \right] \tag{3.8}
\]

\[
= \frac{1}{R N^2} \tag{3.9}
\]

We next employ a rule of thumb which states

\[
\frac{NA^2W_g}{\gamma_N} \geq 100 \tag{3.10}
\]

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

\[
\frac{A^2}{RN\gamma_N} \geq 100 \tag{3.11}
\]

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

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

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 \( E[\hat{f}_c - f_c] = 0 \).  

28
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

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

(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

$$\max(f_c - \hat{f}_c) < \frac{f_s}{N}$$  \hspace{1cm} (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
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.
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 our 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.
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

<table>
<thead>
<tr>
<th>PG</th>
<th>$F_c$ MHz</th>
<th>40 dB</th>
<th>45 dB</th>
<th>50 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>32.60</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32.65</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>32.70</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>32.80</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>32.60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32.65</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>32.70</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32.80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>32.60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32.65</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32.70</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32.80</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* 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

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

(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
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 $C_b/N_0 = 40$ 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 negligible 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
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
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 $\pm 50$ 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 ver-
sion 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 of 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
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 DSP56xEVJ, 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 $x(n) \in \mathbb{R}$ 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.
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

% 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 DAHA 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_DAHA_DATA.M to visualize data

% Notes:
% 1) For n MA users DAMA-to-MA power ratio is
% 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 Fc_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 +60% max
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)
069 \( R_c \cdot MA = 3000000; \) \( \% \) chip rate for MA (chips/s) 300K

061 \% -------------------------------
062 % Frequency Estimation Parameters
063 \% -------------------------------
064 \% Estimation of \( F_c \) occurs over a bandwidth of...
065 \( F_c = \max\_Doppler - \text{main\_lobe\_BW}/2 < F_{\text{DAMA}} < F_c + \max\_Doppler + \text{main\_lobe\_BW}/2 \)
066 \( 0 < f < 328 \)
067 \( N = 512; \) \% length data block to take FFT over
068 window = 0; \% if 0 assume rectangular window, 1 Hamming Window
069 if (window)
070 \( w = \text{hamming}(N); \) \% Hamming window used in FFT
071 end;
072
073 \%----------------------
074 % Simulation Parameters
075 \%----------------------
076 \text{number\_of\_estimates} = 30; \% number of frequency estimates to perform
077 \% for each SNR typically 10000
078 \text{num\_FFTs} = 8; \% average 8 FFTs per estimate;
079 \text{snr} = [-6]'; \% vector of SNRs (in dB) to simulate typically from -2.2 to 4.7dB
080
081 \%----------------------
082 % Simulation
083 \%----------------------
084 \text{est\_data} = \text{zeros}(\text{length}(\text{snr}),\text{N}/2); \% malloc storage for estimation data
085
086 for \( m = 1:\text{length}(\text{snr}); \) \% m is index for SNRs
087 \text{disp}(['\%\%\% SNR \= ',\text{snr(m)}]);
088 \text{max\_pos\_est\_error} = 0; \% reset variables
089 \text{max\_neg\_est\_error} = 0;
090 \text{rand('seed',0); \% reset uniform generator seed}
091 \text{randn('seed',0); \% reset normal generator seed}
092
093 for \( n = 1:\text{number\_of\_estimates}; \) \% n is index for estimate #
094 \text{n;}
095 \text{sum\_mag\_X\_sqrd} = \text{zeros}(\text{N}/2,1); \% malloc storage for accumulation of FFT data
096 \text{sum\_mag\_S\_sqrd} = \text{zeros}(\text{N}/2,1); \% malloc storage for accumulation of FFT data
097 \% ----------------------
098 \% Synthesize SS signal
099 \% ----------------------
100 \% Generate digital message
101 \text{data} = \text{round}((\text{rand}(\text{ceil}(\text{num\_FFTs}\cdot\text{N}/\text{samples\_per\_bit}),1)); \% generate data (bits) to modulate
102 \% Direct Sequence Spreading
103 start = 0;
104 \text{stop} = \text{samples\_per\_bit} - 1;
105 \text{msg} = [];
106 for \( i = 1:\text{length}(\text{data})\)
107 \text{n = [start:stop];}
108 if ('\text{data(i)}') \% if 0 multiply by -1
109 \text{msg} = ['msg' '(-1)*sin(2*pi*n*Fc/Fs)']; \% build up message signal
110 else \% if 1 multiply by 1
111 \text{msg} = ['msg' 'sin(2*pi*n*Fc/Fs)']; \% build up message signal
112 end;
113 \text{start} = \text{stop} + 1;
114 \text{stop} = \text{min}((\text{start} + \text{samples\_per\_bit} - 1),\text{num\_FFTs}\cdot\text{N});
115 \text{end};
116 \% DBPSK Modulation
117 \text{msg} = \text{round}((\text{rand}(\text{ceil}(\text{num\_FFTs}\cdot\text{N}/\text{samples\_per\_chip}),1));
120 \text{PN\_code} = \text{round}((\text{rand}(\text{ceil}(\text{num\_FFTs}\cdot\text{N}/\text{samples\_per\_chip}),1));

58
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)*mag(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)));
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;
else
    k = find(MF_mag_X == max(MF_mag_X));
end;
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

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

save dama_data est_data
end; % snr loop

% -----------------------------
% Save Data in MATLAB 4.x format
% -----------------------------

x = version;
if (x(1)==63) % 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_sss.m

001 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
002 % MF_BPSK
003 %
004 % Description: This function computes the matched filter for the PSD of the
005 % BPSK-SS frequency-domain waveform. The matched filter coefficients are
006 % stored in a file for use in the DAMA.M code. We may be able to use the
007 % theoretical PSD.
008 %
009 % Programmer: Phillip De Leon
010 %
011 % Creation Date: Sept. 25, 1998
012 % Last Revision: Oct. 24, 1998
013 %
014 % Version History:
015 % 1.0 - MF for the BPSK-SS
016 % 1.1 - New Vectorization Methods - will reduce run time.
017 % 1.2 - Added Crash Recovery - ALL data is saved to temp.mat prior to
018 % next major algorithm step.
019 %
020 % Required subroutines:
021 %
022 % Notes:
023 % 1) N must be the same as in DAMA.M
024 %
025 % Approximate Runtime:
026 % P200 win98 : ~22 minutes
027 % Mac : Not performed
028 % Pxx Linux : Not performed
029 %
030 % References:
031 %
032 % Copyright (c) 1998 by Phillip De Leon, All Rights Reserved
033 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
034 ON = 1; % Used to turn options on or off.
035 OFF = 0;
036 ERR = 0; % Set ERR to level achieved before crash - results
037 % are saved in temp.mat
038
039 if ERR == 0 disp('Operation Level One:')
040 % ----------------------
041 % Code Parameters
042 % ----------------------
043 spreading_opt = ON; % ON to spread - OFF to just BPSK modulate.
044 window = OFF; % if OFF assume rectangular window, ON Hamming Window
045
046 % ----------------------
047 % DAMA User's Parameters
048 % ----------------------
049 Fc_DAMA = 200e3; % DAMA user's carrier frequency (after shifting) in Hz
050
051 % ----------------------
052 % Communication System Parameters
053 % ----------------------
054 Fs = 800000; % output signal sampling freq. (samples/s)
055 Fc = Fc_DAMA; % assume unshifted DAMA carrier
056
057 Rb_DAMA = 1000; % data rate for DAMA (bits/s) Fs/Rb_DAMA must be integer
058 Rc_DAMA = 50000; % chip rate for DAMA (chips/s) 100K?
059 if rem(Fs,Rb_DAMA) ~= 0

61
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
Fe - max_Doppler - main_lobe_BW/2 < F_DAMA < Fe + max_Doppler + main_lobe_BW/2

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 \rightarrow 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/Fa); \% BIG vector - not for the weak!!

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,...
upamp((-1).^round(rand(cell(num_FFTs*N/samples_per_chip),1)));
end

s = msg.*PN_code;  % s is the BPSK modulated spread spectrum signal.

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
    n = [start:stop];
temp = v.*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

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

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
185  % ERR
186  % spreading_opt
187  % window
188  % Fc_DAMA
189  % Fs
190  % Fc
191  % Fs
192  % Fc
193  % Rb_DAMA
194  % Rc_DAMA
195  % samples_per_bit
196  % samples_per_chip
197  % N
198  % w
199  % num_FFTs
200  % M
201  % msg
202  % data
203  % s
204  % PN_code
205  % start
206  % stop
207  % l
208  % h
209  % num_mag_S_sqrd
210  % S
211  % mag_S_sqrd
212  % avg_mag_S_sqrd
213  % MF
214  % vtype
215  % fname
216  % mver

Program Control
Program Control
Program Control
Program Parameter
Program Parameter
Program Parameter
Program Parameter
Program Control
Program Parameter
Program Parameter
Program Parameter
Program Parameter
Program Parameter
Program Parameter
Program Control
Program Control
Program Control
Program Control
Program Parameter
Program Control
Program Control
Program Control

Used in disaster recovery
Used to turn ON/OFF spreading
Used to turn ON/OFF windowing
DAMA Carrier Frequency - for the purposes of this program it is used
to set the reference index.
Sampling Frequency
Carrier Frequency - in this program it is set to the value of Fc_DAMA.
Bit Rate
Chip Rate
Calculated Parameter Each bit is represented by # of samples
Calculated Parameter Each chip is represented by # of samples.
Length of FFT block
Calculated Parameter window coefficients
Number of Iterations
Calculated Parameter The number of bits required in the data vector.
Calculated Vector The actual sinusoidal signal - later it is BPSK modulated.
Calculated Vector Binary data stored as (-1,1)
Calculated Vector BPSK SS carrier
Calculated Vector The pseudo-random noise vector to cause spreading.
Program Parameter Used in indexing
Program Parameter Used in indexing
Used in FOR loop
Program Control Handle to figure
Holds sum of magnitude squared data from FFT.
Holds FFT data for current iteration
Holds magnitude squared data from FFT for current iteration.
Holds the average mag_S_sqrd vector.
Frequency reversed - think of it as time reversed data.
Type of window employed for inclusion into file name.
Filename
mver(1) holds matlab version.

plot_dama_data.m

001 %******************************************************************************
002 % PLOT_DAMA_DATA plot_dama_data
003 %
004 % Description: This function provides a variety of tools for DAMA data
005 % visualization using the file saved in DAMA.M
006 %
007 % Call Syntax: plot_dama_data
008 %
009 % Programmer: Phillip De Leon
010 %
011 % Creation Date: December 31, 1995
012 % Last Revision: July 1, 1997
013 %
014 % Required subroutines:
015 %
016 % Notes: Assume workspace (DAMA_DATA.MAT) has already been loaded.
017 %
018 % References:
019 %
020 % Copyright (c) 1998 by Phillip De Leon, All Rights Reserved
021 %******************************************************************************
022
023 clg;
024
025 selection = menu('Select a Plot', ...
026 'Estimation Ranges', ... % 1)
027 'Estimation Histograms', ... % 2)
028 'Estimation Accuracy', ... % 3)
029 'Exit');
030
031 % ---------------------
032 % Estimation range
033 % ---------------------
034 if (selection == 1)
035 i = find(est_data(1,:)); % find non-zero elements
036 max_neg_est_error = ((min(i) - 1) * Fs / N) - Fe; % 1 <= i <= N but need 0 <= i <= N-1
037 max_pos_est_error = ((max(i) - 1) * Fs / N) - Fe;
038 plot([snr(1) snr(1)], [max_neg_est_error max_pos_est_error]);
039 hold on
040 for m = 2:length(snr)
041 i = find(est_data(m,:)); % find non-zero elements
042 max_neg_est_error = ((min(i) - 1) * Fs / N) - Fe;
043 max_pos_est_error = ((max(i) - 1) * Fs / N) - Fe;
044 plot([snr(m) snr(m)], [max_neg_est_error max_pos_est_error]);
045 end;
046 plot([-100 100], [est_tolerance est_tolerance],'--');
047 plot([-100 100], [-est_tolerance -est_tolerance],'--');
048 hold off;
049 ylabel('Estimation error (Hz)');
050 xlabel('SNR (dB)');
051 if (window)
052 title(['Frequency Estimation Ranges (Rc=',sprintf('%.3d',Rc_DAMA), ...
053 ', N=',sprintf('%.3d', N),'.', Hamming window')]);
054 else
055 title(['Frequency Estimation Ranges (Rc=',sprintf('%.3d',Rc_DAMA), ...
056 ', N=',sprintf('%.3d', N),'.', Rectangular window')]);
057 end;
058 [i,j] = find(est_data);
059 Ymin = min(j);
060 Ymax = max(j);
axis([[snr(1)-10) (snr(length(snr)))*10] 1.1*min(Ymin,-est_tolerance) ...  
1.1*max(Ymax,est_tolerance));

% --------------------
7. Estimation histogram
% --------------------
elseif (selection = 2)
    for m = 1:length(snr)
        plot([0:N/2-1]*Fs/N,est_data(m,1:N/2));
        hold on;
        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:');
        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),', Hamming window)']);
    else
        % title(['Frequency Estimation for SNR of ',sprintf('%g', snr(m)), ...  
            % 'dB (Rc=',sprintf('%d',Rc_DAMA),', Rectangle window)']);
        title(['Frequency Estimation for SNR of ',sprintf('Xg', snr(m)), ...  
            'dB']);
    end;
    axis([0.96*(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;
% ----------------------
7. Statistical Accuracy
% ----------------------
elseif (selection == 3)
    lower_index_bound = ceil(N*(Fc-est_tolerance)/Fs)+1 % 1 <= lower_index_bound <= N
    upper_index_bound = floor(N*(Fc+est_tolerance)/Fs)+1 % 1 <= upper_index_bound <= N
    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

001 % White Sands Data Collection Code ver 4.0
002 % Started on 6/26/98
003 % Fill in comments. --> Still need to comment
004 %
005 % Revision History:
006 %
007 % Ver 1.0: Baseline, one file generated per user input
008 % Ver 2.0: Four "useful" files generated at once, no user
009 % input. Some code cleaned up (vectorized).
010 % Ver 3.0: More vectorization, utilizes MATLAB's filter
011 % function. Vectors are pre-allocated.
012 % Ver 3.01: Changed code so that # of bits may be individually
013 % selected.
014 % Ver 4.0: To allow for larger data file generation, a menu
015 % system is implemented so that only one data file
016 % is generated. This compensates for memory
017 % problems. Edit file for different spread/data
018 % rates.
019 % Ver 4.01: "2047" NASA PN sequence took to long to generate
020 % so the sequence is generated offline for faster
021 % performance. This restricts resulting binary file
022 % length to 2^20.
023 %
024 % Last Update: 7/13/98
025 % Written by: Bradley James Scaife
026 %
027 timer = cputime;
028 preamble_length = 5000;
029
030 choice = menu('Please Select File to be
031 Generated:','1k_100','1k_20','1k_10','1k_40');
032
033 if choice == 1
034 % Generate file one
035 %
036 msg_len1 = input('Enter Message Length
037 (bits): '); 
038 % Restricted value - only change if spread vector code is altered.
039 msg_len1 = 20e3; 
040
041 % HSG 1 and Spreading Vector Generation
042
043 % Message One Parameters
044 %
045 %msg_len1 = input('Enter Message Length One (bits): '); 
046
047 % Calculated Parameters: Message One
048 %
049 %msg_len1 = zeros(msg_len1,1); msg1 = 
050
051 % Preallocate MSG1 vector
052 %
053 meg_len1 = 20e3; % Restricted value - only change if spread vector code is altered.
054 meg_len1 = 10000;
055
056 msg1_init = zeros(meg_len1,1); msg1 = 
057 % Preallocate MSG1 vector
058
059 % MSG 1 and Spreading Vector Generation
060

67
061 rand('seed',0);
062
063 msg1_init = round(rand(msg_len1,1));
064
065 msg1 = upsamp(msg1_init,spread_factor1)';
066 msg1 = filter(ones(spread_factor1,1),1,msg1);
067 clear msg1_init
068
069 load svec
070
071 % Generate s1
072 %************************
073 disp('Generating s1...')
074
075 s1 = 256*xor(msg1,spreading_vector)';
076
077 % Binary File Generation
078 %************************
079 disp('Generating binary file...')
080 disp('Inserting preamble...')
081 disp('Inserting signal vector...')
082
083 data_out1 = [zeros(1,preamble_length) s1];
084 fid = fopen('1k_100.bin','wb');
085 fwrite(fid,data_out1,'int8');
086 fclose(fid);
087
088 elseif choice == 2
089 %************************
090 % Generate File 2: Per definition in header above.
091 %************************
092 %msg_len2 = input('Enter Message Length Two (bits): ');  
093 msg_len2 = 100e3; %Restricted value - only change if spread vector code is altered.
094 Rb2 = 1000; Rc2 = 20000;
095
096 % Calculated Parameters
097 %************************
098 spread_factor2 = ceil(Rc2/Rb2);
099
100 % Preallocate MSG2 vectors
101 %************************
102 msg2_init = zeros(msg_len2,1);
103 msg2 = zeros(msg_len2*spread_factor2,1);
104 spreading_vector2 = zeros(msg_len2*spread_factor2,1);
105
106 % Message and Spreading Vector Generation
107 %************************************************************************
108 msg2_init = round(rand(msg_len2,1));
109 msg2 = upsamp(msg2_init,spread_factor2)';
110 msg2 = filter(ones(spread_factor2,1),1,msg2);
111 clear msg2_init
112
113 load svec
114
115 % Generate s2
116 %************************
117 disp('Generating s2...')
118
119 s2 = 256*xor(msg2,spreading_vector)';
120
121 % Binary File Generation
122 %************************
123 disp('Generating binary file...')
124 disp('Inserting preamble...')
125 disp('Inserting signal vector...')
126
127 data_out2 = [zeros(1,preamble_length) s2];
128
129 fid = fopen('1k_20.bin','wb');
130 fwrite(fid,data_out2,'int8');
131 fclose(fid);
132
133
134 elseif choice == 3
135     X•••••••••••••••••••••••••••••••••••••••••••••••••••
136     %••••···············································
137     %msg_len3 = input('Enter Message Length Three (bits): ');
138     msg_len3 = 200e3; %Restricted value - only change if spread vector code is altered.
139     Rb3 = 1000; Rc3 = 10000;
140
141     X•••••••••••••••••••••••••••••••••••••••••••••••••••
142     % % Calculated Parameters
143     %•••••••••••••••••••••••••••••••••••••••••••••••••••
144     spread_factor3 = ceil(Rc3/Rb3);
145
146     %••••····················
147     msg3_init = zeros(msg_len3,1);
148     msg3 = zeros(msg_len3•spread_factor3,1);
149     spreading_vector3 = zeros(msg_len3•spread_factor3,1);
150
151     X•••••••••••••••••••••••••••••••••••••••••••••••••••
152     % Message and Spreading Vector Generation
153     %•••••••••••••••••••••••••••••••••••••••••••••••••••
154     msg3_init = round(rand(msg_len3,1));
155     msg3 = zeros(msg_len3•spread_factor3,1));
156     spreading_vector3 = zeros(msg_len3•spread_factor3,1);
157     msg3 = filter(ones(spread_factor3,1),1,msg3);
158     clear msg3_init
159
160     X•••••••••••••••••••••••••••••••••••••••••••••••••••
161     % Generate s3
162     X•••••••••••••••••••••••••••••••••••••••••••••••••••
163     disp('Generating s3...')
164     s3 = 255•xor(msg3,spreading_vector);  
165
166     X•••••••••••••••••••••••••••••••••••••••••••••••••••
167     % Binary File Generation
168     X•••••••••••••••••••••••••••••••••••••••••••••••••••
169     disp('Generating binary file...')
170     disp('Inserting preamble...')
171     disp('Inserting signal vector...')
172     data_out3 = [zeros(1,preamble_length) s3];
173     fid = fopen('1k_10.bin','wb');
174     fwrite(fid,data_out3,'int8');
175     fclose(fid);
176
177
178 else
179     X•••••••••••••••••••••••••••••••••••••••••••••••••••
180     %msg_len4 = input('Enter Message Length Four (bits): ');
181     msg_len4 = 50e3; %Restricted value - only change if spread vector code is altered.
182     Rb4 = 1000; Rc4 = 40000;
183
184
69
Calculated Parameters

spread_factor4 = ceil(Rc4/Rb4);

Preallocate MSG4 vectors

c4 = zeros(msg_len4,1);

c4 = zeros(msg_len4*spread_factor4,1);

c4 = zeros(msg_len4*spread_factor4,1);

% Message and Spreading Vector Generation

c4 = round(rand(msg_len4,1));

c4 = upsamp(c4_init,spread_factor4)';

c4 = filter(ones(spread_factor4,1),1,c4);

clear c4_init

load avec

generate_s4

disp('Generating s4...')

e = 255 xor(c4,spreading_vector)';

% Binary File Generation

disp('Generating binary file...')

disp('Inserting preamble...')

disp('Inserting signal vector...')

data_out4 = [zeros(1,preamble_length) s4];
id = fopen('1k_40.bin','wb');
fwrite(fid,data_out4,'int8');
close(fid);

disp('All Done :)')

cputime = cputime - timer
cap_mf.m

001 % Title:
002 % cap_mf.m
003 %
004 % Purpose:
005 % The purpose of this code is to generate estimation data similar to
006 % dama_mt with the exception that this code is tailored to use data
007 % captured during the White Sands Complex (WSC) experiment.
008 %
009 % Revision:
010 % 1.0 -- 11/20/98
011 %
012 % Revision History:
013 % none => baseline
014 %
015 % Author:
016 % Brad Scaife
017 %
018 % Notes:
019 % This code requires as input data that has been prepared by vsco_d2.m which
020 % prepares the captured data and generates a matrix of usable vectors (see
021 % vsco_d2 comments for more information). This code also relies on an index
022 % matrix that is generated by in_prp.m. In both cases these files have already been
023 % prepared and may simply be loaded following prescribed naming conventions. In
024 % the event that these data files have been lost though it was thought helpful to
025 % comment on how to regenerate them.
026 %
027 % Average runtime:
028 % 8 minutes
029 %
030 %******************************************************************************
031 clear all;
032
033 ON = 1; OFF = 0;
034
035 % Code Options
036 %:-----------------------:
037 window = OFF; % Use to toggle use of hamming window
038 matched_filter = ON; % Use to toggle use of matched filter
039 insight = OFF; % Use to gain instantaneous frequency estimation
040
041 % Code Parameters
042 %:-----------------------:
043 Fs = 100e6; % DO NOT CHANGE - this was the sample rate used.
044 snr = 1.5 % As per dama_mf definition - not vsco definition.
045 est_tolerance = 3e3; % +/- range of acceptable error in Hz.
046 N = 65536; % FFT block length.
047 number_of_estimates = 1000; % # of estimation attempts. Don't modify!!!
048 fft_avg = 8; % Number of FFT's to average over - changing this will
049 require changes to code mentioned in documentation above
050 % as well as the rerunning of this code.
051
052 alpha = 2; % Constant used to obtain consistent data points - does not
053 % add to statistical meaning nor detract from it.
054 f_lo = 32.50e6;
055 f_hi = 32.90e6;
056
057 lower_bound_index = ceil(f_lo*N/Fs)+1;
058
059 starting_bound = lower_bound_index;
060

71
upper_bound_index = floor(f_hi*N/Fs)+1;

Rb_DAMA = 1e3;

% Program Flow
X:::::::::::::::
if window
   vtype = ['h'];
else
   vtype = ['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, SNR3, F1',... % Selection 6
'SF=10, SNR3, F2',... % Selection 7
'SF=10, SNR3, F3',... % Selection 8
elseif menu_sel == 1
   data_filename = ['c:\research\code\matlab\whites-2\wscmat-1\snr1_f2' vtype];
   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);
   Xsnr = -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' vtype];
   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);
   Xsnr = -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' vtype];
   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);
   Xsnr = 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' vtype];
   eval(['load ' data_filename]);
num_col = size(x,2);
index_filename = [e:\researeb\eode\matlab\wbites\2\emb8_' 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 == 6
data_filename = [c:\researeh\eode\matlab\wbites\2\emb8_' int2str(num_col)];
eval(['load ' data_filename]);
index_filename = [e:\researeh\eode\matlab\wbites\2\emb8_' 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 == 7
data_filename = [c:\researeh\eode\matlab\wbites\2\emb8_' int2str(num_col)];
eval(['load ' data_filename]);
index_filename = [c:\researeh\eode\matlab\wbites\2\emb8_' 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:\researeh\eode\matlab\wbites\2\emb8_' int2str(num_col)];
eval(['load ' data_filename]);
index_filename = [c:\researeh\eode\matlab\wbites\2\emb8_' 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.
X = zeros(N,fft_avg);
XX = zeros(N/2,fft_avg);
X_mag_squared = zeros(length(lower_bound_index:upper_bound_index),fft_avg);
XX_mag_squared = zeros(N/2,fft_avg);
X_avg = zeros(length(lower_bound_index:upper_bound_index),fft_avg);
XX_avg = zeros(length(snr),N/2);
eat_data = zeros(length(snr),length(lower_bound_index:upper_bound_index));
eat_data = zeroa(length(anr),N/2);
for n = 1:iterations
    temp = x(:,index(n,:));
    X = fft(temp);
    I = X(length(lower_bound_index:upper_bound_index,:));
    % Select current index pattern.
    % Perform 8 FFT's
    % Restrict to search bound
    % 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
    end
    if matched_filter
        k = find(MF_mag_X(1:length(MF_mag_X)) == max(MF_mag_X(1:length(MF_mag_X))));
    end
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))));
end

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

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

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 Fe Fs N N1 est_data est_tolerance number_of_estimates ...
    starting_bound smr window vtype2
else % running on MATLAB 4.x
    save cap_dat Fe Fs N N1 est_data est_tolerance number_of_estimates ...
    starting_bound smr window vtype2
wsco.d2.m

001 % DAMA Test Signal Analysis
002 %
003 % Purpose:
004 % This code is to be used as driver code for wsco.m. This code
005 % converts captured whitesands files and converts them into
006 % usable length N blocks. Not entirely automated. Please read
007 % info in wsco.m for details on project purpose.
008 %
009 % Input:
010 % This code requires the DAMA test files named as s1,s2,...sn.
011 %
012 % Output:
013 % Undecided at this point.
014 %
015 % Revision History:
016 %
017 % ver 1.0: baseline
018 % ver 2.0: Changed scope of code. This program is now to
019 % to be used to drive other code only.
020 % ver 2.5: Restructured the format data is to be saved in.
021 %
022 %
023 % Current Version - Date:
024 % 2.5 - 8/4/98
025 %
026 % Author:
027 % Brad Scaife
028 %
029 % Date:
030 % 7/23/98
031 %
032 % Notes:
033 % Remember to change the save filename at the end of the code. Sorry
034 % got to do a little work to run this one.
035 %
036 clear all
037 ON = 1;
038 OFF = 0;
039 window = OFF;
040
041 % Change to working directory
042 cd "c:\research\data\damaus\1\lk_10\snr2\f3"
043
044 % Code Options
045 %
046 % capture_option = 0; % 0 for 50k sample else 1 (for 100k)
047 %
048 % Code Parameters
049 %
050 % num_files = 6; % 6 - for snr1_f3; 9 - for snr2_f3; 12 - for snr3_f2
051 % half = 65536;
052 % short_cap = 50002;
053 % long_cap = 100002;
054 % load s1;
055 % if length(s1) == long_cap
056 % capture_option = 1;
057 % else
058 % capture_option = 0;
clear s1;
pack

% Load raw signals
%

k = 0;

h = waitbar2(0,'Loading files...');

while 1
    k = k+1;
    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\whites2

if capture_option
    h = waitbar2(0,'Parsing 100 kSample Files...');
    index = 1;
    if window
        W = hamm(half);
    else
    end
    for k = 1:2:2*num_files
        waitbar2(index/num_files);
        varname = ['s' int2str(index)];
        temp = eval([varname]);
        x(:,k) = temp(1:half).*W;  % For others
        x(:,k+1) = temp(length(temp)-half+1:length(temp)).*W;  % For others
    end
end

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

    end

end

close(h);
waitbar2(k/num_files);

varname = ['s' int2str(k)];

temp = eval([varname]);

temp = temp.*W;

temp = [temp;zeros(half - length(temp),1)];

x(:,k) = temp;

close(h);

end

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

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

001 % Title:
002 % prepcap.m
003 %
004 % Purpose:
005 % The purpose of this code is to prepare DAMA captured signals for
006 % comparison against simulated results.
007 %
008 % Revision:
009 % 1.0 -- 7/26/98
010 % 1.5 -- 8/2/98
011 % 2.0 -- 11/2/98
012 %
013 % Revision History:
014 % none
015 %
016 % 1.5 -- Altered code to accept new signal matrix generated
017 % in wsco_d.m
018 %
019 % Author:
020 % Brad Scaife
021 %
022 % Date:
023 % 7/26/98
024 %
025 % Notes:
026 % This code requires input signals matrices that have been prepared
027 % by wsco_d.m and vector sequence matrices prepared by goob.m. Due to
028 % limited test set, the code makes some approximations that may or may
029 % not be valid. Therefore, this code should not be considered in any
030 % way a proof but rather a "ballpark" type of justification: Could it
031 % work in the real world?
032 %
033 %**********************************************************************
034 clear all
035 ON = 1;
036 OFF = 0;
037 %
038 % Code Parameters
039 %************************
040 snr = 1:7;        % SNR Ranges from S/N estimate.
041 Fs = 32.7e6;     % Measured DAMA carrier - from WSC.
042 Fs = 100e6;      % Burr-Brown Sampling Rate.
043 est_tolerance = 3e3;  % +/- acceptable error.
044 Hz = 65536;      % FFT block length.
045 number_of_estimates = 1000;  % # of estimation attempts
046 fft_avg = 8;     % Number of FFT's to perform estimation over.
047 alpha = 2;       % Correction to number of estimates.
048 insight = 0;
049 lower_bound_index = 21291;  % Index of lower search bound.
050 upper_bound_index = 21504;  % Index of upper search bound.
051 starting_bound = lower_bound_index;
052 window = OFF;
053 matched_filter = ON;
054
055 % Preallocation
056 %************************
057 X = zeros(N,fft_avg);
058
059 X_mag_squared = zeros(length(lower_bound_index:upper_bound_index),fft_avg);
060 \texttt{X}_{\text{avg}} = \texttt{zeros(length(lower\_bound\_index:upper\_bound\_index),fft\_avg)};
061 \texttt{est\_data} = \texttt{zeros(length(snr),length(lower\_bound\_index:upper\_bound\_index))};
062 if \texttt{window} \\
063 \texttt{load c:\research\code\matlab\damane-1\pg10ha} \\
064 else \\
065 \texttt{load c:\research\code\matlab\damane-1\pg10re} \\
066 \texttt{end} \\
067 \texttt{end} \\
068 \\
069 \texttt{Load SNR1_F2 Data Matrix (65536x12) and Combination Matrix (496x8)}
070 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
071 \texttt{load c:\research\code\matlab\vhites-2\snr1_f2} \\
072 \texttt{load c:\research\code\matlab\vhites-2\cmb8_12 h =} \\
073 \texttt{waitbar2(0, 'Formulating SNR1 Estimation for f2...');} \\
074 \texttt{for n = 1:495} \\
075 \texttt{index = cmbo3;} \\
076 \texttt{waitbar2(n/495);} \\
077 \texttt{temp = \texttt{x(:,index(n,:))};} \\
078 \texttt{X = \texttt{fft(temp)};} \\
079 \texttt{X = X(lower\_bound\_index:upper\_bound\_index,:);} \\
080 \texttt{X\_mag\_squared = (X.*conj(X)).'} \\
081 \texttt{X\_avg = (sum(X\_mag\_squared) ./ fft\_avg).'} \\
082 \texttt{k = find(X\_avg(1:length(X\_avg)) \& max(X\_avg(1:length(X\_avg))));} \\
083 if \texttt{insight} \\
084 \texttt{disp ('Frequency found to be:')} \\
085 \texttt{(k+20791)*100e6/64e3} \\
086 \texttt{pause(10)} \\
087 \texttt{end} \\
088 \texttt{if (length(k) > 1)} \\
089 \texttt{k = median(k);} \\
090 \texttt{disp ('oops')} \\
091 \texttt{end} \\
092 \texttt{est\_data(1,k) = est\_data(1,k) + 1;} \\
093 \texttt{end} \\
094 \texttt{end} \\
095 \texttt{est\_data(1,:) = est\_data(1,:).*alpha;} \\
096 \texttt{clear x} \\
097 \texttt{clear index;} \\
098 \texttt{close(h);} \\
099 \texttt{clear} \\
100 \texttt{Load SNR1_F3 Data Matrix (65536x12) and Combination Matrix (496x8)}
101 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
102 \texttt{load c:\research\code\matlab\vhites-2\snr1_f3} \\
103 \texttt{load c:\research\code\matlab\vhites-2\cmb8_12 h =} \\
104 \texttt{waitbar2(0, 'Formulating SNR1 Estimation for f3...');} \\
105 \texttt{for n = 1:495} \\
106 \texttt{index = cmbo3;} \\
107 \texttt{waitbar2(n/495);} \\
108 \texttt{temp = \texttt{x(:,index(n,:))};} \\
109 \texttt{X = \texttt{fft(temp)};} \\
110 \texttt{X = X(lower\_bound\_index:upper\_bound\_index,:);} \\
111 \texttt{X\_mag\_squared = (X.*conj(X)).'} \\
112 \texttt{X\_avg = (sum(X\_mag\_squared) ./ fft\_avg).'} \\
113 \texttt{k = find(X\_avg(1:length(X\_avg)) \& max(X\_avg(1:length(X\_avg))));} \\
114 if \texttt{insight} \\
115 \texttt{disp ('Frequency found to be:')} \\
116 \texttt{(k+20791)*100e6/64e3} \\
117 \texttt{pause(10)} \\
118 \texttt{end} \\
119 \texttt{if (length(k) > 1)} \\
120 \texttt{k = median(k);} \\
121 \texttt{disp ('oops')} \\
122 \texttt{end} \\
123 \texttt{est\_data(1,k) = est\_data(1,k) + 1;} \\
124 \texttt{est\_data(1,:) = est\_data(1,:).*alpha;} \\
125 \texttt{clear x} \\
126 \texttt{clear index;} \\
127 \texttt{clear} \\
128 \texttt{close(h);} \\
129 \texttt{clear} 

80
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('oops')
end

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

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

clear x

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;
    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('oops')
    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)
X = load('c:\research\code\matlab\white2\snr2_f2');
load c:\research\code\matlab\white2\cmb8_12
h = waitbar2(0,'Formulating SNR2 Estimation for f2...');

for n = 1:495
    index = cmbo3;
    temp = x(:,index(n,:));
    X = fft(temp);
    X = X(lover_bound_index:upper_bound_index,:); % Restrict to search bound
    X_mag_squared = (X.*conj(X)).';
    X_avg = (sum(X_mag_squared) ./ fft_avg).';
    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(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)
X = load('c:\research\code\matlab\white2\snr2_f3');
load c:\research\code\matlab\white2\mc cmb2
h = waitbar2(0,'Formulating SNR2 Estimation for f3...');

for n = 1:1000
    temp = x(:,index(n,:));
    X = fft(temp);
    X = X(lover_bound_index:upper_bound_index,:); % Restrict to search bound
    X_mag_squared = (X.*conj(X)).';
    X_avg = (sum(X_mag_squared) ./ fft_avg).';
    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(4,k) = est_data(4,k) + 1; % Increment bin
end

82
247 k = median(k);
248 disp('oops')
249 end
250 est_data(5,k) = est_data(5,k) + 1; % Increment bin
251 end
252
clear x;
close(h);
255
256 % Load SNR2_F5 Data Matrix (65536x12) and Combination Matrix (495x8)
257 %---------------------------------------------------------------------------------
258 load c:\research\code\matlab\vhites-2\snr2_F5
259 load c:\research\code\matlab\vhites-2\cmb8_12
260 b = waitbar2(0, 'Formulating SNR2 Estimation for f5...');
261
262 for n = 1:495
263    index = cmbo3; % Load unique index pattern matrix.
264    waitbar2(n/495);
265    temp = x(:,index(n,:)); % Select current index pattern.
266    X = fft(temp); % Perform 8 FFT's
267    X = X(1:lower_bound_index:upper_bound_index,:); % Restrict to search bound
268    X_mag_squared = (X.*conj(X)).';
269    X_avg = (sum(X_mag_squared) ./ fft_avg).'; % Sum and average.
270    k = find(X_avg(1:length(X_avg)) == max(X_avg(1:length(X_avg))));
271    if k
272        disp('Frequency found to be:')
273        (k+20791)*100e6/64e3
274        pause(10)
275    end
276    if (length(k) > 1)
277        k = median(k);
278        disp('oops')
279    end
280
281    est_data(6,k) = est_data(6,k) + 1; % Increment bin
282 end
283
est_data(6,:) = est_data(6,:) .* alpha; % Sponge data to look like a larger estimate
284
285 clear x
286 clear index;
close(h);
289
290 % Load SNR3_F2 Data Matrix (65536x12) and Combination Matrix (495x8)
291 %---------------------------------------------------------------------------------
292 load c:\research\code\matlab\vhites-2\snr3_F2
293 load c:\research\code\matlab\vhites-2\cmb8_12
294 b = waitbar2(0, 'Formulating SNR3 Estimation for f2...');
295
296 for n = 1:495
297    index = cmbo3; % Load unique index pattern matrix.
298    waitbar2(n/495);
299    temp = x(:,index(n,:)); % Select current index pattern.
300    X = fft(temp); % Perform 8 FFT's
301    % Spon
\[ X = X(\text{lower_bound_index:upper_bound_index},:) \]; % Restrict to search bound
\[ X_{\text{mag squared}} = (X.*\text{conj}(X)).'; \]
\[ X_{\text{avg}} = (\text{sum}(X_{\text{mag squared}}) ./ \text{fft} \_\text{avg}).'; \] % Sum and average.
\[ k = \text{find}(X_{\text{avg}}(1:length(X_{\text{avg}})) == \text{max}(X_{\text{avg}}(1:length(X_{\text{avg}})))); \]
\[ \text{if} \ \text{insight} \]
\[ \text{disp}'\text{Frequency found to be:}' \]
\[ (k+20791)*100e6/64e3 \]
\[ \text{pause}(10) \]
\[ \text{end} \]
\[ \text{if} \ \text{length} (k) > 1 \]
\[ k = \text{median}(k); \]
\[ \text{disp}'\text{ooops}' \]
\[ \text{end} \]
\[ \text{est} \_\text{data}(7,k) = \text{est} \_\text{data}(7,k) + 1; \] % Increment bin
\[ \text{end} \]
\[ \text{est} \_\text{data}(7,:) = \text{est} \_\text{data}(7,:) .* \alpha; \] % Sponge data to look like a larger estimate
\[ \text{clear} \ x \]
\[ \text{clear} \ \text{index}; \]
\[ \text{close}(b); \]
\[ \text{mlver} = \text{version}; \]
\[ \text{if} \ \text{mlver}(1) == 5 \] % Running on MATLAB 5.x
\[ \text{save} \ -v4 \ \text{dama} \_\text{cap}, \text{Fs} \ F_c \ \text{snr} \ \text{est} \_\text{tolerance} \ \text{N} \ \text{number} \_\text{of} \_\text{estimates} \ldots \]
\[ \text{est} \_\text{data} \ \text{lower} \_\text{bound} \_\text{index} \ \text{upper} \_\text{bound} \_\text{index} \]
\[ \text{else} \] % Running on MATLAB 4.x
\[ \text{save} \ \text{dama} \_\text{cap}, \text{Fs} \ F_c \ \text{snr} \ \text{est} \_\text{tolerance} \ \text{N} \ \text{number} \_\text{of} \_\text{estimates} \ldots \]
\[ \text{est} \_\text{data} \ \text{lower} \_\text{bound} \_\text{index} \ \text{upper} \_\text{bound} \_\text{index} \]
\[ \text{end} \]
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

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; stime = (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.
Klow = ceil(f_lo*N/fs); % Lower index bound.
k_hi = floor(f_hi*N/fs); % Upper index bound.
```matlab
if N == Nbase
    k_hi = k_lo;
end
res_factor = N/L;  % - Resolution Factor

success = zeros(1,length(snr));
mse_calc = zeros(1,length(snr));
rmse = zeros(1,length(snr));
rmse_calc = zeros(1,length(snr));
mae = zeros(1,length(snr));
rmse_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_freq = find(S_mag == max(S_mag))-1
f_true = true_freq/N*fs
true_ind = floor(f_true*N/fs)

disp(sprintf('True Frequency: %10.6f 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))
pause('Press a key to continue ... ') pause

for n = 1:length(snr)
    snr_mod = sqrt(cov(s)/(10-(snr(n)/10)));
    No = Ts*snr_mod-2;
    mse_calc(n) = (6*No*Jw)/(2*pi*(2 + snr_mod)^2);
    rmse_calc(n) = sqrt(mse_calc(n));
    randn('seed',0);
end
```
Y = fft(y,L);
Y_mag = abs(Y);
YMag = 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
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);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
002 %
003 % Experiment Two: Finding Probabilities and MSE estimates of estimating
004 % a real sinusoid in real noise.
005 %
006 % Purpose: The purpose of this code is to determine and prove the
007 % relationship between the DAMA curves and actual MSE of
008 % estimation. This baseline test will provide insight into the
009 % simpler case of a complex sinusoid in noise which will then
010 % be extended to the more complex DAMA carrier case.
011 %
012 % Programmer: Brad Scaife
013 % Date: 3/18/99
014 % Revision Date: 3/18/99
015 % Current Revision: 1.0
016 % Revision History: 1.0 - Baseline
017 %
018 % Notes: See Porat. As per indicated in Porat, the results of
019 % estimation are valid only when the "rule of thumb" are
020 % satisfied. Thus any processing of snr's below the RUT are
021 % not valid with the theoretical curve.
022 %
023 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
024 clear all clc
025
026 % Program Parameters:
027 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
028 N = 65536; % Normally set to 512.
029 Nbase = 65536; % Basic DFT resolution.
030 fs = 800e3; % Sampling frequency.
031 Ts = 1/fs; % Sampling interval.
032 f = 200e3 + fs/(4*N); % Sinusoid frequency.
033 stime = 0; etime = (N-1)/fs; %
034 phi = 0; % Phase offset.
035 A = 1; % Amplitude of sinusoid.
036 Ju = 1; % Set as window function - see Porat.
037 D = Nbase*Ts; % As per Porat.
038
039 snr = [-20:.2:10]';
040 number_estimates = 100;
041
042 % Calculated Program Parameters
043 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
044 est_range = floor(fs/Nbase)/2; % Frequency estimation range as a function of
045 % basic DFT resolution.
046 f_lo = f - est_range; % Lower "accurate" estimation bound.
047 f_hi = f + est_range; % Upper "accurate" estimation bound.
048 k_lo = ceil(f_lo*N/fs); % Lower index bound.
049 k_hi = floor(f_hi*N/fs); % Upper index bound.
050
051 if N == Nbase
052 k_hi = k_lo;
053 end
054
055 res_factor = N/Nbase; % Resolution Factor
056
057 % Preallocate
058 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
059 success = zeros(1,length(snr));
% 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: %10.6f Hz',f_true))
% disp(sprintf('Base Points: %d',Nbase))
% disp(sprintf('DFT Points: %d',N))
% disp(sprintf('Base DFT Resolution: %7.10.5f Hz', fs/Nbase))
% disp(sprintf('Calculation DFT Resolution: %7.10.5f Hz', fs/N))
% disp(sprintf('Resolution Factor: %7.d',res_factor))
% disp(sprintf('Estimation Range: %7.10.5f Hz', est_range))
% disp(sprintf('Lower Index Bound: %d',k_lo))
% disp(sprintf('Upper Index Bound: %d',k_hi))
% pause
% for n = 1:length(snr)
% snr_mod = sqrt(cov(s)/(10-(snr(n)/10))); % Standard Deviation of noise
% Xe = Ts * (2 + snr_mod) * 2;
% No = Ts * snr_mod * 2;
% mse_calc(n) = (24*No*Jw)/((2*pi)-2*A-3); % Current SNR to be tested.
% % Remove me
% % Remove me
% % Remove me
% % Remove me
% % Remove me
% % Remove me
% 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

eend

% mse(n) = (err_sum / number_estimates)^2;
mse(n) = mse_sum / number_estimates;
rmse(n) = sqrt(mse(n));
err_sum = 0;
rmse_calc = sqrt(mse_calc);
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:
- 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.

%----------------------
% Program Parameters
%----------------------
N = 512;
L = 2*0*N;
A = 1;
Jw = 1;
fft_avg = 26;
%----------------------
% Communication System Parameters
%----------------------
Fs = 800e3; % output signal sampling freq. (samples/s)
Fc = 178e3; % BPSK carrier frequency in Hz. (cycles/sec)
k = 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);
zero = zeros(N,1);
x = zeros(N,1);

%----------------------
% Display Info
%----------------------
disp(sprintf('Sampling Frequency Fs: %6.16f',Fs));
disp(sprintf('Carrier Frequency Fc: %5.15f',Fc));
disp(sprintf('FFT Resolution (Data Supported): %5.16f',Fs/N));
disp(sprintf('FFT Computational Resolution: %5.15f',Fs/L));
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);
        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;
    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

001 ; REV 3.0
002
003 ; Just for convenience - delete later!!! Turns on/off D/A codec
004 ; bclr $19,x:M_CRBO ;Disable Rx on A-codec
005 ; bclr $18,x:M_CRBO ;Used to disable Tx interrupt
006
007
008         opt now
009
010 ; nlist
011 include 'ioequ.asm'
012 include 'intequ.asm'
013 include 'ada_equ.asm'
014 include 'vectors.asm'
015 include '7819equ.asm'
016 ;list
017
018 ;**********************************************************************************************************
019 ; Initial Layouts: This section of code sets up the D/A memory resources,
020 ; the program memory resources and defines the FFT macro.
021 ;**********************************************************************************************************
022 include 'CS4216.asm' ;D/A Memory Resources
023 include 'fftr2cn.asm' ;FFT Macro
024 include 'convm.asm' ;Convolution Macro
025 include 'mlayout.dat' ;Memory Layout
026
027 ;**********************************************************************************************************
028 ; Fast Interrupt - IRQB
029 ;**********************************************************************************************************
030         org pli:I_IRQB
031         movep y:BB7819_DR,x:(r0)+
032         org pli:I_IRQB=1
033         bset $0,x:FLAGS
034
035         org p:$100
036         START
037         main
038
039 ;**********************************************************************************************************
040 ; Set Operating Frequency
041 ;**********************************************************************************************************
042         movep #CLK_RATE,x:M_PCTL ;Set PLL and Chip Operating Frequency
043
044 ;**********************************************************************************************************
045 ; Set Operating Parameters of DSP66303
046 ;**********************************************************************************************************
047         move #OP_MODE,omr ;Set Operating Mode of 303
048
049 ;**********************************************************************************************************
050 ; Setup Stack
051 ;**********************************************************************************************************
052         movec #0,sp ;clear hardware stack pointer
053         move #STACK,r6 ;initialize stack pointer
054         move #-1,m6 ;linear addressing
055
056
057
058 ;**********************************************************************************************************

94
Set AAR Wait States for External Memory (32k) and A/D Codec

include 'vs_set.asm' ; Set Wait States

Set Up IRQB Interrupt Parameters. IRQB is the interrupt designated to the A/D Codec.

include 'core_ipl.asm' ; Set IRQB Interrupt Parameters

ori #003, 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 'vsc.asm'
include 'avg_fft.asm'
include 'init.asm'
include 'get_bin.asm'
include 'sinwgid.asm'
include 'ada_init.asm'

echo end
mlayout.dat

001 ;***************************************************************************
002 ;mlayout.DAT: This data file is used with rev1.ASM to lay things out in memory
003 ;
004 ; Notes: For use with rev 2.1 code.
005 ;***************************************************************************
006 ; Equates:
007 ;***************************************************************************
008 016 CLK_RATE equ $040004 ;Chip Operating Clock - See Section 9.3 300FM
009 017 OP_MODE equ $389 ;Chip Operating Mode
010 018 ;Please use either 389 or 3C9 for proper operation. Please see DAMA Programming Notes
011 019 ;and 303UM:3-13 for details.
012 020 FS equ 32000 ;Please set the same as D/A sample rate.
013 021 ;***************************************************************************
014 ; DAMA Project Memory Settings. PLEASE DO NOT CHANGE!! CODE WILL LIKELY NOT
015 ; FUNCTION. THE MEMORY HAS BEEN SPECIFICALLY SETUP UTILIZING ALL ON-CHIP
016 ; MEMORY.
017 ;***************************************************************************
018 026 POINTS equ 512 ;Number of Points (samples)
019 027 TABLE_SIZE equ 512 ;Sine Wave Lookup Table Size (Will adjust output)
020 028 ITERS equ 8 ;FFT Iterations
021 029 OFFSET equ 128 ;Correction from Spectral Smearing due to Convolution.
022 030 ;OUTPUT_SEC equ 2 ;Please enter duration of output in seconds.
023 031 ;OUT_LENGTH equ 64 ;Please DO NOT CHANGE!!! CODE WILL LIKELY NOT
024 ; FUNCTION. THE MEMORY HAS BEEN SPECIFICALLY SETUP UTILIZING ALL ON-CHIP
025 ; MEMORY.
026 ;***************************************************************************
027 org l:$000a
028 "SAHPLE_DATA dam
029 "FFT_DATA dsm
030 "FFT_RESULT dsm
031 "COEFF dsm
032 "SAMPLE_DATA ds 1 ;SAMPLE_DATA Pointer Storage
033 "FT_DATA_PTR ds 1 ;FT_DATA Pointer Storage
034 "IFFT_PTR ds 1 ;Imaginary FFT Data Pointer Storage
035 "IFFT_MOD ds 1 ;Imaginary FFT Data mod Storage
036 "FFT_COUNTER ds 1 ;Sample Counter
037 ;***************************************************************************
038 org x:$000a
039 ;***************************************************************************
040 org x:$000a ;see ADA_INIT.ASM for why we start at x:$000a
041 ;***************************************************************************
042 ;***************************************************************************
043 ;***************************************************************************
044 ;***************************************************************************
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 ; Delta for Carrier Reconstruction.
RO_STORE ds 0
R1_STORE ds 0
R2_STORE ds 0
R3_STORE ds 0
R4_STORE ds 0
R5_STORE ds 0
R7_STORE ds 0
M0_STORE ds 0
M1_STORE ds 0
M2_STORE ds 0
M3_STORE ds 0
M4_STORE ds 0
M5_STORE ds 0
M6_STORE ds 0
M7_STORE ds 0
MO_STORE ds 0
M1_STORE ds 0
M2_STORE ds 0
M3_STORE ds 0
M4_STORE ds 0
M5_STORE ds 0
M6_STORE ds 0
M7_STORE ds 0
FLAGS ds 0 ; User Defined Flag Register
OUT_COUNTER ds 0 ; Output Sample Counter
CNV_MEM equ * ; Beginning of Stack
STACK ds 0
org x:$800

MAG_SQ_DATA dam POINTS/2
org x:$A00

TAB dam TABLE_SIZE
include 'sintab.asm'
sintab TABLE_SIZE,TAB

Y Memory:
org y:$0

HAMM dam POINTS ; Hamming Window table.
include 'hamming.asm'
hamming POINTS,HAMM

HANN dam POINTS ; Hamming Window table.
include 'sintab.asm'

SMF dam POINTS/2
include 'smf20.dat'

org y:SMF
include 'smf20.dat'
123
124 ; Convolution Output
125 ;*************************
126 CNV_OUT dsm 2*POINTS-1
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 with Window, scale, and iteration adjustments

Alters: a, b, r0, r3, r4, x0, 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/1 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
01 ;*****************************************************************
02 ; COMPUTE FFT Subroutine
03 ;
04 ; Purpose: The purpose of this subroutine is to compute the FFT of the input
05 ; signal and store it in memory.
06 ;
07 ; In: r0, r1, r2, r3, r4, r5, m0, m1, m2, m3, m4, m5
08 ; Out: x(r1), r0, r1, r2, r3, r4, r5, m0, m1, m2, m3, m4, m5
09 ; Alters: Everything
10 ;*****************************************************************
11 COMPUTE_FFT
12 move r0, x:R0_STORE
13 move r1, x:R1_STORE
14 move r2, x:R2_STORE
15 move r3, x:R3_STORE
16 move r4, x:R4_STORE
17 move r5, x:R5_STORE
18 move r7, x:R7_STORE
19 move m0, x:MO_STORE
20 move m1, x:M1_STORE
21 move m2, x:M2_STORE
22 move m3, x:M3_STORE
23 move m4, x:M4_STORE
24 move m5, x:M5_STORE
25 move m7, x:M7_STORE
26 move n5, x:N5_STORE
27 move n7, x:N7_STORE
28 fftr2cn POINTS, SAMPLE_DATA, FFT_DATA, COEFF
29 move x:N7_STORE, n7
30 move x:N6_STORE, n6
31 move x:M7_STORE, m7
32 move x:M5_STORE, m5
33 move x:M7_STORE, m7
34 move x:M5_STORE, m5
35 move x:M4_STORE, m4
36 move x:M3_STORE, m3
37 move x:M2_STORE, m2
38 move x:M1_STORE, m1
39 move x:M0_STORE, m0
40 move x:R7_STORE, r7
41 move x:R5_STORE, r5
42 move x:R4_STORE, r4
43 move x:R3_STORE, r3
44 move x:R2_STORE, r2
45 move x:R1_STORE, r1
46 move x:R0_STORE, r0
47 rts
avg_fft.asm

01 ; for use with rev 2.1 code only AVG_FFT
02
03 move x:(r1)+,x0 y:(r7)+,y0
04 do #POINTS/2,END_TLOOP
05 mpy x0,x0,a x:(r2),y1
06 macr y0,y0,a
07 add y1,a
08 move x:(r1)+,x0 y:(r7)+,y0
09 move a,x:(r2)+
10
11 END_TLOOP
12 clr b
13 move x:FFT_COUNTER,b0
14 dec b
15 move b0,x:FFT_COUNTER
16 tst b
17 js eq GET_MAX_BIN
18
19 ;************************* Prepare to perform next FFT iteration
20 ;*************************
21
22 move #POINTS,xi ;Reload sample counter for next sample
23 move xi,x:SAMPLE_COUNTER ;buffering.
24
25 move #FFT_RESULT,r2 ;re-Setup FFT Result ptr
26 move #MAG_SQ_DATA,r2 ;re-Setup Mag Squared data ptr
27 move #FFT_DATA,r1 ;Setup FFT Data ptr
28 move r1,r7 ;Imag. Pointer to FFT Buffer
29
30 move #0,x0
31 do #POINTS,CLR_DAT ;Clear FFT Data buffer
32  move x0,x:(r1) ;Real
33  move x0,y:(r1)+ ;Imaginary
34
35 do #POINTS,CLR_SMP
36  move x0,x:(r0)
37  move x0,y:(r0)+
38
39 CLR_SMP
40 rts
get_bin.asm

01 ;*********************************************************************************************
02 GET_MAX_BIN Subroutine
03 ;
04 ; Purpose: The purpose of this subroutine is to determine the frequency bin
05 ; that has the largest component and then to determine the delta for the
06 ; sine wave generation routine.
07 ;
08 ; In:     x:(r2)
09 ; Out:    b
10 ; Alters: b,x1,y1,r2
11 ;
12 ; Notes: For use with rev 2.1 code only!!
13 ;*********************************************************************************************
14 GET_MAX_BIN
15 ;*********************************************************************************************
16 ; Clean Up From GET_BIN Subroutine
17 ;*********************************************************************************************
18 move #POINTS,x1 ;Reload sample counter for next sample
19 move x1,x:Sample_counter ;buffering.
20 move #FFT_DATA,r1 ;re-Setup FFT Data ptr
21 move r1,r7 ;Imag. Pointer to FFT Buffer
22 move #FFT_RESULT,r2 ;re-Setup FFT Result ptr
23 move #MAG_SQ_DATA,r2 ;re-Setup Mag Sq Data ptr
24 move POINTS/2-l,MAG_SQ_DATA,SMF,CNV_OUT,CNV_MEM
25 ;*********************************************************************************************
26 ; Perform SMF Convolution
27 ;*********************************************************************************************
28 move r0,x:RO_STORE
29 move r4,x:R4_STORE
30 move r1,x:R1_STORE
31 move m0,x:MO_STORE
32 move m4,x:M4_STORE
33 move m0,x:M1_STORE
34 convm POINTS/2-1,MAG_SQ_DATA,SMF,CNV_OUT,CNV_MEM
35 move x:RO_STORE,r0
36 move x:R4_STORE,r4
37 move x:R1_STORE,r1
38 move x:M0_STORE,m0
39 move x:M4_STORE,m4
40 move x:M1_STORE,m1
41 move x:M4_STORE,m4
42 move #CNV_OUT,r7
43 move #POINTS/2-2,m7
44 clr b
do #POINTS-1,ND_MAX
45 move x:(r7),x1 ;Bin Comparison
46 cmp x1,b ;b-x1
47 jlt NEW_MAX ;b will always hold max
48 jmp DUMMY
49
50
51
52
53 NEW_MAX
54 move x1,b ;Store New Max Location
55 move r7,x:MAX_LOCATION
56 DUMMY
57 move (r7)+
58 nop
59 ND_MAX
60 move x:MAX_LOCATION,b ;Subtract max location from base

102
123 ;
124 NOP  ;- Reserved
125 NOP
126 ;
127 jmp * ;- SCI Receive Data
128 NOP
129 ;
130 jmp * ;- SCI Receive Data w/ Exception Status
131 NOP
132 ;
133 jmp * ;- SCI Transmit Data
134 NOP
135 jmp * ;- SCI Idle Line
136 NOP
137 jmp * ;- SCI Timer
138 NOP
139 jmp *
140 NOP  ;- Reserved
141 ;
142 NOP
143 NOP
144 ;
145 NOP  ;- Reserved
146 NOP
147 ;
148 NOP  ;- Reserved
149 NOP
150 ;
151 ;
152 jmp * ; Host receive data full
153 NOP
154 ;
155 ;
156 jmp * ;- Host transmit data empty
157 NOP
158 ;
159 jmp * ; Available for Host Command
160 NOP
161 jmp * ; Available for Host Command
162 NOP
163 jmp * ; Available for Host Command
164 NOP
165 jmp * ; Available for Host Command
166 NOP
167 jmp * ; Available for Host Command
168 NOP
169 jmp *
170 NOP  ; Available for Host Command
171 jmp *
172 NOP  ; Available for Host Command
173 jmp *
174 NOP  ; Available for Host Command
175 jmp *
176 NOP  ; Available for Host Command
177 jmp *
178 NOP  ; Available for Host Command
179 jmp *
180 NOP  ; Available for Host Command
181 jmp *
182 NOP  ; Available for Host Command
183 jmp *
184 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 ;
jar SINWGID ; and program
move #>ITERS,x1 ; Init FFT Counter
move x1,x:FFT_COUNTER ; Reset FFT counter for next iteration.
RTS
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.

INITIALIZATION: D/A Codec and Setup Control Words. Only initialize the first time.

; Initialization: D/A Codec and Setup Control Words. Only initialize the first time.

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

; Repeat control words

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

; Repeat control words

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

; Repeat control words

send_loop
jeq tl,x:FLAGS,send_loop ;Skip after initial pass
jset tl,x:FLAGS,send_loop ;Skip after initial pass
jclr tl,x:FLAGS,send_loop ;Skip after initial pass

; Test for end of duration of samples out phase.

move x:OUT_COUNTER,a0 ;samples out phase.
dec a
move a0,x:OUT_COUNTER
move a0,x:OUT_COUNTER
move a0,x:OUT_COUNTER
ejq 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

01 ;*****************************************************************************
02 ; Wait State Parameter Settings Routine
03 ;
04 ; Purpose: The purpose of this routine is to set the Bus Control Register (BCR)
05 ; to the proper number of wait states required by each AAR device. In the
06 ; DAMA Project, the AAR devices are:
07 ;
08 ; 32k SRAM
09 ; Burr-Brown Codec (Operating @ 800kHz for DAMA Project)
10 ;
11 ;*****************************************************************************
12 ;Wait State Settings:
13 ;*****************************************************************************
14 DEFAULT_WS equ $0f ;default are wait states
15 SRAM_WS equ $0f ;32kW SRAM
16 FLASH_WS equ $00 ;FLASH
17 PERIPH_WS equ $0f ;A/D Peripheral board
18
19 AREA0 equ SRAM_WS
20 AREA1 equ FLASH_WS
21 AREA2 equ PERIPH_WS
22 AREA3 equ SRAM_WS
23
24 BBS equ $0 ;Bus State
25 BLH equ $0 ;Bus Lock Hold
26 BRH equ $0 ;Bus Request Hold
27
28 BCR equ (BBS<<21)+(BLH<<22)+(BRH<<23)+(DEFAULT_WS<<16+M_BDFW)\n29 + (AREA3<<13&M_BAW)+(AREA2<<10&M_BA2W)+(AREA1<<6&M_BA1W)+(AREA&M_BAGW)
30
31 movep #BCR,:M_BCR ;Initialize Bus Control Register
convm.asm

convm macro length,xcoefs,hcoefs,result,conv_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 DSP66xxx 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,m0,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

106
mac x0,y1,a x:(r0)-,x0 y:(r4)+,y1

END_F

inc b
move acoefs,r4
macr x0,y1,a
move b0,n0
move a,x:(r1)+
move acoefs,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 #acoefs+1,r4
move x1,n4
move b0,x:(r6)+
move x1,b0
inc b
macr x0,y1,a
move b0,x1
move #acoefs+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

01 ;---Buffer for talking to the CS4215
02
03   org x:0
04 RX_BUFF_BASE equ *
05 RX_data_1_2 ds 1 ;data time slot 1/2 for RX ISR
06 RX_data_3_4 ds 1 ;data time slot 3/4 for RX ISR
07 RX_data_5_6 ds 1 ;data time slot 5/6 for RX ISR
08 RX_data_7_8 ds 1 ;data time slot 7/8 for RX ISR
09
10 TX_BUFF_BASE equ *
11 TX_data_1_2 ds 1 ;data time slot 1/2 for TX ISR
12 TX_data_3_4 ds 1 ;data time slot 3/4 for TX ISR
13 TX_data_5_6 ds 1 ;data time slot 5/6 for TX ISR
14 TX_data_7_8 ds 1 ;data time slot 7/8 for TX ISR
15
16 RX_PTR ds 1 ; Pointer for rx buffer
17 TX_PTR ds 1 ; Pointer for tx buffer
18
19 TONE_OUTPUT EQU HEADPHONE_EN+LINEOUT_EN+(4*LEFT_ATTN)+(4*RIGHT_ATTN)
20 TONE_INPUT EQU MIC_IN_SELECT+(15*MONITOR_ATTN)
21 CTRL_WD_12 equ NO_PREAMP+HI_PASS_FILT+SAMP_RATE_32+STEREO+DATA_16 ;CLB=0
22 CTRL_WD_34 equ IMMED_STATE+XTAL1_SELECT+BITS_64+CODEC_MASTER
23 CTRL_WD_56 equ $000000
24 CTRL_WD_78 equ $000000

108
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.

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
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
move #FFT_RESULT,r2 ;Setup FFT Result ptr
move #MAG_SQ_DATA,r2 ;
do #POINTS/2-1,m2 ;Setup FFT Result mod
move x0,x:(r2) ;Real
move x0,y:(r2)+ ;Imaginary
move r7,x:IFFT_PTR ;Store IFFT for r7 reuse.
move m7,x:IFFT_MOD ;
do #2+POINTS-2,m7 ;Setup Conv. Output mod.
do #2+POINTS-1,CLEAR_CNV ;Clear Conv. Output
move x0,y:(r7)+
do #HAMI,r4 ;Setup Hamming ptr
move #POINTS-1,m4 ;Setup Hamming mod
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
61  move  x0,x:FLAGS ;Clear User Defined Flag Register
63
64 ;  move  #OUT_LENGTH,x1 ;Set output duration counter.
65 ;  move  x1,x:OUT_COUNTER ;
67  rts
hamming.asm

01 hamming macro points,hamm_loc
02 hamming ident 1,2
03
04 py equ 3.141592654
05 FREQ_INC equ 2.0*py/Gcvf(points-1) ;frequency increment
06 SCALE_SHIFT equ GCVI(Glog(Gcvf(points))/Glog(2.0)) ;Shifts to Produce 1/POINTS
07 ;ITERATIONS equ 2 ;Number of FFT Iterations to perform. There
08 ;exists a limit before overflow.
09 ;SCALE_FAC equ Gcvf(points)*Gcvf(ITERATIONS) ;Scale Factor
10
11 org y:hamm_loc
12 N set 0
13 dup points
14 dc (0.54-0.46*cos(FREQ_INC*Gcvf(N)))/Gcvf(points/2)
15 N set N+1
16 endm
17 endm ;end of hamming macro
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 6 to 3 are relevant. The following table suggests the proper settings:

<table>
<thead>
<tr>
<th>Enabled</th>
<th>No</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

IBL2: 0 for level triggering, 1 for edge triggering (DAMA uses edge)

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.

---

```assembly
; 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

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

01 sintab macro tasiz,tab_loc
02 sintab ident 1,2
03
04 pie equ 3.141592654
05 TAB_INC equ 2.0*pi/Gcvf(tasiz)
06
07 org x:tab_loc
08 N set 0
09 dup tasiz
10 dc @sin(TAB_INC*Gcvf(N))/2.0
11 N set N+1
12 endm
13 endm
14 ;end sine table generation macro
7819equ.asm

1 BB_ADR    equ  $0 ;DIP Switch Address SW1 ($0 - $3f)
2 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 DISCLAIMER 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

061: Do first and second Radix 2 FFT passes, combined as 4-point butterflies
062;
063 move x: (r0)+n0, x0
064 tfr x0, a x: (r0)+n0, y1
065 do n0, _twopass
066 tfr y1, b x: (r0)+n0, y0
067 add y0, a x: (r0), x1 ; ar+cr
068 add x1, b r0, r4 ; br+dr
069 add a, b (r0)+n0 ; ar'= (ar+cr)+(br+dr)
070 sub b, a b, x: (r0)+n0 ; br'= (ar+cr)-(br+dr)
071 tfr x0, a a, x0 y: (r0), b
072 sub y0, a y: (r4)+n4, y0 ; ar-cr
073 sub y0, b x0, x: (r0) ; bi-dj
074 add a, b y: (r0)+n0, x0 ; cr'= (ar-cr)+(bi-dj)
075 sub b, a b, x: (r0) ; dr'= (ar-cr)-(bi-dj)
076 tfr x0, a a, x0 y: (r0), b
077 add y0, a y: (r0)+n0, y0 ; bi+di
078 sub b, a b, (r0) ; bi-di
079 add a, b y: (r0)+n4 ; ai+ci
080 sub b, a b, x: (r0) ; di-br
081 add a, b x: (r0) ; ar+cr
082 tfr x0, a a, x0 y: (r4)+n4
083 sub y0, a y: (r4)+n4, y0 ; ai-ci
084 sub y0, b x: (r0)+n0, x0
085 add a, b b, y: (r4)+n4 ; bi+di
086 sub b, a b, x: (r4)+n4 ; ai-ci
087 tfr x0, a a, y: (r4)+
088 _twopass
089;
090 ; Perform all next FFT passes except last pass with triple nested DO loop
091;
092 move #points/8, n1 ; initialize butterflies per group
093 move #4, n2 ; initialize groups per pass
094 move #1, m2 ; linear addressing for r2
095 move #0, m7 ; initialize C address modifier for
096 ; reverse carry (bit-reversed) addressing
097 do #cvi(#log(points)/#log(2)-2.5), _end_pass : -1 ??? example: 7 passes for 1024 pt. FFT
098 move #data, r0 ; initialize A input pointer
099 move r0, r1
100 move n1, r2
101 move r0, r4 ; initialize A output pointer
102 move r1, r5
103 move (r1)+n1 ; initialize B input pointer
104 move r1, r6 ; initialize B output pointer
105 move #coefs, r7 ; initialize C input pointer
106 lua (r2)+, n0 ; initialize pointer offsets
107 move n0, n4
108 move n0, n5
109 move (r2)- ; butterfly loop count
110 move x: (r1), x1 y: (r7), y0 ; xl=br,y0=vi,lookup -sine and -cosine values
111 move x: (r7)+n7, x0 y: (r0), b ; b=ai, x0=w, update C pointer, preload data
112 mac x1, y0, b y: (r1)+, y1 ; y1=bi,b=ai+br+wi,
113 macr -x0, y1, b y: (r0), a ; a=ai, b=ai+br+wi=bi+wr=bi'
114 do n2, _end_grp
115 do r2, _end_bfy ; loop BF-1 times and start Radix 2 DIT BF kernel
116 subl b, a x: (r0)+n0, y0 ; br=ar, ai=br+wi+br=ai', PUT bi'
117 mac -x1, x0, b x: (r0)+a y: (r5) ; a=br, b=ar+br+wi, PUT ai'
118 macr -y1, y0, b x: (r1), x1 ; b=ar-(br+wr+wi-ri)+wri, x1=next br
119 subb b, a b, x: (r0)+y ; y: (r0), b ; b=ai, a=2ar+ar+br+wr+br+wi+ar', PUT br'
120 mac x1, y0, b y: (r1)+, y1 ; y1=bi, b=ai+br+wi
121 macr -x0, y1, b b, x: (r5)+ y: (r0), a ; a=ai, b=ai+br+wi-ri=br+wi=bi' PUT ar'
123  _end_bfy
124  move (rl)+n1 ;points to first B in next group
125  subl b,a x:(r0),b b,y:(r4) ;PUT last bi' in a group
126  mac -x1,x0,b x:(r0)+n0,a a,y:(r5) ;PUT last ai' in a group and update A pointer
127  macr -y1,y0,b x:(r1),x1 y:(r7),y0
128  subl b,a b,x:(r4)+n4 y:(r0),b ;PUT last br' in a group and update A' pointer
129  mac x1,y0,b x:(r7)+n7,x0 y:(r1)+,y1 ;update W pointer
130  macr -x0,y1,b a,x:(r3)+n5 y:(r0),a ;PUT last ar' in a group and update B' pointer
131  _end_grp
132  move m1,b1
133  lsr b n2,a1 ;divide butterflies per group by two
134  lsl a b1,n1 ;multiply groups per pass by two
135  move a1,n2
136  _end_pass
137
138  ; Do last FFT pass
139
140  move #2,n0 ;initialize pointer offsets
141  move n0,n1
142  move #points/4,n4 ;output pointer A offset
143  move n4,n5 ;output pointer B offset
144  move #data,r0 ;initialize A input pointer
145  move #data,r4 ;initialize A output pointer
146  move r4,r2 ;save A output pointer
147  lua (r0)+,r1 ;initialize B input pointer
148  lua (r2)+n2,r5 ;initialize B output pointer
149  move #0,n4 ;bit-reversed addressing for output ptr. A
150  move n4,n5 ;bit-reversed addressing for output ptr. B
151  move #coef,r7 ;initialize C input pointer
152  move (r5)+n5 ;predecrement output pointer
153  move x:(r1),x1 y:(r7),y0 ;x1=br, y0=wi
154  move x:(r5),a y:(r0),b ;a=? b=ai
155
156  do n2,_lastpass ;Radix 2 DIT butterfly kernel with one butterfly per group
157  mac x1,y0,b x:(r7)+n7,x0 y:(r1)+n1,y1 ;b=aci+br=wi, x0=wr, y1=bi
158  subl b,a a,x:(r6)+n5 y:(r0),a ;b=ai+br=wi+wr=bi',a=ai, PUT previous ar'
159  mac -x1,x0,b x:(r0)+n0,a a,y:(r5) ;b=ar-br=wr,a=ar, PUT ai'
160  mac -y1,y0,b x:(r1),x1 y:(r7),y0 ;b=br',x1=br,y0=wi
161  subl b,a b,x:(r4)+n4 y:(r0),b ;a=ar',b=ai, PUT br'
162  _lastpass
163  move a,x:(r5)+n5 ; PUT ar'
164  endm
sincos.asm

01 ;
02 ; This program originally available on the Motorola DSP bulletin board.
03 ; It is provided under a DISCLAIMER OF WARRANTY available from
04 ; Motorola DSP Operation, 8501 Wm. Cannon Drive W., Austin, Tx., 78735.
06 ;
06 ; Sine-Cosine Table Generator for FFTs.
07 ;
08 ; Last Update 25 Nov 86  Version 1.2
09 ;
10 sincos macro points,coef
11 sincos ident 1,2
12 ;
13 ; sincos - macro to generate sine and cosine coefficient
14 ; lookup tables for Decimation in Time FFT
15 ;
16 ;
17 ; points - number of points (2 - 32768, power of 2)
18 ;
19 ;
20 ; negative cosine value in X memory
21 ;
22 ; Latest revision - 25-Nov-86
23 ;
24
25 pi equ 3.141592654
26 freq equ 2.0*pi/cvf(points)
27
28 org x:coef
29 count set 0
30 dup points/2
31 dc -@cos(cvf(count)*freq)
32 count set count+1
33 endm
34
35 org y:coef
36 count set 0
37 dup points/2
38 dc -@sin(cvf(count)*freq)
39 count set count+1
40 endm
41
42 endm ;end of sincos macro
ada_equ.asm

001 page 132,60
002
003 ;******************************
004
005 ; ADA_EQU.ASM
006
007 ; Initialization constants to facilitate initialization of the CS4215
008
009 ;
010
011 ; Copyright (c) MOTOROLA 1996
012
013 ;
014 ;
015 ; Semiconductor Products Sector
016 ;
017 ;
018 ;
019 ;******************************
020
021 ;
022
023
024
025 NO_PREAMP equ 100000
026
027 LO_OUT_DRV equ 080000
028
029 HI_PASS_FILT equ 008000
030
031 SAMP_RATE_9 equ 003800  ; 9.6 kHz sample rate
032
033 SAMP_RATE_48 equ 003000  ; 48 kHz sample rate
034
035 SAMP_RATE_32 equ 001800  ; 32 kHz sample rate
036
037 SAMP_RATE_27 equ 001000
038
039 SAMP_RATE_16 equ 000800
040
041 SAMP_RATE_8  equ 000000
042
043 STEREO    equ 000400
044
045 DATA_8LIN equ 200300
046
047 DATA_9A   equ 200200
048
049 DATA_8U   equ 200100
050
051 DATA_16   equ 200000
052
053 IMMED_3STATE equ 800000
054
055 XTAL1_SELECT equ 100000  ; 24.576 MHz
056
057 XTAL2_SELECT equ 200000  ; 16.9344 MHz
058
059 BITS_64 equ 000000
060

119
061 BITS_128    equ $040000
062
063 BITS_256    equ $080000
064
065 CODEC_MASTER equ $020000
066
067 CODEC_TX_OFF equ $010000
068
069
070
071 ;CTRL_WD_12 equ NO_PREAMP+HI_PASS_FILT+SAMP_RATE_48+STEREO+DATA_16 ;CLB=0
072
073 ;CTRL_WD_34 equ IMMED_3STATE+XTAL1_SELECT+BITS_64+CODEC_MASTER
074
075 ;CTRL_WD_56 equ $000000
076
077 ;CTRL_WD_78 equ $000000
078
079
080
081 HEADPHONE_EN  equ $800000
082
083 LINEOUT_EN    equ $400000
084
085 SPEAKER_EN    equ $004000
086
087 MIC_IN_SELECT equ $100000
088
089 LEFT_ATTN     equ $010000 ;63*LEFT_ATTN = -94.5 dB, 1.5 dB steps
090
091 RIGHT_ATTN    equ $001000 ;63*RIGHT_ATTN = -94.5 dB, 1.5 dB steps
092
093 LEFT_GAIN     equ $010000 ;16*LEFT_GAIN = 22.5 dB, 1.5 dB steps
094
095 RIGHT_GAIN    equ $001000 ;16*RIGHT_GAIN = 22.5 dB, 1.5 dB steps
096
097 MONITOR_ATTN  equ $001000 ;15*MONITOR_ATTN = mute, 6 dB steps
098
099 ;OUTPUT_SET   equ HEADPHONE_EN+LINEOUT_EN+(LEFT_ATTN*4)
100
101 ;INPUT_SET    equ MIC_IN_SELECT+(15+MONITOR_ATTN)+(RIGHT_ATTN*4)
vectors.asm

001 ;
002 page 132,60
003 ;******************************************************************************
004 ; VECTORS.ASM
005 ; Vector table for the 56303
006 ;
007 ; Copyright (c) MOTOROLA 1996
008 ; Semicondutor Products Sector
009 ; Digital Signal Processing Division
010 ;
011 ;******************************************************************************
012 ;
013 ORG P:0
014 ;
015 vectors JMP START ; Hardware RESET
016 ;
017 jmp *
018 NOP ; Stack Error
019
020 jmp *
021 NOP ; - Debug Request Interrupt
022 ;
023 jmp *
024 NOP ; - Debug Request Interrupt
025 ;
026 jmp *
027 NOP ; - Trap
028 ;
029 jmp *
030 NOP ; - NMI
031 ;
032 NOP ; - Reserved
033 NOP
034 ;
035 NOP ; - Reserved
036 NOP
037 ;
038 jsr main ; - IRQA
039 ;
040 jmp *
041 NOP ; - IRQB
042 ;
043 jmp *
044 NOP ; - IRC
045 ;
046 jsr echo ; - IRQD
047 ;
048 jmp *
049 NOP ; - DMA Channel 0
050 ;
051 jmp *
052 NOP ; - DMA Channel 1
053 ;
054 jmp *
055 NOP ; - DMA Channel 2
056 ;
057 jmp *
058 NOP ; - DMA Channel 3
059 ;
060 jmp *

121
061 NOP ;= DMA Channel 4
062 
063 jmp * ;= DMA Channel 5
064 NOP
065 
066 jmp *
067 NOP ;= Timer 0 Compare
068 
069 jmp *
070 NOP ;= Timer 0 Overflow
071 
072 jmp *
073 NOP ;= Timer 1 Compare
074 
075 jmp *
076 NOP ;= Timer 1 Overflow
077 
078 jmp *
079 NOP ;= Timer 2 Compare
080 
081 jmp *
082 NOP ;= Timer 2 Overflow
083 
084 jsr ssi_rx_isr ;= ESSIO Receive Data
085 
086 jsr ssi_rx_isr ;= ESSIO Receive Data w/ Exception Status
087 
088 jsr ssi_rxls_isr ;= ESSIO Receive last slot
089 
090 jsr ssi_tx_isr ;= ESSIO Transmit Data
091 
092 jsr ssi_txe_isr ;= ESSIO Transmit Data w/ Exception Status
093 
094 jsr ssi_txls_isr ;= ESSIO Transmit last slot
095 
096 NOP ;= Reserved
097 NOP
098 
099 NOP ;= Reserved
100 NOP
101 
102 jmp *
103 NOP ;= ESSI1 Receive Data
104 
105 jmp *
106 NOP ;= ESSI1 Receive Data w/ Exception Status
107 
108 jmp *
109 NOP ;= ESSI1 Receive last slot
110 
111 jmp *
112 NOP ;= ESSI1 Transmit Data
113 
114 jmp *
115 NOP ;= ESSI1 Transmit Data w/ Exception Status
116 
117 jmp *
118 NOP ;= ESSI1 Transmit last slot
119 
120 
121 NOP ;= Reserved
122 NOP

122
185 jmp * ; Available for Host Command
186 NOP
187 jmp * ; Available for Host Command
188 NOP
189 jmp * ; Available for Host Command
190 NOP
191 jmp * ; Available for Host Command
192 NOP
193 jmp * ; Available for Host Command
194 NOP
195 jmp * ; Available for Host Command
196 NOP
197 jmp * ; Available for Host Command
198 NOP
199 jmp * ; Available for Host Command
200 NOP
201 jmp * ; Available for Host Command
202 NOP
203 jmp * ; Available for Host Command
204 NOP
205 jmp * ; Available for Host Command
206 NOP
207 jmp * ; Available for Host Command
208 NOP
209 jmp * ; Available for Host Command
210 NOP
211 jmp * ; Available for Host Command
212 NOP
213 jmp * ; Available for Host Command
214 NOP
215 jmp * ; Available for Host Command
216 NOP
217 jmp * ; Available for Host Command
218 NOP
219 jmp * ; Available for Host Command
220 NOP
221 jmp * ; Available for Host Command
222 NOP
223 jmp * ; Available for Host Command
224 NOP
225 NOP
226 jmp * ; Available for Host Command
227 NOP
228 jmp * ; Available for Host Command
229 NOP
230 jmp * ; Available for Host Command
231 NOP
232 jmp * ; Available for Host Command
233 NOP
234 jmp * ; Available for Host Command
235 NOP
236 jmp * ; Available for Host Command
237 NOP
238 jmp * ; Available for Host Command
239 NOP
240 jmp * ; Available for Host Command
241 NOP
242 jmp * ; Available for Host Command
243 NOP
244 jmp * ; Available for Host Command
245 NOP
246 jmp * ; Available for Host Command

124
247  NOP
248  jmp *
249  NOP
250  jmp *
251  NOP
252  jmp *
253  NOP
254  jmp *

256  NOP
257  jmp *
258  NOP
259  jmp *
260  NOP
261  JMP
262  jmp *

264  NOP
265  jmp *
266  NOP
267  jmp *
268  NOP
269  JMP
270  jmp *

272  NOP
273  jmp *
274  NOP
275  NOP
276  jmp *

278  NOP
279  jmp *
280  NOP
281  NOP
282  jmp *

284  NOP
285  jmp *
286  NOP
287  NOP
288  NOP
289  NOP
290  jmp *

292  jmp *
293  NOP
294  jmp *
295  NOP
296  jmp *
297  NOP
298  jmp *

300  jmp *
301  NOP
302  jmp *
303  NOP
304  jmp *

306  NOP
307  NOP
308  jmp *
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>309</td>
<td>NOP</td>
<td></td>
<td>; Available for Host Command</td>
</tr>
<tr>
<td>310</td>
<td>jmp</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>311</td>
<td>NOP</td>
<td></td>
<td>; Available for Host Command</td>
</tr>
<tr>
<td>312</td>
<td>jmp</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>313</td>
<td>NOP</td>
<td></td>
<td>; Available for Host Command</td>
</tr>
<tr>
<td>314</td>
<td>jmp</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>315</td>
<td>NOP</td>
<td></td>
<td>; Available for Host Command</td>
</tr>
<tr>
<td>316</td>
<td>jmp</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>317</td>
<td>NOP</td>
<td></td>
<td>; Available for Host Command</td>
</tr>
<tr>
<td>318 ;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>319 ;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
intequ.asm

01:******************************************************************************
02:
03: EQUATES for ONYX 56302 interrupts
04:
05: Last update: June 11 1995
06:
07:******************************************************************************
08:
09: page 132,56,0,0,0
10: opt mex
11: intequ ident 1,0
12:
13: if #DEF(I_VEC)
14: ;leave user definition as is.
15: else
16: I_VEC equ $0
17: endif
18:
19: ; Non-Maskable interrupts
20:---------------------------------
21:
22: _RESET equ I_VEC+$00 ; Hardware RESET
23: _STACK equ I_VEC+$02 ; Stack Error
24: _ILL equ I_VEC+$04 ; Illegal Instruction
25: _DBG equ I_VEC+$06 ; Debug Request
26: _TRAP equ I_VEC+$08 ; Trap
27: _NMI equ I_VEC+$0A ; Non Maskable Interrupt
28:
29: ; Interrupt Request Pins
30:---------------------------------
31:
32: _IRQA equ I_VEC+$10 ; IRQA
33: _IRQB equ I_VEC+$12 ;IRQB
34: _IRQC equ I_VEC+$14 ;IRQC
35: _IRQD equ I_VEC+$16 ;IRQD
36:
37: ; DMA Interrupts
38:---------------------------------
39:
40: _DMA0 equ I_VEC+$18 ; DMA Channel 0
41: _DMA1 equ I_VEC+$1A ; DMA Channel 1
42: _DMA2 equ I_VEC+$1C ; DMA Channel 2
43: _DMA3 equ I_VEC+$1E ; DMA Channel 3
44: _DMA4 equ I_VEC+$20 ; DMA Channel 4
45: _DMA5 equ I_VEC+$22 ; DMA Channel 5
46:
47: ; Timer Interrupts
48:---------------------------------
49:
50: _TIM0C equ I_VEC+$24 ; TIMER 0 compare
51: _TIM0CF equ I_VEC+$26 ; TIMER 0 overflow
52: _TIM1C equ I_VEC+$28 ; TIMER 1 compare
53: _TIM1F equ I_VEC+$2A ; TIMER 1 overflow
54: _TIM2C equ I_VEC+$2C ; TIMER 2 compare
55: _TIM2F equ I_VEC+$2E ; TIMER 2 overflow
56:
57: ; ESSI Interrupts
58:---------------------------------
59:
60: _SIORD equ I_VEC+$30 ; ESSI0 Receive Data
61 I_SIORDE EQU I_VEC+$32 ; ESSIO Receive Data With Exception Status
62 I_SIORSLS EQU I_VEC+$34 ; ESSIO Receive last slot
63 I_SIOCTD EQU I_VEC+$36 ; ESSIO Transmit data
64 I_SIOCTDE EQU I_VEC+$38 ; ESSIO Transmit Data With Exception Status
65 I_SIOCTLS EQU I_VEC+$3A ; ESSIO Transmit last slot
66 I_SIIKDE EQU I_VEC+$40 ; ESSII Receive Data
67 I_SIIKDE EQU I_VEC+$42 ; ESSII Receive Data With Exception Status
68 I_SIIKLS EQU I_VEC+$44 ; ESSII Receive last slot
69 I_SIIKTDL EQU I_VEC+$46 ; ESSII Transmit data
70 I_SIIKDE EQU I_VEC+$48 ; ESSII Transmit Data With Exception Status
71 I_SIIKLS EQU I_VEC+$4A ; ESSII Transmit last slot
72
73 ;------------------------------------------------------------------------
74 ; SCI Interrupts
75 ;------------------------------------------------------------------------
76 I_SCIRD EQU I_VEC+$60 ; SCI Receive Data
77 I_SCIRDE EQU I_VEC+$62 ; SCI Receive Data With Exception Status
78 I_SCITD EQU I_VEC+$64 ; SCI Transmit Data
79 I_SCILD EQU I_VEC+$66 ; SCI Idle Line
80 I_SCITM EQU I_VEC+$68 ; SCI Timer
81
82 ;------------------------------------------------------------------------
83 ; HOST Interrupts
84 ;------------------------------------------------------------------------
85 I_HRD T EQU I_VEC+$60 ; Host Receive Data Full
86 I_HTED EQU I_VEC+$62 ; Host Transmit Data Empty
87 I_HC EQU I_VEC+$64 ; Default Host Command
88
89 ;------------------------------------------------------------------------
90 ; INTERRUPT ENDING ADDRESS
91 ;------------------------------------------------------------------------
92 I_INTEND EQU I_VEC+$FF ; last address of interrupt vector space
93
94 LIST
sincos.asm

01 ;
02 ; This program originally available on the Motorola DSP bulletin board.
03 ; It is provided under a DISCLAIMER OF WARRANTY available from
04 ; Motorola DSP Operation, 6501 Wm. Cannon Drive W., Austin, Tx., 78735.
05 ;
06 ; Sine-Cosine Table Generator for FFTs.
07 ;
08 ; Last Update 25 Nov 86 Version 1.2
09 ;
10 sincos macro points,coef
11 sincos ident 1,2
12 ;
13 ; sincos - macro to generate sine and cosine coefficient
14 ; lookup tables for Decimation in Time FFT
15 ; twiddle factors.
16 ;
17 ; points - number of points (2 - 32768, power of 2)
18 ; coef - base address of sine/cosine table
19 ; negative cosine value in X memory
20 ; negative sine value in Y memory
21 ;
22 ; Latest revision - 25-Nov-86
23 ;
24 25 pi equ 3.141592654
26 freq equ 2.0*pi/Gcvf(points)
27 28 org x:coef
29 count set 0
30 dup points/2
31 dc -Gcos(Gcvf(count)*freq)
32 count set count+1
33 endm
34
35 org y:coef
36 count set 0
37 dup points/2
38 dc -Gsin(Gcvf(count)*freq)
39 count set count+1
40 endm
41
42 endm ;end of sincos macro
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:

;----reset delay for codec ----
do #1000,delay_loop
rep #2000 ; 100 us delay (assuming 40MHz VCD)
beet #0,x:M_PDRC ; sends out a 1 on pd0 (rst_codec=1)
movep #0008,x:M_IPRP ; set interrupt priority level for ESSIO to 1
andi #FC,wr ; enable interrupts

; The following data sets up the CS4215 control mode data:
; (CTS = Control Time Slot, U/LN = upper/lower Nibble)

<table>
<thead>
<tr>
<th>CTS1-UN:</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>MLB</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTS1-LN:</td>
<td>CLB</td>
<td>CLB</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

 CLB = 0

<table>
<thead>
<tr>
<th>CTS2-UN:</th>
<th>HPF</th>
<th>X</th>
<th>DFR2</th>
<th>DFR1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTS2-LN:</td>
<td>DFR0</td>
<td>ST</td>
<td>DF1</td>
<td>DFO</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

 CLB = 0

<table>
<thead>
<tr>
<th>CTS3-UN:</th>
<th>ITS</th>
<th>MCK2</th>
<th>MCK1</th>
<th>MCK0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTS3-LN:</td>
<td>BSEL1</td>
<td>BSELO</td>
<td>XCLK</td>
<td>XEN</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CTS4-UN:</th>
<th>TEST</th>
<th>TEST</th>
<th>TEST</th>
<th>TEST</th>
<th>TEST</th>
<th>(TEST MUST BE 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTS4-LN:</td>
<td>TEST</td>
<td>TEST</td>
<td>ENL</td>
<td>DAD</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

 x0 = $8800xx

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

----- 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 #0003C,x:M_PCRC

-----

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

-----

movep #0000,x:M_PCRC ; disable ESSIO

-----

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_CRAO ; 40MHz/16 = 2.5MHz SCLK, WL=16 bits, 4W/F
movep #FF310C,x:M_CRBO ; scck and fsync (sc02) as inputs
movep #0003,x:M_PDRC ; D/C pin = 0 => data mode
movep #003C,x:M_PCRC ; turn on ESSIO except for sc0 and sc2

rts
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_SSISRO
mov dx:-(r6),r6
mov r0,x:(r6)+
mov m0,x:(r6)+
mov #3,m0
mov x:TX_PTR,r0
mov x:-(r6),m0
mov x:-(r6),r0
rti

;******************** SSI TRANSMIT LAST SLOT ISR ****************
ssi_txls_isr
bclr #4,x:M_SSISRO
mov dx:-(r6),r6
mov r0,x:(r6)+
mov m0,x:(r6)+
mov #3,m0
mov x:TX_PTR,r0
mov x:-(r6),m0
mov x:-(r6),r0
rti

;******************** SSI RECEIVE ISR ********************
ssi_rxe_isr
bclr #5,x:M_SSISRO
mov dx:-(r6),r6
mov r0,x:(r6)+
mov m0,x:(r6)+
mov #3,m0
mov x:RX_PTR,r0
nop
movp x:(r0)+,x:M_RX0
mov r0,x:RX_PTR
mov x:-(r6),m0
mov x:-(r6),r0
rti
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>185</td>
<td>;************************** SSI receive last slot ISR **************************</td>
<td></td>
</tr>
<tr>
<td>186</td>
<td>ssi_rxls_isr</td>
<td></td>
</tr>
<tr>
<td>187</td>
<td>move r0,x:(r6)+</td>
<td>; Save r0 to the stack.</td>
</tr>
<tr>
<td>188</td>
<td>move #RX_BUFF_BASE,r0</td>
<td>; Reset rx buffer pointer just in case it was corrupted.</td>
</tr>
<tr>
<td>189</td>
<td>move r0,x:RX_PTR</td>
<td>; Update rx buffer pointer.</td>
</tr>
<tr>
<td>190</td>
<td>move x:-(r6),r0</td>
<td>; Restore r0.</td>
</tr>
<tr>
<td>191</td>
<td>rti</td>
<td></td>
</tr>
</tbody>
</table>

```
References


