REAL-TIME ADAPTIVE CANCELLATION OF SATELLITE INTERFERENCE IN RADIO ASTRONOMY

by

Andrew J. Poulsen

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

Brigham Young University

August 2003

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Andrew J. Poulsen

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Dr. Brian D. Jeffs, Chair
Date	Dr. A. Lee Swindlehurst
Date	Dr. Karl F. Warnick

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Andrew J. Poulsen in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Dr. Brian D. Jeffs
Chair, Graduate Committee

Accepted for the Department

Dr. Michael A. Jensen
Graduate Coordinator

Accepted for the College

Dr. Douglas M. Chabries
Dean, College of Engineering and Technology

ABSTRACT

REAL-TIME ADAPTIVE CANCELLATION OF SATELLITE INTERFERENCE IN RADIO ASTRONOMY

Andrew J. Poulsen

Department of Electrical and Computer Engineering

Master of Science

Radio astronomy is the science of observing the heavens at radio frequencies, from a few kHz to approximately 300 GHz. In recent years, radio astronomy has faced a growing interference problem as radio frequency (RF) bandwidth has become an increasingly scarce commodity. A programmable real-time DSP least-mean-square interference canceller was developed and demonstrated as a successful method of excising satellite down-link signals from both an experimental platform at BYU, and the Green Bank Telescope at the National Radio Astronomy Observatory in West Virginia. A performance analysis of this cancellation system in the radio astronomy radio frequency interference (RFI) mitigation regime constitutes the main contribution of this thesis. The real-time BYU test platform consists of small radio telescopes, low noise RF receivers, and a state-of-the-art DSP platform. This programmable real-time radio astronomy RFI mitigation tool is the first of its kind. Basic tools needed for radio astronomy observations and the analysis and implementation of interference mitigation algorithms were also implemented in the DSP platform, including a power spectral density estimator, a beamformer, and an array signal correlator.

ACKNOWLEDGMENTS

Many people have contributed to the successful completion of this thesis. I would like to express appreciation to my thesis and academic advisor, Dr. Brian D. Jeffs, for his guidance and advice. I have also received much help from the other members of my graduate committee, Dr. A. Lee Swindlehurst and Dr. Karl F. Warnick. The Green Bank, WV field experiments would not have been possible, nor successful without the extensive assistance of Rick Fisher, Jeff Acree and Bill Shank. Others who have given significant help include Chad Hansen, Jared Jones and Jacob Campbell.

I am also indebted to my parents, Robert and Jane Poulsen, for being friends and examples. They have always encouraged me to excel in life. Above all, I appreciate the loving support and devotion of my wife, Katherine, and sweet daughter, Alyssa. I dedicate this thesis to them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Radio Astronomy RFI Mitigation

Astronomers utilize an extremely wide range of the electromagnetic spectrum for deep space observations. Because observing techniques vary greatly depending on the frequency of choice, many distinct disciplines exist, including gamma, X-ray, ultraviolet, optical, infrared, millimeter wave, and radio astronomy [1]. As the name suggests, radio astronomy is the science of observing the heavens at radio frequencies, from a few kHz to approximately 300 GHz.

In recent years, radio astronomy has faced a growing interference problem as radio frequency (RF) bandwidth has become an increasingly scarce commodity. Pertinent frequency band allocation restrictions and designated radio quiet zones have historically been preserved by the FCC, ITU and other regulatory agencies. Some of this protection has been undermined as the global community launches high-powered satellite communication systems. Unfortunately, some countries do not always respect the strict guidelines for protection of bands reserved for passive radio astronomy use. In addition, many desired astronomical sources exist at frequencies outside of the traditionally protected observational bands. Space-based satellite signals are especially problematic because they cannot respect radio quiet zone or remote location policies, and are often spatially closer to the desired signal look direction than ground based interferers. Particularly problematic systems include the Russian Federation Global Navigation Satellite System (GLONASS), and the Iridium satellite phone system. Due to an unfortunate choice of transmission bands, fragile spectral lines important

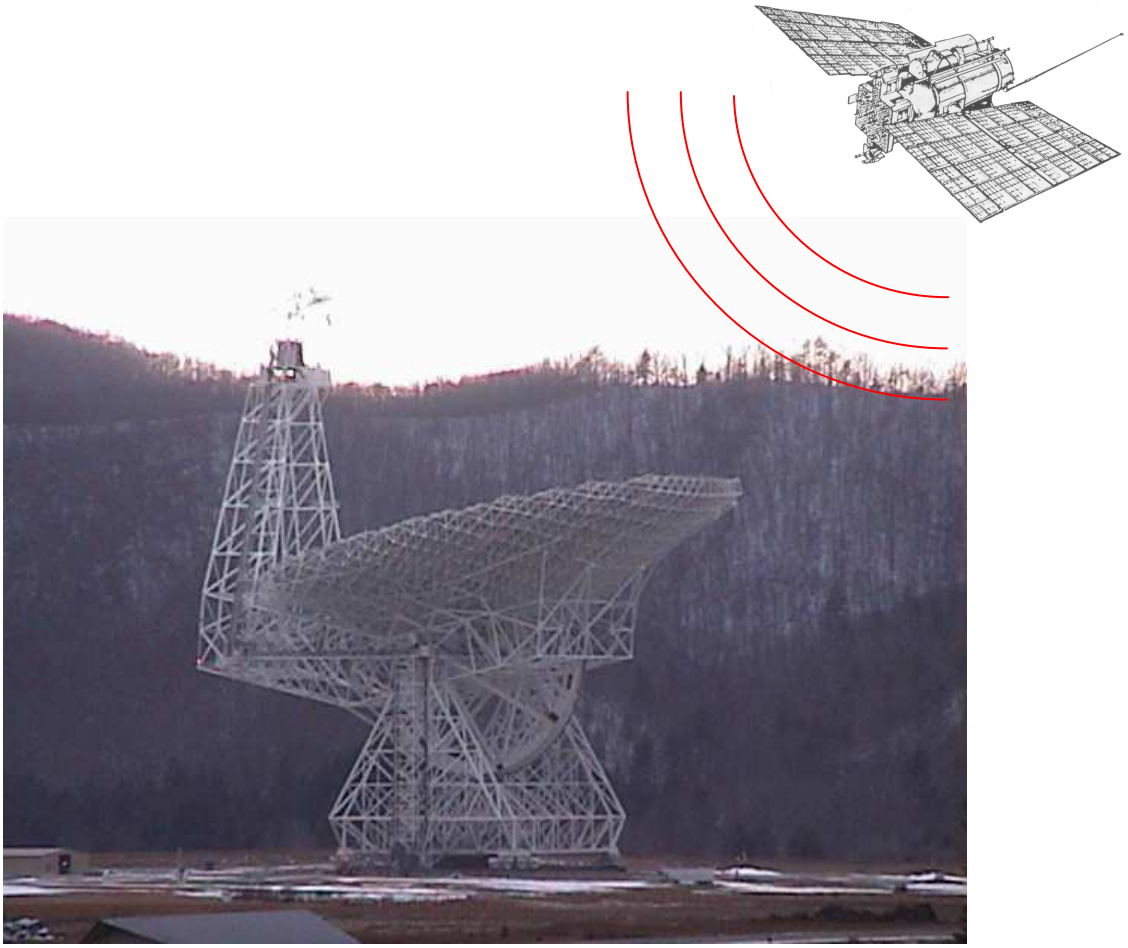


Figure 1.1: Illustration of a GLONASS downlink signal entering the sidelobes of the 100 meter Green Bank Telescope. GLONASS downlink signals are especially problematic because their transmissions corrupt many 1612 MHz hydroxyl (OH) spectral line observations (GLONASS satellite illustration [2]).

to radio astronomy are corrupted, effectively blinding sensitive radio telescopes. Currently, much data is simply discarded due to a lack of effective interference mitigation algorithms.

Radio frequency interference (RFI) mitigation approaches which are customized to a specific observation instrument will be essential to success because of the diverse characteristics of individual radio telescopes. Radio astronomers explore the cosmos by utilizing many platforms such as the Very Large Array (VLA) in New Mexico,

the Green Bank Telescope (GBT) in West Virginia, the Westerbork Array in the Netherlands, and the Very Long Baseline Array (VLBA) dispersed from Hawaii to the Virgin Islands. The next generation of radio telescopes will be even more sensitive and consequently more vulnerable to corruption by RF and microwave sources. The Atacama Large Millimeter Array (ALMA) and the Square Kilometer Array (SKA) are two of these newest projects in development [3, 4]. Each platform is susceptible to potentially very different interference environments due to location, geography, and hardware design. Despite many similarities, an interference mitigation approach valid for one platform may be unsatisfactory for another. Modifications must be explored to customize general solutions to individual platforms.

Interference such as satellite downlink signals must be removed without significantly affecting the telescope main beam shape or attenuating the signal of interest. Satellites sweep across a telescope's field of view transmitting impulsive and broadband signals. A few traditional attempts at interference cancellation include notch filtering [5], blanking [6], and shielding [7]. Despite partial success, each of these approaches carries its own limitations: inadequate cancellation, inability to adapt to time varying characteristics of the interferer, exorbitant amounts of post-processing, or simultaneous cancellation of the desired signal. Consequently, many conventional techniques for excising interference are not practical. Adaptive filtering is a more promising approach [8]. Effective solutions will include real-time mitigation of radio astronomy interference using both adaptive temporal and spatial cancellation techniques. This has only recently been considered a viable approach. Previously, technology could not support the required computational load.

1.2 Current Research in Radio Astronomy Interference Mitigation

Several observatories are actively involved in RFI mitigation development for radio astronomy, including the National Radio Astronomy Observatory (NRAO), the Australia Telescope National Facility (ATNF), and the Netherlands Foundation for Research in Astronomy (NFRA). The Brigham Young University (BYU) radio astronomy research group actively collaborates with the NRAO.

The NFRA, collaborating with Delft University of Technology, is investigating methods of spatial filtering with radio astronomy imaging [9]. The NFRA is also developing phased-array adaptive nulling techniques with the One Square Meter Array [10, 11]. Blanking techniques have been implemented by the National Astronomy and Ionosphere Center (NAIO) at the Aricebo Observatory [12] and by the University of Orleans, France [13]. The ATNF, collaborating with Ohio State University, has developed a technique for subtracting GLONASS interference from hydroxyl (OH) spectral line observations using parametric modelling methods [14]. Berkeley University is investigating several methods for RFI suppression at the Allen Telescope Array, including adaptive cancelling, interferometric nulling, and time blanking [15]. Berkeley and the ATNF have evaluated voltage and power domain post-processing adaptive cancellation techniques of satellite interference with the Rapid Prototyping Array [16]. Other related work is reported in [17, 18, 19, 20].

The BYU radio astronomy research group is actively involved with several approaches to RFI mitigation. This includes an extensive analysis of adaptive array algorithm performance for satellite interference cancellation [21]. It has been demonstrated that adaptive array algorithms are capable of much better performance when assisted by an auxiliary antenna [22, 23]. Other active BYU RFI research topics include cancellation using subspace projection methods, multifeed spatial filtering [24], and time blanking of air traffic control radar interference using a Kalman estimator.

1.3 Proposed Approach

Typical satellite interferers present a non-stationary signal due to their relative motion with respect to both the desired space source and the antenna. Thus prospective RFI mitigation solutions should be temporally and/or spatially adaptive to properly excise the interference. One solution to real-time RFI mitigation is the least-mean-square (LMS) algorithm, a temporally adaptive filter [25, 26]. This algorithm has become the most commonly used adaptive filtering method due to its simplicity, robustness, and good performance in a wide range of applications.

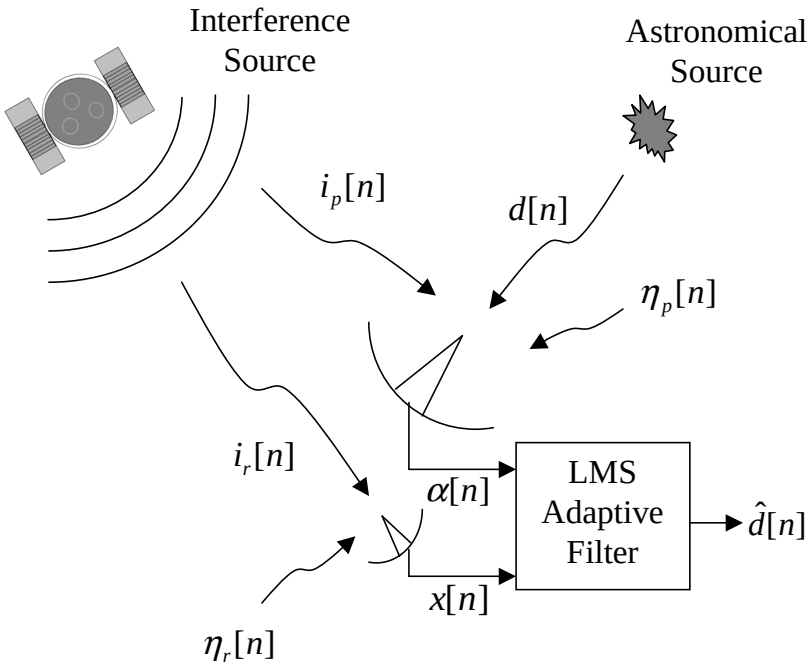


Figure 1.2: LMS adaptive filter used for radio astronomy RFI mitigation

Figure 1.2 illustrates the implementation of the LMS filter in the radio astronomy interference cancellation regime. The radio telescope not only receives the desired signal, $d[n]$, from an astronomical source through the antenna main beam, but also the interfering signal, $i_p[n]$, seen through the antenna response side lobes. There will also always be a white noise component, $\eta_p[n]$, due to cosmic background noise, thermal receiver noise, atmospheric noise, and ground noise from antenna spillover. The net signal received at the feed of the telescope is the sum of these three signals, $\alpha[n] = d[n] + i_p[n] + \eta_p[n]$. Due to higher reference antenna gain in the direction of the interferer, the reference input to the adaptive filter, $x[n]$, contains a higher interference-to-noise ratio (INR) copy of the interferer, $i_r[n]$, along with a white noise component, $\eta_r[n]$. Note that the subscripts p and r stand for the primary and reference antennas, respectively. The LMS filter adaptively estimates and subtracts the interference in the telescope signal to produce a clean estimate of the desired astronomical source, $\hat{d}[n]$.

The real-time LMS adaptive filter proved to be a successful method of excising GLONASS satellite down-link signals from both the BYU experimental platform and NRAO's Green Bank Telescope (see Chapter 4). A performance analysis of the LMS adaptive filter in the radio astronomy RFI mitigation regime constitutes the main contribution of this thesis.

The interference mitigation development effort at BYU, including that reported in this thesis, consists of three separate phases. First, adaptive filtering algorithms are evaluated using computer simulations and synthesized data. Second, because computer simulations cannot sufficiently model all phase, calibration, and other signal and system characteristics, a real-time test platform consisting of three 10-foot diameter radio telescopes and a real-time DSP processor is used for additional algorithmic testing. Third, the final experiments and implementation occur at facilities of the National Radio Astronomy Observatory (NRAO). The test platform used in phases two and three includes low noise microwave receivers and a state-of-the-art DSP platform featuring four processors with four channels of 65 M sample/second digital receivers. This programmable real-time radio astronomy RFI mitigation tool is the first of its kind. Multiprocessor DSP programming is especially demanding since it involves real-time programs, communication with specialized hardware, and painstaking optimization of code. Hence, only the most promising of the simulated algorithms are implemented in the DSP platform.

Several basic tools are needed for radio astronomy observations and the analysis and implementation of interference mitigation algorithms. Three tools have been implemented in our DSP platform: a PSD estimator, a beamformer, and an array signal correlator. The beamformer and array signal correlator will facilitate future analysis of real-time spatially adaptive algorithms.

1.4 Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2, *Experimental Platform*, describes the setup of the test platform used for algorithmic development and research in RFI mitigation. Specifications,

schematics, and performance details are provided for each of the following subsystems: the BYU Very Small Array (VSA), the custom RF low noise receiver, and the DSP platform.

Chapter 3, *Tools for Radio Astronomy Observations and RFI Mitigation*, describes the background, DSP implementation, and performance of three basic tools needed for radio astronomy observations and the analysis and implementation of interference mitigation algorithms. These include a PSD estimator, a beamformer, and an array signal correlator.

Chapter 4, *Real-time RFI Cancellation with an LMS Adaptive Filter*, addresses the real-time implementation of the LMS algorithm for RFI suppression. Extensive descriptions of test setups and other important considerations are included. The results and corresponding analysis of tests performed both with the BYU VSA and the Green Bank Telescope (GBT) are presented.

Chapter 5, *Conclusion*, outlines the thesis' contributions to science along with suggestions of possible future research.

Appendix A, *Source Code*, contains the LMS adaptive filter simulation code used to produce the results in Section 4.4, the antenna steering software described in 2.2, and the C code for the DSP applications described in Sections 3.1.1, 3.2.3, and 4.5.

Appendix B, *Experimental Platform*, gives additional schematics and details concerning the experimental platform described in Chapter 2. These include the results of BYU RF receiver local oscillator (LO) tests for enabling clean OH observations, the 6216 digital receiver decimation anti-aliasing filters, etc.

Appendix C, *Green Bank Test Details*, provides additional details about the Green Bank field experiments. These include tables of highly red-shifted 1612 MHz OH maser sources, scheduling calendars, data realignment parameters, etc.

Chapter 2

Experimental Platform

2.1 The BYU Very Small Array

As mentioned previously, prospective adaptive filtering algorithms are evaluated using computer simulations and synthesized data. As these simulations cannot satisfactorily model all phase, calibration, and other signal and system characteristics, a real-time test platform array of three radio telescopes was developed as part of this thesis research to facilitate additional algorithmic studies. We have nicknamed this array the BYU Very Small Array (VSA) as the counterpart to New Mexico's Very Large Array (VLA). Despite their small aperture, these antennas are ideal for inexpensive analysis of prospective interference mitigation algorithms.

The three 10-foot diameter telescope antennas used in the BYU VSA were based on a design used for radio astronomy educational purposes by the Massachusetts Institute of Technology (MIT) Small Radio Telescope (SRT) research initiative and manufactured by Kaul-Tronics Inc. [27]. As compared to the SRT, our system included a different feed design, a custom RF receiver, modifications to hardware and software to permit multiple antenna array operation, and the real-time DSP platform. Table 2.1 details a few of the important antenna specifications. The positioning of each 3 meter antenna is driven by dual azimuth/elevation rotors. The main advantage of the SRT system is its relatively low cost. In exchange, however, we did have to dedicate significant effort to coordinating proper positioning and tracking control; this will be discussed in Section 2.2.

Figure 2.1: The BYU Very Small Array

Table 2.1: VSA antenna specifications. The 3 dB beamwidth, gain, and highest sidelobe level are from simulations using a circular waveguide feed at 1612 MHz. The sidelobe levels are most likely higher.

Location	Provo, UT
Latitude	40.25°
Longitude	111.65°
Diameter (D)	120 inches
Focal Length (F)	45.6 inches
F/D Ratio	0.38
3 dB Beamwidth at 1.6 GHz	5°
Gain	29 dBi
Highest Sidelobe Level	-4 dBi

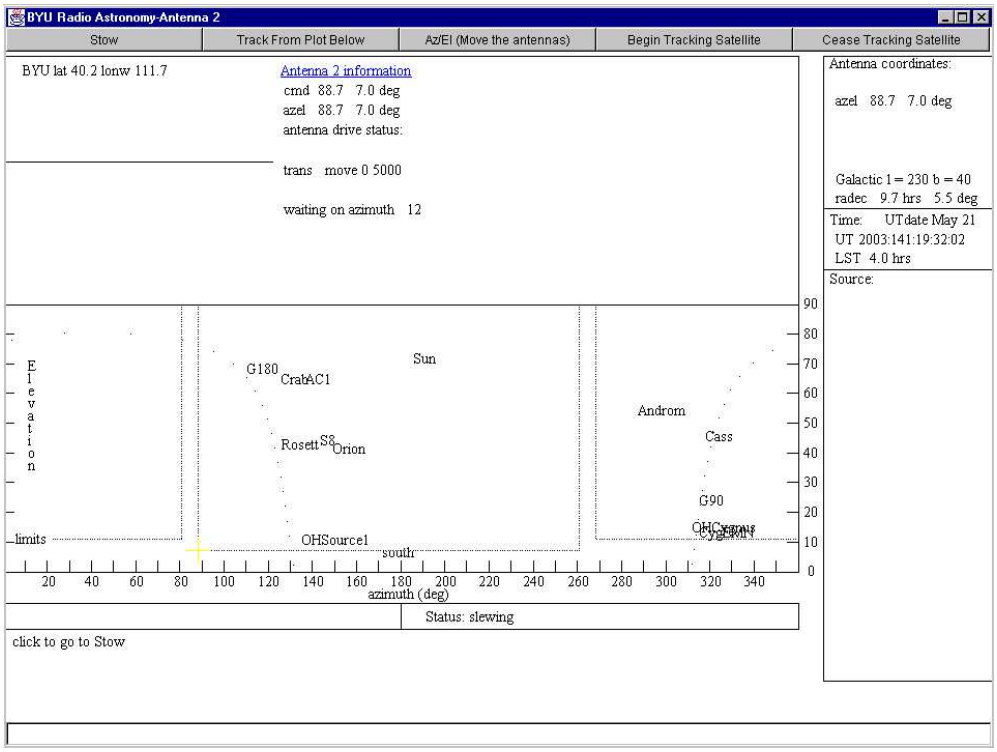
2.2 Antenna Control

Antenna positioning hardware and software were developed in order to track satellite interference sources and celestial objects in real-time. This involved pinpointing and resolving the problems encountered in the complex interface between our host computer, the positioning hardware, and the dual azimuth/elevation rotors.

The antenna steering software is an adaptation of a java application developed by the MIT SRT research initiative [27]. Several changes were made to the code provided by MIT. All functions relating to the SRT receiver were removed, while retaining its ability to track any celestial source given its right ascension and declination (1950 epoch). We added the capability to track satellites by creating an interface between the java software and Nova for Windows, a powerful satellite tracking application [28].

The positioning hardware also came from a design by MIT (see Figure B.1). Because it was not yet available for purchase, a BYU technician built the hardware to the SRT design. As with the antenna control software, we also modified the original hardware design. We removed a few unnecessary components which controlled a SRT noise calibration system and added the capability to use an external power supply for faster antenna drive slew rates.

(a)



(b)

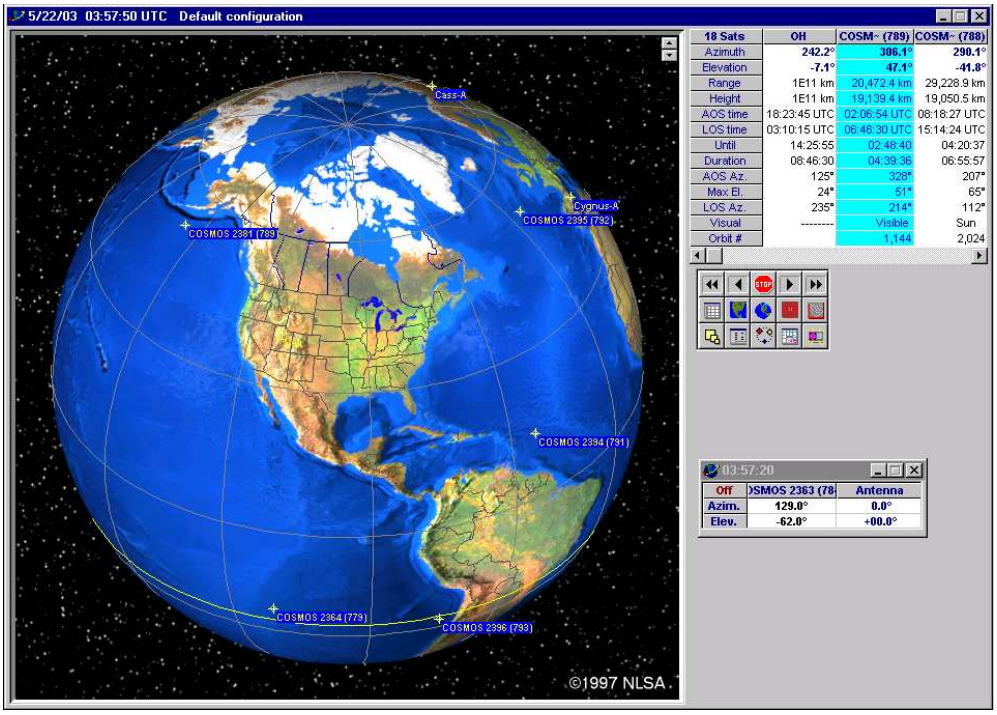


Figure 2.2: (a) Java software used to position the VSA, (b) Nova software used for tracking satellites

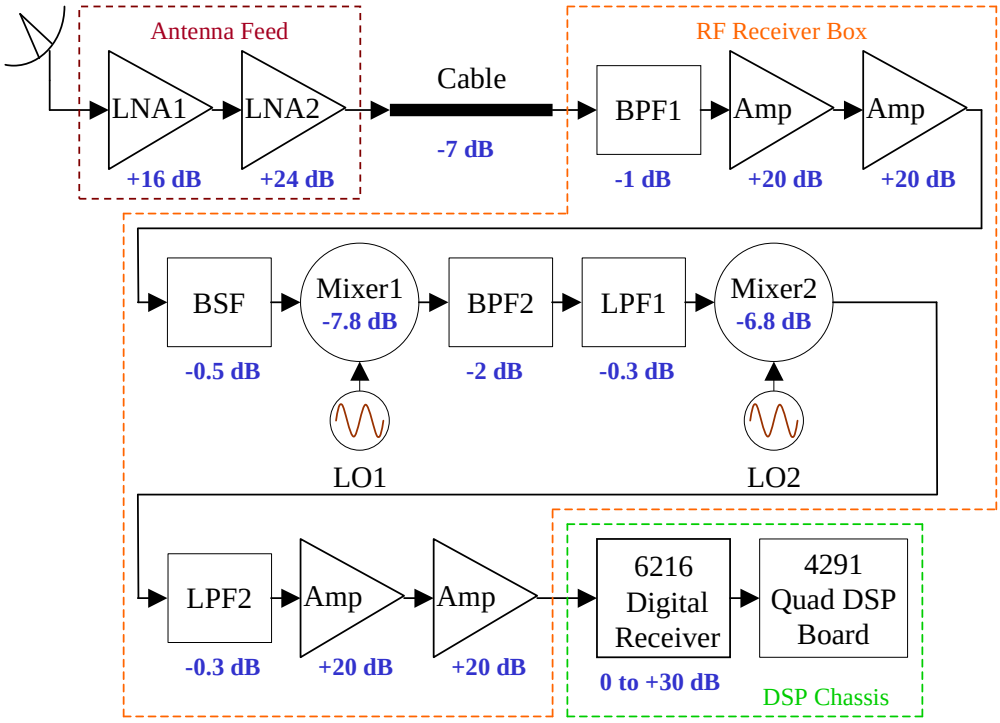


Figure 2.3: The block diagram of the radio telescope VSA receiver, including the power gain or loss of each element. Several filters (BP=band pass, LP=low pass, and BS=band stop) are included in the design for signal image and out-of-band interference rejection. BPF2 is a surface acoustic wave (SAW) filter. The 6216 digital receiver (discussed in Section 2.4) provides 30 dB of tunable gain.

2.3 RF Receiver

Brett Walkenhorst initially designed and built the radio frequency (RF) receiver [29]. Chad Hansen performed most of the subsequent revisions and receiver work. A system block diagram and specifications are shown in Figure 2.3 and Table 2.2, respectively.

A low system noise temperature design, necessary for observing weak radio astronomy signals, was implemented with two front end low noise amplifiers (LNAs) located at the feed of each VSA antenna. The primary device used for LNA1 is a wide-band 16 dB amplifier with a noise temperature of 50° Kelvin over the entire operational frequency of 1.3–1.7 GHz. An alternate narrow-band 16 dB gain device

Table 2.2: RF receiver specifications

Operational Frequency	1.3–1.7 GHz
Final IF Center Frequency	8–24 MHz
IF Bandwidth	16 MHz
System Gain	94-124 dB
System Noise Temperature (Calculated)	52° K (wide-band LNA) 31° K (1612 MHz narrow-band LNA)

has also been used for LNA1. This unit has a noise temperature of $< 28°$ Kelvin, optimized for observations of the 1612 MHz OH maser line.

A few modifications have been made to Brett Walkenhorst's receiver design. The most significant was the addition of a bandpass filter with a 1.3–1.7 MHz pass-band (BPF1). When attempting to make 1612 MHz OH observations, we encountered overwhelming interference at the output of the RF receiver due to interference both internal and external to the receiver system. The addition of BPF1 significantly decreased, but did not eliminate all interference. Some of this residual interference was due to the first local oscillator (LO1) harmonics mixing into the processing bandwidth through the second LO (LO2). Additionally, much of the interference was buried well below the observational noise floor, but became significant with time integration. With an antenna pointed to zenith, we monitored a 4 MHz band centered at 1612 MHz with many combinations of LO1, LO2 and the final DSP IF. Several combinations produced a relatively interference-free spectrum, including LO1=2458 MHz, LO2=834 MHz, and the DSP IF = 12 MHz. For reference, the results of these tests are included in Appendix B.2. They can be used to determine proper LO settings for clean OH maser line observations.

The two stage mix-down involves both high side (Mixer1) and low side (Mixer2) mixing. As a result, the desired spectrum is inverted at the input to the DSP. This is corrected by the 6216 digital receiver in the DSP platform (see Section 2.4).

A radio astronomer relies on integration in order to detect signals buried deep below the noise floor. Therefore, a radio telescope receiver should be stable over time. The stability of the BYU VSA receiver system is discussed in Section 3.1.5.

2.4 Real-time DSP Platform

A major part of the test platform is the real-time subsystem, which is comprised of four parallel digital signal processors (DSPs) with four channels of 65 M sample/second analog to digital conversion, and dedicated digital receiver chips as the interface with the analog RF receivers. The purpose of the DSP platform is to evaluate real-time performance of cancellation algorithms. Real-time implementation of these algorithms on a state-of-the-art DSP platform is an extremely effective method of measuring interference mitigation performance. Multiprocessor DSP programming is especially demanding since it involves real-time programs, communication with specialized hardware, and painstaking optimization of code.

Our DSP system is shown in Figure 2.4. The four RF inputs are fed into two Pentek 6216 digital receiver boards. Each channel contains an analog variable gain amplifier, with a dynamic range of $G_{var} = 0$–30 dB, followed by a low-pass anti-aliasing filter with a 25 MHz cutoff frequency. The user can elect to bypass these two components and directly enter a 12-bit A/D converter. The sample clock, F_c, is provided by either the internal 64 MHz crystal oscillator or an external source up to 65 MHz. Furthermore, the sample clock can be decimated by a factor of $D_s = 1,\ 2,\ 4,\ 6,\ 8$ or 10. Multiple channels, including separate 6216s, can be configured to share a common sample clock and synchronize phase across all channels via a sync bus.

After A/D conversion, the data can either be sent directly to the DSP board, or to a digital down converter. The down convertor involves conversion to complex baseband representation, band selection, signal decimation by D, and bandpass filtering to achieve the desired complex sampling rate. This process is highly customizable. A specified intermediate frequency f_c in the range of $\pm\frac{1}{2}F_c$ is mapped to DC (complex baseband representation); in this process, the spectrum can be inverted by setting

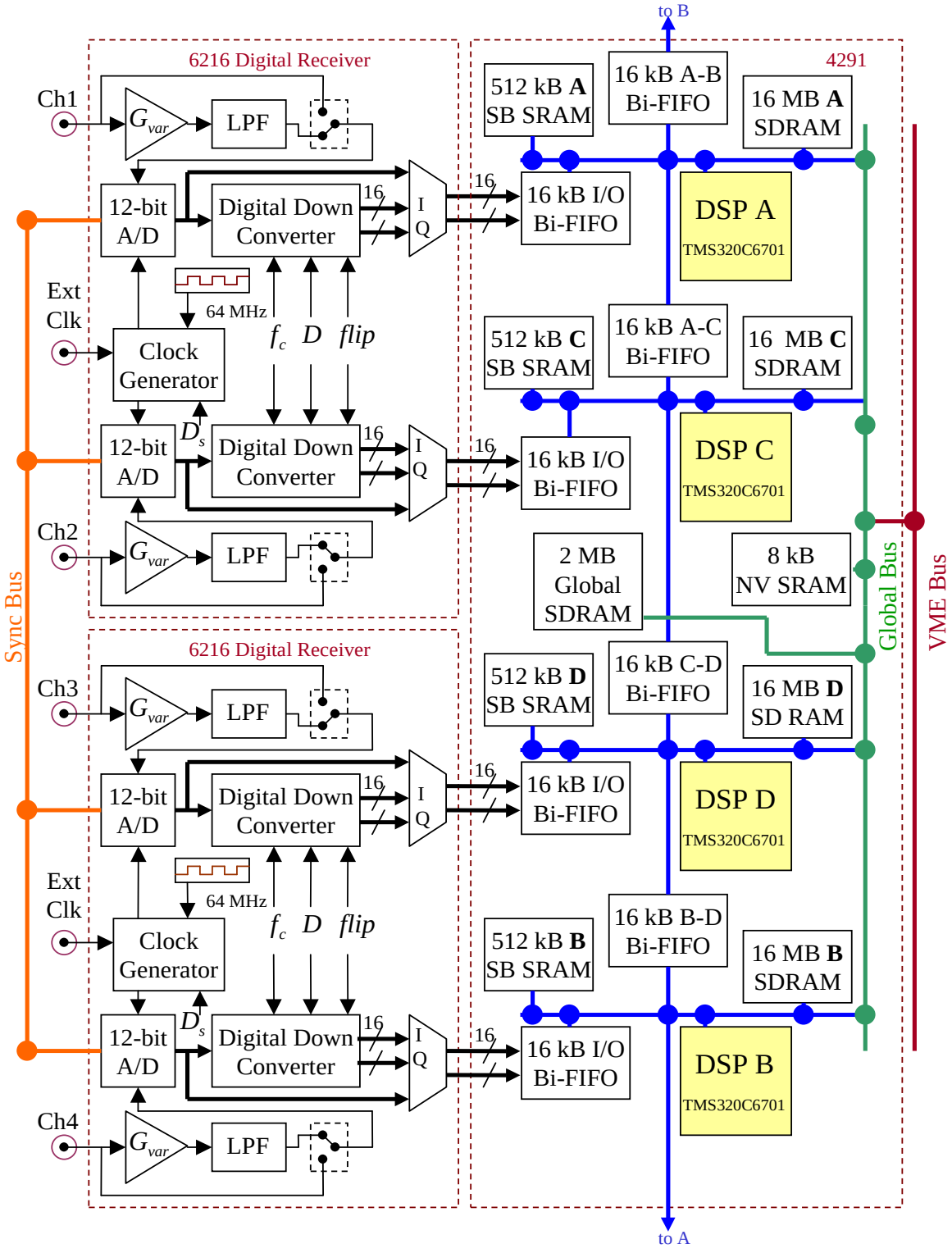


Figure 2.4: The DSP platform architecture. For simplicity, I have not included some of the components and control signals. More detailed Pentek schematics of the 6216 Digital Receiver and 4291 Quad TMS320C6701 Processor VME Board are found in Figures B.2 and B.3 [30].

the $flip$ flag. D can be assigned one of six values: 2, 4, 8, 16, 32 or 64. In order to avoid aliasing upon decimation, the signal is also bandpass filtered by one of six digital FIR filters, each corresponding to a specific decimation rate. These filters, along with their respective magnitude responses, are shown in Appendix B.3.

Following the digital receiver, each input signal enters the Pentek 4291 quad DSP board via a 16 kB I/O Bi-FIFO (bidirectional first in first out register). This method of data transfer is extremely efficient and customizable–especially with the automated direct memory access (DMA) data transfer available on each Texas Instrument (TI) processor. Each processor is also connected to two of the three other processers via additional 16 kB inter-processor Bi-FIFOs. Ideally, this is also performed with DMA, enabling memory access and communication between processors without a significant penalty on DSP processing capacity. Each Bi-FIFO boasts a maximum data transfer rate of 333 MB/sec.

Each TI TMS320C6701 processor has a clock speed of 167 MHz, contains six separate ALUs (arithmetic logic units), and is capable of 1 GFLOPS (billion floating point operations per second) for a total peak board capacity of 4 GFLOPS. On the 4291 DSP board, each processor has access to a host of different memory types. The fastest is the on-chip 256 kB SRAM (static random access memory). Externally, each processor has access to 512 kB SB (synchronous burst) SRAM and 16 MB of SDRAM (synchronous dynamic random access memory) with access speeds of 667 MB/sec and 333 MB/sec, respectively. All four processors share 2 MB of global SRAM and 8kB of fast non-volatile (NV) SRAM.

Data transfer from the 4291 Quad TMS320C6701 Processor VME Board to the host PC is accomplished via a separate VME card manufactured by SBS Technologies, Inc. The maximum data transfer speed from DSP to PC is 1 MB/sec.

Specific applications implemented in the DSP platform are described in Section 3.1.1 (a PSD estimator), Section 3.2.3 (a beamformer and correlation estimator), and Section 4.5 (the least-mean-square (LMS) adaptive filtering algorithm).

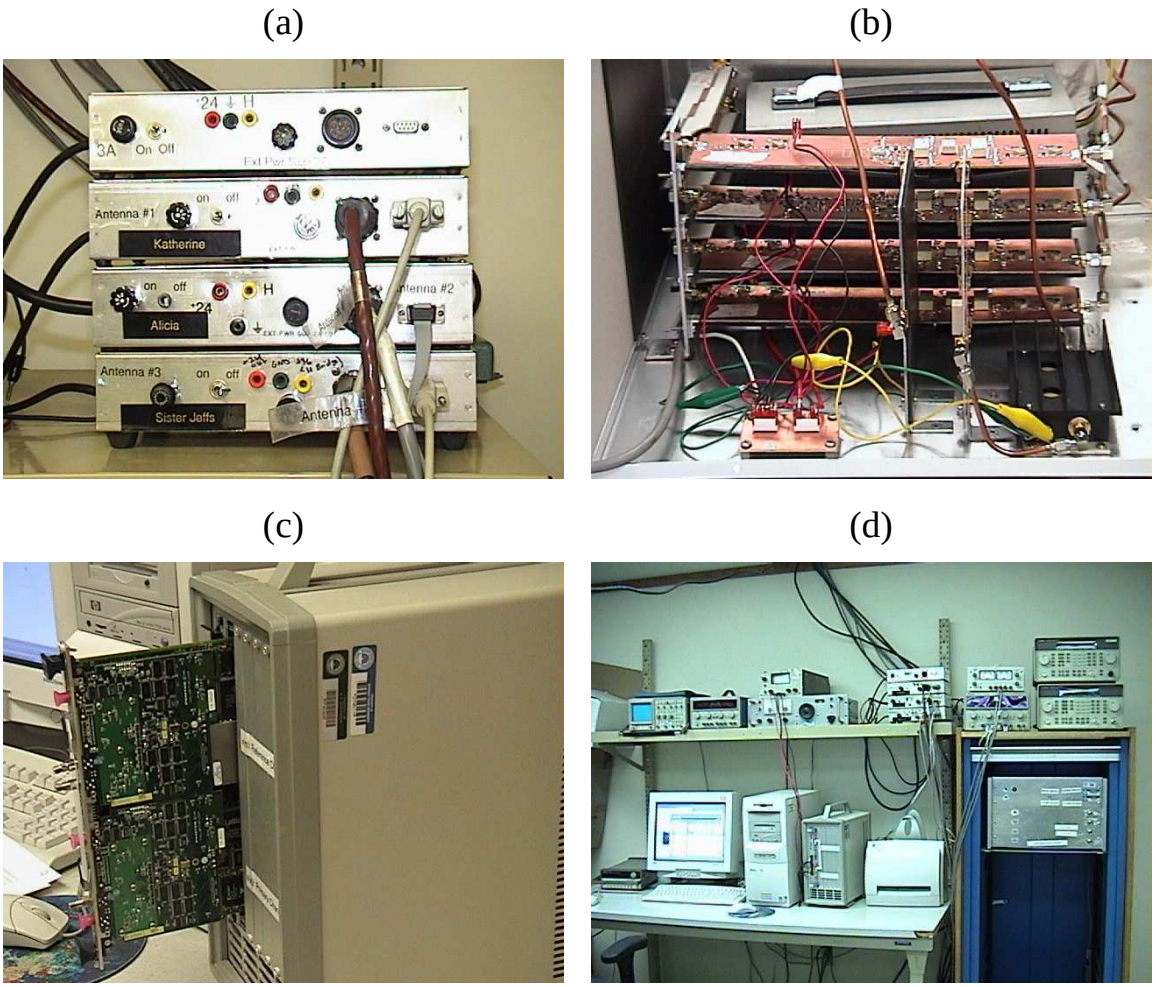


Figure 2.5: (a) Antenna positioning hardware used to steer the VSA antennas, (b) Four channel RF receiver system, (c) Real-time DSP platform, (d) BYU VSA control/receiver station

2.4.1 Portability

After successful algorithmic testing with the VSA, it may be desirable to perform tests with a commissioned radio telescope. Two separate field experiments have been performed with the Green Bank Telescope (GBT–see Section 4.7). As each telescope maintains its own receiver system, only the DSP platform and peripheral devices need be transported to the test location.

The DSP platform is housed in a seven slot VME card cage. This chassis can be packaged into a custom foam protected package which can easily fit into a carry-on overhead bin on a commercial aircraft. The host PC chassis must also be shipped because it contains a necessary PCI card and software for proper control and analysis of the DSP platform.

Chapter 3

Tools for Radio Astronomy Observations and RFI Mitigation

Several basic tools are needed for radio astronomy observations and the analysis and implementation of interference mitigation algorithms. Three of the most important, including a PSD estimator, a beamformer, and an array signal correlator, were implemented in real-time DSP code for the VSA. The background, DSP implementation, performance, and analysis of these three tools are discussed in this chapter.

3.1 PSD Estimation

Estimation of the power spectral density (PSD) of a signal is a necessary tool for radio astronomers. Many different techniques and algorithms exist for estimating the PSD, including modifications of the periodogram: Bartlett's method of periodogram averaging [31], Welch's method [32], and Blackman-Tukey method of periodogram smoothing [33]. Other approaches to spectrum estimation include the minimum variance method [34], the maximum entropy method [35], the multiple signal classification method (MUSIC) [36], and a variety of parametric methods. An extensive introduction to the discipline of spectrum estimation can be found in several textbooks [37].

For our PSD estimates, we chose the simple Bartlett's method, which utilizes periodogram averaging of non-overlapping rectangular windows. For a random process $x[n]$, Bartlett's method is defined as

$$\hat{S}_x[\omega_k] = \frac{1}{M} \sum_{m=0}^{M-1} \left| \sum_{n=0}^{N-1} x[n + mN] e^{-jn\omega_k} \right|^2 , \tag{3.1}$$

where estimates are found at N discrete frequencies, $\omega_k = \frac{2\pi k}{N}$ (N is the length of the DFT), and M is the number of averaged windows. M determines the length of signal integration.

Bartlett's method was selected for several reasons. Compared with other approaches, it was relatively simple to implement in our real-time DSP platform. DFT calculations are very efficient using FFT algorithms available in hand optimized assembly code. Moreover, this approach does not require additional calculations for windowing the data. By not overlapping the rectangular windows, the PSD estimator can process a larger bandwidth for a fixed processor capacity. Furthermore, the averaged periodogram is an asymptotically unbiased and consistent estimate, so that

$$\lim_{M\to\infty} E\{\hat{S}_x[\omega_k]\} = S_x(\omega_k), \tag{3.2}$$

$$Var\{\hat{S}_x[\omega_k]\} \approx \frac{1}{M} S_x^2(\omega_k), \tag{3.3}$$

$$\text{and } \lim_{M\to\infty} Var\{\hat{S}_x[\omega_k]\} = 0. \tag{3.4}$$

Most signals of interest to a radio astronomer are buried deep in the observational noise, $\eta[n]$. The signal received by the telescope, $x[n]$, contains both the signal of interest, $d[n]$, and additive noise:

$$x[n] = d[n] + \eta[n]. \tag{3.5}$$

Since $d[n]$ and $\eta[n]$ are statistically independent, the power spectral density of $x[n]$ is

$$S_x(\omega) = S_d(\omega) + S_\eta(\omega). \tag{3.6}$$

Estimation of the desired spectrum $S_d(\omega)$ is obtained by separately measuring both $\hat{S}_x[\omega_k]$ and $\hat{S}_\eta[\omega_k]$, and forming the difference

$$\hat{S}_d[\omega_k] = \hat{S}_x[\omega_k] - \hat{S}_\eta[\omega_k]. \tag{3.7}$$

The estimate of $\hat{S}_\eta[\omega_k]$ can be obtained by pointing the telescope to a location near the desired coordinates, but one in which $d[n]$ falls out of the telescope main lobe; in many cases, the statistics of $\eta[n]$ will be almost identical in the two pointing directions.

Alternatively, an RF absorber can be positioned to block the aperture of the feed. This method of obtaining $\hat{S}_\eta[\omega_k]$ is effective if $\eta[n]$ is spectrally white. It is important to note, however, that the noise power/temperature will typically increase when an absorber blocks the feed. Since $\hat{S}_x[\omega_k]$ and $\hat{S}_\eta[\omega_k]$ are estimated using separate data sets, their corresponding spectral variance or estimation errors are independent of one another.

In order to produce a reliable estimate of the desired signal spectrum, $\hat{S}_d[\omega_k]$, the variance of $\hat{S}_x[\omega_k]$ due to $\eta[n]$ must be decreased to a level less than the important features of $S_d[\omega_k]$. If $d[n]$ is buried deep below the noise floor of $S_\eta(\omega_k)$, then Eq. 3.3 can be approximated as

$$Var\{\hat{S}_x[\omega_k]\} \approx \frac{1}{M} S_x^2(\omega_k) \approx \frac{1}{M} S_\eta^2(\omega_k) \tag{3.8}$$

because the contribution of d[n] to the variance is negligible. From Eqs. 3.7 and 3.8, it can be shown that

$$Var\{\hat{S}_d[\omega_k]\} = Var\{\hat{S}_x[\omega_k]\} + Var\{\hat{S}_\eta[\omega_k]\} \approx \frac{2}{M} S_\eta^2(\omega_k). \tag{3.9}$$

Standard deviation (σ) is often used instead of spectral variance due to its physical meaning of units of power. Hence, for reliable detection,

$$\sqrt{\frac{2}{M}} S_\eta(\omega_k) \ll S_d(\omega_k). \tag{3.10}$$

As a rule of thumb, sure detection requires at least a 10σ change in the spectral baseline [27],

$$10\sqrt{\frac{2}{M}} S_\eta(\omega_k) < S_d(\omega_k). \tag{3.11}$$

3.1.1 DSP Implementation

DSP implementation of the PSD estimator was challenging despite our choice of the straightforward Bartlett's method of periodogram averaging. The finished product is a useful tool for spectral radio astronomy observations.

Fortunately, a hand-optimized floating-point FFT algorithm developed by Texas Instruments was available for the most computationally intensive portion of

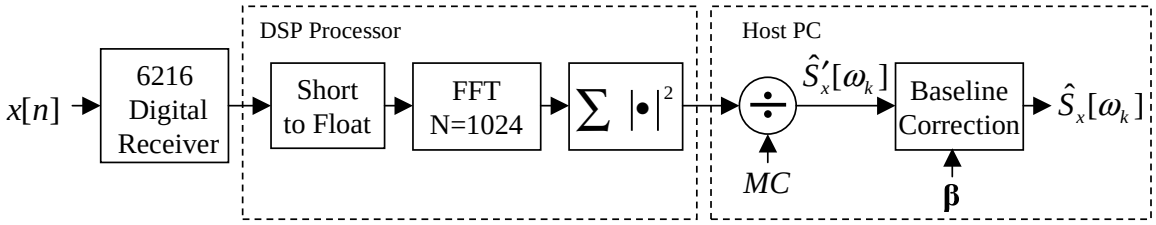


Figure 3.1: DSP implementation of the PSD estimator

PSD estimation, the DFT calculation. This routine implements the decimation-in-time radix-2 FFT algorithm. Unfortunately, the assembly code is written for a "little endian platform" (referring to byte MSB, LSB ordering), while ours is set to big endian. Ultimately, we were able to edit the code to work properly with our big endian setup.

Figure 3.1 illustrates the general structure of the DSP implementation of the PSD estimator. The input signal x[n] is sampled by the 6216 digital receiver and sent to one of four processors. This process not only involves an A/D convertor, but also conversion to a complex baseband representation, band selection, signal decimation, and bandpass filtering to achieve the desired complex sampling rate. Upon arrival at the DSP processor, the signal is converted from fixed to floating point format in preparation for processing. Because other applications, including beamforming and LMS adaptive filtering, would be using floating point computation, this was also the format chosen for PSD estimation.

The DFT of $x[n]$ is computed in non-overlapping blocks of 1024 samples, independent of the sample rate. Next the magnitude squared/power calculation of the DFT output is made. Due to data streaming limitations between the DSP platform and host PC, some integration must be completed in the DSP in order to reduce the DSP to host PC data transfer rate. The user can specify a desired DSP integration and total program run time. For example, if the DSP integration is set to 2 seconds and total run time to 10 minutes, every 2 seconds a noncumulative 2 second integration is sent to the host PC for a total of 300 disjoint 2 second integrations. These can

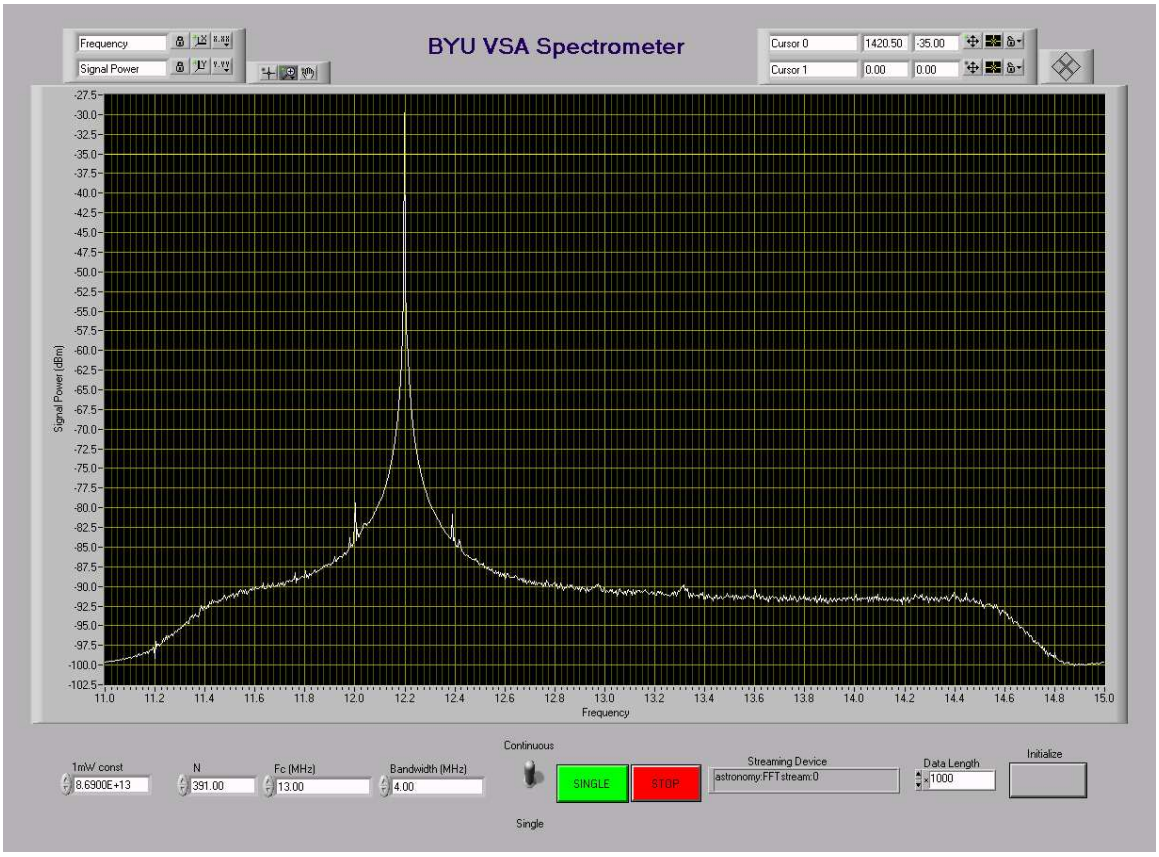


Figure 3.2: A real-time LabView DSP PSD estimator displays the input spectrum on the host PC. A -30 dBm, 12.2 MHz sinusoid was the input to the 6216 digital receiver.

then be added together in post processing for longer integration times and variance convergence analysis. Normalization by M (integration time) and C (power calibration constant–see Section 3.1.2) occur in the host PC. Additionally, the spectral ripple introduced by the signal decimation bandpass filter in the 6216 digital receiver is removed in the host PC (see Section 3.1.3).

The DSP PSD estimator has two modes of operation. Either the data is written to a file for post-processing, or displayed real-time using a LabView application on the host PC. An example of the real-time output is shown in Figure 3.2. This tool displays a calibrated spectrum in dBm with several available analysis tools, including zoom and an option of using cursors to monitor specific frequency bins.

3.1.2 Power Calibration

As shown in Eq. 3.1, the averaged periodogram is normalized only by M, the number of DFT blocks used for integration. It is also desirable to produce a calibrated PSD plot displaying realistic power levels at the input to the DSP. A few variables determine the magnitudes measured by the DSP platform. These include the digital receiver variable gain setting (G_{var}), the mapping used in the A/D convertor from voltage to a fixed point digital signal, the additional gain introduced by the decimation bandpass filter, and a gain of 1024 introduced by the DFT calculation.

In order to calibrate the PSD plots, a precision signal generator was used as an input to the DSP. A 0 dBm sinusoid was placed at the center of a DFT bin and its peak magnitude was measured by the PSD estimator. Three of these tests, taken consecutively for a total of 6 hours, are presented in Figure 3.3. The power convergence during the first 20 minutes of the first of these tests, 12.00056 MHz, is due to the signal generator warming up. Based on the average DSP magnitude for these tests, a power calibration constant of 2.35×10^{13} gave accurate estimates of the input signal power. By including the digital receiver variable gain parameter, G_{var}, I obtained the final expression for C used in all spectral plots of this thesis,

$$C = \left(2.35 \times 10^{13}\right) \left(10^{\frac{G_{var}}{10}}\right). \tag{3.12}$$

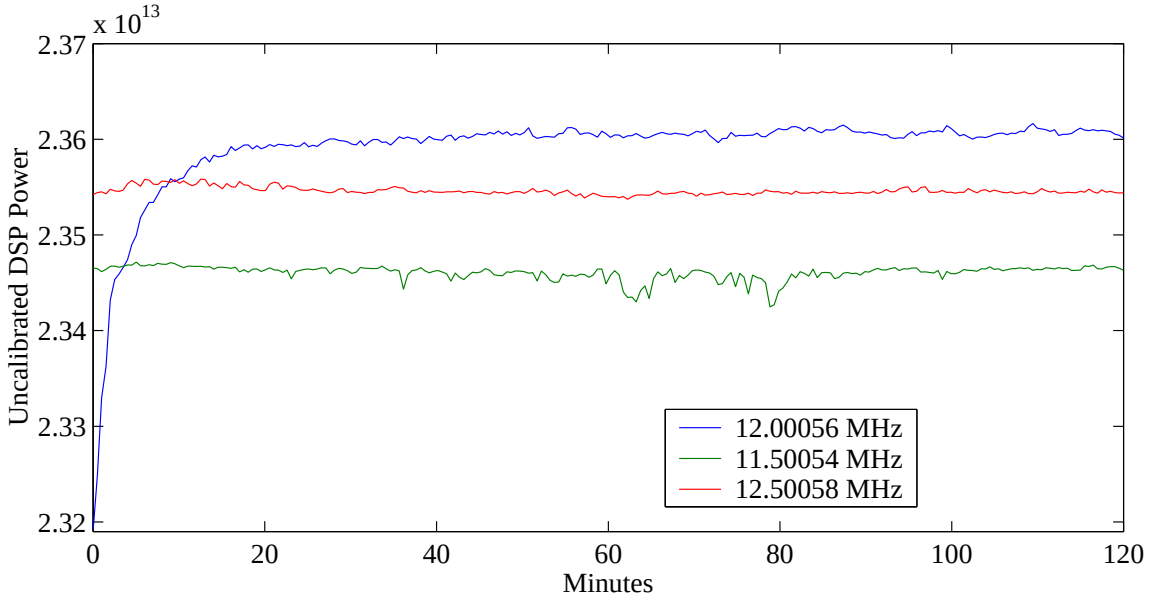


Figure 3.3: Three tests used to determine the DSP power calibration constant, C. For each two hour test, a 0 dBm sinusoid was placed at the center of a DFT bin. The power was monitored in 30 second averaged segments, each made up of 117180 DFT blocks.

This constant applies to any processing bandwidth or choice of G_{var}. As illustrated in Figure 3.1, power calibration is accomplished through division by MC in the host PC.

3.1.3 Baseline Corrections

Not only is it important to calibrate the power levels of the PSD estimator, but frequency dependent receiver gain artifacts must be removed from $\hat{S}_x[\omega_k]$. As a part of the decimation process in the 6216 digital receiver, the data is bandpass filtered to select the desired spectrum and to avoid aliasing. There are six digital filters, one for each of the programmable decimation rates: 2, 4, 8, 16, 32 and 64. In order to obtain the extremely low -80 dB sidelobes characteristic of these filters with a lower filter order, a ripple of approximately 0.13 dB exists in each passband. This ripple is very evident in the output spectrum at the scale needed to detect weak

signals buried in noise. The signals of interest may vary from the noise baseline by much less than 0.13 dB.

Because these anti-aliasing decimation bandpass filters are implemented in the digital domain, each filter's magnitude response is constant. Therefore, by computing these passband characteristics, the 0.13 dB ripple can be removed from each PSD estimate. The filter tap values were obtained by corresponding directly with Graychip, Inc [38].

As shown in Figure 3.1, in-band ripple is removed using a baseline correction vector, $\boldsymbol{\beta}$, by computing

$$\hat{S}_x[\omega_k] = \frac{\hat{S}'_x[\omega_k]}{\beta_k}, \tag{3.13}$$

where

$$\beta_k = \left\{ \begin{array}{cc} 1, & N_L \leq k \leq N_U \\ \left|H_D(e^{j\omega_k})\right|^2, & \text{otherwise} \end{array} \right\} \tag{3.14}$$

is the k^{th} element of $\boldsymbol{\beta}$. Here, $\left|H_D(e^{j\omega_k})\right|^2$ is the magnitude squared response of the filter $h_D[n]$ after decimation, as illustrated in Figure 3.4 for a decimation rate of $D = 64$. N_L and N_U are the FFT bin indices corresponding to the filter lower and upper passband edges. For example, with our $N = 1024$ PSD estimator, bins 148–880 are preserved from the normalized 1024 point $\left|H_D(e^{j\omega})\right|^2$ after decimation, while the others are set to one. The resulting baseline smoothing vectors are shown in Figure 3.5.

Thorough testing demonstrates drastic improvement to PSD estimates after baseline corrections by $\boldsymbol{\beta}$. Figure 3.6 illustrates three different PSD estimates without baseline smoothing. The location, magnitude and even existence of the hydrogen spectral lines are severely distorted in the PSD estimates of the (a) Cygnus and (b) Cassiopoeia hydrogen lines. After removing the ripple, the structure of all lines can be clearly seen (see Figure 3.7). A similar phenomenon would occur with all PSD estimates seen throughout this thesis in the absence of proper baseline smoothing.

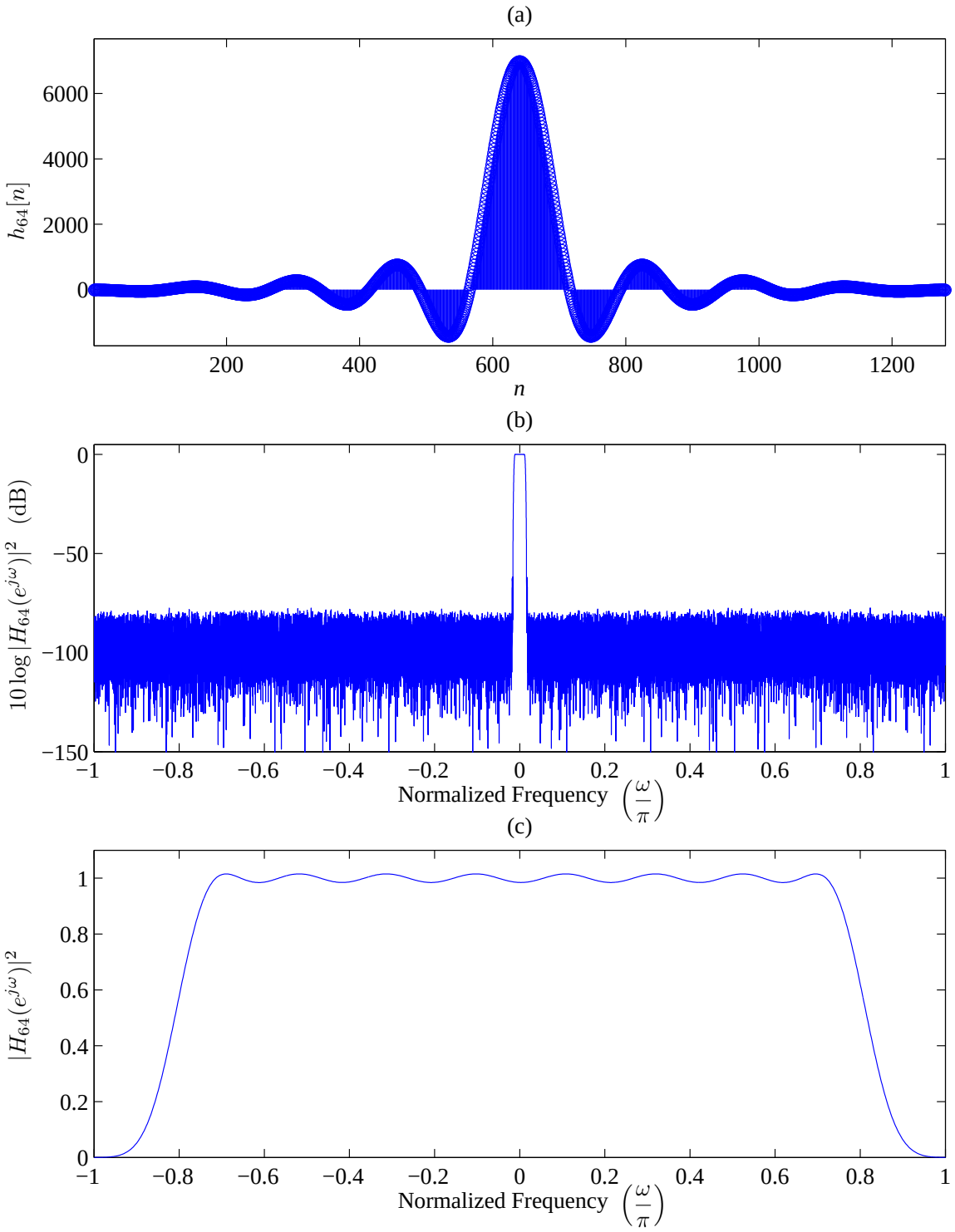


Figure 3.4: (a) 1280 tap decimate-by-64 anti-aliasing bandpass filter; (b) Normalized frequency response of $h_{64}[n]$ in dB, $10\log|H_{64}(e^{j\omega})|^2$; (c) Normalized frequency response of $h_{64}[n]$ after decimation by 64, $|H_{64}(e^{j\omega})|^2$. Without normalization, $h_{64}[n]$ has a passband gain of 114.3924 dB. Results for all six 6216 digital receiver bandpass filters are included in Appendix B.3.

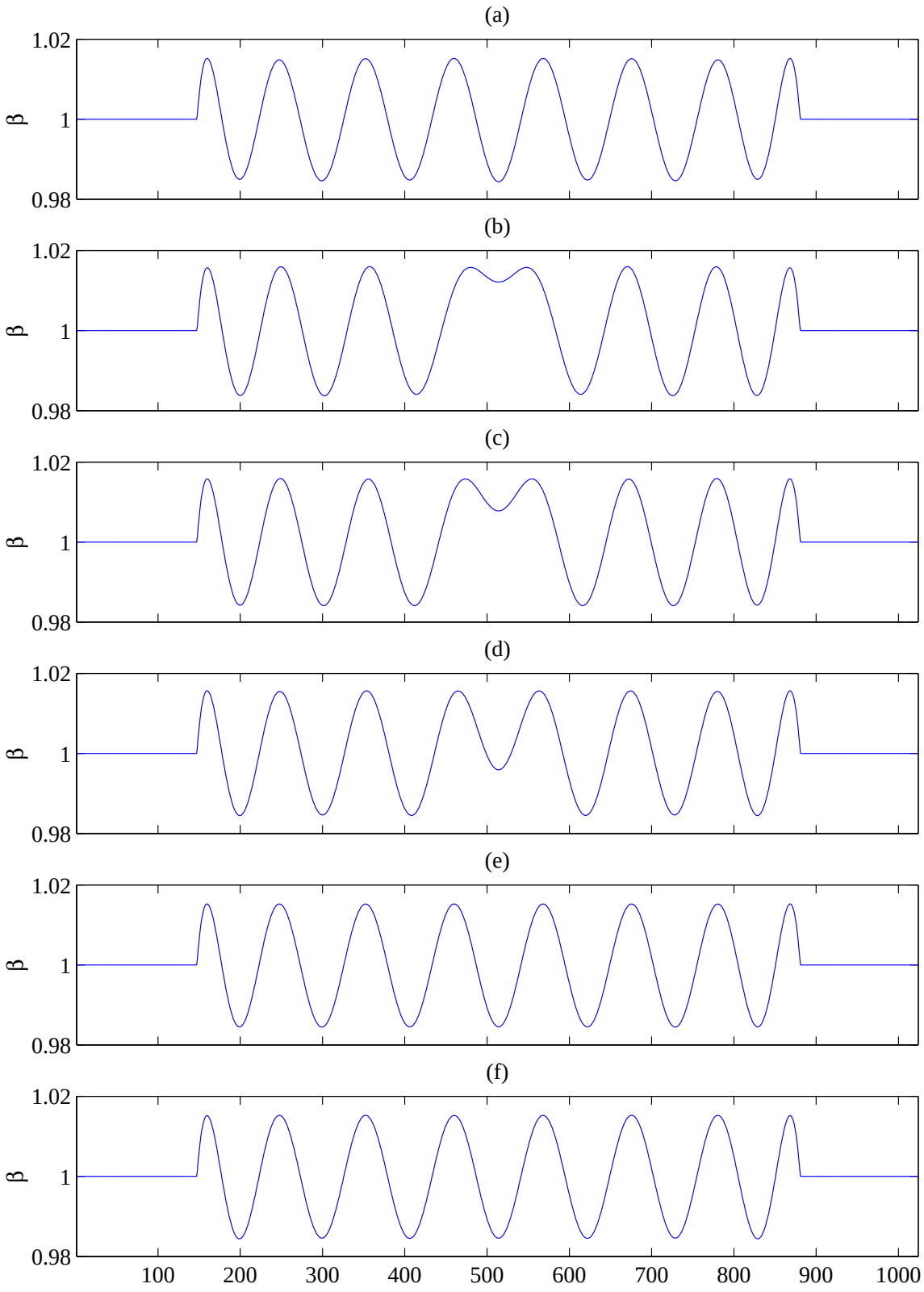


Figure 3.5: PSD baseline smoothing vectors ($\boldsymbol{\beta}$) for (a) $D = 2$, (b) $D = 4$, (c) $D = 8$, (d) $D = 16$, (e) $D = 32$, and (f) $D = 64$.

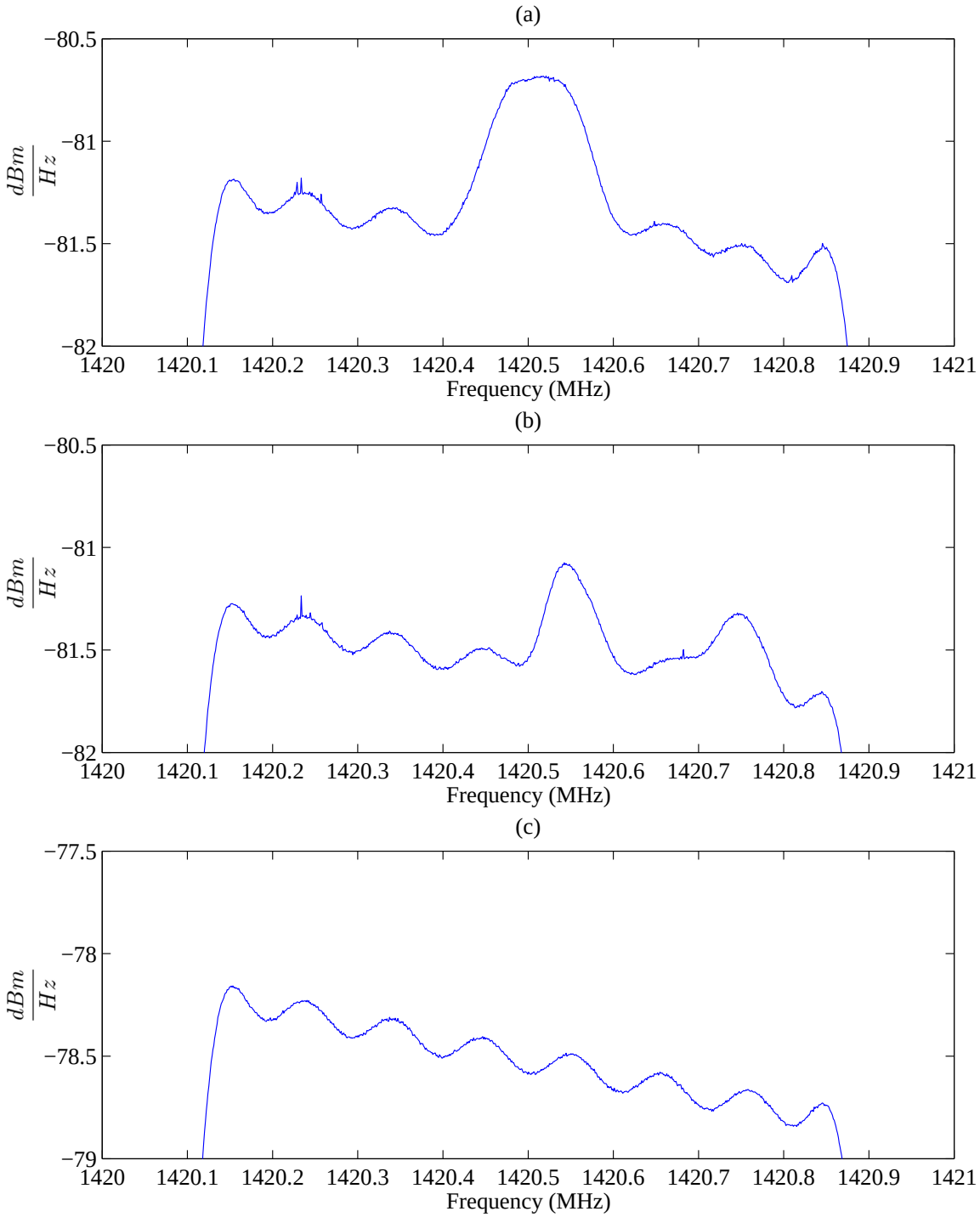


Figure 3.6: Figure 3.7 without baseline smoothing by $\boldsymbol{\beta}$. The spectrum is significantly distorted without removing the passband ripple.

3.1.4 Results

The DSP PSD estimator has proven to be an accurate and robust tool for many tests and applications. These include spectral line observations with both the VSA and Green Bank Telescope (GBT). The PSD estimator also gave invaluable insight when evaluating interference cancellation with the LMS adaptive filter. The spectrums of the primary channel, reference channel, and primary channel after filtering were all simultaneously computed (see Section 4.5).

Successful observations of hydrogen spectral lines have been made with the VSA. Figure 3.7 illustrates a couple of these observations. Spectrum (a) is a PSD estimate of the Cygnus hydrogen line with a white noise component, $\hat{S}_{x_1}[\omega_k] = \hat{S}_{Cyg}[\omega_k] + \hat{S}_{\eta_1}[\omega_k]$. Spectrum (b) is a PSD estimate of the Cassiopoeia hydrogen line with corresponding white noise component, $\hat{S}_{x_2}[\omega_k] = \hat{S}_{Cass}[\omega_k] + \hat{S}_{\eta_2}[\omega_k]$. Spectrum (c), $\hat{S}_{\eta_3}[\omega_k]$, is a separate noise estimate made by covering the aperture of the feed with RF absorber foam. Notice that the baseline of $\hat{S}_{\eta_3}[\omega_k]$ made with the absorber is approximately 3 dB higher than either $\hat{S}_{\eta_1}[\omega_k]$ or $\hat{S}_{\eta_2}[\omega_k]$ seen with $\hat{S}_{Cyg}[\omega_k]$ and $\hat{S}_{Cass}[\omega_k]$ in (a) and (b), respectively. This is due to the higher temperature of the absorber compared to the thermal sky noise. In all cases, we assume the noise is white and that spectral variations are due to our system response, which we are attempting to compensate for.

As indicated by Eq. 3.7, both $\hat{S}_{Cyg}[\omega_k]$ and $\hat{S}_{Cass}[\omega_k]$ can be estimated by subtracting a separate estimate of $S_\eta(\omega_k)$ from those spectrums displayed in Figure 3.7 (a) and (b). $\hat{S}_{\eta_3}[\omega_k]$, however, must be scaled by a constant, c_1 or c_2, before subtraction, i.e.

$$\hat{S}_{Cyg}[\omega_k] = \hat{S}_{x_1}[\omega_k] - c_1\hat{S}_{\eta_3}[\omega_k], \text{ and} \tag{3.15}$$

$$\hat{S}_{Cass}[\omega_k] = \hat{S}_{x_2}[\omega_k] - c_2\hat{S}_{\eta_3}[\omega_k]. \tag{3.16}$$

The scaling constants were estimated using the following least squares fit approach. For $p = \{1,\ 2\}$, let

$$\mathbf{x}_p \equiv \left\{\hat{S}_{x_p}[\omega_k]\ \forall\ k \in \mathbb{S}\right\}, \tag{3.17}$$

$$\mathbf{n} \equiv \left\{\hat{S}_{\eta_3}[\omega_k]\ \forall\ k \in \mathbb{S}\right\}, \tag{3.18}$$

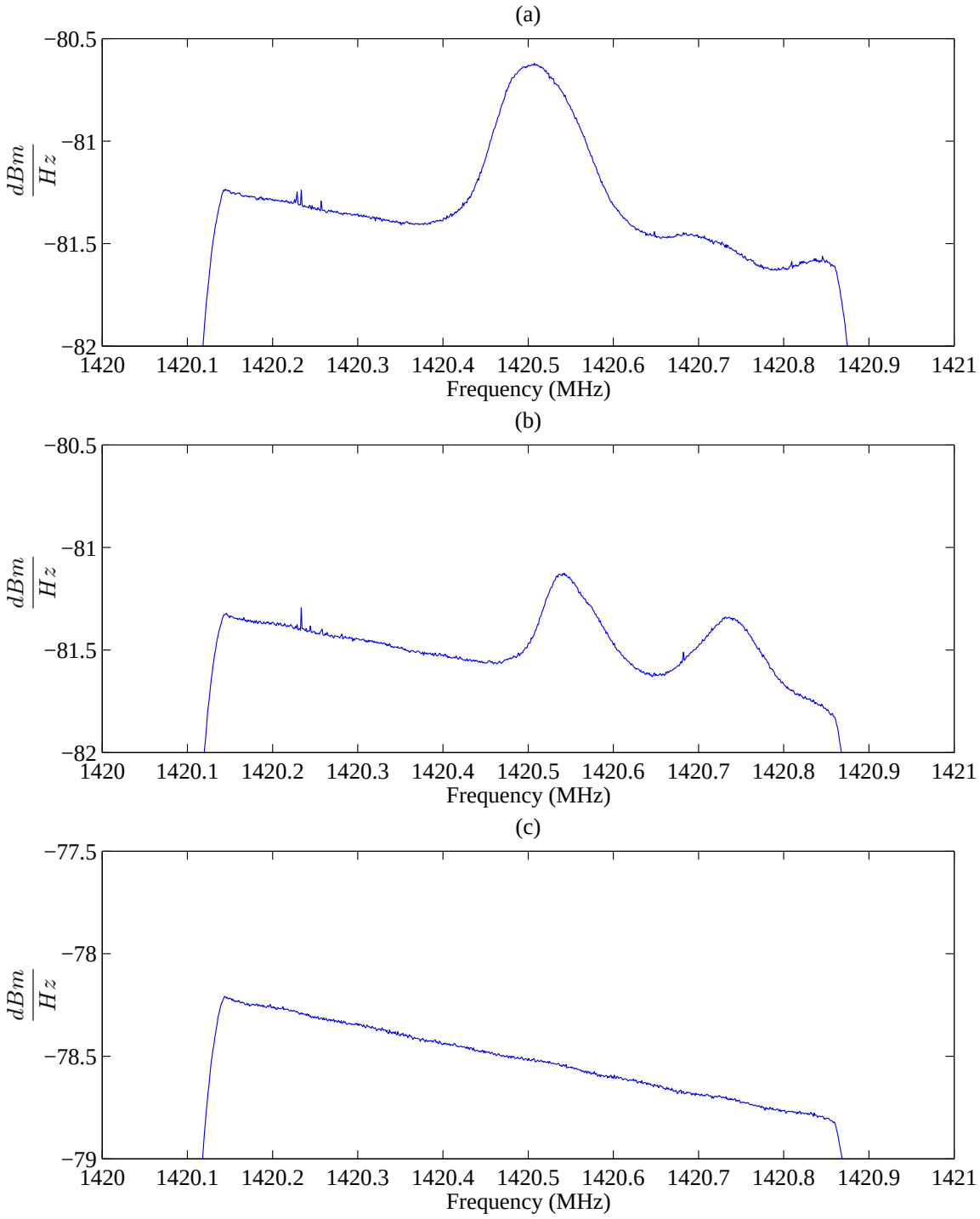


Figure 3.7: Hydrogen line detections using the VSA. (a) Cygnus: $\hat{S}_{x_1}[\omega_k] = \hat{S}_{Cyg}[\omega_k] + \hat{S}_{\eta_1}[\omega_k]$; (b) Cassiopoeia: $\hat{S}_{x_2}[\omega_k] = \hat{S}_{Cass}[\omega_k] + \hat{S}_{\eta_2}[\omega_k]$; (c) Estimate of $\hat{S}_{\eta_3}[\omega_k]$ made by blocking the aperture of the feed with RF absorber foam. Each PSD estimate was made with 20 minutes of integration and were taken consecutively.

$$\mathbf{e}_p \equiv \mathbf{x}_p - \mathbf{n}, \tag{3.19}$$

and $\mathbb{S}$ be the set of all frequency bin indices where $S_{x_p}[\omega_k]$ contains no known signal component. The least squares solution which minimizes $\mathbf{e}_p^{\mathrm{T}}\mathbf{e}_p$ is given by

$$c_p = \frac{\mathbf{n}^{\mathrm{T}}\mathbf{x}_p}{\mathbf{n}^{\mathrm{T}}\mathbf{n}}. \tag{3.20}$$

This method was used to obtain the spectra in Figure 3.8 [39].

Successful OH maser line observations were made with the GBT. Some of these are shown in Figure 3.9. The first two observations, (a) and (b), are both of VY CMa, one of the highest power OH sources. This is a prime candidate for OH detection with the VSA. This source required virtually no integration for detection with the GBT; it was 14 dB above the integrated noise floor. The other sources shown are relatively low power sources. IRAS #17560-2027 (f) is a highly red-shifted source (see Appendix C.3 and Section 4.7.5).

The real-time DSP PSD estimator was selected as an observational tool over a commercially available spectrum analyzer for many reasons. First, the BYU PSD estimator produces a higher quality PSD estimate of the extremely low power signals found in radio astronomy observations. It is capable of much longer and more stable integration than a typical spectrum analyzer (see Section 3.1.5). Often, commercially available analyzers use frequency scanning techniques to produce a spectral estimate. When using this technique, much integration time is lost because the analyzer sweeps across and integrates only one frequency bin at a time. The DSP system, however, simultaneously monitors each frequency bin with each periodogram, block DFT sample. Additionally, the PSD estimator is capable of baseline removal through an independent observation of $\hat{S}_\eta[\omega_k]$. Moreover, since the PSD estimator is implemented in the digital domain (DSP), it is capable of providing real-time analysis of interference mitigation algorithms.

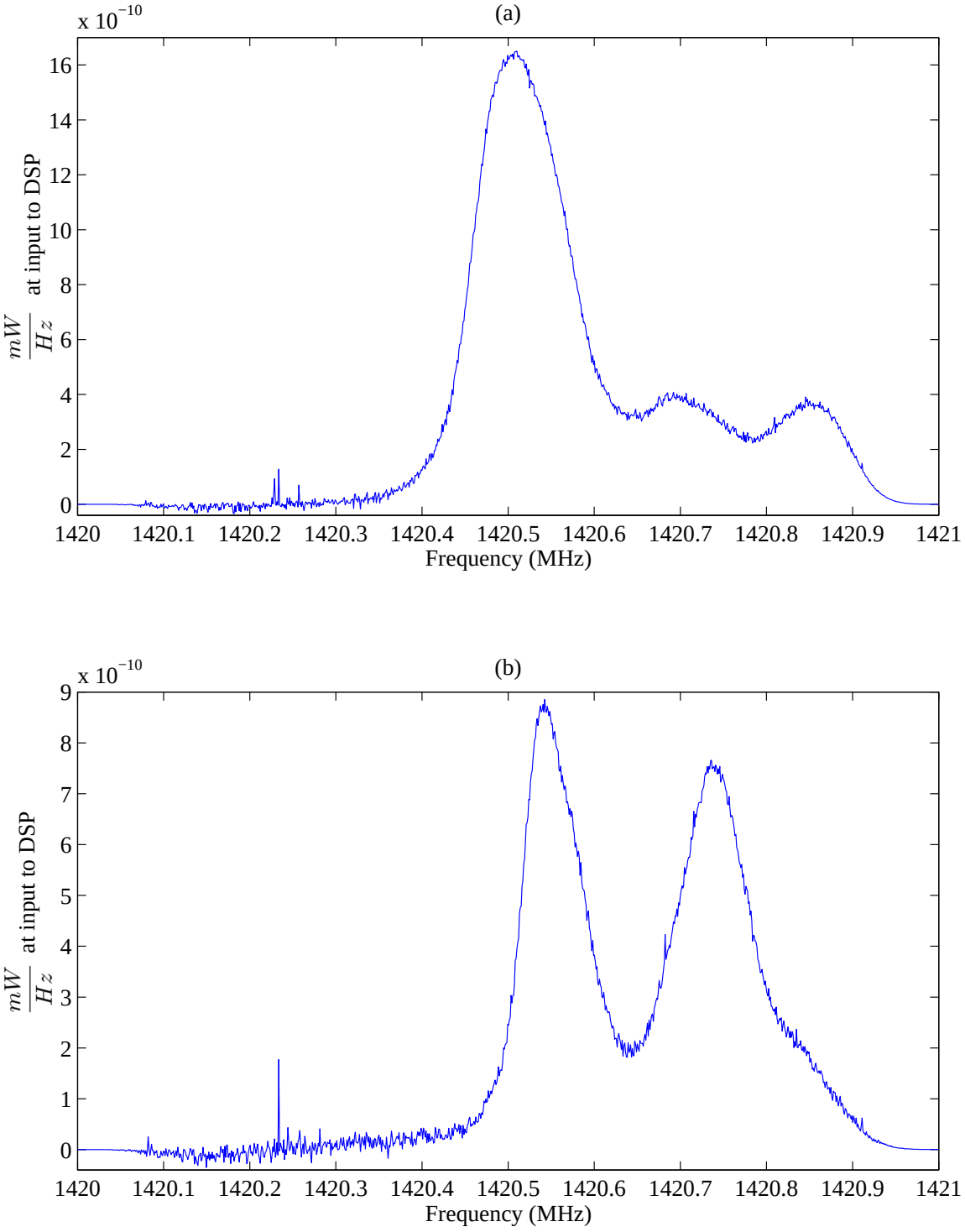


Figure 3.8: Hydrogen line detections using the VSA with $\hat{S}_\eta[\omega_k]$ subtracted off. (a) Cygnus: $\hat{S}_{Cyg}[\omega_k]$; (b) Cassiopoeia: $\hat{S}_{Cass}[\omega_k]$. These are the same observations shown in Figure 3.7.

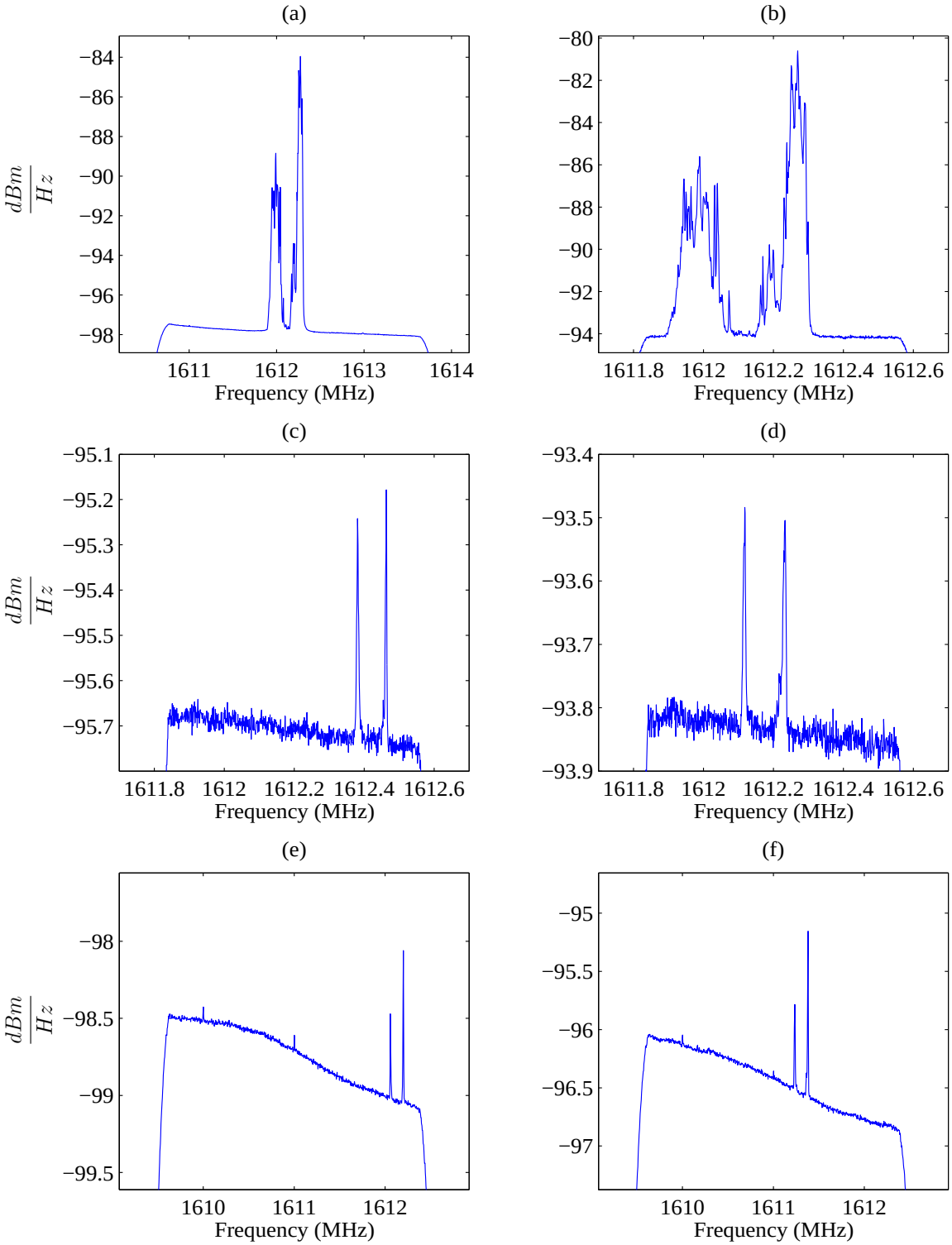


Figure 3.9: A few OH maser line observations made with the Green Bank Telescope. (a) VY CMa (4 MHz bandwidth), 220 sec.; (b) VY CMa (1 MHz bandwidth), 18 sec.; (c) Low power OH line, 116 sec.; (d) Low power OH line, 116 sec.; (e) IRAS #16260+3454, 58 sec.; (f) IRAS #17560-2027, 48 sec.

3.1.5 Estimation of Variance with Integration Time

A radio astronomer relies on integration in order to detect signals buried deep below the noise floor. Theoretically, the variance and standard deviation should decrease with integration, as described by Eqs. 3.8–3.11. In order to verify proper decline of variance for our instrument, it was necessary to obtain an estimate of the spectral variance as a function of integration time.

Consider a random variable X with corresponding pdf (probability density function) $f_X(x)$, mean μ, and variance σ^2. The unbiased, consistent estimator of the variance, $\hat{\sigma}^2$, is given by

$$\hat{\sigma}^2 = \frac{1}{k-1} \sum_{i=1}^{k} (X_i - \hat{\mu})^2 , \tag{3.21}$$

with k i.i.d. observations $X_1, \ldots, X_k$ of X using the sample mean estimator [40]

$$\hat{\mu} = \frac{1}{k} \sum_{i=1}^{k} X_i. \tag{3.22}$$

In the case of the PSD estimate $\hat{S}_x[\omega_k]$ of a random process $x[n]$, the mean will most likely be frequency dependent (unless $x[n]$ is spectrally white), written as $\mu[\omega_k]$. If the portion of $\hat{S}_x[\omega_k]$ used to estimate $\hat{\sigma}^2$ is relatively white, we can assume that $\mu[\omega_k]$ will contain only low frequency characteristics from bin to bin, not the high frequency structure introduced by the sample variance of $\hat{S}_x[\omega_k]$. By taking advantage of the correlation between adjacent bins, we can obtain a more accurate estimate $\hat{\mu}[\omega_k]$. This is accomplished by first computing the sample mean of each frequency bin, which is $\hat{S}_x[\omega_k]$ after integration. Next, the spectrum can be filtered with the low pass FIR filter $h[n]$ along frequency bins to reduce the high frequency deviation due to the residual variance–effectively smoothing out $\hat{\mu}[\omega_k]$ with the filter,

$$\hat{\mu}[\omega_k] = \hat{S}_x[\omega_k] * h[k], \tag{3.23}$$

where $*$ is the convolution operator. It was determined experimentally that a 144 tap FIR filter developed with the Parks-McClellan algorithm gave fine results, as illustrated in Figure 3.10.

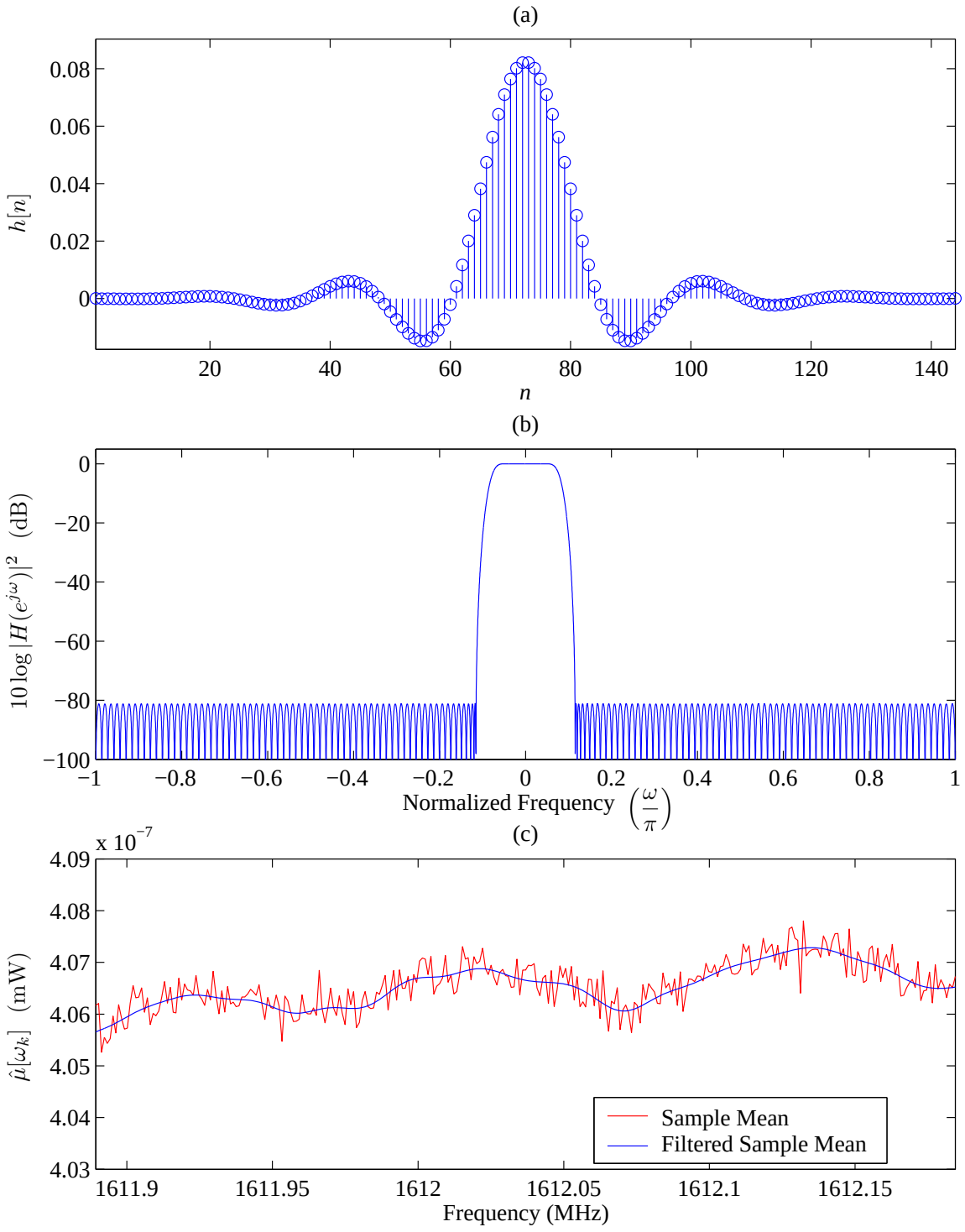


Figure 3.10: Estimation of the PSD mean. (a) $h[n]$, the 144 tap FIR filter used to remove the high frequency variance from $\hat{\mu}[\omega_k]$; (b) Frequency response of $h[n]$; (c) Sample mean and filtered sample mean for 301 frequency bins from over 40 minutes of integrated, interference-free GBT data. These 301 bins were used to create Figure 3.11.

Any receiver system will have at least a small degree of gain drift with time. This is usually not a problem for PSD estimation as long as the gain drift is not frequency dependent. If unaccounted for, however, it can still introduce an undesired bias when estimating $\hat{\sigma}^2$. A slight gain variation can be removed by scaling the periodograms averaged to create $\hat{S}_x[\omega_k]$ before subtracting $\hat{\mu}[\omega_k]$ to estimate $\hat{\sigma}^2$. If there is a frequency dependent gain drift, then the non-stationary mean $\hat{\mu}_n[\omega_k]$ can be re-estimated at every stage of the integration:

$$\hat{\mu}_n[\omega_k] = \hat{S}_{x,n}[\omega_k] * h[k], \tag{3.24}$$

where $\hat{S}_{x,n}[\omega_k]$ represents the PSD estimate up to integration time n. In order to have a good PSD estimate, however, the system should be as stationary as possible. It is desirable to observe a proper decline in $\hat{\sigma}^2$ with integration, without needing to recompute $\hat{\mu}[\omega_k]$.

An interference-free 40 minute reference data set was collected using the GBT and corresponding receiver system. The standard deviation as a function of integration time for this clean data is found in Figure 3.11 and succeeding plots demonstrating decline of variance with integration time. Figure 3.11 highlights $\hat{\sigma}$, estimated both with a constant mean and time varying mean. As expected, the estimates made with a time varying mean are generally slightly lower than those made with the constant mean. Fortunately, both estimates closely follow the theoretical slope. This suggests that in this case, $x[n]$ does have a stationary mean. Unless stated otherwise, a constant mean is used for all plots illustrating decline of $\hat{\sigma}^2$ with integration time.

The results shown in Figure 3.11 demonstrate the *linearity* and *stability* of the DSP real-time PSD estimator over long integration. These qualities are essential ingredients of a high quality radio astronomy observational instrument. If the 6216 digital receivers were not stable with long integration, then the variance of this PSD estimate would not decrease as predicted by theory.

Unfortunately, the BYU VSA RF receiver is not as stable as the GBT receiver. Figure 3.12 (a) displays an estimate of the standard deviation of the PSD estimate of the output of a VSA receiver through 8 hours of integration. RF absorber foam

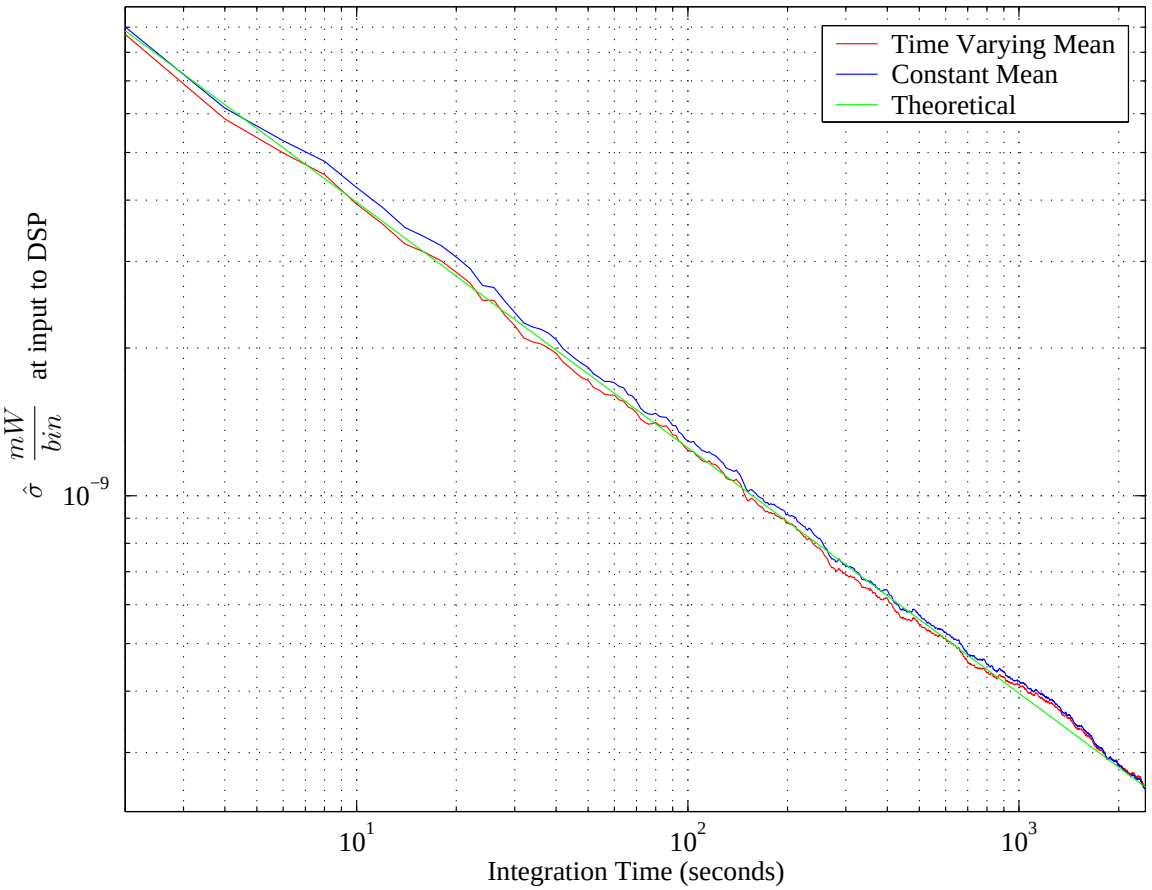


Figure 3.11: Decline of standard deviation with integration time for 40 minutes of interference-free data collected with the Green Bank Telescope. Notice that $\hat{\sigma}$ converged as predicted by theory, whether or not I assumed a stationary mean ($\hat{\sigma}$ was estimated with disjoint 2 second integration PSD samples). The GBT receiver and DSP platform are very stable systems.

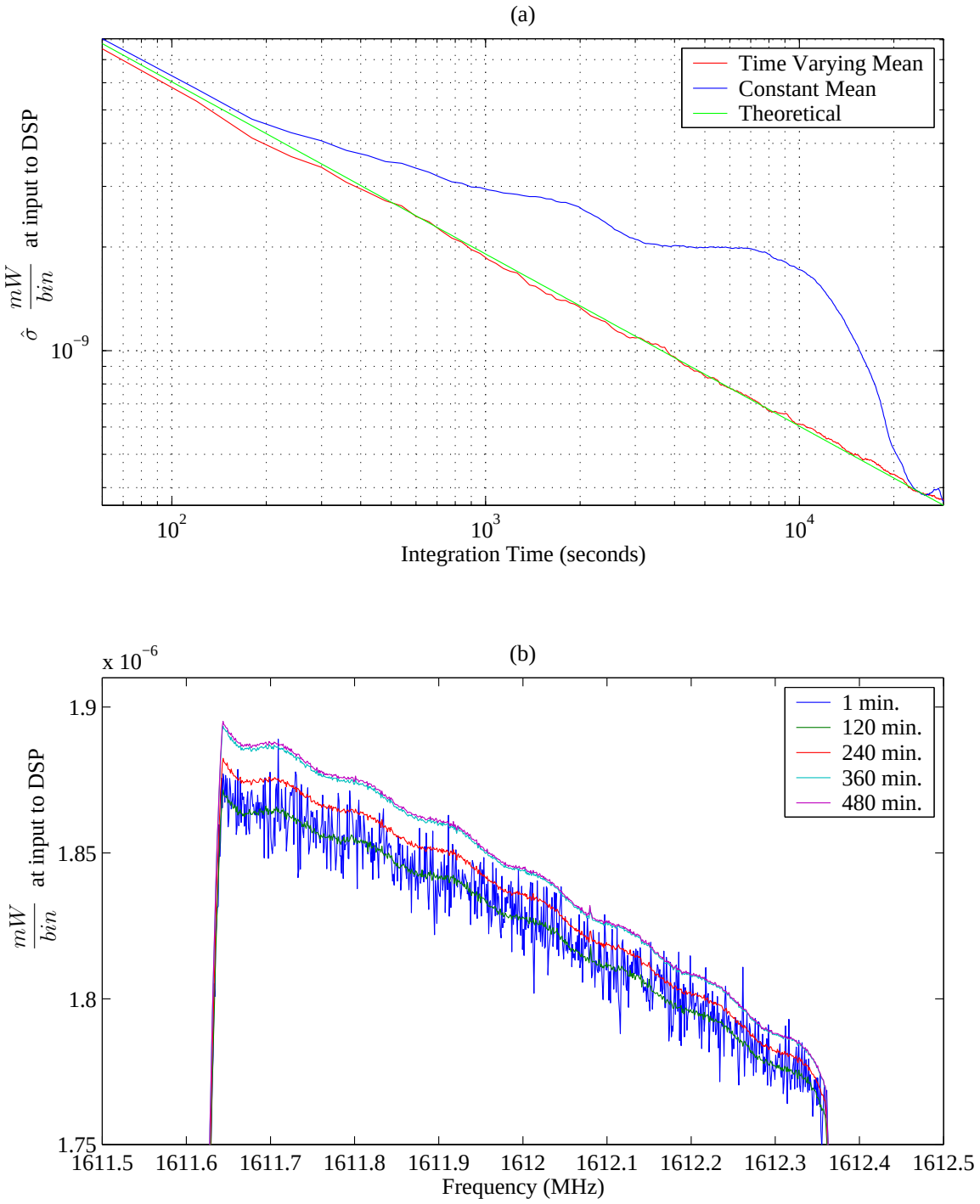


Figure 3.12: (a) Decline of standard deviation with 8 hours of integration using one VSA RF receiver ($\hat{\sigma}$ was estimated with disjoint 60 second integration PSD samples). RF absorber foam covered the aperture of the feed. (b) Snapshots of the PSD estimate every two hours. The gain of the receiver rose more quickly at the lower frequencies, causing the non-stationary mean. The standard deviation was estimated using 350 frequency bins from 1611.69–1612.04 MHz.

was placed over the feed aperture to produce a stable input signal. The mean for the signal at the output of the receiver is non-stationary; the frequency dependent gain drift is illustrated in Figure 3.12 (b).

3.2 Complex Beamformer/Correlation Estimator

Two important tools for sensor array processing, and thus radio astronomy, include beamforming and correlation estimation. This section presents a short background for each of these real-time tools, along with details of a DSP application integrating both beamforming and correlation estimation.

3.2.1 Complex Beamformer

In the simplest terms, a beamformer linearly combines separate observations of a signal $x[n]$ which have been spatially sampled by an array of antennas. Beamforming performs spatial filtering to direct a high gain response toward a desired signal, and to attenuate interferers in other directions. This is helpful in cases where two signals occupy the same frequency band, but originate from different directions.

Although beamforming can be applied to both narrow and broadband signals, this brief background will focus primarily on the beamforming of narrowband signals. Please refer to other sources for an extensive introduction to beamforming and its applications [41, 42]. As suggested by Figure 3.13, a beamformer performs the following computation:

$$y[n] = \mathbf{w}^{\mathrm{H}}\mathbf{x}[n], \tag{3.25}$$

where $\mathbf{w} = \begin{bmatrix} w_1 & w_2 & \cdots & w_\rho \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{x}[n] = \begin{bmatrix} x_1[n] & x_2[n] & \cdots & x_\rho[n] \end{bmatrix}^{\mathrm{T}}$ are vectors containing respectively the beamforming weights and spatially sampled signals from ρ sensor elements (antennas). Each complex weight w_i is capable of introducing both a change in magnitude and phase to its corresponding $x_i[n]$.

When determining the appropriate weight vector, $\mathbf{w}$, for use in a beamformer, it is important to understand the differences in gain and phase response between each element in the array for a given signal DOA, θ, and frequency, ω. This information is contained in the array response vector, $\mathbf{a}(\theta, \omega)$, which for the case of identical element

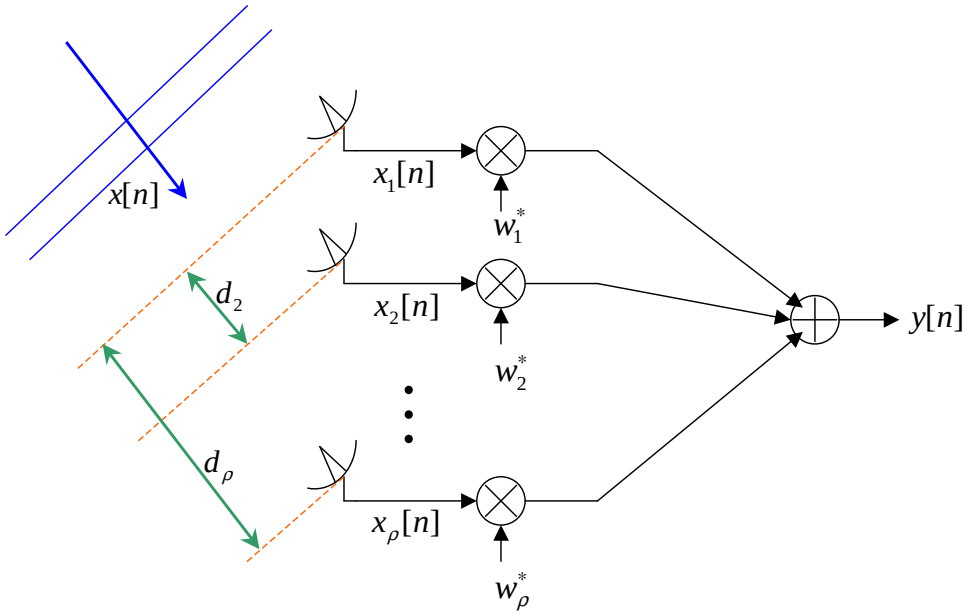


Figure 3.13: A beamformer linearly combines separate instances of a signal $x[n]$ arriving at ρ array sensors. The incident plane wave propagates different relative distances between array elements, introducing phase differences between $x_1[n],\ x_2[n], \ldots,\ x_\rho[n]$.

responses is given by

$$\mathbf{a}(\theta, \omega) = r(\theta, \omega)\mathbf{d}(\theta, \omega), \tag{3.26}$$

where $r(\theta, \omega)$ is the complex response of each (identical) sensor element, and $\mathbf{d}(\theta, \omega)$ is the steering vector which expresses relative phase information. Dropping the notational dependence on θ and ω, the steering vector is given by

$$\mathbf{d} = \begin{bmatrix} 1 & e^{-j\omega t_2} & \cdots & e^{-j\omega t_\rho} \end{bmatrix}^{\mathrm{T}}, \tag{3.27}$$

where t_i is the relative wave propagation delay between the reference element and the i^{th} element. This relative wave propagation delay is $t_i = \frac{d_i}{c}$, where d_i is the propagation distance, dependent on θ, between the two antennas and c is the speed of light.

Using the sensor geometry, it can be shown that

$$d_i = r_{xi}\cos\theta_{el}\sin\theta_{az} + r_{yi}\cos\theta_{el}\cos\theta_{az} + r_{zi}\sin\theta_{el} = \mathbf{r}_i{}^{\mathrm{T}}\mathbf{s}, \tag{3.28}$$

where $\mathbf{r}_i = \begin{bmatrix} r_{xi} & r_{yi} & r_{zi} \end{bmatrix}^{\mathrm{T}}$ is the 3-dimensional location of the i^{th} antenna, with the reference antenna at the origin. I adopt a coordinate system with the $+x$ direction being due east, the $+y$ direction due north, and the +z direction corresponding to positive elevation. θ_{az} and θ_{el} are the azimuth and elevation coordinates defining the signal DOA, with $\theta_{az} = 0°$ defined to be north, increasing clockwise, and $\theta_{el} = 90°$ as zenith. The unit vector, $\mathbf{s} = \begin{bmatrix} \cos\theta_{el}\sin\theta_{az} & \cos\theta_{el}\cos\theta_{az} & \sin\theta_{el} \end{bmatrix}^{\mathrm{T}}$, points to the DOA. After combining Eqs. 3.27 and 3.28, it is evident that the steering vector is a function of the signal DOA/frequency and array geometry,

$$\mathbf{d} = \begin{bmatrix} 1 & e^{-\frac{j\omega \mathbf{r_2}^{\mathrm{T}}\mathbf{s}}{c}} & \cdots & e^{-\frac{j\omega \mathbf{r}_\rho{}^{\mathrm{T}}\mathbf{s}}{c}} \end{bmatrix}^{\mathrm{T}}. \tag{3.29}$$

In many environments, however, the desired signal is coincident in frequency with an interferer. Array beamforming is capable of simultaneously steering a beam in the direction of the desired signal and a null to the interferer. In a stationary scenario with a fixed DOA for the desired and interfering signals, a fixed $\mathbf{w}$ can be very effective. Often, however, the desired signal and/or interferer is moving, requiring an adapting $\mathbf{w}$. For example, radio astronomers often encounter interference from a non-stationary satellite while tracking a desired signal which is also moving with respect to the observer. At times, it may also be necessary to steer multiple nulls to multiple stationary and/or moving interferers.

A large number of adaptive beamforming algorithms have been developed for a wide range of environments. A few of these algorithms include the multiple sidelobe canceller (MSC), linearly constrained minimum variance (LCMV), recursive least squares (RLS), maximum signal to noise ratio (Max SNR), and others [41, 42].

3.2.2 Correlation Estimation

An important statistical measure for both radio astronomy imaging arrays and interference mitigation is the array autocorrelation matrix

$$\mathbf{R}_{xx}[n] = E\left\{\mathbf{x}[n]\mathbf{x}[n]^{\mathrm{H}}\right\}. \tag{3.30}$$

As before, the incident signal $x[n]$ is spatially sampled by ρ array elements such that $\mathbf{x}[n] = \begin{bmatrix} x_1[n] & x_2[n] & \cdots & x_\rho[n] \end{bmatrix}^{\mathrm{T}}$.

Typically, $\mathbf{R}_{xx}[n]$ is unknown, requiring an estimate. If $x[n]$ is approximately ergodic and relatively stationary over a time interval N, then the sample estimate may be formed as

$$\hat{\mathbf{R}}_{xx}[n] = \frac{1}{N} \sum_{k=0}^{N-1} \mathbf{x}[n+k]\mathbf{x}[n+k]^{\mathrm{H}}. \tag{3.31}$$

$\hat{\mathbf{R}}_{xx}[n]$ is useful for radio astronomy synthesis imaging, computation of adaptive beamforming weights, array auto-calibration, and other applications.

3.2.3 DSP Implementation

A three channel beamformer and correlation estimator was implemented in the DSP. Despite being a straightforward calculation, coding presented significant challenges. These included the interface to the digital receivers, inter-processor communication, proper data alignment, and code optimization.

Figure 4.5 illustrates the general structure of the DSP beamforming/correlation estimation application. The inputs $x_1[n]$, $x_2[n]$ and $x_3[n]$ are sampled by the 6216 digital receivers and sent to processors C, A and B, respectively. This process not only involves an A/D convertor, but also conversion to a complex baseband representation, band selection, signal decimation, and bandpass filtering to achieve the desired complex sampling rate.

The signals are converted to floating point in preparation for beamforming. Floating point computations enable higher degrees of cancellation than fixed point when steering nulls with adaptive weights. The beamforming calculations are divided among each processor. One complex multiply (four real multiplies) is required for each sample of $x_1[n]$, $x_2[n]$ and $x_3[n]$. As shown in the diagram, processor C applies w_1 to $x_1[n]$, then passes the output to processor A, who in turn applies w_2 to $x_2[n]$ and adds its output to the output from processor C. This process continues in processor B who sends the beamformer output to processor D. An additional beamforming leg could easily be added to the fourth channel. When this application was first developed, the VSA consisted of only three antennas. Soon a fourth antenna will be added, requiring a four channel beamformer/correlator.

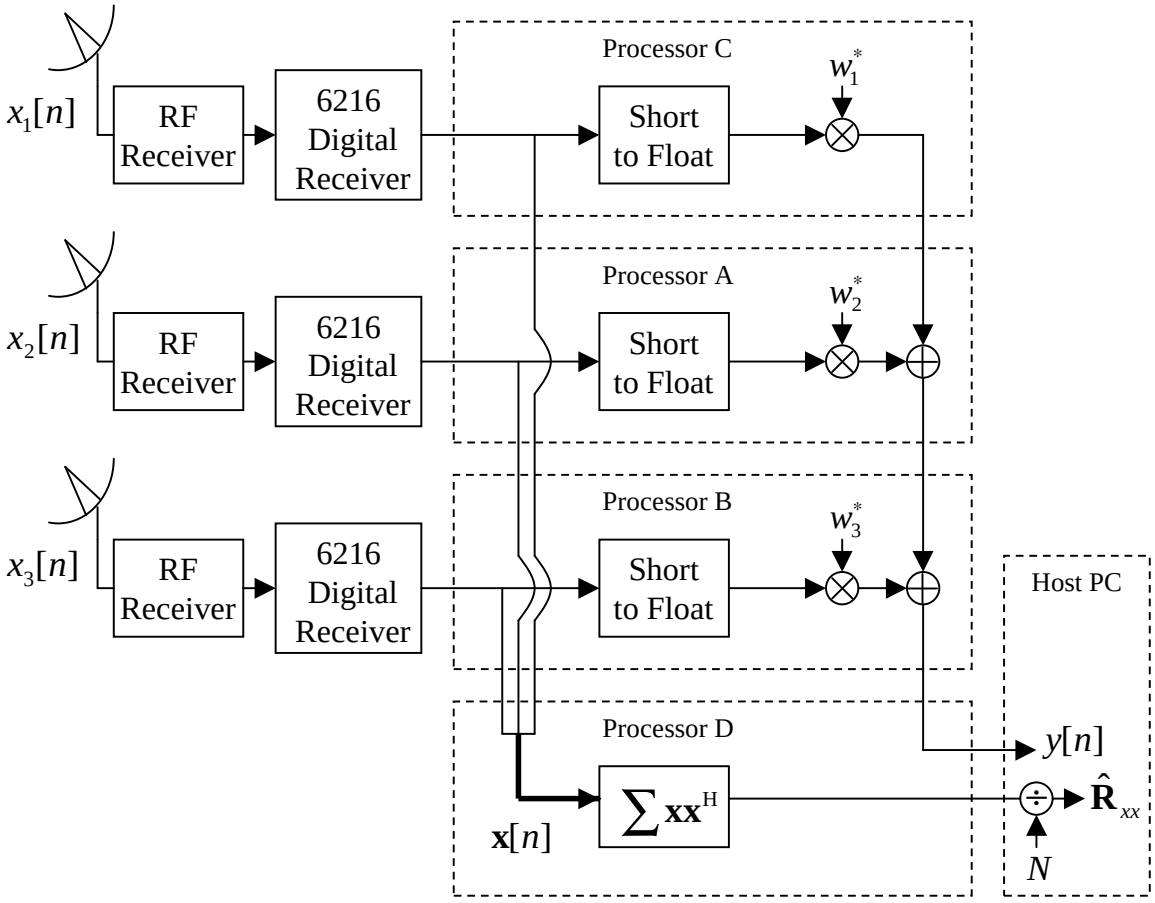


Figure 3.14: DSP beamformer and correlation estimator

Along with beamforming data, the unprocessed inputs $x_1[n]$, $x_2[n]$ and $x_3[n]$ in fixed point format are also cascaded between processors. One processor must simultaneously have access to all array signals in order to calculate $\hat{\mathbf{R}}_{xx}$. As shown in the diagram, this calculation occurs in processor D. The output is accessed on the host PC, where normalization by N occurs.

With a very limited data transfer rate from DSP to host PC, $y[n]$ cannot be streamed to the PC in its entirety. Therefore, only bursts of contiguous samples can be sent. Since $\mathbf{R}_{xx}[n]$ is computed using integration over time, it is possible to perform full real-time transfer to the host PC.

Currently, the complex beamforming weights are constants, set when the code is compiled. Later, both the steering vector $\mathbf{d}$ and $\hat{\mathbf{R}}_{xx}$ will be used to calculate adaptive beamforming weights. This code could also be used and adapted for radio astronomy synthesis imaging.

3.2.4 Results

The beamforming/correlation estimator DSP application was our first attempt at real-time array processing. Similarly, it was also the first to utilize inter-processor communication, which alone proved to be a difficult hurdle. Additionally, each sample of $y[n]$ must be a linear combination of *synchronous* samples of $x_1[n]$, $x_2[n]$ and $x_3[n]$. Needless to say, inter-processor communication is not an instantaneous transfer of information. Furthermore, the three channels need to arrive simultaneously to processor D for proper $\hat{\mathbf{R}}_{xx}[n]$ calculation. The solution required the addition of varying delays in each digital receiver, giving each processor access to synchronous samples from all necessary signals.

It is imperative that unwanted relative phase and bulk time delays not be introduced by the DSP platform. In order to verify synchronous sampling of $x_1[n]$, $x_2[n]$ and $x_3[n]$ along with proper inter-processor communication, the beamforming weights were all set to unity ($\mathbf{w} = \mathbf{1}$). The resulting calculation is a direct sum of the three channels:

$$y[n] = x_1[n] + x_2[n] + x_3[n]. \tag{3.32}$$

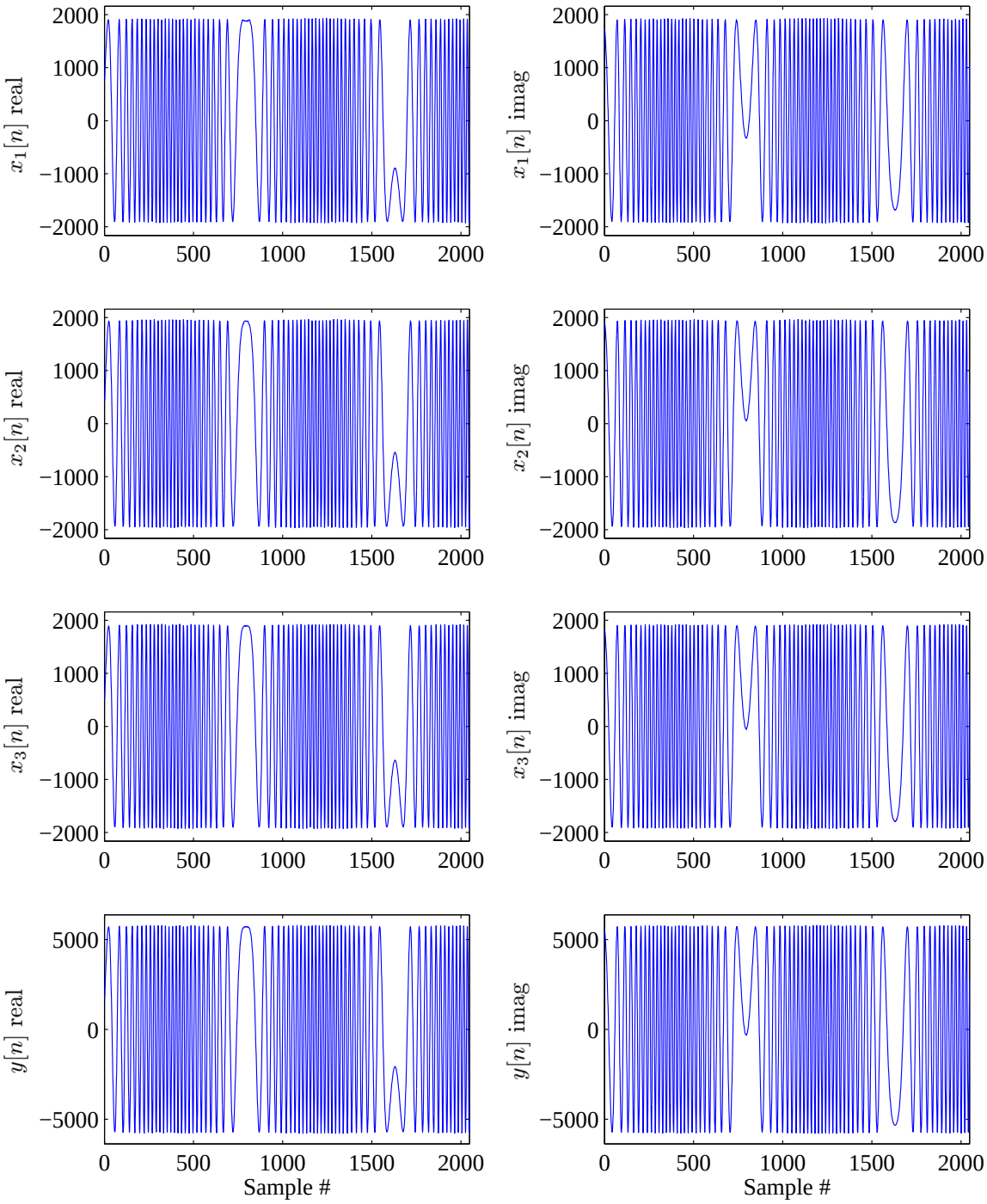


Figure 3.15: Example results of the DSP beamformer and correlation estimator. In this case, $x[n]$ is an FM sweep signal centered at 16 MHz, then basebanded and decimated in the digital receiver for a final sample rate of 2 MHz ($x[n] \approx x_1[n] \approx x_2[n] \approx x_3[n]$). With $\mathbf{w} = \mathbf{1}$, the output, $y[n] \approx 3x[n]$, demonstrates proper synchronous sampling of $x_1[n]$, $x_2[n]$ and $x_3[n]$.

A signal generator was split three ways and sent directly into the 6216 digital receivers, i.e. $x[n] \approx x_1[n] \approx x_2[n] \approx x_3[n]$. If timing and synchronous sampling issues were properly resolved, then even if $x[n]$ were a broadband signal, $y[n] = 3x[n]$ (neglecting very small gain and phase differences in each channel due to the power splitter).

Figure 3.15 illustrates the results of such an experiment. An FM sweep signal was chosen to demonstrate proper performance at both low and high frequencies. Notice that the output $y[n]$ is approximately three times the amplitude of any one of $x_1[n]$, $x_2[n]$ and $x_3[n]$, independent of the frequency. The autocorrelation matrix,

$$\hat{\mathbf{R}}_{xx} = \begin{bmatrix} 3.663 & 3.623 - j0.528 & 3.651 - j0.758 \\ 3.623 + j0.528 & 3.797 & 3.720 + j0.223 \\ 3.651 + j0.758 & 3.720 - j0.223 & 3.659 \end{bmatrix} \times 10^6, \tag{3.33}$$

was simultaneously estimated by the DSP with $N = 199680$ samples from each channel. The diagonal elements of $\hat{\mathbf{R}}_{xx}$ are the estimated powers of $x_1[n]$, $x_2[n]$ and $x_3[n]$. Therefore, if each signal is synchronously sampled with $\mathbf{w} = \mathbf{1}$, then

$$|y[n]| \approx \text{trace}\left\{\sqrt{\hat{\mathbf{R}}_{xx}}\right\} = 5775. \tag{3.34}$$

By using only the 2048 samples of $y[n]$ shown in Figure 3.15, $|y[n]|$ was estimated to be 5754, which is very close to the expected amplitude provided by Eq. 3.34.

As of this writing, the VSA has not been properly phase calibrated for extensive beamforming/correlation estimation experiments. Hence, the only tests performed have been with a signal generator as the input.

Chapter 4

Real-time RFI Cancellation with an LMS Adaptive Filter

4.1 Background

One potentially useful approach to real-time RFI mitigation is the least-mean-square (LMS) algorithm. This algorithm, first proposed by Widrow and Hoff [26], has become the most commonly used adaptive filtering method due to its simplicity, robustness, and good performance in a wide range of applications.

The LMS adaptive filter is an approximation to a steepest descent minimization of the mean-square error

$$\xi[n] = E\{|e[n]|^2\}, \tag{4.1}$$

where the error sequence is defined to be

$$e[n] = \alpha[n] - y[n], \tag{4.2}$$

and

$$y[n] = \mathbf{h}_n^{\mathrm{T}} \mathbf{x}_n. \tag{4.3}$$

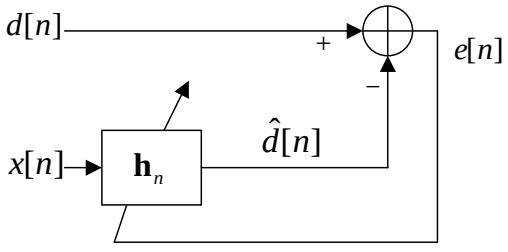


Figure 4.1: Illustration of a "generic" setup of the steepest descent and LMS adaptive filters

The steepest descent update for an FIR filter weight vector, $\mathbf{h}_n$, is given by (see Figure 4.1)

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \mu E\{e[n]\mathbf{x}_n^*\}, \tag{4.4}$$

where μ is the filter step size, p is the filter order, $x[n]$ is a data sequence, and the data vector $\mathbf{x}_n$ is defined as

$$\mathbf{x}_n = \begin{bmatrix} x[n] \\ x[n-1] \\ \vdots \\ x[n-p+1] \end{bmatrix}. \tag{4.5}$$

Typically the expectation $E\{e[n]\mathbf{x}_n^*\}$ is unknown and must be estimated. An unbiased and commonly used estimate for the expectation operator is the sample mean,

$$\widehat{E\{e[n]\mathbf{x}_n^*\}} = \frac{1}{L}\sum_{l=0}^{L-1} e[n-l]\mathbf{x}_{n-l}^*. \tag{4.6}$$

When substituted into Eq. 4.4, we obtain the update equation

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \frac{\mu}{L}\sum_{l=0}^{L-1} e[n-l]\mathbf{x}_{n-l}^*. \tag{4.7}$$

As a special case, the one-point sample mean ($L = 1$)

$$\widehat{E\{e[n]\mathbf{x}_n^*\}} = e[n]\mathbf{x}_n^* \tag{4.8}$$

leads to the LMS weight vector update equation [37]

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \mu e[n]\mathbf{x}_n^*. \tag{4.9}$$

Since the LMS filter is an approximation to the steepest descent optimization algorithm, it also minimizes the error signal, $e[n]$, in the mean-square sense.

The LMS adaptive filter can be used in a variety of applications, but methods of implementation will vary. The weight vector update defined in Eq. 4.9 is generally applicable, but forming the error sequence, $e[n]$, and desired sequence, $\alpha[n]$, often requires creativity. In many textbooks, the LMS algorithm is introduced as shown in Figure 4.1. The input signal, $x[n]$, is filtered by $\mathbf{h}_n$ to create the output estimate, $y[n]$,

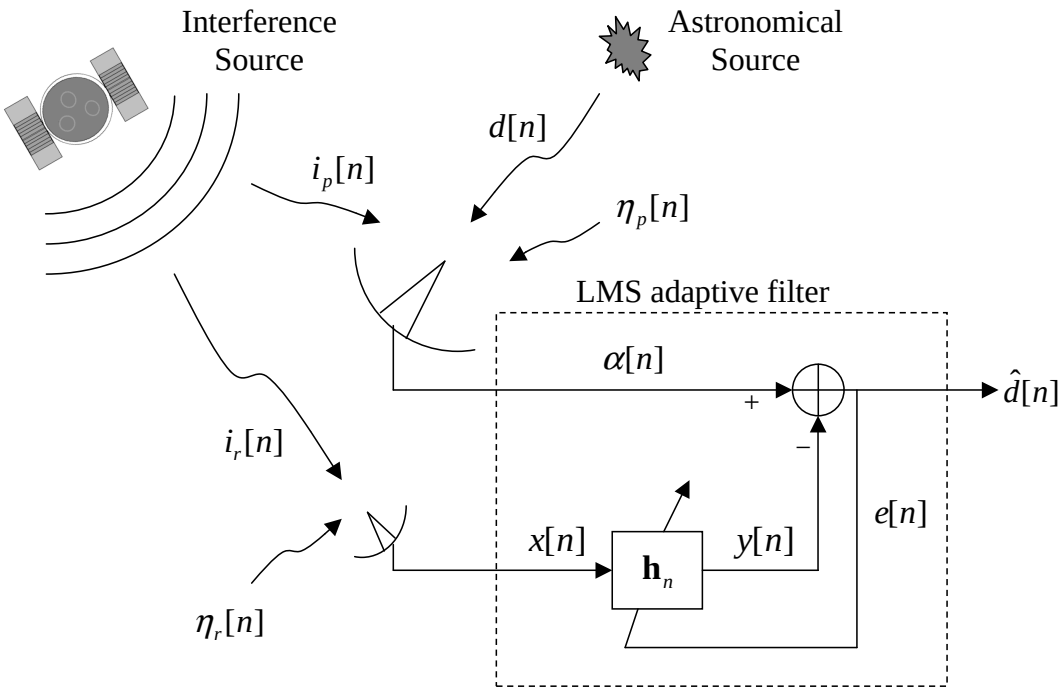


Figure 4.2: LMS adaptive filter used for radio astronomy RFI mitigation

of the desired sequence, $\alpha[n]$. At convergence, $y[n] \approx \alpha[n]$. The error sequence, $e[n]$, is used as a control to update the weight vector, $\mathbf{h}_n$, as described in Eq. 4.9. Application of the LMS adaptive filter to interference mitigation in the radio astronomy regime is described in the next section.

4.2 Radio Astronomy Scenario

The implementation of the LMS filter in the radio astronomy interference regime is shown in Figure 4.2. The radio telescope (primary antenna) not only receives the desired signal, $d[n]$, from an astronomical source through the antenna main beam, but also the interfering signal, $i_p[n]$, seen through the antenna response side lobes. There will also always be a white noise component, $\eta_p[n]$, due to cosmic background noise, thermal receiver noise, atmospheric noise, and ground noise from antenna spill-over. The net signal received at the feed of the telescope, $\alpha[n]$, is the sum of these

Table 4.1: Summary of the LMS adaptive filter.

Inputs:	$\alpha[n] = d[n] + i_p[n] + \eta_p[n]$: signal received by radio telescope
	$d[n]$: desired astronomical source
	$i_p[n]$: interference at primary channel
	$\eta_p[n]$: white noise at primary channel
	$x[n] = i_r[n] + \eta_r[n]$: signal received by reference antenna
	$i_r[n]$: interference at reference channel
	$\eta_r[n]$: white noise at reference channel
Output:	$\hat{d}[n]$: estimate of desired signal
Parameters:	μ: filter step size
	p: filter order
	$\mathbf{h}_n$: filter weight vector
	$\mathbf{h}_{n+1}$: filter weight vector update
Internal signals:	$e[n]$: error sequence
	$y[n] = \hat{i}_p[n] + \eta_h[n]$
	$\hat{i}_p[n]$: estimate of $i_p[n]$
	$\eta_h[n]$: $\eta_r[n]$ shaped by $\mathbf{h}_n$
Initialization:	$\mathbf{h}_0 = \mathbf{0}$
Computation:	$y[n] = \mathbf{h}_n^{\mathrm{T}}\mathbf{x}_n$
	$\hat{d}[n] = e[n] = \alpha[n] - y[n] = d[n] + (i_p[n] - \hat{i}_p[n]) + \eta_p[n] - \eta_h[n]$
	$\approx d[n] + \eta_p[n]$
	$\mathbf{h}_{n+1} = \mathbf{h}_n + \mu e[n]\mathbf{x}_n^*$

three signals,

$$\alpha[n] = d[n] + i_p[n] + \eta_p[n]. \tag{4.10}$$

Due to higher auxiliary antenna gain in the direction of the interferer, the reference input to the adaptive filter, $x[n]$, contains a higher interference-to-noise ratio (INR) copy of the interferer, $i_r[n]$, along with a white noise component, $\eta_r[n]$:

$$x[n] = i_r[n] + \eta_r[n]. \tag{4.11}$$

Note that the subscripts p and r stand for the primary and reference antennas, respectively.

The filter weight vector, $\mathbf{h}_n$, adaptively manipulates the reference channel to match the interference incident to the primary channel. The result, $y[n]$, contains

both the approximation, $\hat{i}_p[n]$, and a noise component, $\eta_h[n]$, which is $\eta_r[n]$ colored by $\mathbf{h}_n$,

$$y[n] = \mathbf{h}_n^{\mathrm{T}}\mathbf{x}_n = \hat{i}_p[n] + \eta_h[n]. \tag{4.12}$$

Because the interfering signal passes through different channel transfer functions, $i_p[n]$ and $i_r[n]$ may have very different properties. Channel differences could include propagation distances, antenna beam patterns, signal multipath structure, etc. Filtering $i_r[n]$ to closely resemble $i_p[n]$ is only possible if there is high correlation between the two signals. For example, if the two signals are severely misaligned in time, cancellation may not occur (see Section 4.3).

In the radio astronomy noise cancellation scenario, the error signal, $e[n]$, approximates $d[n] + \eta_p[n]$. This approach is possible because $d[n]$ and $x[n]$ are statistically uncorrelated; the auxiliary antenna does not receive any of the astronomical signal. Hence, the relationship between $e[n]$ and $d[n]$ is shown to be

$$e[n] = \alpha[n] - y[n] = d[n] + (i_p[n] - \hat{i}_p[n]) + \eta_p[n] - \eta_h[n] = \hat{d}[n] \approx d[n] + \eta_p[n]. \tag{4.13}$$

This is valid contingent upon two properties of the test environment. First, an accurate approximation to $i_p[n]$ must be found, making

$$i_p[n] - \hat{i}_p[n] \approx 0. \tag{4.14}$$

Second, the white noise incident to the auxiliary channel, $\eta_r[n]$, must not result in a significant filtered noise component, $\eta_h[n]$, being added to the primary channel. If $i_r[n]$ is a high interference-to-noise copy compared to $i_p[n]$, then $x[n]$ will be attenuated to create $\hat{i}_p[n]$. With enough attenuation, the residual filtered noise from the reference channel is insignificant compared to the dominant noise term, $\eta_p[n]$, making

$$\eta_h[n] \approx 0. \tag{4.15}$$

4.3 Data Alignment

As stated previously, there must be a high correlation between the interference components found in $\alpha[n]$ and $x[n]$ in order to realize significant cancellation. If such

correlation exists, but the primary and auxiliary antennas are widely separated, the signal propagation delay between channels may exceed the reach of a finite length LMS filter. Similarly, differences often exist in cable transmission delays from the two antenna feeds to a central data collection location. The resulting data misalignment can prevent interference cancellation. In other words, $\mathbf{h}_n$ cannot introduce enough delay in $x[n]$ to align it for maximum correlation with $\alpha[n]$. It is also possible that the interference term seen in $\alpha[n]$ is advanced relative to that in $x[n]$, making cancellation impossible. The two channels may be realigned by simply delaying one of the channels.

In order to determine which channel to delay, we must calculate the relative wave propagation delay, t_w, and cable transmission delay, t_c, referenced from the primary to the auxiliary channel. The net delay, t_{net}, can then be calculated as

$$t_{net} = t_w + t_c. \tag{4.16}$$

The relative wave propagation delay is calculated by

$$t_w = \frac{d_w}{c}, \tag{4.17}$$

where d_w is the propagation distance between the two antennas and c is the speed of light. Using the sensor geometry, it can be shown that

$$d_w = r_x \cos\theta_{el} \sin\theta_{az} + r_y \cos\theta_{el} \cos\theta_{az} + r_z \sin\theta_{el} = \mathbf{r}^{\mathrm{T}}\mathbf{s}, \tag{4.18}$$

where $\mathbf{r} = \begin{bmatrix} r_x & r_y & r_z \end{bmatrix}^{\mathrm{T}}$ is the 3-dimensional location of the auxiliary antenna referenced to the primary antenna at the origin. I adopt a coordinate system with the $+x$ direction being due east, the $+y$ direction due north, and the +z direction corresponding to positive elevation. θ_{az} and θ_{el} are the azimuth and elevation coordinates of the interferer with $\theta_{az} = 0^\circ$ defined to be north, increasing clockwise, and $\theta_{el} = 90^\circ$ as zenith. The unit vector, $\mathbf{s} = \begin{bmatrix} \cos\theta_{el}\sin\theta_{az} & \cos\theta_{el}\cos\theta_{az} & \sin\theta_{el} \end{bmatrix}^{\mathrm{T}}$, points to the interferer. This same development of geometric delay is also used for array beamforming in Section 3.2.1.

The relative cable transmission delay is calculated by

$$t_c = \frac{L_p - L_r}{v_c}, \tag{4.19}$$

where L_p and L_r are the respective cable lengths of the primary and reference channels, and v_c is the cable transmission speed.

The approximate number of samples one of the channels must be delayed, D, can be calculated by

$$D \approx t_{net} f_s, \tag{4.20}$$

where f_s is the sampling frequency (note that D must be an integer). If D is positive, then the auxiliary channel should be delayed. Conversely, if D is negative, the primary should be delayed. Due to the adaptive nature of the LMS filter, as long as D approximately compensates for the bulk propagation delay between the two channels, the filter should converge to an effective solution.

In order to obtain maximum correlation between $x[n]$ and $\alpha[n]$, it is often advantageous to align the two channels to create a pseudo-symmetric filter, with the largest taps (highest correlation lags) found in the center of the filter. Once the signals are aligned, this can be accomplished by delaying $\alpha[n]$ by half the filter order ($\frac{p}{2}$). This is achieved by altering Eq. 4.20 as shown,

$$D \approx t_{net} f_s - \frac{p}{2}, \tag{4.21}$$

rounded to the nearest integer.

4.4 Simulations

In order to verify proper performance of the radio astronomy scenario setup shown in Figure 4.2, simulations and initial tests were conducted using MATLAB. Contrasted with the difficult and cumbersome task of implementing the algorithm in the complex DSP environment, development in MATLAB is relatively simple and adaptable. In order to facilitate conversion from MATLAB to a DSP environment, the code was written with program structure matching as closely as possible the potential DSP C code, avoiding some of the built-in MATLAB functions not available in C.

First tests involved synthesizing the input signals $\alpha[n]$ and $x[n]$. Initially, sinusoids were used to represent $d[n]$, $i_p[n]$, and $i_r[n]$. Effective cancellation was demonstrated.

After demonstrating performance using synthetic signals, the MATLAB code was tested using real signals obtained using the BYU VSA platform described in Chapter 2. One antenna tracked a GLONASS satellite, providing the reference signal, $x[n]$. A second antenna, positioned approximately 7.5° away from the incident interference direction, served as the primary channel. Approximately one second of raw data was simultaneously collected from each channel. As there was not an astronomical signal of interest within the processing bandwidth, a sinusoidal $d[n]$ was synthetically added to the primary channel to create $\alpha[n]$.

Results from this test are shown in Figures 4.3 and 4.4. Notice that the injected sinusoid at 1609.1 MHz is untouched by the filtering process, and the GLONASS interference has been eliminated. Figure 4.4 illustrates the adaptation of $\mathbf{h}_n$ with time. At a complex sample rate of 4 MHz, there were over 4×10^6 filter weight vector updates. Notice that one filter tap is significantly larger than the others. This suggests that in this case, the main difference between the interference in the primary and reference channels is an amplitude and phase shift. The two antennas were closely spaced with very little, if any multipath structure.

While developing the software prototype in MATLAB, I found that results were very dependent on the filter step size μ. In order to insure stability and better convergence properties to the algorithm, the normalized LMS filter was used. It is equivalent to the original LMS filter in all respects except for the vector update equation

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \frac{\mu}{\|\mathbf{x}_n\|^2} e[n]\mathbf{x}_n^*, \tag{4.22}$$

where $\|\mathbf{x}_n\|^2 \equiv {\mathbf{x}_n}^{\mathrm{H}}\mathbf{x}_n$. Note that normalization by $\|\mathbf{x}_n\|^2$ only changes the magnitude of the vector update, not its direction. This ensures that the filter will converge in the mean-square sense if $0 < \mu < 2$ [37]. This is not true for the original vector update equation given by Eq. 4.9.

The normalization term can be recursively computed by

$$\|\mathbf{x}_{n+1}\|^2 = \|\mathbf{x}_n\|^2 + |x(n+1)|^2 - |x(n-p)|^2, \tag{4.23}$$

requiring just two squaring operations, one addition, and one subtraction. Initially, I

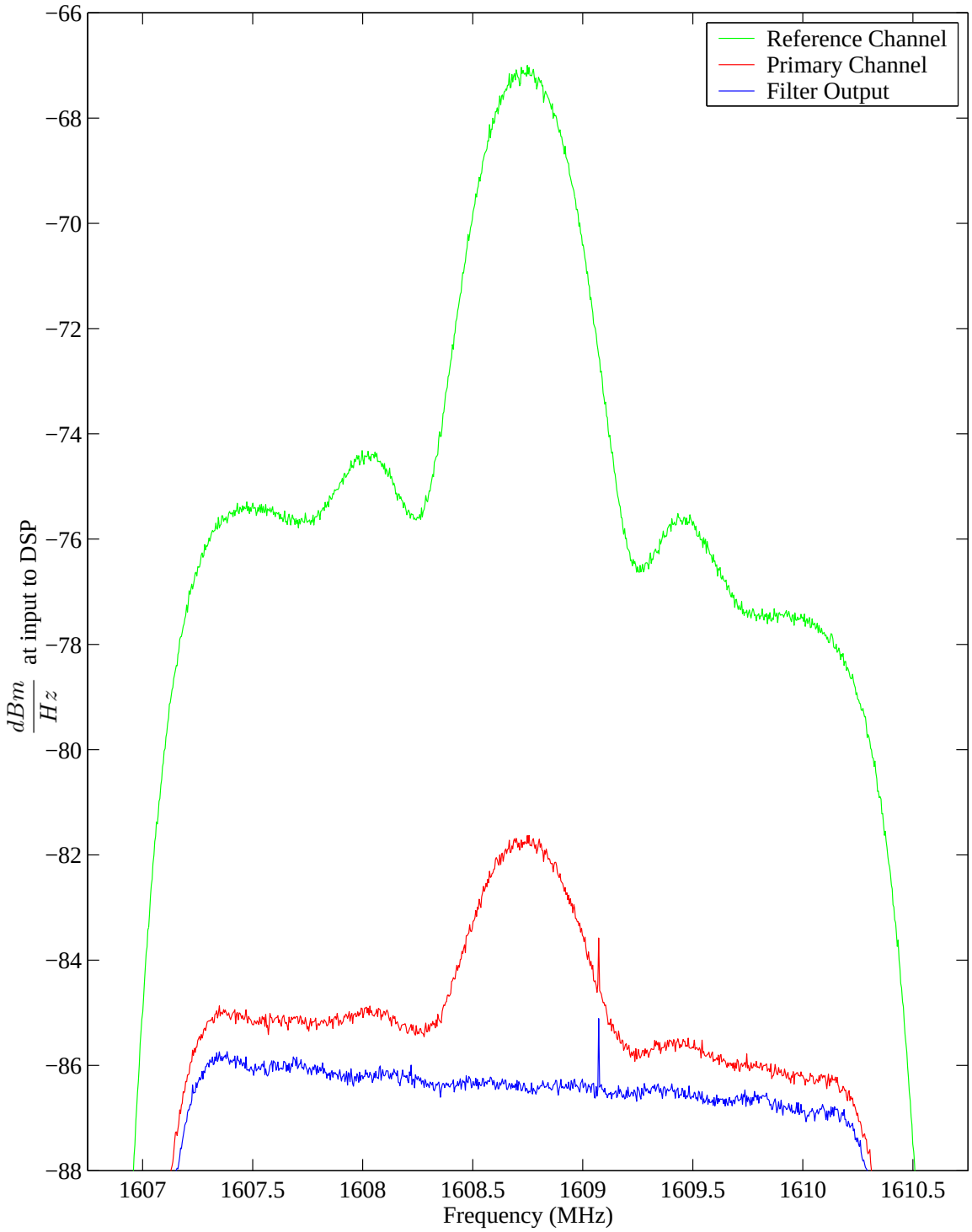


Figure 4.3: Cancellation of GLONASS 789 using the normalized LMS adaptive filter. Raw data was collected using the digital receiver and post-processed in MATLAB. This is a PSD estimate of the reference channel $x[n]$, primary channel $\alpha[n]$, and filter output $\hat{d}[n]$ with $p = 20$ and $\mu = 0.001$. The test tone injected at 1609.1 MHz was properly preserved.

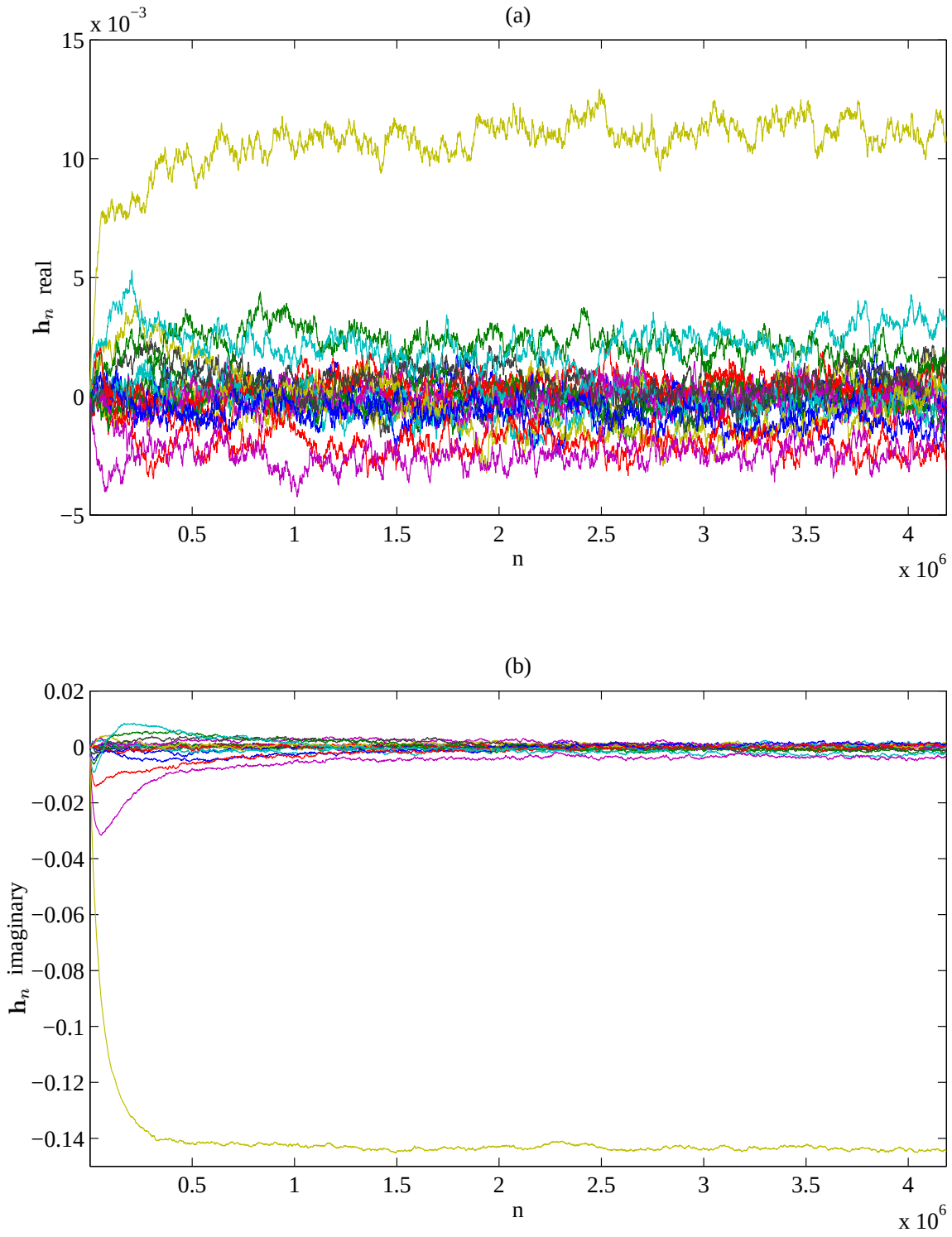


Figure 4.4: Convergence of $\mathbf{h}_n$ with time. (a) Real component of $\mathbf{h}_n$, (b) Imaginary component of $\mathbf{h}_n$

implemented the normalized LMS algorithm in the DSP, but found that I could not optimize the function significantly due to the additional computation required for division by $\|\mathbf{x}_n\|^2$. On the other hand, I was able to optimize the original LMS vector update equation quite significantly, as described in Section 4.5. Although the original LMS algorithm does not self-adjust for large deviations in signal power, its real-time computational performance far outweighs the normalized algorithm's benefits in this application.

4.5 DSP Implementation

After completing simulations and tests using MATLAB, I implemented the LMS algorithm in the DSP. Despite being a straightforward algorithm, coding presented significant challenges. These included the interface to the digital receivers, inter-processor communication, proper data alignment, and code optimization.

For development, it is important not only to filter the data, but to evaluate performance. This involves analysis of spectral content of the inputs, $\alpha[n]$ and $x[n]$, as well as the output, $\hat{d}[n]$ (see Section 3.1). Power spectral density (PSD) estimates help determine the magnitude of interference cancellation, the quality of the reference signal, and other characteristics of the RFI mitigation scenario. PSD estimates of $e[n]$ ($\hat{d}[n]$), $\alpha[n]$, and $x[n]$ are computed in real-time. This is necessary because streaming of the high data rate time-domain signals involved in this application are impossible, given our limited data transfer interface between the DSP and host PC. The PSD estimator performs significant data reduction so the resulting spectral data can be streamed to the PC host.

Figure 4.5 illustrates the general structure of the DSP LMS filtering application. The inputs $\alpha[n]$ and $x[n]$ are sampled by the 6216 digital receivers and sent to processors C and A, respectively. This process not only involves an A/D convertor, but also conversion to a complex baseband representation, band selection, signal decimation, and bandpass filtering to achieve the desired complex sampling rate.

If $\alpha[n]$ and $x[n]$ need to be realigned due to reasons detailed in Section 4.3, this is accomplished immediately after receiving the data in processors C and A; $\alpha[n]$ is

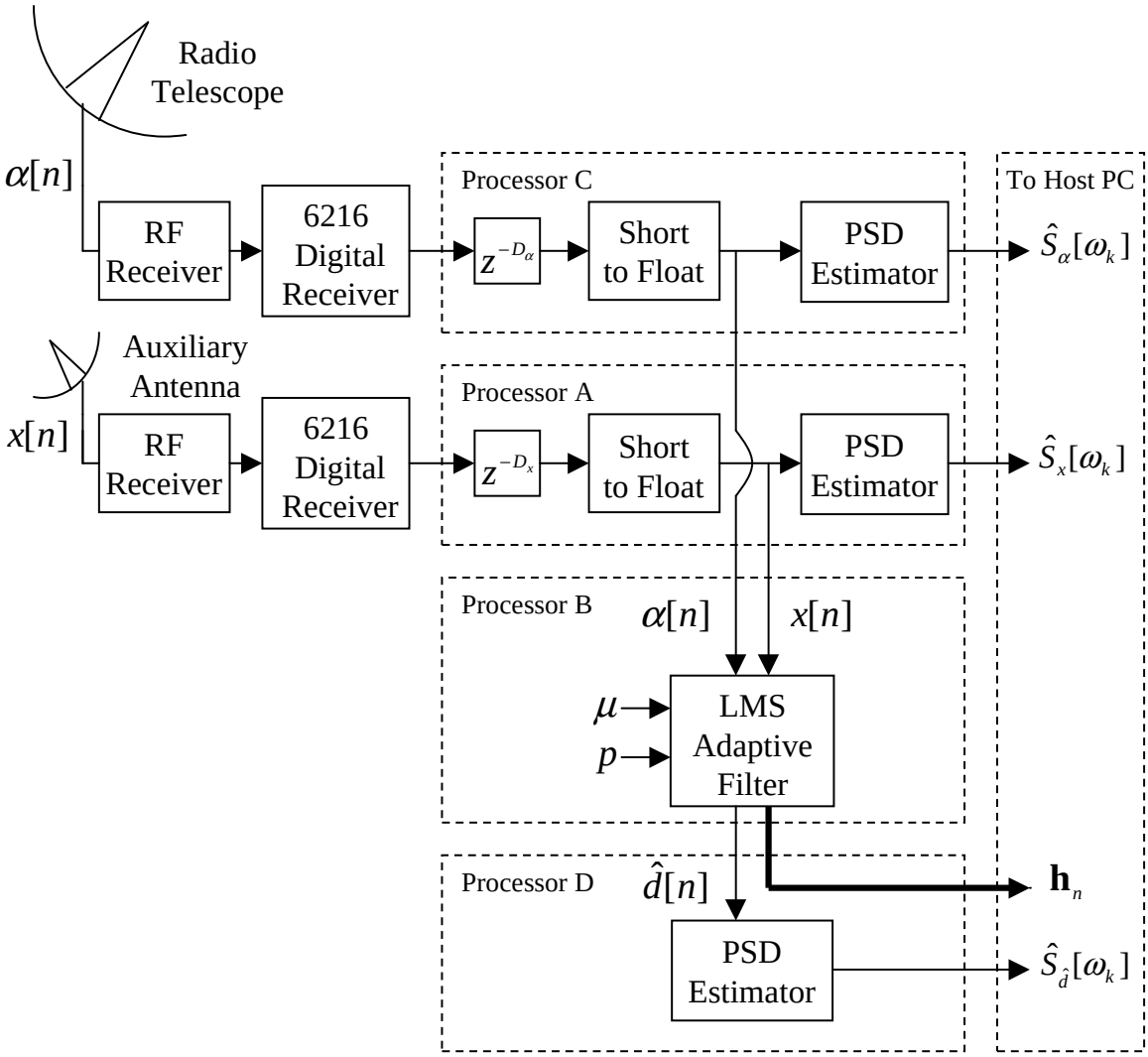


Figure 4.5: DSP implementation of the LMS adaptive filter

delayed by D_α, and $x[n]$ by D_x. Once the signals are roughly aligned, the signals are converted to floating point in preparation for processing. Floating point computations enable higher degrees of cancellation than fixed point. As shown in Figure 4.5, the PSD estimates of $\alpha[n]$ and $x[n]$, $\hat{S}_\alpha[\omega_k]$ and $\hat{S}_x[\omega_k]$, also occur in processors C and A. The PSD estimators utilize the same techniques outlined in Section 3.1.

After conversion to floating point, the signals are forwarded to processor B via inter-processor communication, where the adaptive filtering occurs. As the computations required for implementation of the LMS adaptive algorithm are quite heavy, processor B is the limiting factor in determining the processing bandwidth, filter order, etc. Therefore, as many calculations as possible were pushed into the other three processors, including conversion to floating point and spectral analysis.

One of the most time consuming elements of implementing the LMS filter in the DSP environment was the optimization of the filtering and tap update function. The Texas Instruments compiler is powerful, but performance can greatly vary with seemingly insignificant changes in the C code. Some of these changes included ordering of expressions, parameter designation (e.g. "volatile"), memory allocation, structure of processing loops, etc. I also obtained different results depending on the compile options. In a few cases, I was able to create faster code by specifying lower levels of automatic compiler optimization. With so many possible permutations, my only option was to compile the code under many different circumstances. In order to determine which was the fastest compilation, I counted the clock cycles of the assembly code produced and/or ran the code while measuring unused processing capacity.

When using the internal digital receiver sample clock of 64 MHz, there is a finite set of possible final sample rates due to available sample clock and sampled signal decimation rates. The sample clock rate can be decimated by 2, 4, 6, 8 or 10, while the sampled signal can be decimated by 2, 4, 8, 16, 32 or 64. As the front-end low-pass anti-aliasing filter has a fixed cutoff frequency of 25 MHz, if the sample frequency drops below 50 MHz, aliasing will occur. Therefore, if the 64 MHz sample clock is decimated by any of the possible decimation rates, aliasing will occur; the higher the decimation, the more severe the aliasing. Consequently,

Table 4.2: Summary of the DSP LMS filter processing bandwidths, sample clock frequencies, signal decimation rates, highest respective filter orders, and number of real floating point multiplies per second (measured using the TI TMS320C6701 167 MHz processor clock speed).

Signal Bandwidth	Sample Clock	Decimation	Filter Order	multiplies/sec
0.412500 MHz	26.4 MHz	64	42	173.2500×10^6
0.534375 MHz	34.2 MHz	64	30	160.3125×10^6
1.006250 MHz	64.4 MHz	64	12	121.1250×10^6
2.225000 MHz	35.6 MHz	16	5	111.2500×10^6
2.562500 MHz	41.0 MHz	16	4	102.5000×10^6
3.250000 MHz	52.0 MHz	16	3	97.5000×10^6
3.843750 MHz	61.5 MHz	16	2	76.8750×10^6

obtaining the desired final sample rate is best achieved by using the sampled signal decimation capabilities of the digital receiver. These constraints only permit two possible processing bandwidths when using the internal 64 MHz sample clock: 1 MHz and 2 MHz. A much larger set of possible processing bandwidths can be achieved by using an external sample clock. A few of these bandwidths, along with their highest respective filter order and number of real multiplies per second, are presented in Table 4.2. Note that obtaining the 0.4125 MHz and 0.534375 MHz final sample rates entails sampling the data below the 50 MHz required to avoid aliasing. Lower sample rates can be used without aliasing the signal by adding an additional low-pass filter with appropriate characteristics to the input of the digital receiver. When implementing these lower sample rates in the Green Bank test scenario, signals were bandlimited to 10 MHz to avoid aliasing.

The real-time DSP platform implementation of the LMS adaptive filter has proven to be an effective interference cancellation tool. Some of the results for both the BYU VSA and the Green Bank Telescope (GBT) are given in the following sections.

Figure 4.6: The BYU VSA in test configuration. The left antenna is pointed at a dipole antenna mounted on a tower. The signal transmitted by the dipole is either an FM sweep interfering signal or a "fake" astronomical source. The right antenna is tracking either a hydrogen source or a GLONASS satellite.

4.6 BYU VSA Tests

Two classes of tests were performed using the BYU VSA. Due to limitations of our test platform, we encountered significant difficulty searching for a satellite interferer coincident in frequency with a detectable astronomical spectral line. As an alternative, tests were performed by restoring clarity to hydrogen lines corrupted with a locally generated FM sweep signal. This was broadcast using a dipole antenna mounted on a tower on the Brigham Young University engineering building. For the second class of tests, GLONASS satellite downlink signals were excised from data containing a weak "fake" astronomical source broadcast from the same tower.

In order to demonstrate proper preservation of a desired astronomical signal while removing interference, the Cygnus 1420 MHz hydrogen line was tracked by the primary channel, $\alpha[n]$, with an interfering FM sweep signal being received through the antenna sidelobes. A second antenna was positioned to receive a high INR copy of the reference signal, $x[n]$. The results proved to be very successful. As can be seen in Figure 4.7, the interference incident to the primary channel contained less power per frequency bin (on average) than the hydrogen line. Therefore, both the signal and interference were below the noise floor. Such weak interference can be difficult to extract from data because noise in the channel can prohibit sufficient convergence of the adaptive filter. This is because the error in the estimate of $E\{e[n]\mathbf{x}_n^*\}$, $\widehat{E\{e[n]\mathbf{x}_n^*\}} = e[n]\mathbf{x}_n^*$ (used to calculate $\mathbf{h}_{n+1}$), increases with the noise power, $\eta_p[n]$ and $\eta_r[n]$, contained in $\alpha[n]$ and $x[n]$, respectively.

As mentioned earlier, I also cancelled a GLONASS satellite downlink signal while preserving a weak "fake" astronomical source. These tests gave fine results; some of these are displayed in Figure 4.8. The "fake" source was either a low power sinusoid or an FM sweep signal featuring a double peak, representing the red- and blue-shifted peaks of a typical OH spectral line. In all cases, the desired signal was not touched by the filtering process.

An interesting phenomenon worth noting was observed when restoring clarity to hydrogen lines corrupted with an FM sweep signal. In some cases, not only the interferer incident to the primary channel, $i_p[n]$, was subtracted from $\alpha[n] = d[n] + i_p[n] + \eta_p[n]$. Instead, a portion of the noise component, $\eta_p[n]$, was also excised in the process. This is illustrated in Figure 4.9. There are only a few possible explanations for the observed behavior. First of all, the assumption was made that $\eta_p[n]$ and $\eta_r[n]$ are statistically uncorrelated, which may be incorrect (this will be discussed later). Second, the observed behavior could result if $\eta_p[n]$ and $i_r[n]$ were correlated; this is highly improbable. Third, an excessively large adaptive constant may be causing the problem. Notice that noise power is added to those frequencies untouched by the interferer, a sign of an overly aggressive adaptive constant (see Section 4.7.7 and Appendix C.5). This is especially apparent in test (c).

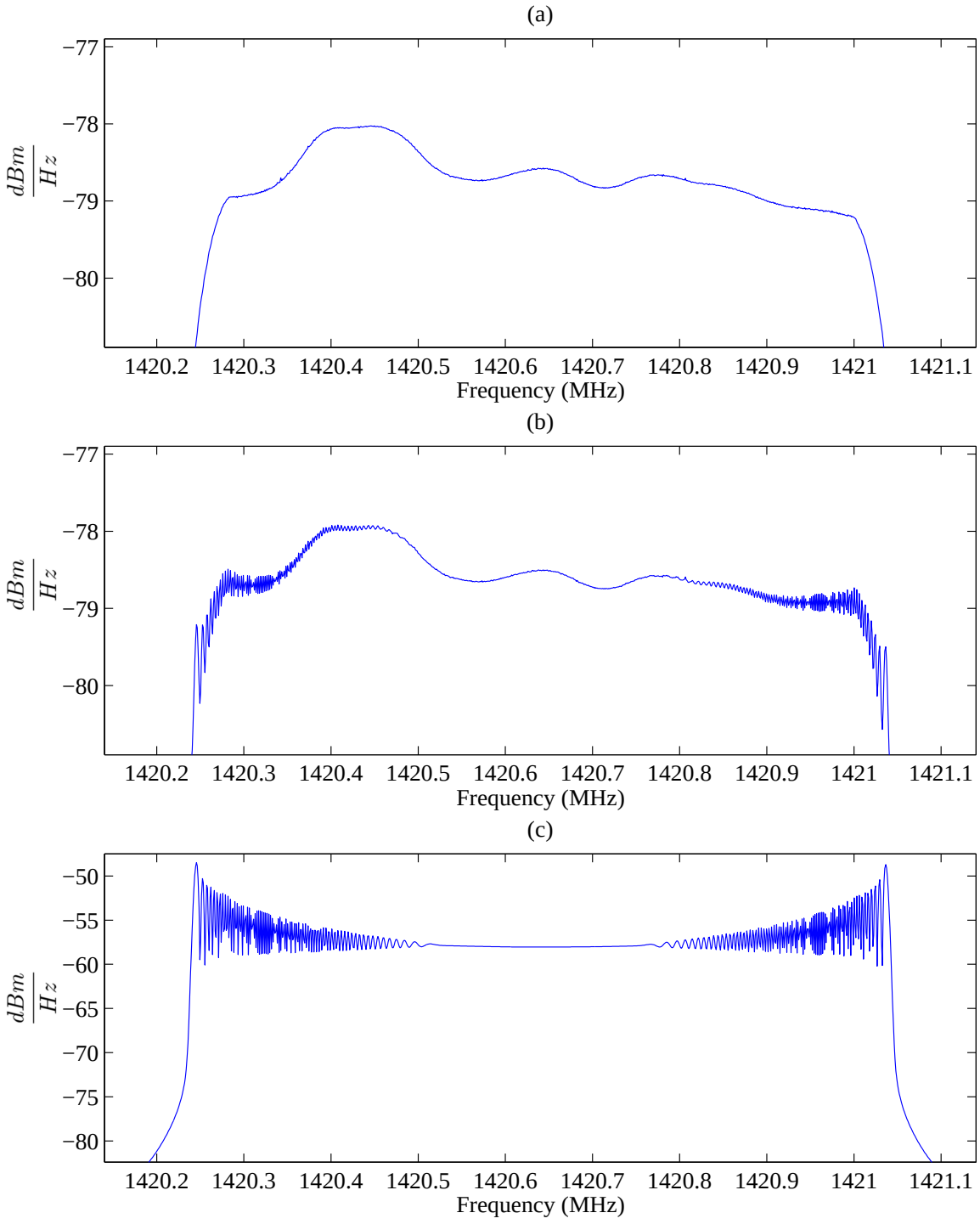


Figure 4.7: Results of one VSA test. Test Parameters: 20 minutes integration, 12 complex taps. (a) PSD estimate, $S_{\hat{d}}[\omega_k]$, of the filter output. Notice that the interference is undetectable in the spectral baseline. (b) PSD estimate, $S_{\alpha}[\omega_k]$, of the primary channel, $\alpha[n]$. The spectral hydrogen line is that of Cygnus corrupted by an FM sweep interferer. (c) PSD estimate, $S_x[\omega_k]$, of the reference channel, $x[n]$, a high INR copy of the interferer.

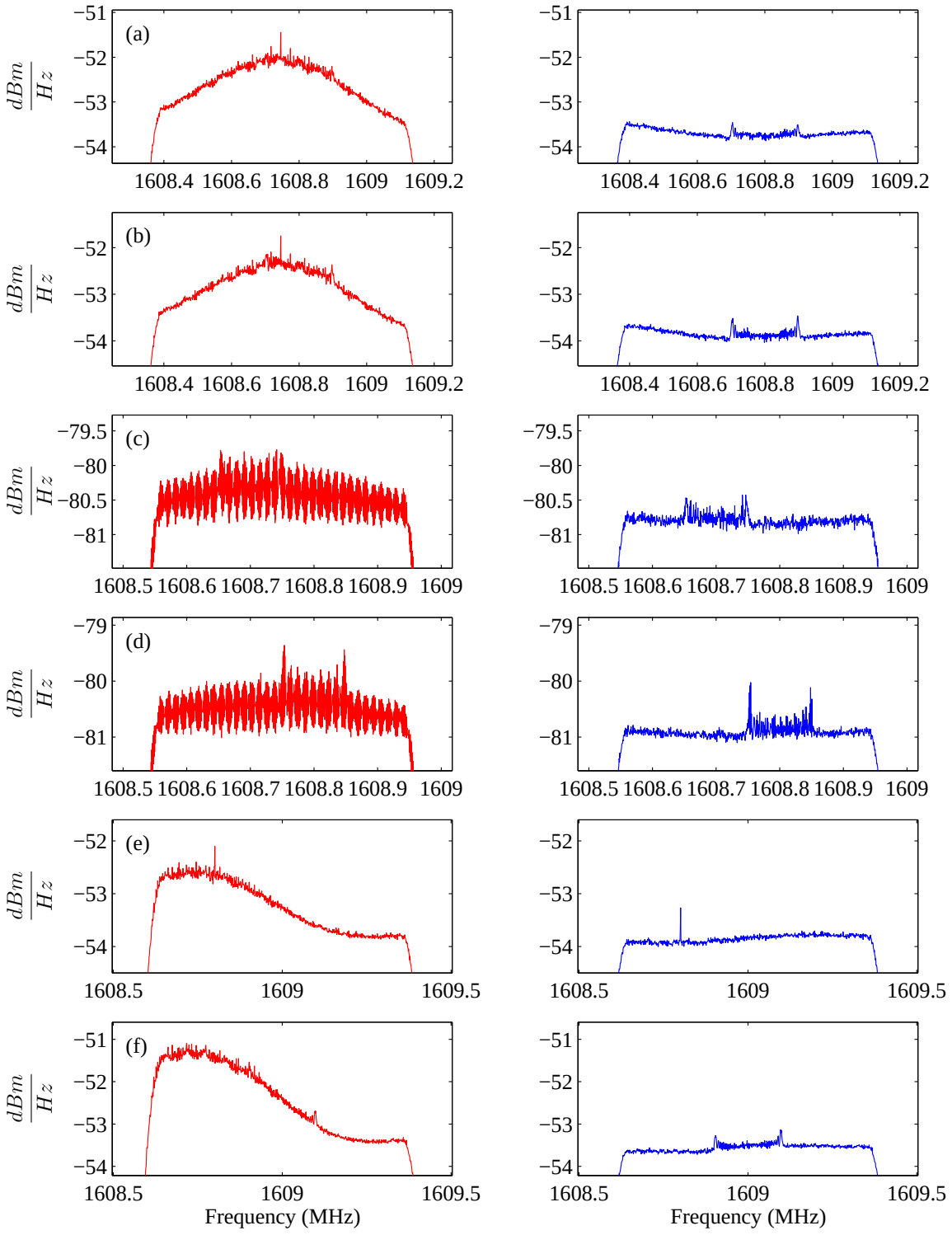


Figure 4.8: BYU VSA signals both before (red) and after (blue) filtering. Tests (c) and (d) utilized a filter order of $p = 30$, while tests (a), (b), (e) and (f) utilized $p = 12$. Each test demonstrates subtraction of GLONASS interference while preserving a weak "fake" astronomical source.

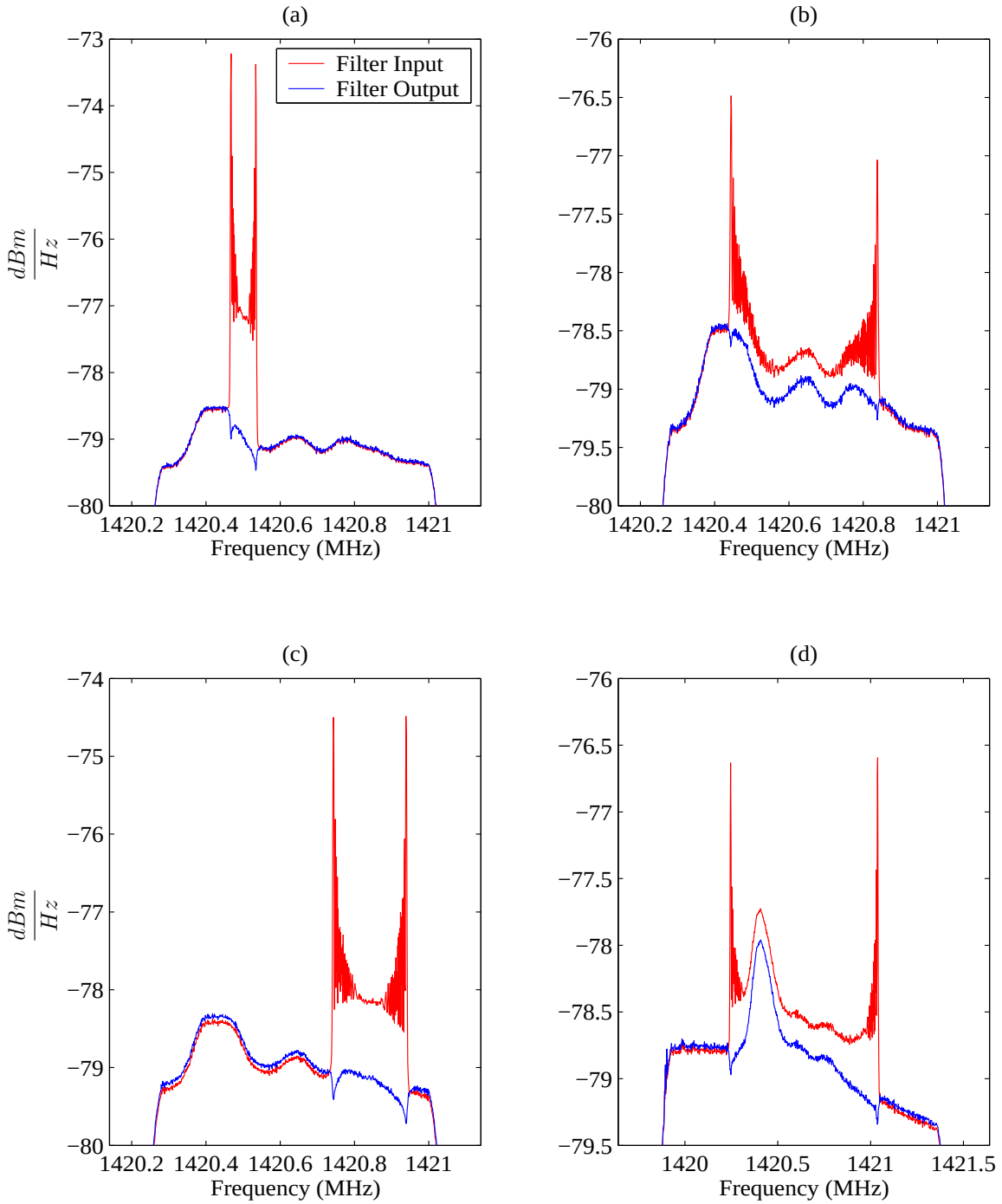


Figure 4.9: Demonstration of an unexpected occurrence observed when cancelling an interferer broadcast by a dipole antenna from a tower on the roof of the BYU engineering building. Instead of just cancelling the interferer incident to the primary channel, $i_p[n]$, a portion of the noise component, $\eta_p[n]$, was also excised in the process. Fortunately this odd behavior was not observed in tests performed with the GBT. Test parameters: (a), (b) & (c) $p = 12$, bandwidth=1 MHz; (d) $p = 5$, bandwidth=2 MHz.

How could $\eta_p[n]$ and $\eta_r[n]$ be statistically correlated causing the overzealous cancellation by the LMS algorithm? As stated previously, $\eta_p[n]$ and $\eta_r[n]$ are made up of several components: cosmic background noise, thermal receiver noise, atmospheric noise, and ground noise from antenna spill-over. With a relatively large antenna 3 dB beamwidth ($\approx 5 - 6^\circ$), perhaps both antennas are receiving correlated noise components such as cosmic background noise, when pointed close to one another. Perhaps there is an undetected, deterministic component to the white noise allowing additional cancellation. Initially, the quantization noise introduced by the DSP was suspect. After some testing, however, it was determined that the quantization noise floor was buried deep below the signal noise floor making it an unlikely culprit. The cause for the behavior observed in Figure 4.9 is unknown; time limitations impeded further investigation. Fortunately this odd behavior was not observed in tests performed with the Green Bank Telescope.

The VSA results demonstrate the high potential for the LMS algorithm in cancelling interference from radio astronomy data. More convincing and in-depth studies were conducted with the 100 meter Green Bank Telescope. As a result, most performance analysis of the LMS adaptive filter is presented in Section 4.7.

4.7 Green Bank Tests

Extensive GLONASS cancellation tests were performed using the Green Bank Telescope in West Virginia. High levels of cancellation and effective preservation of the desired signal were consistently demonstrated.

4.7.1 The Green Bank Telescope

Table 4.3 details many of the telescope specifications. The GBT is the largest fully steerable telescope in the world. It boasts many new innovations. The 100 meter aperture is unblocked, allowing the incident radiation to meet the surface directly. This design increases the telescope efficiency and improves the beam response, which is normally corrupted by reflection and diffraction on the feed support arms. Due to the telescope's enormous size, gravity, temperature and wind can significantly distort

Table 4.3: GBT specifications [43]

Location	Green Bank, WV
Telescope aperture	100 m (effective)
Elevation limits	5° lower limit 95° upper limit
Ultimate frequency coverage	100 MHz—100+ GHz
Slew rates	40°/min Azimuth 20°/min Elevation
Optics	110 m × 100 m unblocked section of a 208 m parent paraboloid off-axis feed arm
Available foci	Prime: $F/D = 0.29$ (208 m parent parabola) Gregorian: $F/D = 1.9$ (100 m effective aperture)
Main reflector	2209 actuated panels Average panel rms = 68 μm
Surface rms (specified)	Phase 1 (passive surface) : 1200 μm Phase 2 (open loop active) : 360 μm Phase 3 (closed loop active) : 240 μm
Pointing accuracy (specified)	Phase 1 : 14” Phase 2 : 3” Phase 3 : 1”
FWHM beamwidth	12.4 arcmin/f(GHz) = 740 arcsec/f(GHz)
Subreflector	8 m reflector with Stewart Platform
Gain	$\approx$ 1.7K/Jy
Polarization	Dual circular, dual linear, or all four simultaneously
Side lobes	First sidelobes 20 to 25 dB below main beam

the shape of the dish, severely degrading measurements at higher frequencies (>15 GHz). To compensate, the dish utilizes a laser metrology system and active reflector segment positioning for maintaining proper surface shape and pointing accuracy. An extensive introduction to the Green Bank Telescope (GBT) can be found in “The GBT User Manual” [43].

Figure 4.10: 3.6 meter reference antenna used for GBT interference cancellation tests. The weatherproof box to the right of the antenna houses the antenna control hardware and power supplies. Notice the GBT feed peeking over the trees.

4.7.2 Green Bank Setup

As a clean (high INR) copy of the interference signal is required for cancellation using the LMS adaptive filter, a reference antenna was needed at the NRAO Green Bank facility for the real-time cancellation experiments. In addition, the NRAO RFI mitigation team at Green Bank needed a directional antenna for conducting RFI surveys (identifying and removing on-site interference sources). For these two purposes, a 3.6 meter Kaul-Tronics Inc. parabolic antenna was installed in October 2002. The only difference between this antenna and those used in BYU's VSA is the larger aperture. Because the same dual azimuth/elevation rotors drive the 3.6 meter antenna and the antennas of the VSA, we were able to utilize all of the control hardware and software developed at BYU.

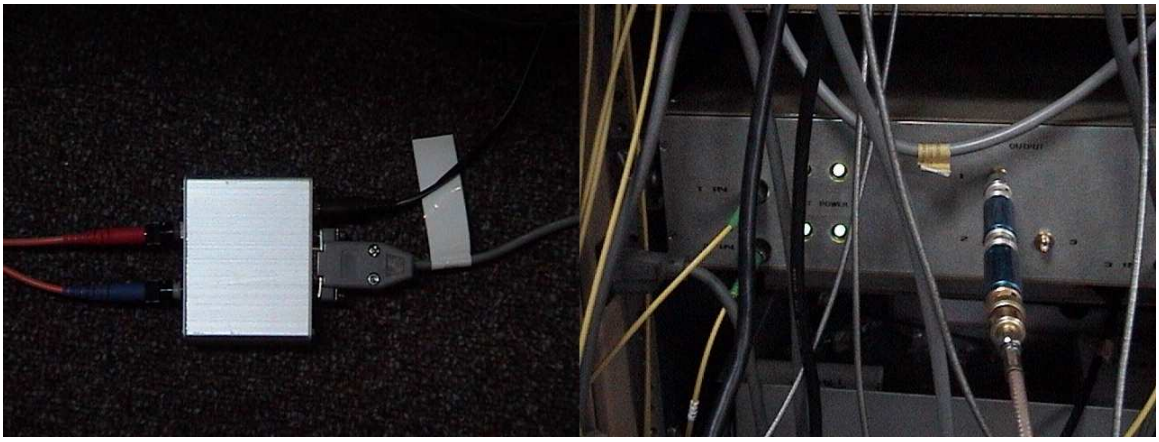


Figure 4.11: Left: RS-232 to optical fiber modem. Right: RF coax to optical modem. Input/Output 1 carries the signal from the 3.6 meter antenna.

A much larger distance separates the GBT control room, where the antenna control PC is located, and the 3.6 meter antenna, than the distance between the BYU control computer and VSA antennas. Consequently, we couldn't directly run a power cable over this distance. As a solution, we used two RS-232 to optical fiber modems to communicate between the control PC and antenna positioning hardware (see Figure 4.11). Additionally, a coaxial cable carrying the RF signal would be much too lossy over such a distance. To remedy this, two coax to optical modems were used to transmit the RF signal; a similar setup is also used to transmit the GBT signal to the GBT control room.

In addition to installing and calibrating the 3.6 meter antenna in October 2002, we were able to run some GLONASS cancellation tests with the GBT and DSP platform. The results were very promising, paving the way for the more extensive tests which occurred in January–February 2003.

4.7.3 Green Bank Data Alignment

For reasons discussed in Section 4.3, the GBT and reference signals must be realigned to realize proper interference cancellation. Data alignment is particulary

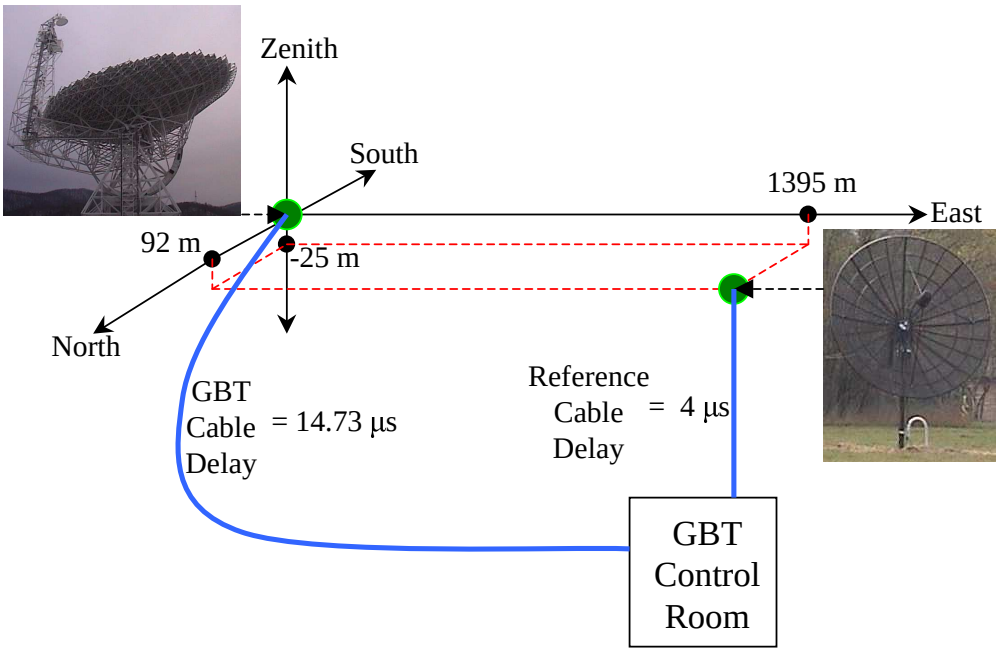


Figure 4.12: Illustration detailing the relative position between the GBT and 3.6 meter antenna, along with respective cable delays to the GBT control room. The GBT is placed at the origin of the coordinate system.

important in the Green Bank test scenario. The GBT and 3.6 meter antenna are separated by 1.4 kilometers; additionally, the GBT RF signal path is over two kilometers longer than the reference signal path. Without realignment, proper cancellation would be impossible when using shorter filter lengths. This section will discuss the methods used for realigning the GBT signal with the 3.6 meter reference signal.

The longitude, latitude and elevation of both the GBT and 3.6 meter reference antenna were used to calculate their relative positions. Precise measurements for the GBT were provided by the Green Bank staff [44]. Measurements for the 3.6 meter dish were recorded by using a hand-held GPS receiver and a topographical map of the Green Bank site. Final measurements are given in Table 4.4. Using this information, the position of the 3.6 meter antenna referenced to the GBT was calculated using the coordinate system described in Section 4.3 [45]. This was found to be approximately $\mathbf{r} = \begin{bmatrix} 1395 & 92 & -25 \end{bmatrix}^{\mathrm{T}}$ m (see Figure 4.12).

Table 4.4: Longitude, latitude and elevation of the GBT and 3.6 meter antenna

	Longitude	Latitude	Elevation
Green Bank Telescope	79° 50′ 23.406″ W	38° 25′ 59.236″ N	855.65 m
3.6 meter antenna	79° 49′ 26″ W	38° 26′ 2″ N	830 m

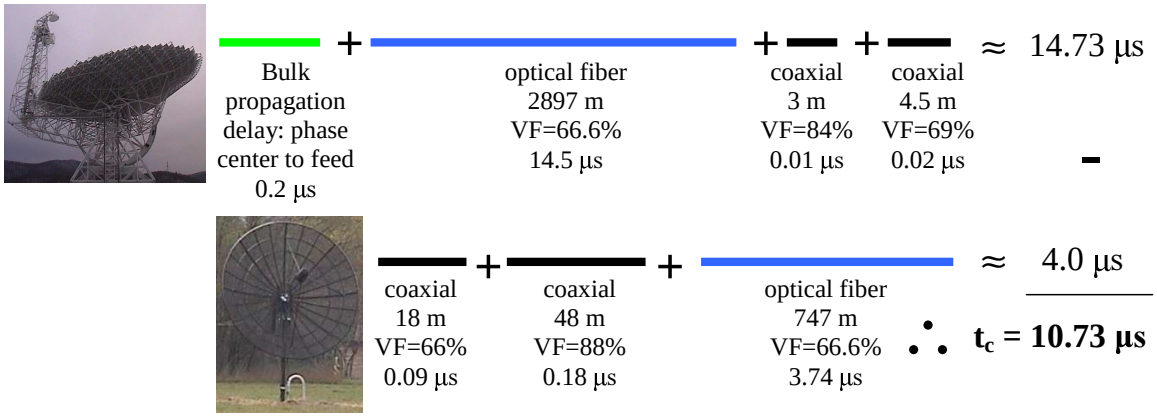


Figure 4.13: Illustration of cable delay for each segment of the GBT and 3.6 meter channels. Calculation of t_c, the relative cable delay, is also shown.

In order to calculate the respective cable propagation delays for the GBT and reference antenna, I collected information including cable lengths, cable velocity factors (VF), and measured cable segment delays. I then used this information to calculate t_c, the relative cable transmission delay, or difference between the absolute cable delay of the GBT and 3.6 meter antenna. The successive cable segments of each channel are shown in Figure 4.13, along with the calculation of absolute and relative cable delays.

Due to the vast aperture of the GBT, the wave propagation delay from the antenna phase center to the feed cannot be ignored if precise data realignment is desired. In Figure 4.13, this term is referred to as "bulk propagation delay: phase center to feed." I calculated this by using a GBT design diagram provided by NRAO, shown in Figure 4.14 [46]. The telescope phase center is the intersection of the

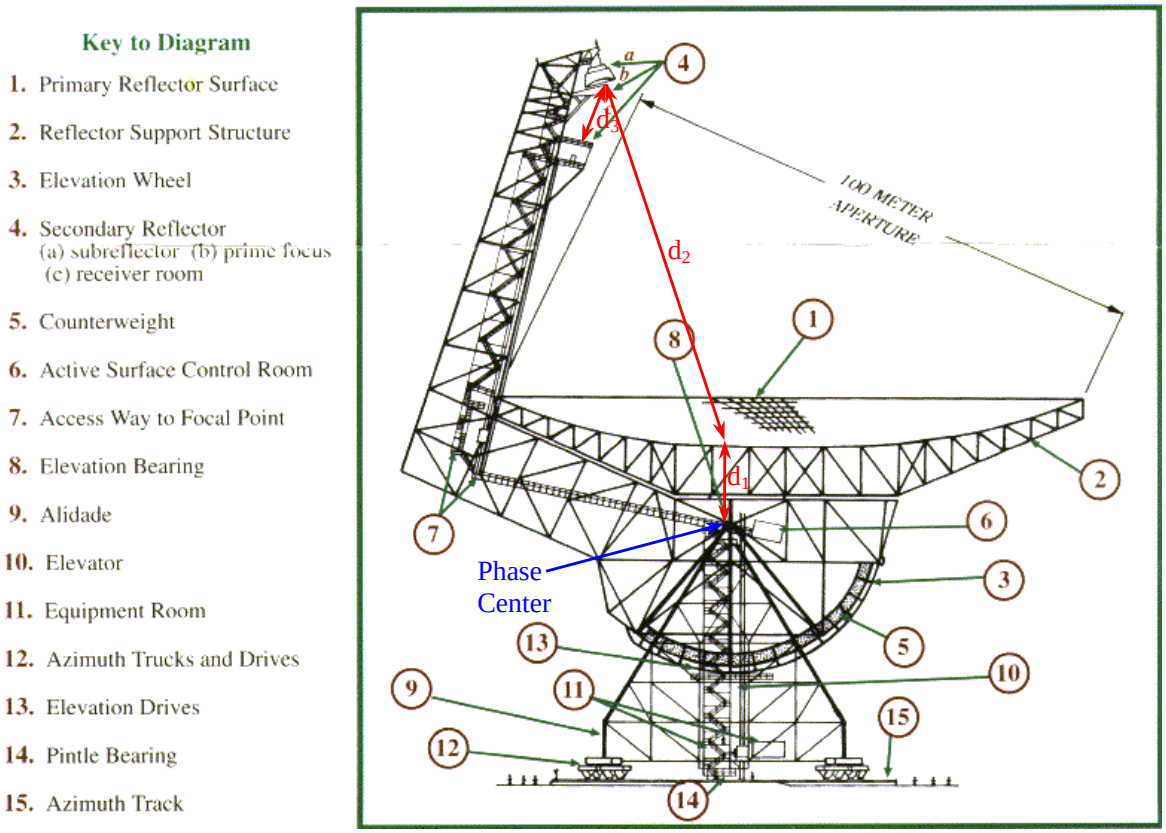


Figure 4.14: Illustration of the GBT design [46]. Distances $d_1 = 13.6$ m, $d_2 = 65.2$ m and $d_3 = 10.6$ m are used to calculate the bulk propagation delay from the GBT phase center to the feed, as described by Eq. 4.24.

center of azimuth and elevation rotation—the location which remains constant for all telescope positions. As this phase center is below the reflector surface, I measured the distance from the phase center to the nearest point on the reflector surface, d_1. Distance from this point on the reflector surface to the secondary reflector, d_2, and from the secondary reflector to the receiver room, d_3, were also estimated. All measurements were calculated by using the 100 meter scaled aperture reference on the diagram. The bulk propagation delay from the phase center to the feed, constant for all telescope positions, is given by the following expression:

$$\nabla t = \frac{d_2 + d_3 - d_1}{c} = \frac{65.2 \text{ m} + 10.6 \text{ m} - 13.6 \text{ m}}{c} \approx 0.2 \ \mu\text{s}. \tag{4.24}$$

This delay can conceptually be treated as a separate cable segment in the GBT signal path (see Figure 4.13).

As positions of satellite interferers can be predicted with satellite tracking software weeks in advance of a test, the necessary realignment parameters can also be precalculated. Calculations only depend on the interferer position and not the desired signal (astronomical source) position. Therefore, when conducting astronomical observations in a frequency band coincident with satellite interferers, one can obtain all necessary realignment data and interferer position information well in advance of observational tests. It may also be possible to schedule observations at times when particularly problematic satellites are below the local horizon.

For my second round of tests, I was allocated six blocks of time on three consecutive days (January 30–February 1, 2003) for running interference mitigation tests with the GBT. Table 4.5 displays the precalculated realignment data for the final block of time on Saturday, February 1.

In practice, the data realignment worked very well. I was able to consistently place the highest correlated sample of the reference signal $x[n]$ in the center of the adaptive filter, independent of the interferer direction of arrival. Figure 4.15 illustrates an instantaneous sample of the adaptive filter, $\mathbf{h}_n$. This was the filter's state at approximately 12:20 p.m. EST on Saturday February 1, 2003. The largest taps lie in the center of the filter, suggesting that the highest correlated sample of $x[n]$ is also in

Table 4.5: Precalculated GBT data realignment parameters for tests during the afternoon of Saturday, February 1, 2003. All times are EST. Az and El are the azimuth and elevation of GLONASS satellite #789 in ten minute intervals. Parameters d_w and t_w are the relative wave propagation distance/delay between channels, while t_{net} is the net delay (further described in Section 4.3). The relative cable delay, $t_c = 10.73$ μs, was used for these computations (see Figure 4.13). Note that the filter orders, p, detailed in the table correspond to their highest respective bandwidths shown in Table 4.2. The final seven columns contain the necessary delay (in samples) for each respective filter order. A positive number suggests the auxiliary channel should be delayed.

Time	Az	El	d_w(m)	$t_w(\mu s)$	$t_{net}(\mu s)$	$p = 42$	$p = 30$	$p = 12$	$p = 5$	$p = 4$	$p = 3$	$p = 2$
11:30	94.4	9.7	1359.83	4.54	15.27	-14.70	-6.84	9.36	31.47	37.12	48.11	57.68
11:40	90.2	12.8	1354.47	4.52	15.25	-14.71	-6.85	9.34	31.43	37.07	48.06	57.61
11:50	85.8	15.6	1339.77	4.47	15.20	-14.73	-6.88	9.29	31.32	36.95	47.90	57.42
12:00	81.1	18.2	1314.97	4.39	15.12	-14.76	-6.92	9.21	31.13	36.74	47.63	57.10
12:10	76.1	20.4	1281.22	4.27	15.00	-14.81	-6.98	9.10	30.88	36.45	47.26	56.67
12:20	70.8	22.1	1239.24	4.13	14.86	-14.87	-7.06	8.96	30.57	36.09	46.81	56.13
12:30	65.3	23.4	1188.49	3.96	14.69	-14.94	-7.15	8.79	30.19	35.65	46.26	55.48
12:40	59.7	24.0	1132.54	3.78	14.51	-15.02	-7.25	8.60	29.78	35.18	45.65	54.76
12:50	54.1	24.0	1071.43	3.57	14.30	-15.10	-7.36	8.39	29.33	34.65	44.99	53.98
13:00	48.7	23.4	1007.62	3.36	14.09	-15.19	-7.47	8.18	28.85	34.11	44.30	53.16
13:10	43.4	22.2	939.88	3.14	13.87	-15.28	-7.59	7.95	28.35	33.53	43.56	52.29
13:20	38.5	20.5	872.10	2.91	13.64	-15.37	-7.71	7.72	27.85	32.95	42.83	51.42
13:30	34.1	18.2	807.53	2.69	13.42	-15.46	-7.83	7.51	27.37	32.40	42.13	50.60
13:40	30.1	15.4	744.59	2.48	13.21	-15.55	-7.94	7.30	26.90	31.86	41.44	49.79
13:50	26.6	12.3	685.33	2.29	13.02	-15.63	-8.04	7.10	26.46	31.35	40.80	49.03
14:00	23.6	8.9	631.19	2.11	12.84	-15.71	-8.14	6.92	26.06	30.89	40.22	48.34

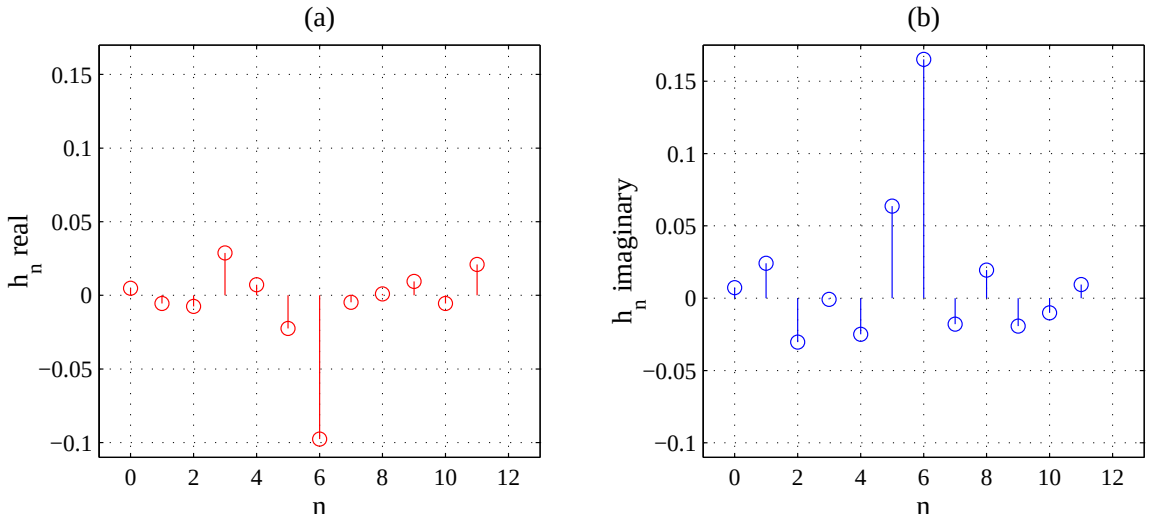


Figure 4.15: An instantaneous sample of the LMS adaptive filter vector, $\mathbf{h}_n$: (a) real taps, (b) imaginary taps. Note in Table 4.5 that for a filter order of 12 (with a corresponding bandwidth of 1.00625 MHz) at 12:20 p.m., the reference channel should be delayed by approximately 9 samples.

the center. Because correlation tends to decrease with increasing lag, this will ensure that the highest correlated samples are being filtered, creating the best available $\hat{i}_p[n]$ for subtraction from $\alpha[n]$.

4.7.4 Green Bank Spectral Processing

In order to properly cancel an interferer, the GBT and 3.6 meter reference signals must to be mixed down to frequencies usable by the DSP. The DSP 6216 digital receivers have an adjustable sample rate of up to 65 MHz; additionally, they have built-in low-pass anti-aliasing filters for each channel with a pass-band cutoff frequency of 25 MHz. Therefore, the IF entering the DSP must be between 1 and 24 MHz.

The Green Bank receiver is a highly complex and customizable system. Since only one polarization of the GBT signal was necessary for my tests, we inserted the reference signal into an alternate polarization receiver path. Hence, both the primary and reference channels passed through identical GBT receiver system channels.

Figure 4.16: The GBT receiver system. Upper Left: Andrew Poulsen holding the 3.6 meter signal cable as it enters the GBT receiver converter racks. Upper Right: All local oscillators are synchronized with a reference signal generated by this hydrogen maser rack. Lower Left: Output of the spectral processor. The GBT signal is carried by the right cable and the reference signal by the left. Lower Right: After leaving the Spectral Processor, the signals directly enter the DSP.

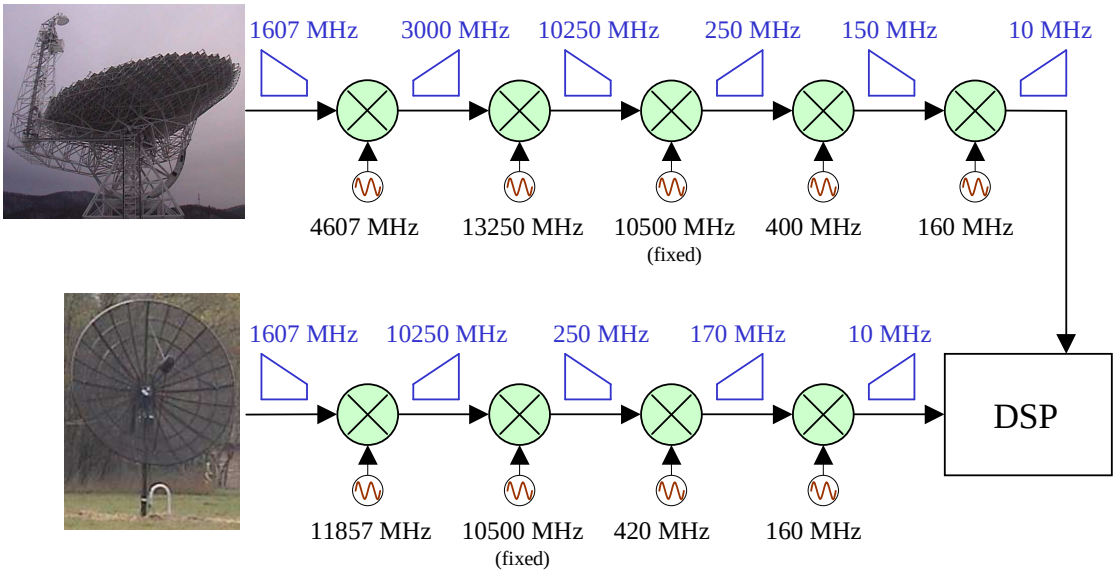


Figure 4.17: Simplified diagram of a typical setup of the GBT receiver system. Details such as filters, amplifiers and attenuators have been omitted. The center frequency and spectrum orientation for the primary and reference signals after each mixing stage are displayed in blue. The first LO in the GBT chain is located in the GBT receiver room at the antenna, and all others in the GBT equipment room.

During the GLONASS cancellation tests, it was often desirable to change the frequencies in the final passband. In order to avoid constantly changing the GBT LO configurations (shown in Figure 4.17) between tests, I would use a constant GBT receiver setup for an extended period of time. The setup shown in the Figure 4.17 gave me an available bandwidth of 20 MHz. This bandwidth included the 1612 OH spectral lines and all available GLONASS transmission frequencies. A desired portion of this spectrum was selected by simply changing the DSP digital baseband mixer center frequency by recompiling the code. I used two configurations for all of the tests: one with a 20 MHz bandwidth, and a second with a 10 MHz bandwidth. The 10 MHz bandwidth configuration was used for tests with an initial DSP sample clock below 40 MHz to avoid aliasing and to satisfy the Nyquist criterion.

4.7.5 Test Preparation

Due to several competing constraints, scheduling the GBT and preparing for these tests proved to be quite complex. Most difficulty involved finding an astronomical source that would place an OH spectral line in the same passband as a GLONASS interferer.

Each GLONASS satellite is assigned one of 24 possible frequency channels from 1602-1615 MHz. When planning the tests, there was one malfunctioning satellite and seven fully operational satellites (for the Russian navigational system to be fully operational, there should be 24). At the time of the tests, none of the GLONASS frequency channels were centered directly over the 1612.231 MHz OH maser line. In fact, the highest carrier frequency available was 1608.75 MHz. The spectrum of this GLONASS signal does spill-over into the 1612 MHz OH maser line, but at a significantly lower power than in the spectral main beam (see Figure 4.18).

Hydroxyl (OH) masers typically contain double spectral lines due to expanding shells from OH stars [1]. The lower line is referred to as the red-shifted peak, while the upper is the blue-shifted peak. Not only are the spectral peaks affected by the expanding shells, but the OH stars are also moving relative to the observer, causing an additional Doppler effect.

The highest frequency GLONASS satellite available (#789) was centered at 1608.75 MHz, while highly red-shifted OH sources are relatively rare and low in power. I searched several different OH catalogs looking for highly red-shifted sources (see Appendix C.3) coincidently above the horizon with GLONASS 789. I found quite a few highly red-shifted sources, but many were typically above the horizon when the satellite was not. Most of these sources are found at right ascensions of 17-18 hours, while GLONASS 789 is typically above the horizon at times mutually exclusive to these sources. In addition, we had to accommodate additional constraints such as available trip schedules and very limited available GBT observation time windows. It is ironic that it was very difficult to find a significant GLONASS spectrum in the same processing bandwidth as an OH maser source, a case of the high priority (to astronomers) interference scenario we are trying to address. It is important to note,

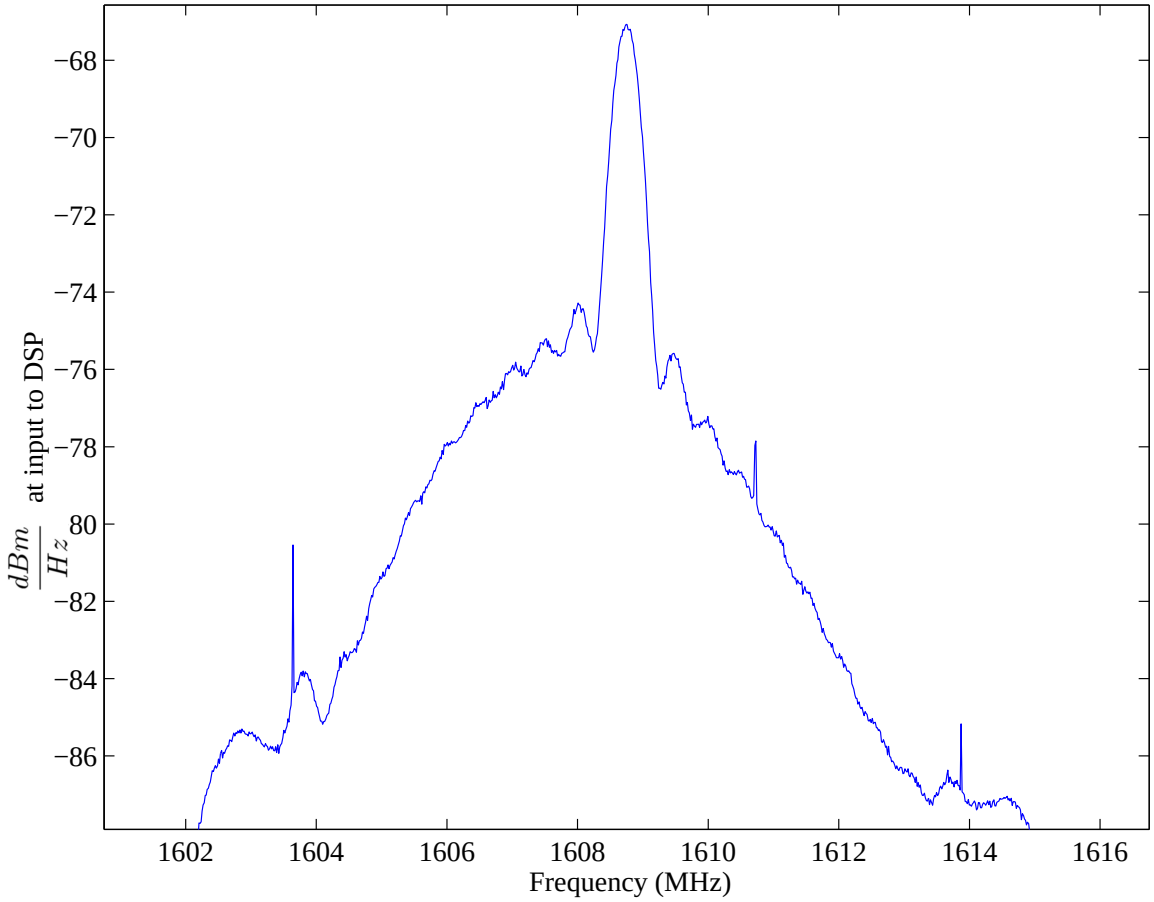


Figure 4.18: The GLONASS 789 spectrum. During test periods both in October 2002 and January–February 2003, this satellite had the highest carrier frequency of all available GLONASS satellites. Notice that the signal power is significantly lower at the OH 1612.231 MHz line. This spectrum was estimated using 0.25 seconds of data collected with the VSA.

however, that the problem of satellite downlink signals corrupting radio astronomy data exists in other frequency bands with satellites other than GLONASS.

There are, however, infrequent periods of time when both the satellite and a few prime sources are simultaneously above the horizon. In order to schedule such times with the GBT, I compiled calendars detailing when GLONASS 789 was above the horizon. These calendars, along with corresponding GBT schedules, are found in Appendix C.1. Initially I had planned to visit Green Bank in November or December 2002, but due to scheduling difficulties the tests were postponed. Ultimately, I was able to schedule tests for January 30–February 1, 2003.

4.7.6 GBT Test Problems

After several months of planning and preparation for the GBT tests, the time finally arrived for executing the tests. I had done everything I could to prepare for a successful run of tests. As typically is the case, a few problems arose at the last moment: problems with both the 3.6 meter reference antenna and the GBT.

A critical weld on the mount of the 3.6 meter antenna we had set up in October 2002 broke under the stress of high wind and excessive snow built up in the reflector. The dish swivelled into the ground, severely damaging a couple of the reflector panels and one of the feed support poles. Even with the weld repaired, the reflector was severely distorted and the feed off-center, producing a degraded reference signal.

I arrived in Green Bank approximately two days before the first run of interference cancellation experiments. After repairing the antenna, we recalibrated the antenna for pointing precision. We found, however, that a much larger slack existed in the elevation gears and other related connections than before the mount weld failed. This "play" in the elevation axis was approximately 6° to 8°. Due to the limited time remaining before the scheduled experiments, we were unable to implement a permanent solution.

The dual axis azimuth/elevation rotors provide almost full azimuth coverage ($\approx 97\%$). This is accomplished by flipping over the antenna in elevation to reach the two azimuth hemispheres. In other words, for a constant degree in elevation, the

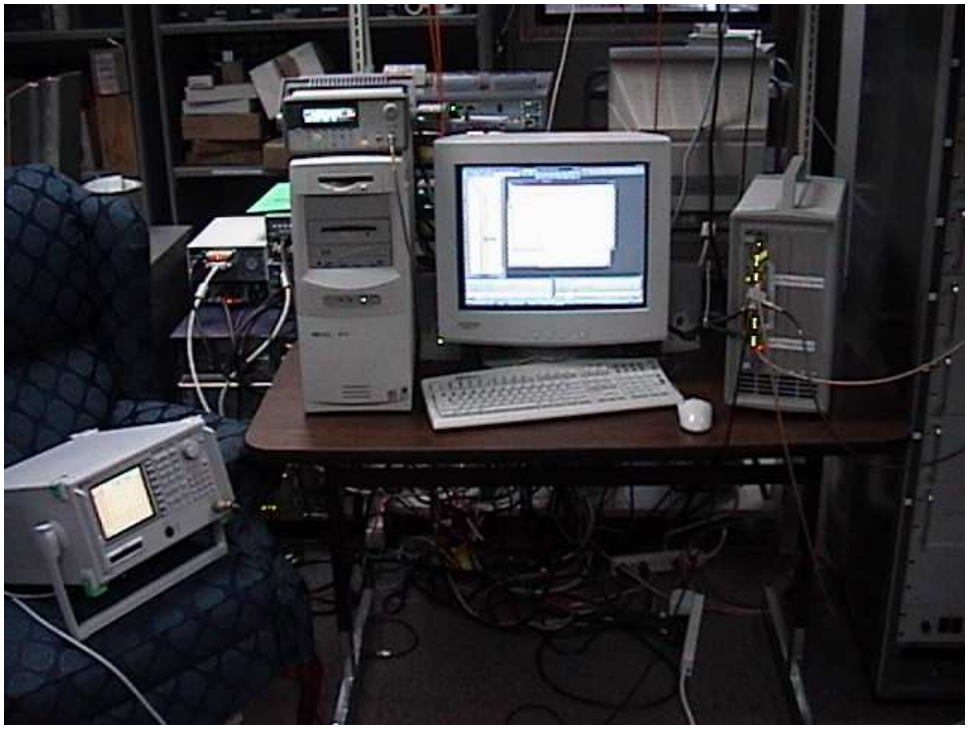


Figure 4.19: The DSP computer setup at Green Bank, WV. The spectrum analyzer on the chair at the left gave real-time analysis of the quality of the GLONASS reference signal. When the power received began to drop, I was able to recalibrate the reference antenna in order to maintain a consistently strong reference signal for the interference cancellation experiments. A power divider at the input to the DSP (at the right) sent a copy of the 3.6 meter signal to the spectrum analyzer. The signal generator sitting on top of the PC chassis served as the external DSP sample clock.

hemisphere spanning from 90° to 270° in azimuth can be reached without moving the elevation rotor. The companion hemisphere (270° to 90°) is reached by flipping the antenna over in elevation. When tracking GLONASS satellites, the excessive slack in the elevation axis severely degraded the interference reference signal. When in opposite azimuth hemispheres, gravity would pull on the mass of the antenna and force it to either limit of the 6° to 8° slack range. When calibrated to one hemisphere, the received GLONASS signal would drop significantly when observing in the opposite hemisphere. As a quick solution, I measured two different pointing calibration settings for each azimuth hemisphere. These calibrations were fine-tuned by tracking several different GLONASS satellites and sweeping the antenna in elevation while observing the received power on a spectrum analyzer. The offset between observed peak and desired pointing direction was recorded and used as a pointing bias correction. In this manner, I was able to consistently obtain a quality GLONASS reference signal.

In addition to the reference antenna difficulties, it was learned on arrival at Green Bank that the test scheduled for the afternoon of Thursday, January 30, 2003 had been cancelled due to inspections which needed to be made on the GBT azimuth track. We would have access to the GBT signal and electronics, but would be unable to steer the GBT. However, we were able to perform some positioning tests during the Thursday morning scheduled time (see GBT OH observations in Figure 3.9).

Later Thursday, inspectors found a large cracked azimuth track wear plate. The crack penetrated 75% of the plate thickness. Due to its severity and position in the plate, if it had completely cracked through, a piece of the track could have broken free causing untold damage to the GBT. Fixing this particular crack required additional tooling, which was unavailable until after all of my scheduled tests. As a result, I was unable to position the GBT during the remainder of my GLONASS cancellation experiments. Additional details are found in Appendix C.2.

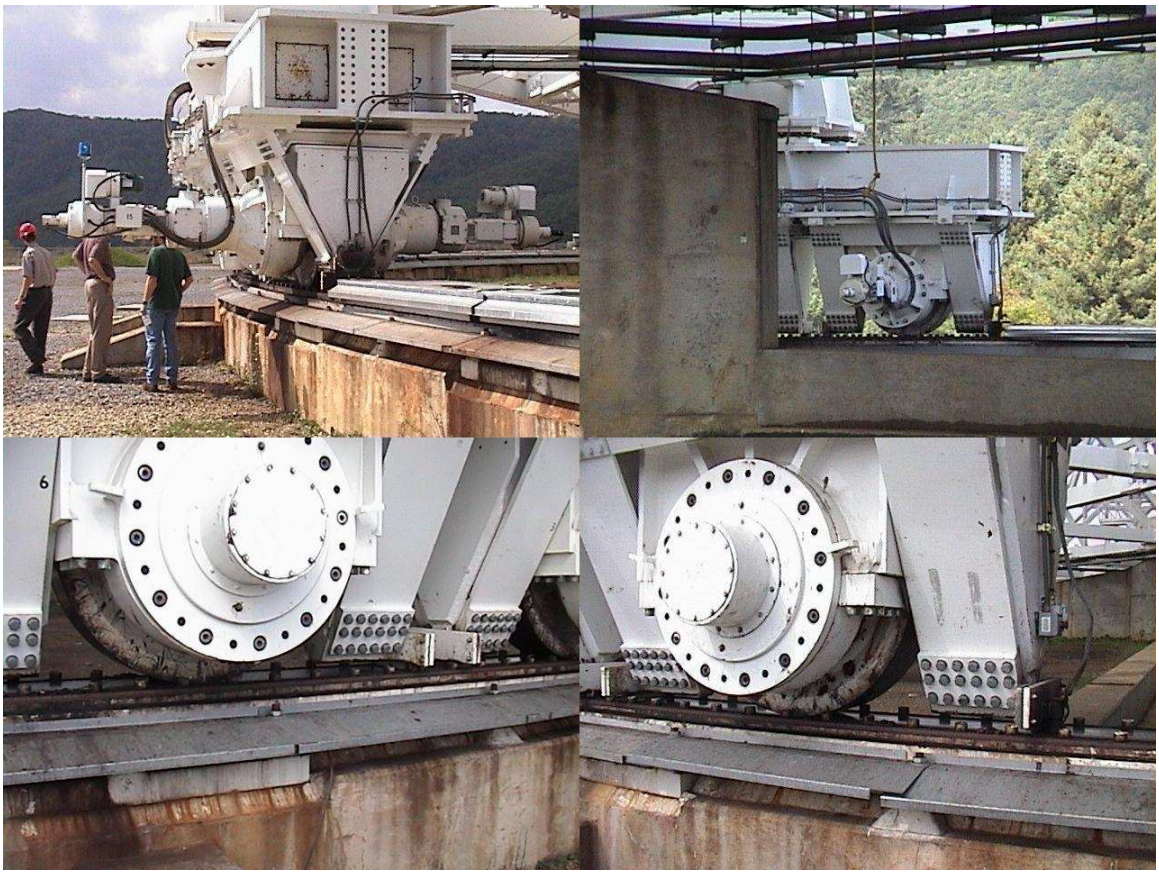


Figure 4.20: The GBT azimuth track. Due to cracks found in the track wear plate, I was unable to steer the GBT during all but the first scheduled block of time early Thursday, January 30, 2003.

4.7.7 GBT Results

Despite being unable to steer the Green Bank Telescope, I was able to run some convincing GLONASS cancellation experiments by injecting a low power simulated astronomical source at the feed of the GBT. Significant interference cancellation was demonstrated while preserving the desired signal. A remotely controlled signal generator in the GBT receiver room provided the "fake" source.

Magnitude of Cancellation

When analyzing interference cancellation algorithms, it is desirable to obtain an estimate of the achieved cancellation depth. Ideally, this would be measured as an attenuation in dB. Unfortunately, once the interference is pushed imperceptibly below the noise floor for a given integration time, the final interference amplitude is unknown. In many cases, both the desired signal and interferer are only detected upon integration; hence, both may lie well below the noise floor. The good news, however, is that as long as the attenuated interference does not emerge above the baseline noise variance for the observation integration time, the final interference amplitude is not needed. In other words, an astronomer will not care about the residual interference if he cannot detect it.

Successful observations depend on driving the variance of the noise floor below the baseline perturbation introduced by the desired signal. Consequently, if the baseline variance can be reduced sufficiently to reveal the desired spectrum absent of any corruption due to interference, the observation is a success. As a result, an appropriate analysis of magnitude of cancellation involves demonstration that the variance of the baseline continues to decrease with integration, as predicted by theory.

As described in Section 3.1, the variance of the PSD estimation error is theoretically inversely proportional to integration time. Similarly, the error standard deviation, which determines the detection threshold for desired signal power, is inversely proportional to the square root of the integration time.

After filtering the primary channel, it is important to verify proper decrease in PSD error variance with integration. As a reference, an interference-free 40 minute

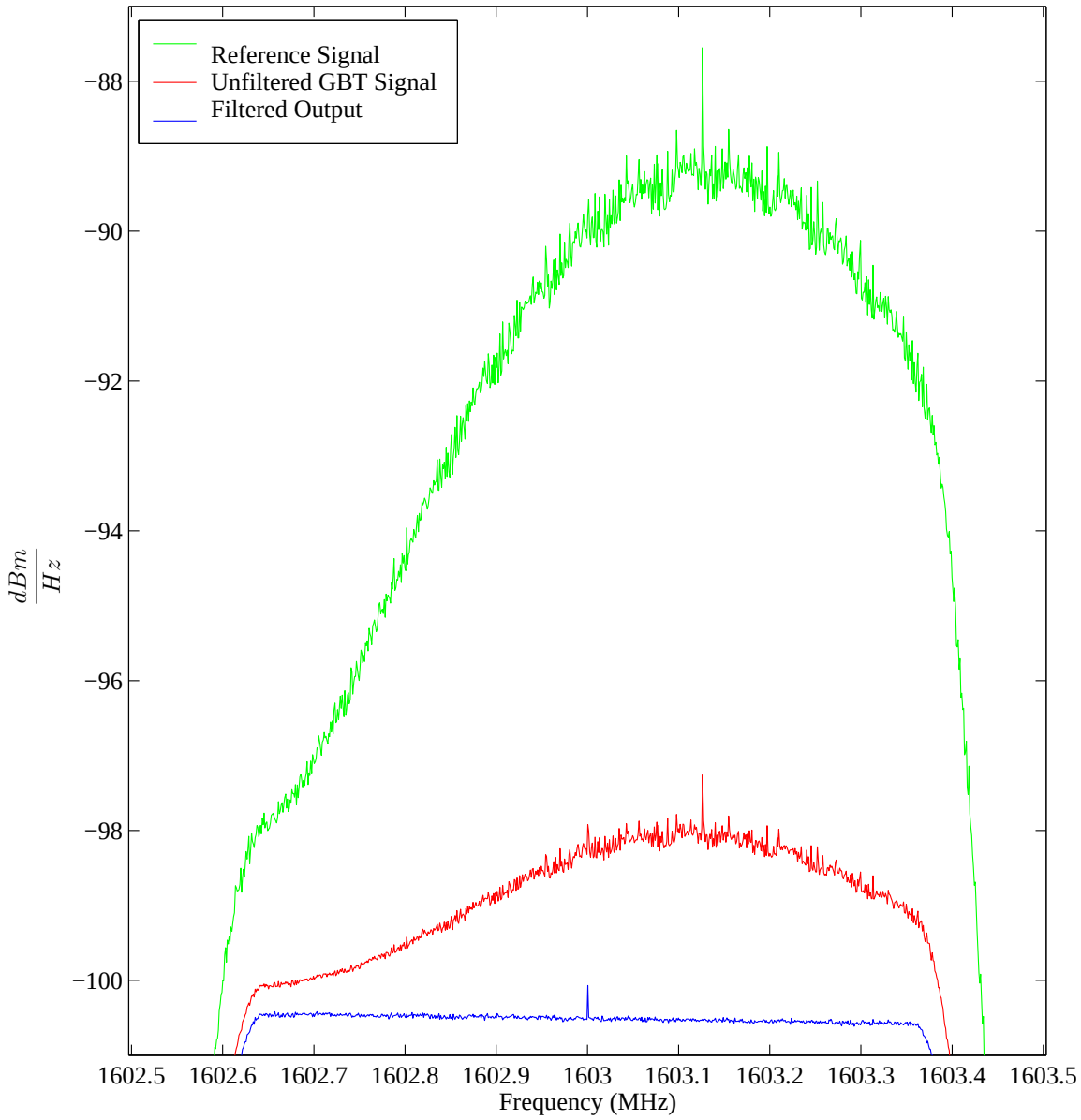


Figure 4.21: The GBT tests demonstrated significant cancellation of GLONASS interference. Due to an inability to steer the GBT to an astronomical source, a "fake" source was injected at the feed, in this case at 1603 MHz.

reference data set was collected using the GBT and corresponding receiver system used for all interference mitigation tests (see Figure 3.11 and Section 3.1.5). The standard deviation of this interference-free, unfiltered data is included as a reference in those plots in this section illustrating decline of variance with integration time. In all cases, standard deviation is shown instead of variance due to the physical meaning of units of power.

Long Integration

In some cases, longer integration is required in order to detect especially weak signals buried deep below the noise floor. Of course, it would be especially desirable to make such observations in the absence of a man-made interferer, but this is often impossible. While at Green Bank, I was able to run a few long integration tests to evaluate filter stability and robustness.

Six of these long integration tests are shown in Figure 4.22. These experiments represent a wide range of interference scenarios. Tests (a) and (b) contained, on average, a relatively low level of interference, often well below the noise floor, with sporadic spurts of higher interference. Data sets (c) and (f) contained average interference levels of up to 1 dB above the integrated noise floor. The remaining two data sets, (d) and (e), contained average interference levels of approximately 3.5 and 13 dB above the integrated noise floor, respectively.

Understandably, those data sets with noticeable residual interference also contained the highest levels of interference. Note in the spectral plots of Figure 4.22, that both (d) and (e) have visible baseline perturbation due to GLONASS interference. Test (e) is the most apparent, while the residual noise in test (d) reveals itself through a noisier estimate of the baseline for those frequency bins coincident with the highest levels of the GLONASS spectral main lobe.

In addition to visual inspection of the PSD plots, additional insight is found by estimating the decline of variance with signal integration time. Figure 4.23 illustrates the standard deviation and magnitude of cancellation for each of the long integration tests shown in Figure 4.22. In each case, the standard deviation was estimated with

frequency bins containing significant levels of interference previous to filtering. In other words, I did not hand pick the "prettiest" regions of the output spectrum to compute the standard deviation.

Standard deviation is proportional to the power in each frequency bin. Since the data sets were taken in widely varying scenarios, the initial standard deviation estimate differs for each case. This is due to changes in the system gain, PSD bin widths, etc.

Note in Figure 4.23 that the output variance of most tests declined linearly with integration time with a slope similar to that of the unfiltered interference-free reference data set. In these cases, the GLONASS interference has been pushed imperceptibly below the integrated noise floor. On the other hand, some residual interference is evident for tests (d) and (e). As expected, the variance drops somewhat linearly with integration until the interference becomes the dominant portion of the baseline variance. At this point, the variance either levels out or decreases at a slower rate.

In cases where the variance levels out, it is possible to estimate the magnitude of cancellation in dB. In other cases where the interference remains buried in the noise, a lower bound on the achieved dB attenuation can be estimated with the available data. These are obtained by taking the ratio of the minimum integrated standard deviation of the output obtained through integration, $\min(\hat{\sigma}_{\hat{d}})$, to the average deviation before filtering, $\text{ave}(\hat{\sigma}_{\alpha})$ (the average interference power encountered during the test),

$$\text{Attenuation} \approx 10 \log \left[\frac{\min(\hat{\sigma}_{\hat{d}})}{\text{ave}(\hat{\sigma}_{\alpha})} \right]. \tag{4.25}$$

Attenuation was calculated for tests (a)–(f) highlighted in Figures 4.22 and 4.23. The results are found in Table 4.6. The variance did not level out or significantly decrease in slope for tests (a), (b), (c) and (f); therefore, the attenuation for these tests is most likely higher than those listed. Due to the wide range of interference levels represented by these tests, it is reasonable to conclude that at least 30 dB of attenuation is a realistically achievable magnitude of cancellation for many test scenarios.

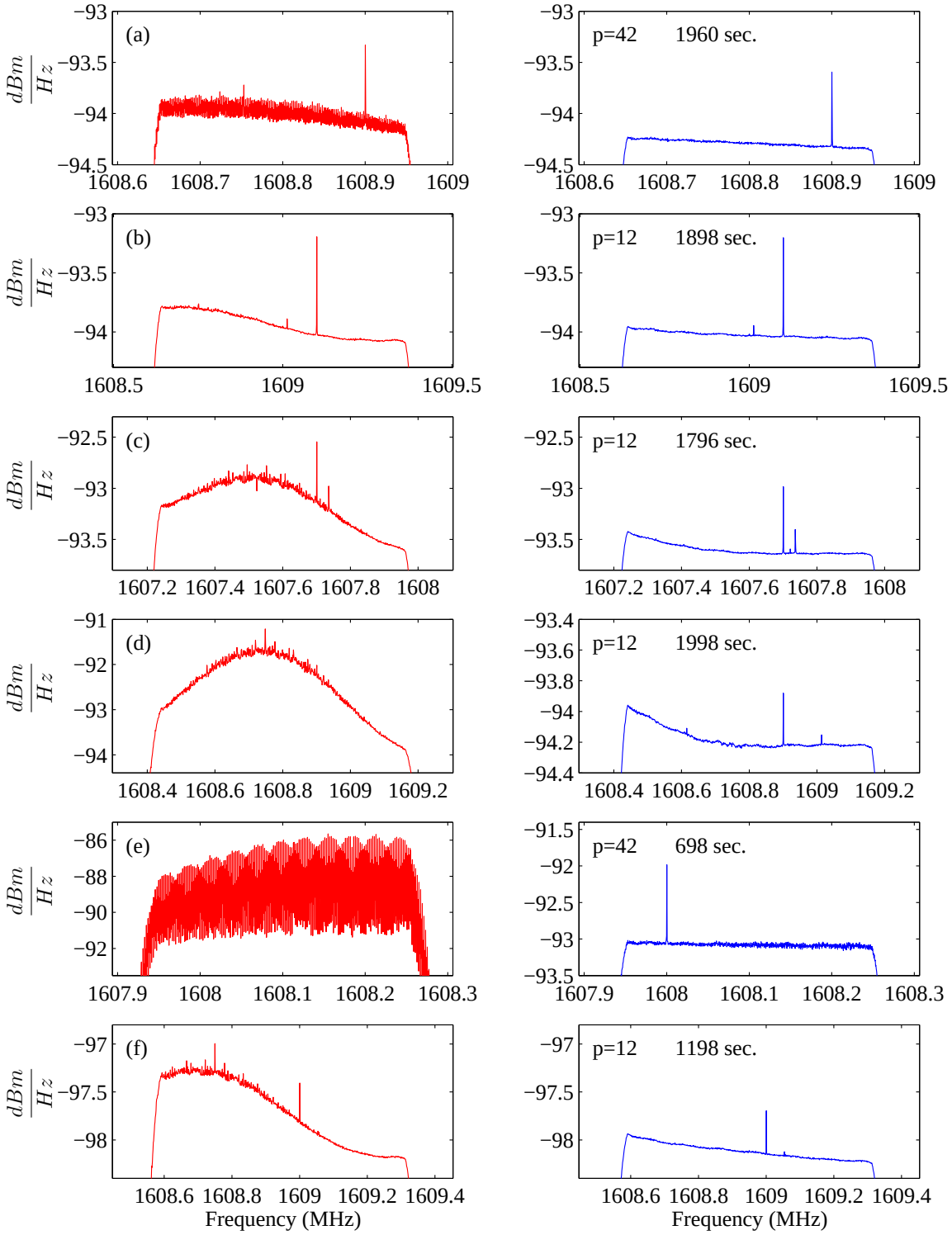


Figure 4.22: GBT signals both before (red) and after (blue) filtering. Each test contains a test tone injected at the feed of the GBT. Included are some of the longest integration data sets collected. All powers are at the input to the DSP platform.

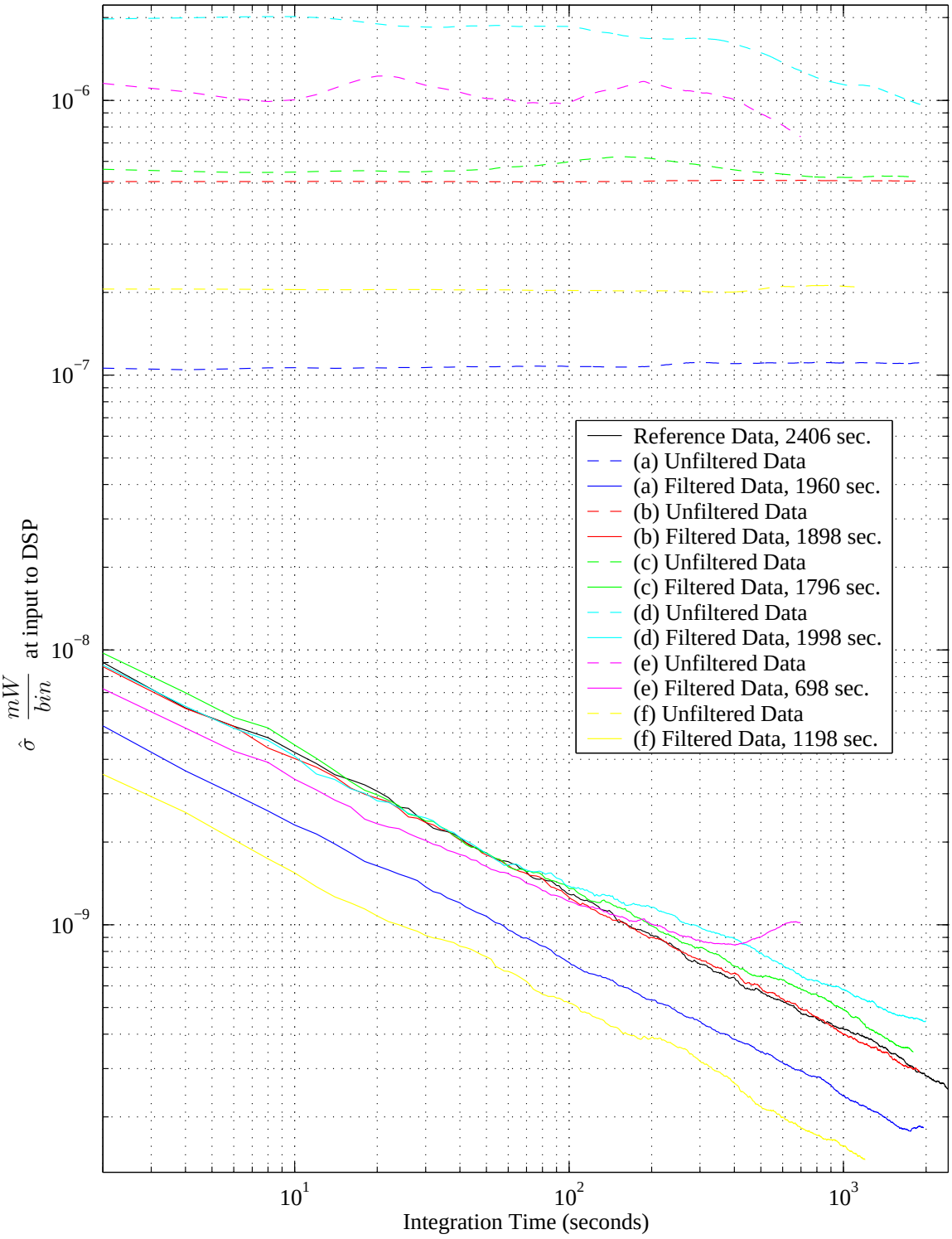


Figure 4.23: Estimates of standard deviation with integration time for those tests shown in Figure 4.22. Residual interference is evident in tests (d) and (e) as the standard deviation either levels out or decreases at a slower rate. The test in black is a 40 minute interference-free sample collected with the GBT (see Figure 3.11).

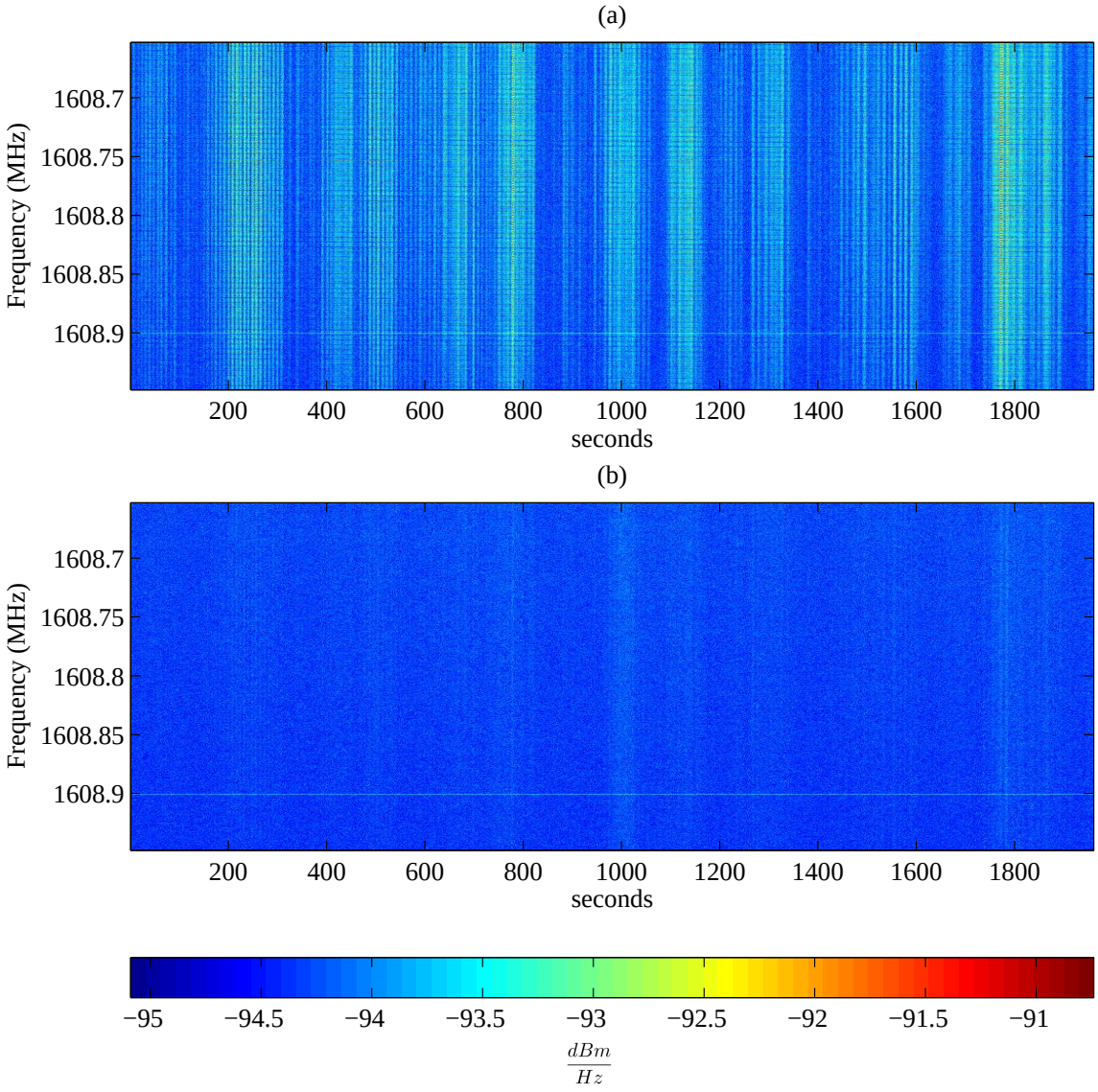


Figure 4.24: These images are an illustration of the time-varying nature of the (a) spectrum before filtering, and (b) spectrum after filtering. They refer to test (a) in Figure 4.22 (without cumulative integration).

Table 4.6: Additional details for those tests highlighted in Figures 4.22 and 4.23. The minimum attenuations listed in the final column were calculated using Eq. 4.25.

Test	p	Bandwidth	Integration Time	Number Bins Used to Estimate $\hat{\sigma}$	Frequency Range Used to Estimate $\hat{\sigma}$	Minimum Attenuation
(a)	42	0.4125 MHz	1960 sec.	541	1608.66–1608.88 MHz	-28.0 dB
(b)	12	1.00625 MHz	1898 sec.	301	1608.69–1608.99 MHz	-32.4 dB
(c)	12	1.00625 MHz	1796 sec.	351	1607.29–1607.64 MHz	-31.8 dB
(d)	12	1.00625 MHz	1998 sec.	201	1608.64–1608.84 MHz	-33.3 dB
(e)	42	0.4125 MHz	698 sec.	401	1608.05–1608.22 MHz	-29.4 dB
(f)	12	1.00625 MHz	1198 sec.	301	1608.64–1608.94 MHz	-31.8 dB

Filter Length Analysis

An important consideration for canceller performance and computational burden is the minimum filter length required for effective cancellation. The "best" length depends on many factors. These include, but are not limited to the presence of multipath, size of the baseline between channels, data alignment accuracy, bandwidth of the interferer, and stationarity.

With limited time allotted for real-time GBT tests and sporadic availability of interference (GLONASS satellites), exhaustive tests were not possible. I did, however, run two sets of experiments by systematically decreasing the filter length while leaving other parameters constant (when possible). As all experiments were run in real-time, each test featured a distinct data set. In order to minimize the effect of non-stationary signals, each series of tests was collected as close together in time as was possible.

The first series of tests has a complex sample rate/signal bandwidth of 0.4125 MHz, 5 minutes of integration, and various filter orders from 42 down to 2 taps. These results are displayed in Figures 4.25, 4.26 and 4.27. Each of the filter lengths tested suppressed the interference to undetectable levels after 5 minutes of integration. Discrepancies for the smaller filter orders may have been evident with longer integration. For filter orders of 2 to 5, the adaptation constant was raised from $\mu = 1 \times 10^{-9}$ to $\mu = 1 \times 10^{-8}$ to insure convergence while tracking signal non-stationarities.

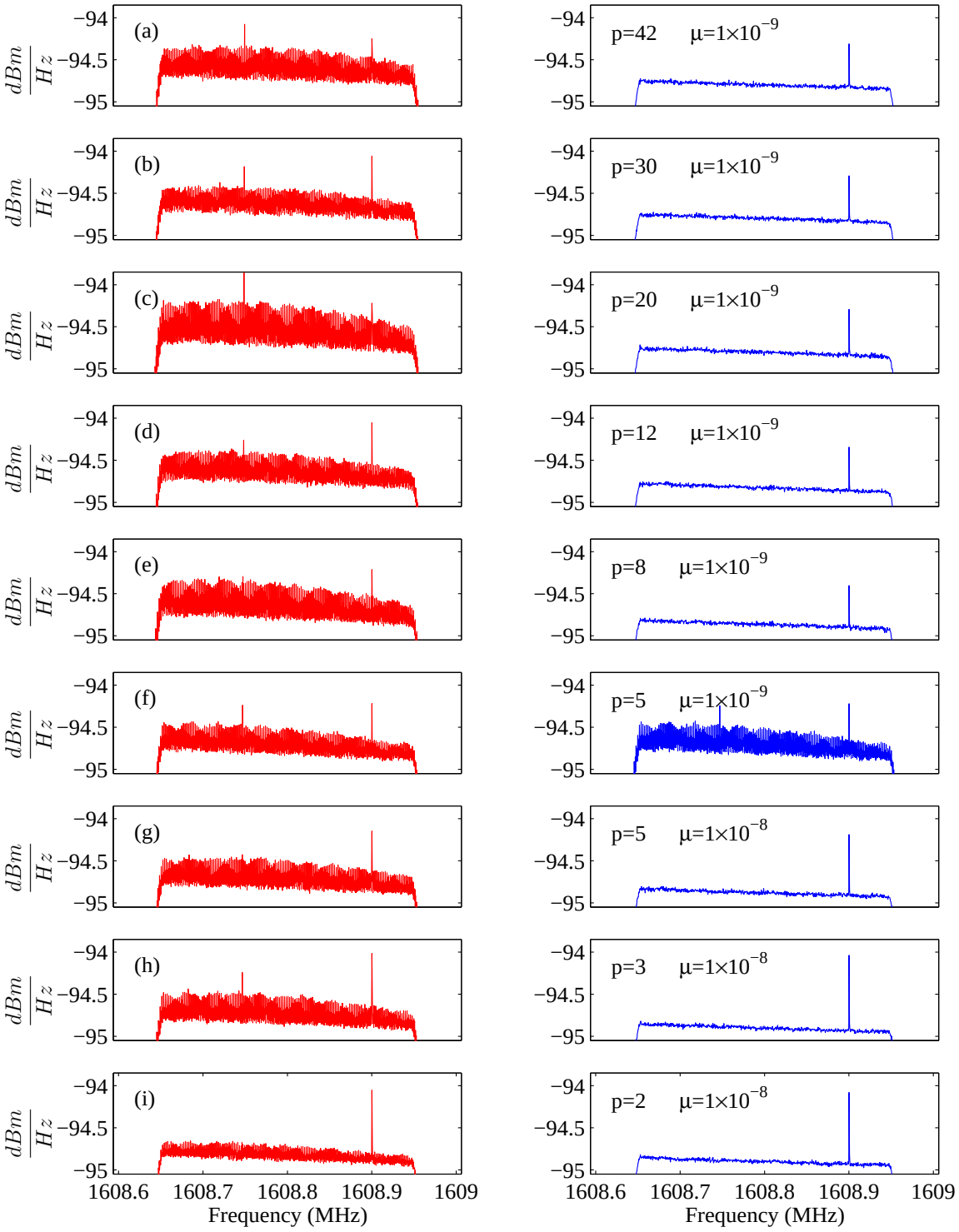


Figure 4.25: GBT signals both before (red) and after (blue) filtering. Each real-time cancellation test represents 298 seconds of integration and a bandwidth of 0.4125 MHz. All tests were run consecutively in order to maintain test continuity. The smallest filter orders required a larger adaptation constant in order to track the dynamic system.

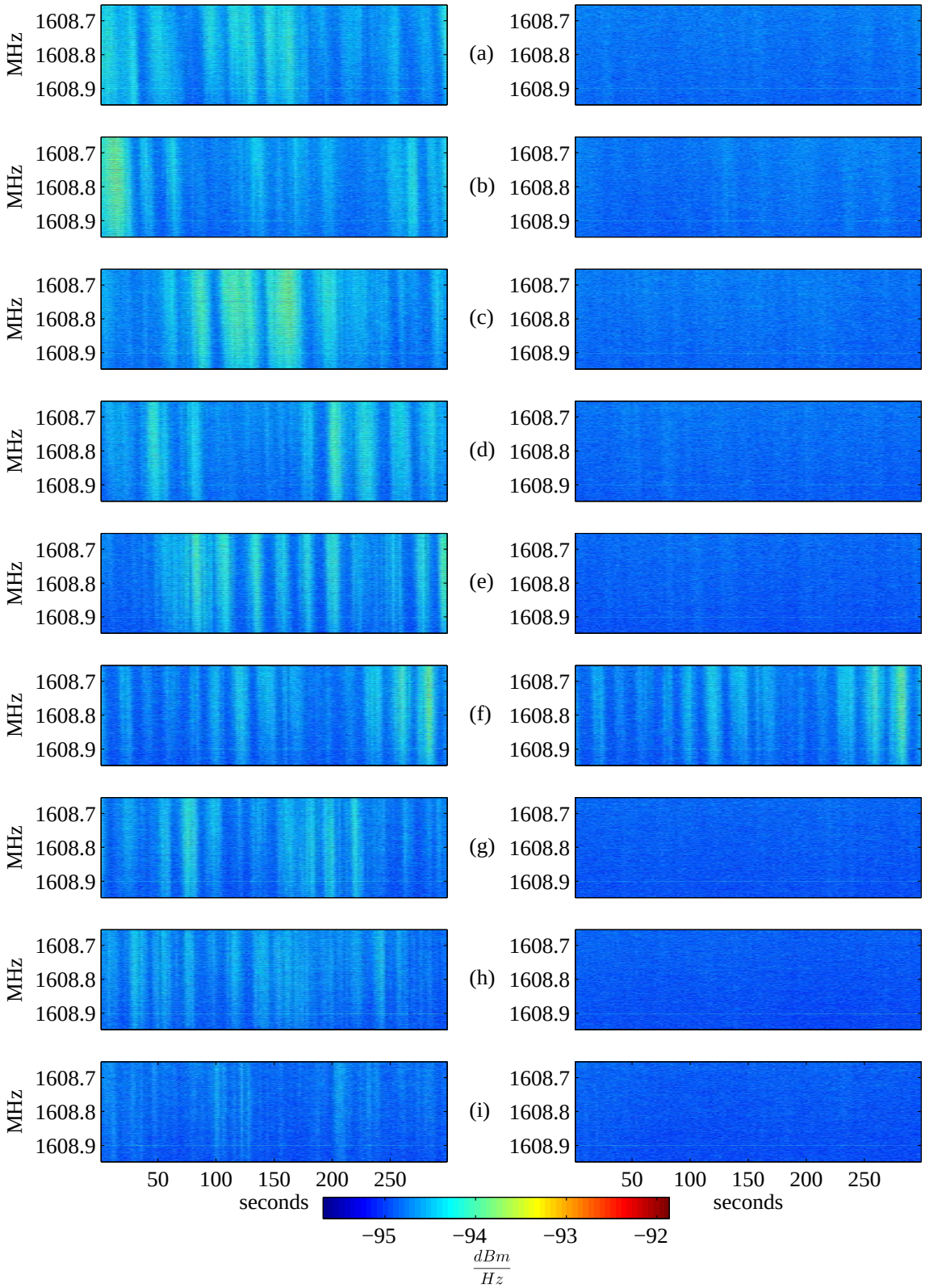


Figure 4.26: Images displaying the time-varying nature of the signal input and output for those tests shown in Figure 4.25.

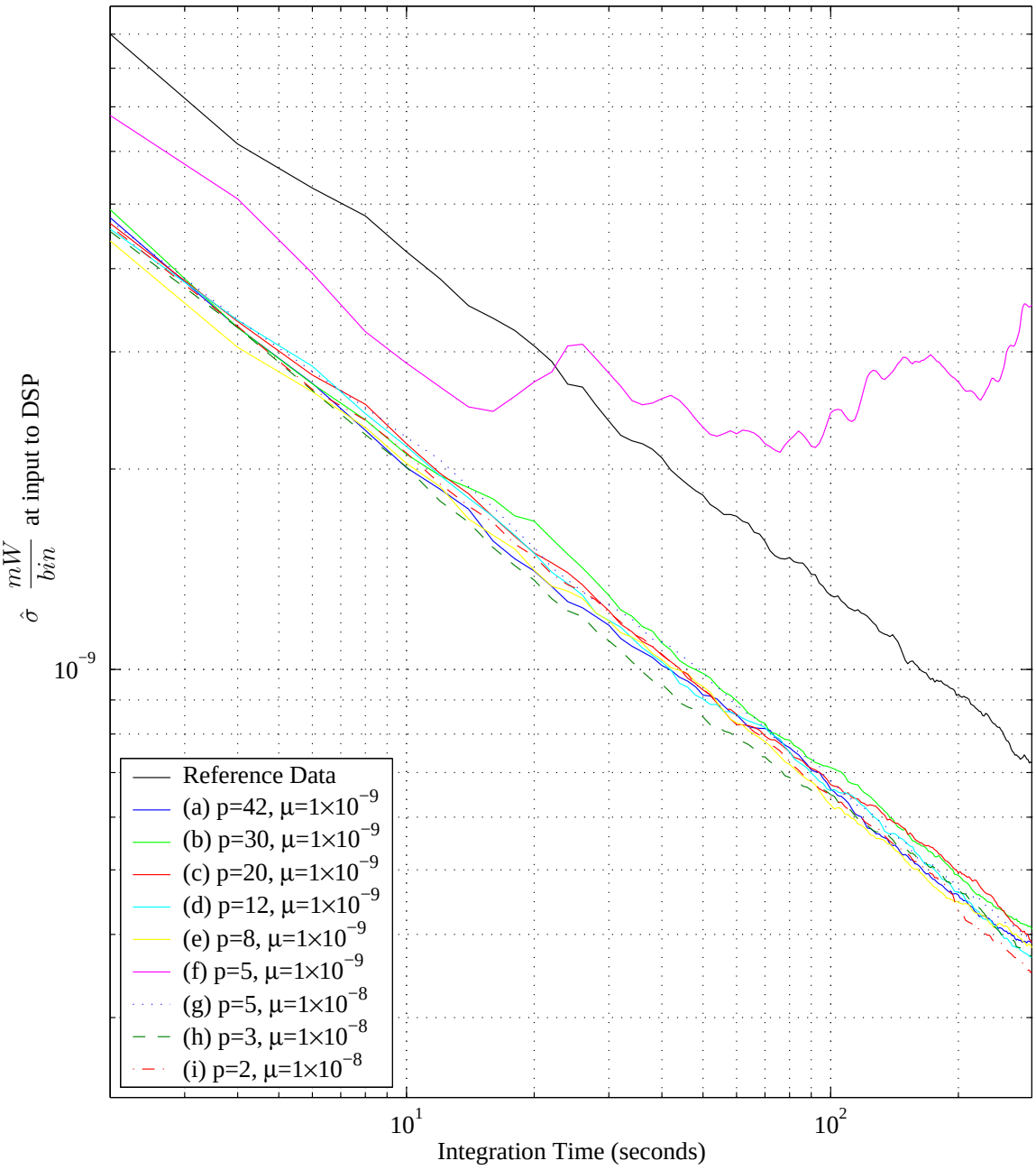


Figure 4.27: Estimates of standard deviation with integration time for those tests shown in Figure 4.25. All tests declined at the appropriate slope except for test (f), where the adaptation constant was too small and cancellation did not occur.

The second series of tests has a complex sample rate/signal bandwidth of 1.00625 MHz, 1 minute of integration, and various filter orders from 12 down to 3 taps. These results are displayed in Figures 4.28 and 4.29. In all tests, the interferer was GLONASS 787 with a center frequency of 1604.8125 MHz. The downlink signal from this satellite was cancelled in all cases.

Unfortunately, there were several other GLONASS satellites simultaneously above the horizon along with GLONASS 787. Consequently, a downlink signal with a higher carrier frequency increasingly entered the passband of this second series of tests. The baseline slowly increased in power at the upper frequencies for each of the tests, as seen in Figures 4.28 and 4.29. The scenario of multiple satellite interferers is discussed below under *Presence of Multiple Interferers*. As was the case previously, the adaptive constant μ needed to be increased for the smallest filter orders.

As demonstrated, successful cancellation can occur with even the smallest filter lengths. There are, however, a few significant drawbacks to using small filter lengths. First, even tiny errors in data realignment can prevent interference attenuation. With longer filter lengths, one can place the highest correlated sample of the reference signal, $x[n]$, a few lags from the center of the filter and still not overstep the end of the finite length filter. In most cases, especially with broadband interferers, it is imperative that the highest correlated samples be included in the span of the filter. Additionally, as the sample rate increases, adjacent samples become temporally tightly packed making realignment increasingly difficult. I ran a number of tests with larger bandwidths, 3.25 and 3.84375 MHz, with 3 and 2 taps, respectively. Conclusive cancellation was difficult to show in these scenarios, as was finding a stable adaptive constant, μ (see Figure C.1).

In general, any large differences that exist between the interference incident to the primary channel, $i_p[n]$, and reference channel, $i_r[n]$, will require a larger filter order. These differences could include frequency dependent gain and/or phase differences in the primary and reference channels, multipath structure, frequency misalignment between channels resulting from improper synchronization of LOs (local oscillators), the presence of other interferers, etc. The adaptive filter tracks any

differences in order to create $\hat{i}_p[n]$ from $i_r[n]$. Often, several taps are required to adequately model contrasting channels. In the special case where the only differences between $i_p[n]$ and $i_r[n]$ are magnitude and phase, one complex tap would be sufficient, assuming the signals are properly aligned. Unfortunately, this is never precisely the case in a real test environment.

Channels with high multipath structure present additional problems for shorter filter lengths. The filter must subtract multiple instances of the interference with potentially very different statistical properties, possibly requiring a very long adaptive filter. Fortunately very little multipath structure is typically part of the channel from a satellite to a radio telescope. Multipath environments are more commonly encountered with ground based signals such as television, radio, and radar transmissions.

12 complex taps seemed sufficient for any GBT test scenario encountered, at least for bandwidths for which the computational power supported 12 taps. The filter order of 42 used for smaller bandwidths proved to be somewhat of an overkill; seemingly equivalent cancellation was obtainable with shorter filter lengths.

In conclusion, there is an unfortunate dilemma of excess available computing power for small bandwidths and a lack thereof for larger bandwidths. Table 4.2 illustrates this fact. For a bandwidths of 0.4125 MHz and 3.84375 MHz, the highest available complex filter orders were 42 and 2 taps, respectively. Ideally, the opposite would be true. For large processing bandwidths with significant channel differences, large filter orders such as 42 or larger would be ideal. With sure improvements in computing capacity in DSP platforms, this will become less of an issue for real-time interference mitigation. For a vast majority OH maser sources, a bandwidth of 1 MHz is sufficient to include both the red- and blue-shifted spectral peaks. For a 1 MHz bandwidth, 12 complex taps was sufficient for cancelling most GLONASS interference scenarios encountered with the GBT.

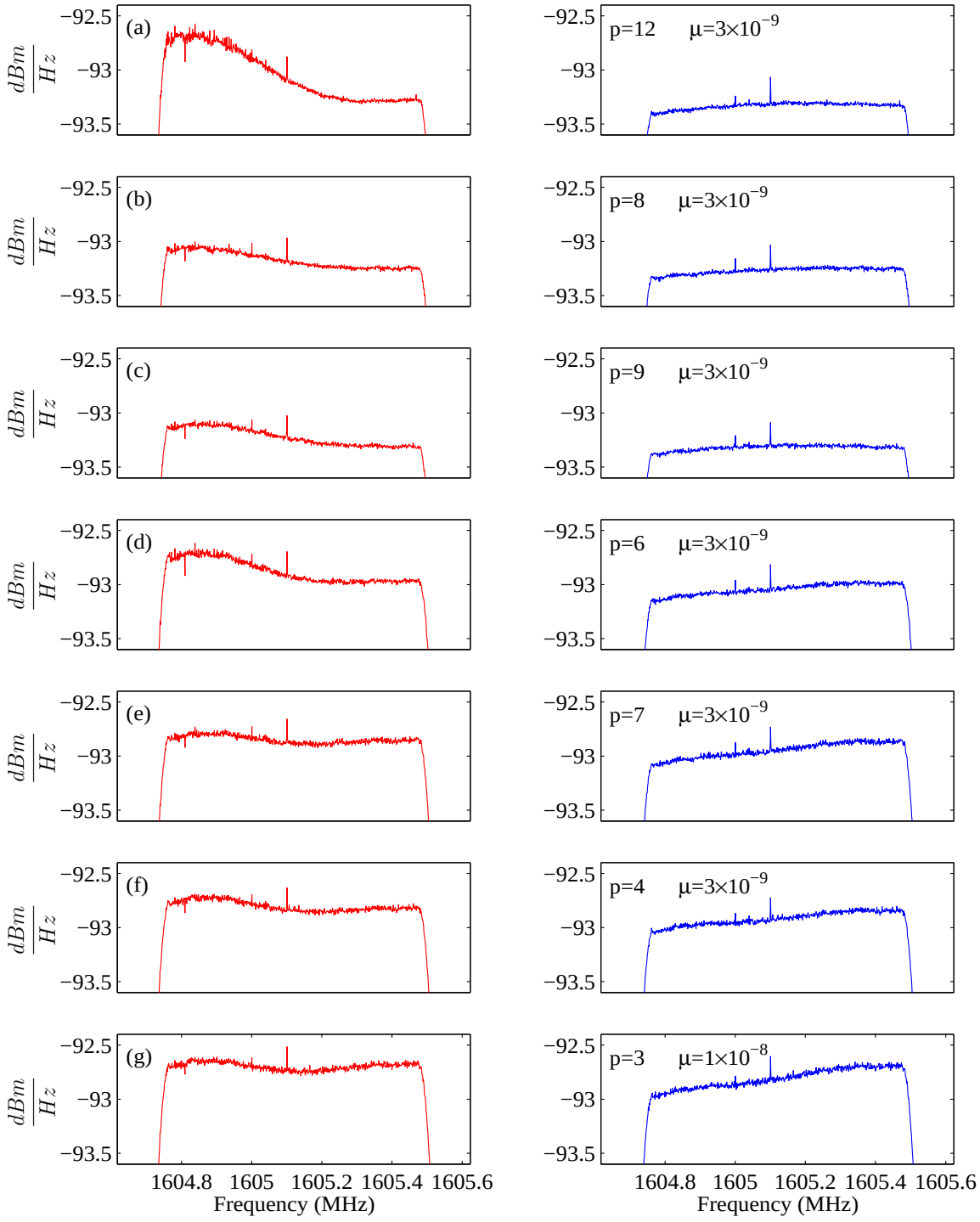


Figure 4.28: GBT signals both before (red) and after (blue) filtering. Each test represents 176 seconds of integration and a bandwidth of 1.00625 MHz. The additional power, especially in the upper frequencies, appears to be from a separate GLONASS satellite not being tracked by the 3.6 meter antenna. As a result, the power in the baseline increased from test to test as this additional satellite downlink signal entered the sidelobes of the GBT.

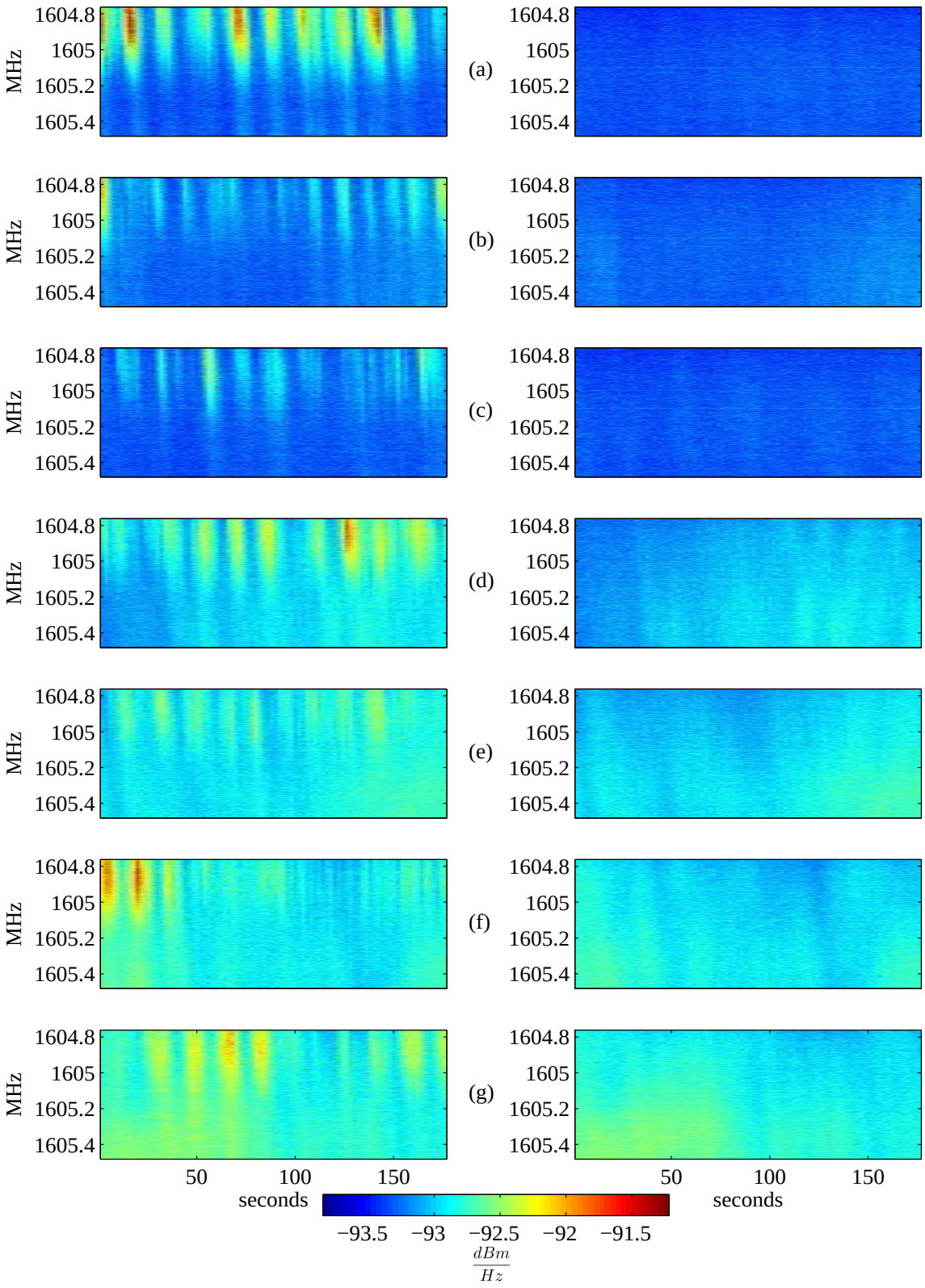


Figure 4.29: Images displaying the time-varying nature of the signal input and output for those tests shown in Figure 4.28

Selection of the Adaptive Constant, μ

Selection of the adaptive constant, μ, is an important aspect of ensuring system stability and performance. This parameter, along with input signal power, dictates the rate of convergence. In a non-stationary test scenario, it is important to track the dynamic properties of the system without sacrificing performance. A rigorous theoretical analysis of stability for the LMS algorithm can be found in many textbooks [37]. This section addresses this problem from a practical viewpoint, detailing a few lessons learned from tests performed with the VSA and GBT. Additional hints for choosing μ are found in Appendix C.5.

As stated previously in Section 4.4, the normalized LMS adaptive algorithm given by

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \frac{\mu}{\|\mathbf{x}_n\|^2} e[n]\mathbf{x}_n^* \tag{4.26}$$

is an ideal method of ensuring system stability and convergence over a wide range of test scenarios. It is immune to large changes in input signal power, as the adaptive constant in the filter update equation, $\frac{\mu}{\|\mathbf{x}_n\|^2}$, is inversely proportional to the estimate of the instantaneous signal power over the span of the finite filter length, $\|\mathbf{x}_n\|^2$.

Initially, I implemented the normalized LMS algorithm in the DSP platform, but found sufficient optimization impossible due to the additional computation required for division by $\|\mathbf{x}_n\|^2$. On the other hand, I was able to "hand tune" μ in the original LMS vector update equation (as described in Section 4.5) to achieve acceptable convergence performance. Although this algorithm does not self-adjust for large deviations in signal power, its real-time computational performance far outweighs the normalized algorithm's benefits in this application. It is important to note that of the hundreds of tests I ran with the VSA and GBT, very few resulted in unstable systems. I found the LMS adaptive filter to be very robust and dynamic in a wide range of signal scenarios.

It can however be difficult from a user's (astronomer's) viewpoint who would prefer a "turnkey" system, to strike the perfect balance between system stability, rate of convergence, and adaptation to the ever changing properties of a test environment.

Aside from the size of telescopes and differences in receiver systems, there are two important contrasting qualities between the VSA and GBT test scenarios. These include the size of the baseline and structure of the antenna sidelobes of the primary antenna through which the interfering signals pass. Obviously, the sidelobes of the 100 meter GBT maintain a much finer structure with many more nulls than those of the 3 meter VSA telescopes. The more dynamic the sidelobe structure, the more the LMS filter must track changes induced by an interferer moving across the primary sidelobes.

Surprisingly, the dominant factor causing system non-stationarity with the Green Bank test scenario was not the GBT antenna sidelobe structure. It was phase rotation due to the huge baseline between antennas. The GBT and 3.6 meter antennas are separated by approximately 1.4 km (see Figure 4.12). Figure 4.30 illustrates the worst-case relative phase rotation between the primary and reference channels due to changes in interferer direction of arrival (DOA). This simulation only takes into account phase drift due to differences in propagation delay between elements, not antenna beam pattern or system phase response issues. In the worst-case GBT test scenario, there are approximately 130 full 360° phase rotations with a one degree movement in interferer DOA. The worst-case VSA scenario is a 118° phase rotation with a one degree movement in interferer DOA. This results in a much more stationary setup at the VSA than with the GBT.

These issues raise the question of how close to place the primary and reference antennas. The fact that non-stationarity increases with the size of the baseline must be weighed against other issues. The GBT is a huge physical structure; if the 3.6 meter reference antenna were placed in close proximity to the GBT, a large portion of the sky would be permanently shielded by the telescope. Even the small 3 meter antennas of the VSA, with such tight placement, can significantly shield one another at lower elevations. One triumph of the tests at Green Bank was demonstration that even with the large baseline, the LMS adaptive filter was capable of reliably and effectively cancelling a moving interferer.

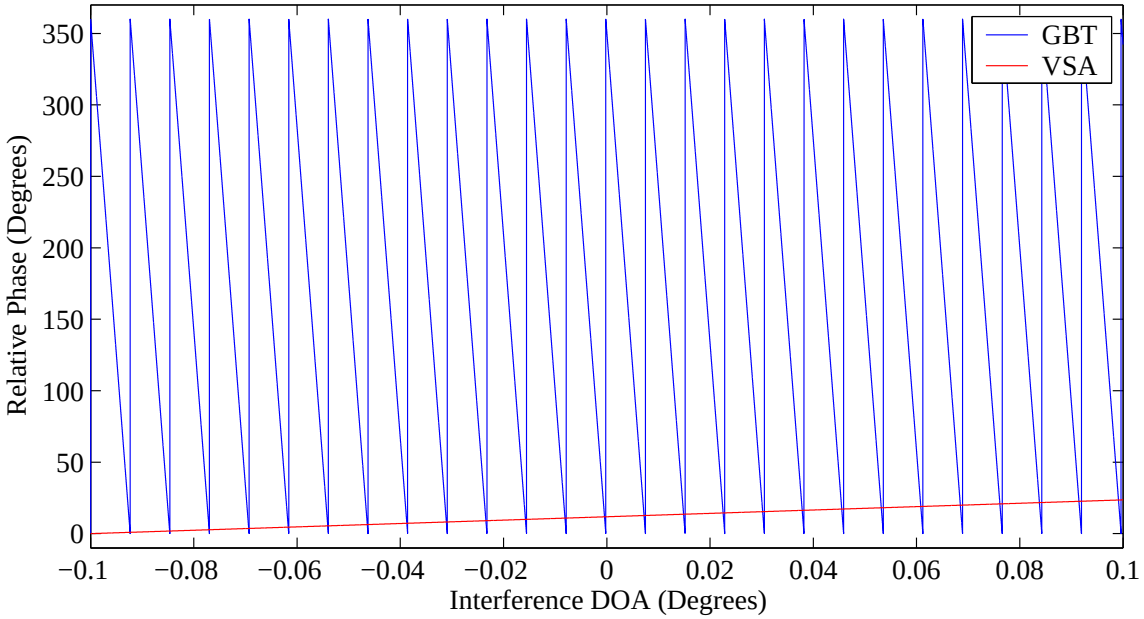


Figure 4.30: Worst-case relative phase rotation between the primary and reference channels for both the VSA and GBT test scenarios. This simulation utilizes the measured baseline estimates of $\mathbf{r} = [1395 \quad 92 \quad -25]^{\mathrm{T}}$ and $\mathbf{r} = [-3.05 \quad 1.76 \quad 0]^{\mathrm{T}}$ for the GBT and VSA, respectively. This simulation only takes into account phase drift due to differences in propagation delay between elements, not antenna or system phase response issues. The x-axis represents deviation in degrees from the interference DOA in the worst-case scenario; 0° on the x-axis refers to $\theta_{az} = 86.23°$ and $\theta_{el} = 88.98°$ for the GBT, and $\theta_{az} = 119.99°$ and $\theta_{el} = 90°$ for the VSA. Relative phase change with DOA is highest when the interferer moves parallel to the baseline. Therefore, the plot presents deviation in θ_{el} of $\pm 0.1°$ from those positions listed above. Movement orthogonal to this would result in no phase change with respect to DOA (the best-case scenario).

Presence of Multiple Interferers

With multiple interferers simultaneously above the horizon, it is highly possible, even probable, that each will spill through the telescope sidelobes with varying intensities. Unfortunately, only those interferers seen by the reference antenna will be adaptively subtracted out, leaving the others untouched. This is not a problem if the secondary interferers fall outside of the desired signal bandwidth. In the case of GLONASS downlink signals, however, adjacent frequency channels lie 0.5625 MHz apart and can be relatively wideband (see Figure 4.18). With less than half of the desired 24 GLONASS satellites currently deployed, this problem could worsen in the future–especially if the upper frequency channels coincident with the OH spectral lines are again utilized.

I encountered multiple interferers on several occasions while running GBT experiments. Figure 4.31 illustrates one of these cases. Two GLONASS satellites centered at 1607.0625 and 1607.625 MHz both lie within the processing bandwidth. The reference antenna collected the interferer centered at 1607.625 MHz, while the other remained untouched through the filtering process. The images detailing the interferer presence with respect to time clearly indicate the need for a second reference antenna.

An approach to cancelling multiple interferers, while utilizing the current DSP software, would be to cascade LMS filters, one for each interferer. Other approaches would likely be more favorable in terms of stability and performance, but this approach would be a simple and possibly successful solution. Each signal could be appropriately delayed according to the interferer's position.

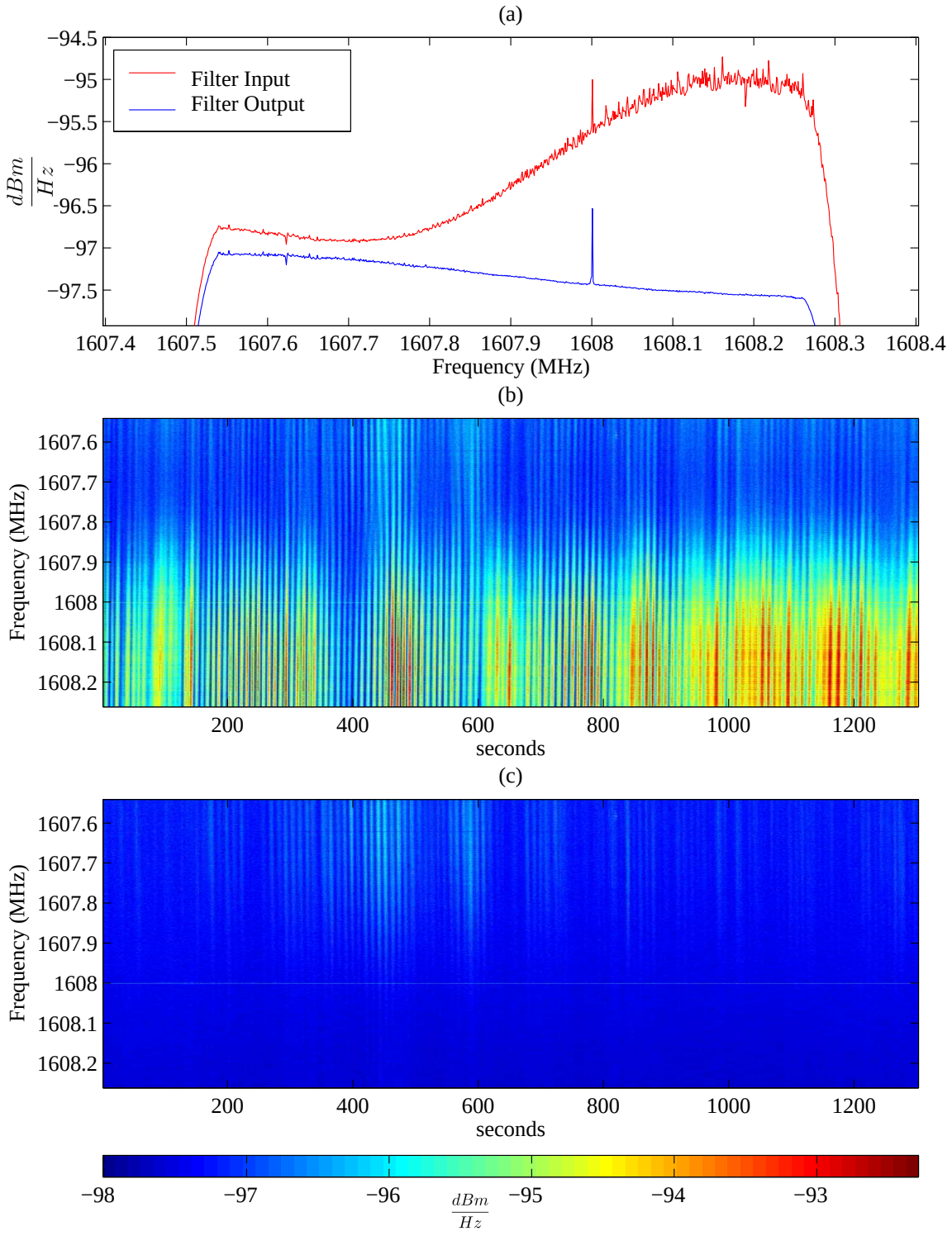


Figure 4.31: Illustration of two GLONASS interferers in the processing bandwidth simultaneously, centered at 1607.0625 and 1607.625 MHz. The reference antenna collected the interferer centered at 1607.625 MHz, while the other remained untouched through the filtering process.

GLONASS Spectral Sidelobes

As seen in Figure 4.18, the GLONASS signal spectrum is broad band, while the power tapers off significantly in the spectral sidelobes. Even if the relatively narrow GLONASS spectral main lobe falls out of the processing bandwidth, the sidelobes can degrade the data.

Cancellation of GLONASS spectral sidelobes can be more challenging without the main lobe in the processing bandwidth. With the main lobe in the processing bandwidth, the reference signal has a significantly higher INR. Additionally, the filter can track phase and amplitude changes in $i_p[n]$ with greater precision. Consequently, even if the GLONASS spectral main lobe is not directly coincident in frequency with the desired signal, there are advantages to including as much of it as possible in the processing bandwidth.

Chapter 5

Conclusion

5.1 Research Contributions

A significant contribution of this thesis was the development of a test platform consisting of three 10-foot diameter radio telescopes, low noise RF receivers [29], and a state-of-the-art DSP platform featuring four processors with four channels of 65 M sample/second digital receivers. This programmable real-time radio astronomy RFI mitigation tool is the first of its kind. Antenna positioning hardware and software were developed in order to track satellite interference sources and celestial objects in real-time. This development involved pinpointing and resolving the problems encountered in the complex interface between our host computer, the positioning hardware, and the dual azimuth/elevation rotors.

Three basic tools needed for radio astronomy observations and the analysis and implementation of interference mitigation algorithms were implemented in the real-time programmable DSP platform. These include a PSD estimator, a beamformer, and an array signal correlator. The multiprocessor DSP programming used to implement these tools was especially demanding since it involved real-time programs, communication with specialized hardware, and painstaking optimization of code.

The real-time PSD estimator produces a quality spectral estimate of the extremely low power signals characteristic of radio astronomy observations. Implementation required creative solutions in order to produce a quality estimate. Absolute power calibration and baseline smoothing methods were developed and successfully

implemented. Proper decline of variance as a function of long signal integration has been demonstrated with the stable GBT receiver. The DSP PSD estimator is capable of much longer and more stable integration than a typical commercial spectrum analyzer.

The real-time three channel beamformer/correlation estimator will be a very useful tool, especially upon successful completion of the VSA phase calibration. This DSP application was our first attempt at real-time array processing. It was also the first application to utilize inter-processor communication, which alone proved to be a difficult hurdle. A significant achievement was proper *synchronous* sampling of each input channel. Furthermore, the data from the three channels need to arrive simultaneously to a single processor for proper $\hat{\mathbf{R}}_{xx}$ calculation. The solution required the addition of varying delays in each digital receiver, giving each processor access to synchronous samples from all necessary signals.

An LMS adaptive filter was also implemented in the programmable real-time DSP platform. Despite being a straightforward algorithm, coding presented significant challenges. These included the interface to the digital receivers, inter-processor communication, proper data alignment, and code optimization. One of the most time consuming tasks of implementing the LMS filter in the DSP environment was the optimization of the filtering and tap update function. There was not a hand optimized assembly routine available for implementing the LMS adaptive filter with complex signals. The optimized routines created represent a significant contribution to the processing capacity of this programmable real-time interference canceller.

In those LMS test scenarios with a large baseline distance between the telescope and reference antenna, the two signals must be realigned to realize proper interference cancellation. As positions of satellite interferers can be predicted with satellite tracking software weeks in advance of a test, the necessary realignment parameters can also be precalculated. Calculations only depend on the interferer position and not the desired signal (astronomical source) position. A significant accomplishment of this thesis was consistent and proper data alignment in the GBT test scenario.

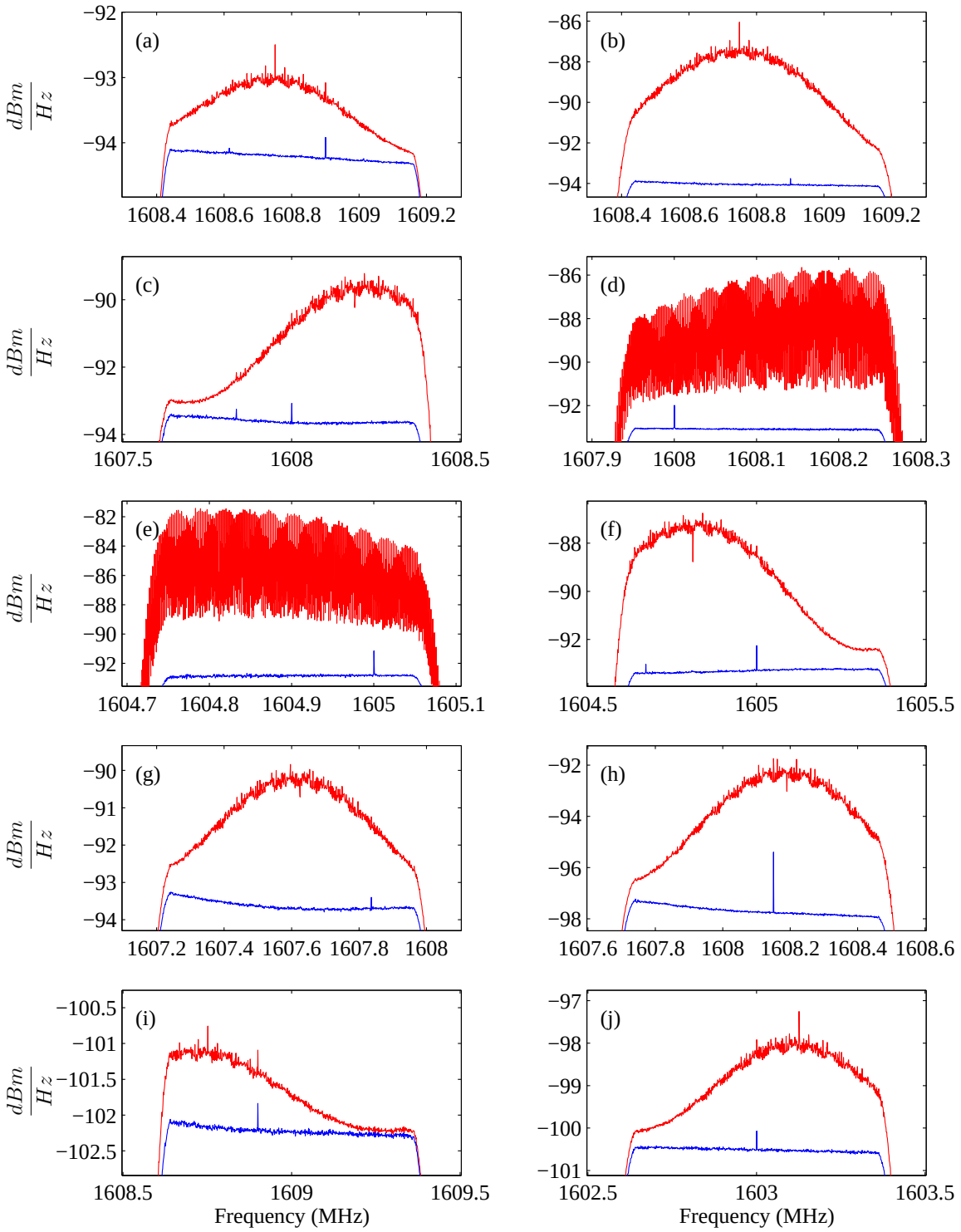


Figure 5.1: Examples of successful cancellation with the GBT. Before filtering, the test tone injected at the feed is hidden in the GLONASS spectrum. After cancellation, the desired signal emerges above a clean baseline. Test parameters: (a)–(c) & (g)–(j) $p = 12$, bandwidth=1.00625 MHz; (d) & (e) p=42, bandwidth=0.4125 MHz.

The real-time LMS adaptive filter proved to be a successful method of subtracting GLONASS satellite down-link signals from radio astronomy data. In many cases with the VSA and GBT, GLONASS interference was pushed undetectably below the noise floor, even with long integration. The interferer was removed without corrupting the desired signal. Methods of determining a lower bound on the achieved dB interference attenuation were demonstrated and implemented. Larger filter orders were more reliable, and less sensitive to data alignment. Significant cancellation did occur, however, with very few taps.

One triumph of the experiments at Green Bank was demonstration that even with a highly non-stationary test scenario, the LMS adaptive filter was capable of reliably and effectively cancelling a moving interferer. It was demonstrated that the dominant factor causing system non-stationarity with the Green Bank test scenario was not the GBT antenna sidelobe structure, but the phase rotation due to the large baseline between antennas.

In spite of a few challenges and obstacles, a majority of the tests run with the Green Bank Telescope were highly successful. Figure 5.1 illustrates cases where the test tone injected at the feed of the GBT is buried or hidden in the GLONASS spectral interference. After cancellation, the test tone emerges above a clean baseline. Undoubtedly, the simplicity, robustness and performance of the LMS adaptive algorithm make it a prime candidate for satellite interference cancellation in radio astronomy.

5.2 Future Research

As described in Section 4.6, an interesting phenomenon was observed when restoring clarity to hydrogen lines corrupted with an FM sweep signal in the BYU VSA test scenario. In some cases not only the interferer, $i_p[n]$, was subtracted from primary channel. Instead, a portion of the observational noise component, $\eta_p[n]$, was also excised in the process. It would be helpful to determine the cause for this unexpected behavior; time limitations impeded further investigation. Fortunately, this odd behavior was not observed in tests performed with the Green Bank Telescope.

A few improvements could be added to the radio astronomy observational and analysis tools developed in the DSP. Additional functionality which could be added to the real-time PSD estimator includes the addition of variable frequency resolution for a fixed processing bandwidth, and the capability to implement non-rectangular and/or overlapping windowing techniques. The real-time beamformer and correlation estimator will need to be modified to account for a fourth channel which will soon be added to the VSA. In addition, the narrow band beamformer design may need to be modified to implement broadband beamforming techniques.

The LMS canceller implemented in our state-of-the-art DSP platform would be immediately beneficial to astronomers making observations in the presence of satellite interference. Scientists at the NRAO have expressed great interest in implementing a similar system for interference cancellation at the Green Bank Telescope. Multiple interferers could be attenuated without new code development by cascading multiple LMS adaptive filters in successive DSP processors, while tracking satellites with multiple reference antennas. Alternately, a multiple reference filter code could be developed.

Appendix A

Source Code

This appendix contains some of the source code pertinent to the research presented in this thesis. The code is written in the MATLAB, C and Java programming languages.

A.1 Antenna Steering Software

Antenna software was developed in order to track satellite interference sources and celestial objects in real-time. The antenna steering software is an adaptation of a java application developed by the MIT SRT research initiative [27]. Several changes were made to the code provided by MIT. All functions relating to the SRT receiver were removed, while retaining its ability to track any celestial source given its right ascension and declination (1950 epoch). We added the capability to track satellites by creating an interface (tracker.cpp) between the java software and Nova for Windows, a powerful satellite tracking application [28].

This section includes the following source code files:

- time.java—page 160
- tracker.cpp—page 162 [47]

srt.cat

```
* first word is key word
* STATION: latitude longitude west in degrees
* SAT: satellite ID then longitude west
* SOU: source ra, dec, name, epoch
STATION 40.25 111.65 BYU
* source coords epoch 1950 unless specified
* SOU 05 31 30  21 58 00 Crab
* SOU 05 32 48  -5 27 00 Orion
* SOU 05 42 00  -1 00 00 S8
SOU 20 27 00  27 00 00 CygX
* SOU 23 21 12  58 44 00 Cass
SOU 00 00 00  00 00 00 Sun
* SOU 17 42 54 -28 50 00 SgrA
* SOU 06 29 12  04 57 00 Rosett
* SOU 18 17 30 -16 18 00 M17
* SOU 20 27 00  41 00 00 CygEMN
SOU 00 00 00  00 00 00 Moon
* SOU 21 12 00  48 00 00 G90
* SOU 05 40 00  29 00 00 G180
* SOU 12 48 00  28 00 00 GNpole
* SOU 00 39 00  40 30 00 Androm
* SOU 05 14 12  18 44 00 AC1
SOU 07 20 54 -25 40 12 VYCMA
* AZLIMITS_1 91.3 262.9 /* mid az range is south (Antenna 1)*/
* AZLIMITS_2 88.7 261.5 /* mid az range is south (Antenna 2)*/
* AZLIMITS_3 87.9 261.8 /* mid az range is south (Antenna 3)*/
* ELLIMITS_1 10.5 171.8  /*  elevation limit south - north (Antenna 1)*/
* ELLIMITS_2 7.0 169.2  /*  elevation limit south - north (Antenna 2)*/
* ELLIMITS_3 12.1 173.6  /*  elevation limit south - north (Antenna 3)*/

AZLIMITS 91.3 262.9 /* mid az range is south (Antenna 1)*/
ELLIMITS 10.5 171.8  /*  elevation limit south - north (Antenna 1)*/

COMM_1 1              /* COM1 */
CALCONS 0.5          /* gain correction constant to put power in units of K */
BEAMWIDTH 5.0        /* 3 dB antenna beamwidth in degrees */
MANCAL 0             /* 0 or absence indicates automated cal vane */
```

cat.java

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.lang.*;
import java.awt.*;
public class cat
// reads catalog file
{
  double dec;
  double ylim = 415.0;
  public void catfile(global global, disp d, Graphics gg)
  {
    try
    {
      BufferedReader in =
      new BufferedReader(new FileReader(new File("srt.cat")));
      double rah,
        ram,
        rass,
        decd,
        decm,
        decss,
        ep,
        glat,
        glon;
      int decsign,
```

```
  i;
global.set_nsou(1);
String str,
  str1,
  str2;
StringTokenizer parser;
while ((str1 = in.readLine()) != null)
{
  if (!str1.startsWith("*"))
  { i = 0;
    str = "";
    while (i < str1.length() && str1.charAt(i) != '/')
    {
     str += str1.charAt(i);
     i++;
    }
    if (str.indexOf("STATION") != -1)
    {
      parser = new StringTokenizer(str);
      str2 = parser.nextToken();
      str2 = parser.nextToken();
      global.set_lat(Double.valueOf(str2).doubleValue() * Math.PI / 180.0);
        str2 = parser.nextToken();
      global.set_lon(Double.valueOf(str2).doubleValue() * Math.PI / 180.0);
        str2 = parser.nextToken();
      global.set_statnam(str2);
    }
    if (str.indexOf("AZLIMITS") != -1)
    {
      parser = new StringTokenizer(str);
      str2 = parser.nextToken();
      str2 = parser.nextToken();
      global.set_azlim1_1(Double.valueOf(str2).doubleValue());
        str2 = parser.nextToken();
      global.set_azlim2_1(Double.valueOf(str2).doubleValue());
    }
    if (str.indexOf("ELLIMITS") != -1)
    {
      parser = new StringTokenizer(str);
      str2 = parser.nextToken();
      str2 = parser.nextToken();
      global.set_ellim1_1(Double.valueOf(str2).doubleValue());
      str2 = parser.nextToken();
      global.set_ellim2_1(Double.valueOf(str2).doubleValue());
    }
    if (str.indexOf("COMM_1") != -1)
    {
      parser = new StringTokenizer(str);
      str2 = parser.nextToken();
      str2 = parser.nextToken();
      global.set_port1((int)Double.valueOf(str2).doubleValue());
    }

    if (str.indexOf("CALCONS") != -1)
    {
      parser = new StringTokenizer(str);
      str2 = parser.nextToken();
      str2 = parser.nextToken();
      global.set_calcons(Double.valueOf(str2).doubleValue());
    }
    if (str.indexOf("TLOAD") != -1)
    {
      parser = new StringTokenizer(str);
      str2 = parser.nextToken();
      str2 = parser.nextToken();
      global.set_tload(Double.valueOf(str2).doubleValue());
    }
    if (str.indexOf("TSPILL") != -1)
    {
      parser = new StringTokenizer(str);
      str2 = parser.nextToken();
      str2 = parser.nextToken();
      global.set_tspill(Double.valueOf(str2).doubleValue());
    }
    if (str.indexOf("BEAMWIDTH") != -1)
```

```
{
  parser = new StringTokenizer(str);
  str2 = parser.nextToken();
  str2 = parser.nextToken();
  global.set_beamw(Double.valueOf(str2).doubleValue());
}
if (str.indexOf("MANCAL") != -1)
{
  parser = new StringTokenizer(str);
  str2 = parser.nextToken();
  str2 = parser.nextToken();
  global.set_mancal((int)Double.valueOf(str2).doubleValue());
}
if (str.indexOf("AXISTILT") != -1)
{
  parser = new StringTokenizer(str);
  str2 = parser.nextToken();
  str2 = parser.nextToken();
  global.set_azaxis_tilt(Double.valueOf(str2).doubleValue());
  str2 = parser.nextToken();
  global.set_tilt_az(Double.valueOf(str2).doubleValue());
  try
  {
    str2 = parser.nextToken();
    global.set_elaxis_tilt(Double.valueOf(str2).doubleValue());
  }
  catch(NoSuchElementException e)
  {
  }
}
if (str.indexOf("SSAT") != -1)   // changes for syncSAT
{
  parser = new StringTokenizer(str);
  str2 = parser.nextToken();
  str2 = parser.nextToken();
  global.set_sounam(str2, global.get_nsou());
  global.set_soutype(4, global.get_nsou());
  str2 = parser.nextToken();
  // ras used to store longitude west
  global.set_ras(Double.valueOf(str2).doubleValue(), global.get_nsou());
  if (global.get_nsou() < 500)
    global.set_nsou(global.get_nsou() + 1);
}
if (str.indexOf("AZEL") != -1)
{
  parser = new StringTokenizer(str);
  str2 = parser.nextToken();
  global.set_soutype(1, global.get_nsou());
  str2 = parser.nextToken();
  // ras,dec used to store az and el
  global.set_ras(Double.valueOf(str2).doubleValue()
   * Math.PI /180.0, global.get_nsou());
  str2 = parser.nextToken();
  global.set_decs(Double.valueOf(str2).doubleValue()
   * Math.PI /180.0, global.get_nsou());
  str2 = parser.nextToken();
  global.set_sounam(str2, global.get_nsou());
  if (global.get_nsou() < 500)
    global.set_nsou(global.get_nsou() + 1);
}
if (str.indexOf("GALACTIC") != -1)
{
  parser = new StringTokenizer(str);
  str2 = parser.nextToken();
  global.set_soutype(0, global.get_nsou());
  str2 = parser.nextToken();
  glon = Double.valueOf(str2).doubleValue();
  str2 = parser.nextToken();
  glat = Double.valueOf(str2).doubleValue();
  global.set_ras(get_ra(glat,glon), global.get_nsou());
  global.set_decs(get_dec(glat,glon), global.get_nsou());
  str2 = parser.nextToken();
  global.set_sounam(str2, global.get_nsou());
  global.set_epoc(2000.0, global.get_nsou());
  if (global.get_nsou() < 500)
```

```
            global.set_nsou(global.get_nsou() + 1);
          }
          if (str.indexOf("Sun") == 0 || str.indexOf("Moon") == 0)
          {    // supports keyword Sun or Moon starting in Col 1
            parser = new StringTokenizer(str);
            str2 = parser.nextToken();
            global.set_sounam(str2, global.get_nsou());
            global.set_ras(0.0, global.get_nsou());
            global.set_decs(0.0, global.get_nsou());
            global.set_epoc(0.0, global.get_nsou());
            global.set_soutype(0, global.get_nsou());
            if (global.get_nsou() < 100)
              global.set_nsou(global.get_nsou() + 1);
          }
          if (str.indexOf("SOU") != -1)
          {
            parser = new StringTokenizer(str);
            str2 = parser.nextToken();
            str2 = parser.nextToken();
            rah = Double.valueOf(str2).doubleValue();
            str2 = parser.nextToken();
            ram = Double.valueOf(str2).doubleValue();
            str2 = parser.nextToken();
            rass = Double.valueOf(str2).doubleValue();
            str2 = parser.nextToken();
            decd = Double.valueOf(str2).doubleValue();
            str2 = parser.nextToken();
            decm = Double.valueOf(str2).doubleValue();
            str2 = parser.nextToken();
            decss = Double.valueOf(str2).doubleValue();
            str2 = parser.nextToken();
            global.set_sounam(str2, global.get_nsou());
//     System.out.println("sou"+str2);
            ep = 1950.0;        /* default */
            try
            {
              str2 = parser.nextToken();
              ep = Double.valueOf(str2).doubleValue();
            }
            catch(NoSuchElementException e)
            {
            }
            if (str.indexOf("-") != -1)
              decsign = -1;
            else
              decsign = 1;
            global.set_ras((rah + ram / 60.0 + rass / 3600.0) * Math.PI / 12.0,
                global.get_nsou());
            global.set_decs(decsign * (Math.abs(decd) + decm / 60.0 + decss / 3600.0)
                 *
                            Math.PI / 180.0, global.get_nsou());
            global.set_epoc(ep, global.get_nsou());
            global.set_soutype(0, global.get_nsou());
            if (global.get_nsou() < 100)
              global.set_nsou(global.get_nsou() + 1);
          }
        }
      }
      in.close();
    }
    catch(NumberFormatException e)
    {
      System.out.println(e);
    }
    catch(FileNotFoundException e)
    {
      System.out.println(e);
      d.dtext(440.0,ylim+40.0,gg,Color.red,"catalog␣file␣srt.cat␣not␣found");
      global.set_fstatus(-99);
    }
    catch(IOException e)
    {
      System.out.println(e);
    }
    global.set_nsoucat(global.get_nsou());
```

```
  }

    public double cmdfile(global g, disp d, Graphics gg, time t) throws IOException
  // reads command file
/* drives the schedule with stop time yyyy:ddd:hh:mm:ss
   or LST:hh:mm:ss  or just : for immediate scheduling
   or :n for scheduling for n secs
   followed by keywords:
   sourcename (any name in catalog)
   azel az_deg el_deg
   radec ra_hh:mm:ss (or decimal) dec_dd:mm:ss (or decimal) [epoch]
   galactic glat_deg glon_deg
   stow
   calibrate
   record (turns on data file if not already on) [filename] [recmode]
   roff (turns off data file)
   freq fcenter_MHz number_of_frequencies [spacing]
   mode n for 25 point b for beamswitch
 */
  {
    RandomAccessFile in = new RandomAccessFile("antenna.txt","r");
    String temp,temp2;
        //long filePosition=0;
        StringTokenizer strChunks;
        String az=null;
        String el=null;

        in.seek(g.get_filepointer());
        temp=in.readLine();
        while(temp!=null) {
                strChunks=new StringTokenizer(temp);
                while(strChunks.hasMoreTokens()) {
                        temp2=strChunks.nextToken();
                        if (temp2.charAt(0)=='A') {
                                if (temp2.substring(0,3).equals("AZ:")) {
                                        az=temp2.substring(3);
                                }
                        }
                        else if (temp2.charAt(0)=='E') {
                                if (temp2.substring(0,3).equals("EL:")) {
                                        el=temp2.substring(3);
                                }
                        }
                }
                g.set_azcmd1(Double.valueOf(az).doubleValue());
                g.set_elcmd1(Double.valueOf(el).doubleValue());
                g.set_track(0);
                //System.out.println(az+" "+el);
                //System.out.println(az+" "+el);
                temp=in.readLine();
        }

        g.set_filepointer(in.getFilePointer());
        in.close();

  //  g.set_cmdf(0);

    return 0;


  }

  public double get_ra(double glat, double glon)
  /* galactic to radec  2000 epoch pole at 12h51.4 27.1 */
  {
    double a,
      xg,
      yg,
      zg,
      xr,
      yr,
      zr,
      d0,
      dp,
      r0,
```

```
      rp,
      ra;
      d0 = -(28.0 + 56.0 / 60.0) * Math.PI / 180.0;
      r0 = (17.0 + 45.5 / 60.0) * Math.PI / 12.0;
      dp = 27.1 * Math.PI / 180.0;
      rp = (12.0 + 51.4 / 60.0) * Math.PI / 12.0;
      zr = Math.sin(d0);
      xr = Math.cos(r0 - rp) * Math.cos(d0);
      yr = Math.sin(r0 - rp) * Math.cos(d0);
      xg = xr * Math.sin(dp) - zr * Math.cos(dp);
      yg = yr;
      a = Math.atan2(yg, xg);
      xg = Math.cos((glon * Math.PI / 180.0) + a) * Math.cos(glat * Math.PI / 180.0);
      yg = Math.sin((glon * Math.PI / 180.0) + a) * Math.cos(glat * Math.PI / 180.0);
      zg = Math.sin(glat * Math.PI / 180.0);
      xr = xg * Math.sin(dp) + zg * Math.cos(dp);
      yr = yg;
      zr = zg * Math.sin(dp) - xg * Math.cos(dp);
      dec = Math.atan2(zr, Math.sqrt(xr * xr + yr * yr));
      ra = Math.atan2(yr, xr) + rp;
      return ra;
  }
  public double get_dec(double glat, double glon)
  {
    return dec;
  }

}
```

checkey.java

```
import java.awt.*;
import java.util.*;
import java.text.*;
import java.lang.*;
public class checkey
// class to make specify action upon commands
{
  private Graphics gg;
  private StringTokenizer parser;
  private String str2,
    str3;
  private double ylim = 415.0;
  void set_graph(Graphics g)
  {
    gg = g;
  }
  void check(int mode, char cmd, global g, disp d, String str)
  {
// mode 1=button or text 2=mouse entered 3=mouse excited
    String str4;
    int i;
    if (mode == 2)
    {                           // Mode 2 is for when the mouse is placed over one of
         the command buttons
                                //  It sends the first char of the label "A" for "
                                    Azimuth/Elevation" in the
                                //  variable cmd.
      g.set_click(0);
      if (cmd == 'S')
        g.set_key(0);
      if (cmd == 'T')
        g.set_key(1);
      if (cmd == 'A')
        g.set_key(2);
      if (cmd == 'B')
        g.set_key(3);
      if (cmd == 'C')
        g.set_key(4);
      if (g.get_key() < 5)
        d.pclear(gg, 8.0, ylim+50.0, 640.0, 100.0);
    }
```

```
    if (g.get_key() == 0)  // Stow antennas
        d.dtext(8.0, ylim+60.0, gg, Color.black,
                "click␣to␣go␣to␣Stow");
    if (g.get_key() == 1) { // Command all antennas to track astronomical source
        d.dtext(8.0, ylim+60.0, gg, Color.black,
                "click␣to␣track␣source");
        d.dtext(8.0, ylim+74.0, gg, Color.black,
                "pointing␣cursors␣are␣yellow␣while␣slewing␣to␣source␣"+
                "and␣turn␣red␣when␣antenna␣reaches␣commanded␣position");
        d.dtext(8.0, ylim+88.0, gg, Color.black,
                "if␣pointing␣offsets␣are␣non␣zero␣cursors␣"+
                "will␣be␣displaced␣from␣source");
                }
    if (g.get_key() == 2 && mode == 2) {  // Azimuth/Elevation
       if(g.get_mainten() == 0)
        d.dtext(8.0, ylim+60.0, gg, Color.black,
                "click␣on␣button␣to␣set␣an␣azimuth␣and␣elevation␣for␣all␣Antennas");
       else
        d.dtext(8.0, ylim+60.0, gg, Color.red,
                "in␣maintenance␣mode␣use␣offsets␣to␣move␣␣␣␣␣");
                }
    if (g.get_key() == 2 && mode == 1)
        d.dtext(8.0, ylim+60.0, gg, Color.black,
                "Please␣click␣on␣text␣area␣to␣enter␣azimuth␣and␣elevation␣for␣
                    Antennas");

    if (mode == 1)
    {
      if (g.get_key() == 2 )
        g.set_click(-1);
      else
        g.set_click(1);
      g.set_cmdstr(str);
    }
  }
  void checky(global g, disp d, Graphics gg)
  {
    str3 = g.get_cmdstr();
// System.out.println("key "+g.get_key()+" str3 "+str3);


    if (g.get_key() == 1)
    { // This is the "track" button
      g.set_stow(0);
      g.set_bsw(0);
      g.set_azoff(0.0);
      g.set_eloff(0.0);
      g.set_clr(1);
      g.set_track(g.get_track() + 1);
      if (g.get_track() >= 2) {
        g.set_track(0);
      }
    }


    if (g.get_key() == 0)
    { // Stow button
      g.set_stow(1);
      g.set_track(0);
      g.set_azcmd1(g.get_azlim1_1());

      g.set_elcmd1(g.get_ellim1_1());

    }


 //   if (g.get_key() == 5)
    if (g.get_key() == 2)
    { // This is the "azel button"
      if (str3.length() > 2)
      {
        parser = new StringTokenizer(str3);
        try
        {
          str2 = parser.nextToken();
```

```
        }
        catch(NoSuchElementException e)
        {
          return;
        }
        try
        {
          g.set_azcmd1(Double.valueOf(str2).doubleValue());

        }
        catch(NumberFormatException e)
        {
          d.dtext(450.0, ylim+60.0, gg, Color.blue, "format␣error␣");
          return;
        }
        try
        {
          str2 = parser.nextToken();
        }
        catch(NoSuchElementException e)
        {
          return;
        }
        try
        {
          g.set_elcmd1(Double.valueOf(str2).doubleValue());

        }
        catch(NumberFormatException e)
        {
          d.dtext(450.0, ylim+60.0, gg, Color.blue, "format␣error␣");
          return;
        }
        d.dtext(450.0, ylim+60.0, gg, Color.blue, "Entered␣az␣" + d.dc(g.get_azcmd1()
            , 5, 1)
                + "␣el␣" + d.dc(g.get_elcmd1(), 5, 1));
      }
      g.set_track(0);
      g.set_stow(0);
      return;
    }

    if (g.get_key() == 3)
    { // Track a satellite from a text file

        g.set_cmdf(1); // execute the command file
        //System.out.println("working");
    }

    if (g.get_key () == 4)
    {
        g.set_cmdf(0);  // turn of auto-satellite tracking
        }


    g.set_cmdstr("");           // set null string

  }
}
```

disp.java

```
import java.awt.*;
import java.awt.event.*;
import java.text.*;
public class disp extends Frame implements ActionListener
// class for Frame, mouse control and graphics
{
  int xoff = 1,
    yoff = 50;
  int gff = 0;
  int w,
    h;
  int fsize = 12;
```

```
Font f;
FontMetrics fm;
Graphics gf;
double sx,
  sy;
global glb;
TextField tf = new TextField();
Button b[] = new Button[20];
checkey che;
MouseEventHandler mhand = new MouseEventHandler();
String but[] =
{
 "Stow", "Track␣From␣Plot␣Below", "Az/El␣(Move␣the␣antennas)","Begin␣Tracking␣
     Satellite","Cease␣Tracking␣Satellite"};
public disp(String title, global g, checkey c)
{
  super(title);
  int i;
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    enableEvents(AWTEvent.KEY_EVENT_MASK);
    enableEvents(AWTEvent.MOUSE_EVENT_MASK);
    enableEvents(AWTEvent.COMPONENT_EVENT_MASK);
    enableEvents(AWTEvent.FOCUS_EVENT_MASK);
    w = Toolkit.getDefaultToolkit().getScreenSize().width;
    h = Toolkit.getDefaultToolkit().getScreenSize().height;
    if(w > 960) w = 960;
    if(h > 720) h = 720;
    sx = (double)w / 800.0;
    sy = (double)h / 600.0;
    setBounds(0, 0, w, h);
    glb = g;
    che = c;
    add(tf, BorderLayout.SOUTH);
    tf.addActionListener(this);
  Panel p = new Panel(new GridLayout(1, 0));
  for (i = 0; i < 5; i++)  // 5 buttons on display panel
  {
    b[i] = new Button(but[i]);
    p.add(b[i]);
    b[i].addActionListener(this);
    b[i].addMouseListener(mhand);
    //System.out.println("i:"+i);  //debugging line to test array bounds
  }
  add(p, BorderLayout.NORTH);
    mhand.mou(glb, this, che);
    setVisible(true);
}
public void set_font(Graphics gg)
{
  f = new Font("Serif", Font.PLAIN, (int)(fsize * sx));
  gg.setFont(f);
  gf = gg;
  gff = 1;
}
public void set_bc(Color color, int n)
{
  b[n].setForeground(color);
}
public void paint(Graphics g, Color color, double xx, double yy)
{
  g.setColor(color);
  g.setPaintMode();
  g.drawLine((int)(xx * sx) + xoff, (int)(yy * sy) + yoff,
             (int)(xx * sx) + xoff, (int)(yy * sy) + yoff);
}
public void lpaint(Graphics g, Color color,
                   double x1, double y1, double x2, double y2)
{
  g.setColor(color);
  g.setPaintMode();
  g.drawLine((int)(x1 * sx) + xoff, (int)(y1 * sy) + yoff,
             (int)(x2 * sx) + xoff, (int)(y2 * sy) + yoff);
}
public void spaint(Graphics g, Color color, double xx, double yy, int siz)
{
```

```
    g.setColor(color);
    g.setPaintMode();
    g.fillOval((int)(xx * sx) + xoff - siz/2,
      (int)(yy * sy) + yoff - siz/2, siz, siz);
  }
  public void rpaint(Graphics g, Color color, double xx, double yy)
  {
    g.setColor(color);
    g.setPaintMode();
    g.fillRect((int)(xx * sx) + xoff, (int)(yy * sy) + yoff, 1 + (int)sx, 1 + (int)sx
      );
  }
  public void pclear(Graphics g, double xx, double yy, double wid, double hi)
  {
    g.clearRect((int)(xx * sx) + xoff, (int)(yy * sy) + yoff,
                (int)(wid * sx), (int)(hi * sy));
  }
  public void ppclear(Graphics g, double xx, double yy, double wid)
  {
    g.clearRect((int)(xx * sx) + xoff, (int)(yy * sy) + yoff
                - g.getFontMetrics().getHeight()
                + g.getFontMetrics().getDescent(), (int)(wid * sx),
                g.getFontMetrics().getHeight());
  }
  public void dtext(double xx, double yy, Graphics g, Color color, String text)
  // print text to screen - with erase to remove prior graphics
  // within text region
  {
    g.setColor(color);
    g.setPaintMode();
    g.clearRect((int)(xx * sx) + xoff, (int)(yy * sy) + yoff
                - g.getFontMetrics().getMaxAscent(),
                g.getFontMetrics().stringWidth(text) +
                g.getFontMetrics().stringWidth("␣␣␣"),
                g.getFontMetrics().getMaxAscent() +
                g.getFontMetrics().getMaxDescent());
//  System.out.println(g.getFontMetrics().stringWidth(text)+" "
//     +g.getFontMetrics().getHeight());
    g.drawString(text, (int)(xx * sx) + xoff, (int)(yy * sy) + yoff);
  }
  public void stext(double xx, double yy, Graphics g, Color color, String text)
  // write text to screen
  {
    g.setColor(color);
    g.setPaintMode();
    g.drawString(text, (int)(xx * sx) + xoff, (int)(yy * sy) + yoff);
  }
  public String dc(double a, int m, int n)
  // make printable floating point equivalent C %m.nf
  // m chars wide n chars after the decimal point
  {
    int i,
      j;
    String str2,
      str3;
    NumberFormat nf = NumberFormat.getInstance();
      nf.setMaximumFractionDigits(n);
      nf.setMinimumFractionDigits(n);
    if (a >= 0.0)
    {
      if (n > 0)
        nf.setMinimumIntegerDigits(m - n - 1);
      else
        nf.setMinimumIntegerDigits(m);
    }
    else
    {
      if (n > 0)
        nf.setMinimumIntegerDigits(m - n - 2);
      else
        nf.setMinimumIntegerDigits(m - 1);
    }
    nf.setGroupingUsed(false);
    str2 = nf.format(a);
    str3 = "";
```

```
    j = 0;
    if (str2.charAt(0) != '-')
    {
      for (i = 0; i < str2.length() - 1; i++)
        if (str2.charAt(i) == '0' && j == 0 && str2.charAt(i + 1) != '.')
          str3 += "␣";
        else
        {
          str3 += str2.charAt(i);
          j = 1;
        }
    }
    else
    {
      for (i = 1; i < str2.length() - 1; i++)
      {
        if (str2.charAt(i) == '0' && j == 0 && str2.charAt(i + 1) != '.')
          str3 += "␣";
        else
        {
          if (j == 0)
            str3 += "-";
          str3 += str2.charAt(i);
          j = 1;
        }
      }
      if (j == 0)
        str3 += "-";
    }
    str3 += str2.charAt(str2.length() - 1);
    return str3;
  }
  public String dcs(String s, int m)
  // fix string to m spaces long by adding spaces
  {
    int i;
    String str2 = s;
    for (i = 0; i < m - str2.length(); i++)
        str2 += "␣";
      return str2;
  }
  protected void processWindowEvent(WindowEvent e)
  {
//   System.out.println("close");
    if (e.getID() == WindowEvent.WINDOW_CLOSING)
    {
      System.exit(0);
    }
    super.processWindowEvent(e);
  }
  protected void processComponentEvent(ComponentEvent e)
  {
    if (e.getID() == ComponentEvent.COMPONENT_RESIZED)
    {
      w = getBounds().width;
      h = getBounds().height;
      if (w > 960) w = 960;
      if (h > 720) h = 720;
      setBounds(0,0,w,h);
//   System.out.println("resized "+w+" "+h+" fsize "+fsize);
      sx = (double)w / 800.0;
      sy = (double)h / 600.0;
      f = new Font("Serif", Font.PLAIN, (int)(fsize * sx));
      if (gff == 1)
                gf.setFont(f);
    }
  }
  protected void processMouseEvent(MouseEvent evt)
  // process mouse and check for nearest source on display
  {
    double dx,
      dy,
      r,
      min;
    int i,
```

```
      j,
      mx,
      my;
    if (evt.getClickCount() >= 1)
    {
      mx = evt.getX();
      my = evt.getY();
      min = 20.0;
      j = -1;
      for (i = 1; i < glb.get_nsou(); i++)
      {
        dx = (double)(mx - xoff - glb.get_xlast(i) * sx);
        dy = (double)(my - yoff - glb.get_ylast(i) * sy);
        r = dx * dx + dy * dy;
        if (Math.sqrt(r) < min)
        {
          min = Math.sqrt(r);
          j = i;
        }
      }
      if (j > 0)
      {
        glb.set_sourn(j);
        glb.set_bsw(0);
        glb.set_clr(1);
      }
     if (my - yoff < 100 * sy)  // check for click on spectrum
        glb.set_xmark((int)((mx-xoff)/sx));
//   System.out.println("mouse "+mx+" "+my+"nfr"+glb.get_nfreq());
    }
  }
  public void actionPerformed(ActionEvent evt)
  {
    String str2;
    char cmd;
//  System.out.println("act "+tf.getText()+"evt= "+evt);
    //  System.out.println("id "+evt.getActionCommand());
      str2 = tf.getText();
    cmd = 0;
//  System.out.println("char="+cmd+" text "+str2);
      che.check(1, cmd, glb, this, str2);
      tf.setText("");
      requestFocus();           // needed so prevent loss of key events

  }
}
class MouseEventHandler extends MouseAdapter
{
  global glb;
  disp dd;
  checkey che;
  public void mou(global g, disp d, checkey c)
  {
    glb = g;
    dd = d;
    che = c;
  }
  public void mouseEntered(MouseEvent evt)
  {
//    System.out.println("mouseent "+ evt.getSource());
    che.check(2, ((Button) evt.getSource()).getLabel().charAt(0),
              glb, dd, "");
  }
  public void mouseExited(MouseEvent evt)
  {
    che.check(3, '0', glb, dd, "");
  }
}
```

geom.java

```
import java.awt.*;
public class geom   // methods for source coordinates
{
```

```
  private double ra,
    dec,
    raout,
    decout,
    azim,
    elev,
    glat,
    glon,
    vel;
  private double x,
    y;
  private double ylim = 415.0;
  private double ylim0 = 200.0;
  private time tim = new time();
  double get_moonra(double time, global g)
  {
    double ttime,
      asnode,
      amon,
      peri,
      em,
      aim,
      aam,
      vsm,
      alamm,
      moonsun,
      evn,
      var;
    double x,
      y,
      z,
      xx,
      yy,
      zz,
      ram,
      decm,
      ha,
      inc;
/* calc moon ra and dec  */
/* see notes and formulae Astronomical Almanac page D2 Moon, 1999 */
      ttime = tim.tosec(1999,0,0,0,0); // Jan 0 1999
      ttime = (time - ttime) / 86400.00;
    /* asnode=long of mean ascending node */
      asnode = 144.452077 - 0.05295377 * ttime;
    /* amon=omga plus mean lunar longitude */
      amon = 69.167124 + 13.17639648 * ttime;
    /* peri=asnode plus mean lunar longitude of perigee */
      peri = 42.524057 + 0.11140353 * ttime;
    /* moonsun is the elongation of moon from the sun */
      moonsun = 149.940812 + 12.19074912 * ttime;
    /* em is the eccentricity of lunar orbit */
      em = 0.054900489;
    /* aim=inclination of lunar orbit to ecliptic */
      aim = 5.1453964 * Math.PI / 180.0;
    /* vsm=true anomaly */
    /* the following are correction terms */
      vsm = 2.0 * em * Math.sin((amon - peri) * Math.PI / 180.0); /* elliptical orbit
          */
      evn = (1.274 / 57.3) * Math.sin((2 * moonsun - (amon - peri)) * Math.PI
          / 180.0); /* evection */
      var = (0.658 / 57.3) * Math.sin(2 * moonsun * Math.PI / 180.0); /* variation */
      alamm = (amon - asnode) * Math.PI / 180.0 + vsm + evn + var;
      x = Math.cos(alamm);
      y = Math.sin(alamm);
      z = 0;
      xx = x;
      yy = y * Math.cos(aim);
      zz = y * Math.sin(aim);
      ram = Math.atan2(yy, xx) + asnode * Math.PI / 180.0;
      decm = Math.atan2(zz, Math.sqrt(xx * xx + yy * yy));
      x = Math.cos(ram) * Math.cos(decm);
      y = Math.sin(ram) * Math.cos(decm);
      z = Math.sin(decm);
      inc = 23.45 * Math.PI / 180.0;
      xx = x;
```

```
      yy = y * Math.cos(inc) - z * Math.sin(inc);
      zz = z * Math.cos(inc) + y * Math.sin(inc);
    /* aam is the semi-major axis of orbit earth radii */
      aam = 60.2665;
      z = zz - Math.sin(g.get_lat()) / aam; /* correct for parallax */
      ha = tim.getGst() - g.get_lon();
      x = xx - Math.cos(g.get_lat()) * Math.cos(ha) / aam;
      y = yy - Math.cos(g.get_lat()) * Math.sin(ha) / aam;
      ra = Math.atan2(y, x);
      dec = Math.atan2(z, Math.sqrt(x * x + y * y));
      return ra;
  }
  double get_moondec()   // make 2nd. call to get moon's declination
  {
    return dec;
  }


  void setsounam(global g, disp d, Graphics gg, time tim)
  {
// plots the catalog sources on the screen
    double has,
      azs,
      elevs,
      sec,
      azss,
      elevss,
      azz,
      ell;
    int j,
      year;
      sec = tim.getTsec(g, d, gg);
      year = (int)tim.get_year();
      azs = elevs = has = 0.0;
    for (j = 1; j < g.get_nsou(); j++)
    { if (g.get_soutype(j) == 0) {
          raout = get_precess_ra(g.get_ras(j), g.get_decs(j), g.get_epoc(j), year);
          decout = get_precess_dec();
      }
      if (g.get_sounam(j).lastIndexOf("Sun") > -1)
      {
        raout = get_sunra(sec);
        decout = get_sundec();
      }
      if (g.get_sounam(j).lastIndexOf("Moon") > -1)
      {
        raout = get_moonra(sec, g);
        decout = get_moondec();
      }
      if (g.get_soutype(j) == 0) {
          has = tim.getGst() - raout - g.get_lon();
          azs = get_radec_az(has, decout, g.get_lat());
          elevs = get_radec_el();
      }
      if (g.get_soutype(j) == 4) {
          azs = get_sattosky_az(g.get_lon(), g.get_ras(j),g);
          elevs = get_sattosky_el();
      }
      if (g.get_soutype(j) == 1) {
          azs = g.get_ras(j);
          elevs = g.get_decs(j);
      }
      if (g.get_sourn() == j)
      {
        azz = azs * 180.0 / Math.PI;
        ell = elevs * 180. / Math.PI;
        g.set_mx(8.0, 3);
        g.set_my(384.0, 3);
        d.dtext(670.0, 200.0, gg, Color.black, d.dcs(g.get_sounam(j), 10));
       if (g.get_sounam(j).lastIndexOf("Sun") == -1
         && g.get_sounam(j).lastIndexOf("Moon") == -1
         && g.get_soutype(j) == 0)
        d.dtext(670.0, 216.0, gg, Color.black,
           get_radecp(g.get_ras(j),g.get_decs(j))
            +"␣"+d.dc(g.get_epoc(j),4,0));
```

```
     else
      d.ppclear(gg, 670.0, 216.0, 130.0);

      d.dtext(670.0, 232.0, gg, Color.black,
        "azel␣" + d.dc(azz, 6, 1)+"␣" + d.dc(ell,6,1)+"␣deg");
      d.dtext(670.0, 56.0, gg, Color.black, "total␣offsets:␣" +
              d.dc(g.get_azoff() + g.get_pazoff(), 6, 1) +
              d.dc(g.get_eloff() + g.get_peloff(), 6, 1));
      d.dtext(670.0, 72.0, gg, Color.black,"pointing␣corr" +
              d.dc(g.get_pazoff(), 5, 1) + d.dc(g.get_peloff(), 5, 1));
      d.dtext(670.0, 88.0, gg, Color.black,"axis␣corr"
        + d.dc(g.get_azcor1(), 5, 1) + d.dc(g.get_elcor1(), 5, 1) +
              "␣␣␣");
      if (g.get_drift() != 0)
      {
        azss = get_radec_az(has + 0.05, decout, g.get_lat());
        elevss = get_radec_el();
        g.set_azoff(azss * 180.0 / Math.PI - azz);
        g.set_eloff(elevss * 180. / Math.PI - ell);
      }
      if ((g.get_track() != 0) || (g.get_drift() != 0))
      {
        g.set_azcmd1(azz + g.get_azoff() + g.get_pazoff());
        g.set_elcmd1(ell + g.get_eloff() + g.get_peloff());
        //g.set_azcmd2(azz + g.get_azoff() + g.get_pazoff()); // Set antenna 2 to
            track with antenna 1
        //g.set_elcmd2(ell + g.get_eloff() + g.get_peloff());
        if (g.get_drift() != 0)
          g.set_track(0);
        g.set_drift(0);
      }
    }
    y = g.get_ylast(j);
    if (y < ylim || elevs > 0.0)
    {
      x = g.get_xlast(j);
      d.spaint(gg, Color.white, x, y, 2);
      d.stext(x, y, gg, Color.white, g.get_sounam(j));
      x = azs * 320.0 / Math.PI;
      y = ylim - elevs * (ylim - ylim0) * 2 / Math.PI;
      if(y < ylim){
       if (g.get_sourn() == j)
       {
         d.stext(x, y, gg, Color.red, g.get_sounam(j));
         d.spaint(gg, Color.red, x, y, 2);
       }
       else
       {
         if (g.get_soutype(j) != 4)
         {
            d.stext(x, y, gg, Color.black, g.get_sounam(j));
            d.spaint(gg, Color.black, x, y, 2);
         }
         if (g.get_soutype(j) == 4)
         {
            d.stext(x, y, gg, Color.black, g.get_sounam(j));
            d.spaint(gg, Color.blue, x, y, 2);
         }
       }
      }
      g.set_xlast(x, j);
      g.set_ylast(y, j);
    }
  }
  galactp(g, d, gg, tim);
}

void galactp(global g, disp d, Graphics gg, time tim)
// plots the galactic plane
{
  double has,
    xg,
    yg,
    xr,
    yr,
```

```
      zr,
      ra,
      dec,
      sec,
      azs,
      elevs;
    int j;
    for (j = 0; j < 360; j += 5)
    {
      xg = Math.cos((j + 2.5) * Math.PI / 180.0);
      yg = Math.sin((j + 2.5) * Math.PI / 180.0);
      xr = xg * Math.sin(27.1 * Math.PI / 180.0);
      yr = yg;
      zr = -xg * Math.cos(27.1 * Math.PI / 180.0);
      dec = Math.atan2(zr, Math.sqrt(xr * xr + yr * yr));
      ra = Math.atan2(yr, xr) + (12.0 + 51.4 / 60.0) * Math.PI / 12.0;
      sec = tim.getTsec(g, d, gg);
      has = tim.getGst() - ra - g.get_lon();
      azs = get_radec_az(has, dec, g.get_lat());
      elevs = get_radec_el();
      y = g.get_gylast(j);
      if (y < ylim || elevs > 0.0)
      {
        x = g.get_gxlast(j);
        d.paint(gg, Color.white, x, y);
        x = azs * 320.0 / Math.PI;
        y = ylim - elevs * (ylim - ylim0) * 2 / Math.PI;
        if(y < ylim)
            d.paint(gg, Color.black, x, y);
        g.set_gxlast(x, j);
        g.set_gylast(y, j);
      }
    }
  }

  double get_sattosky_az(double longw, double sat, global g)
  // coordinate conversion for synchronous orbit satellites
  {
    double re,
      rs,
      satlong,
      ws,
      gs,
      ps,
      wsat,
      gsat,
      psat,
      wss,
      gss,
      pss;
    double west,
      radd,
      zen,
      north;
/* convert from satellite long to sky coords  */
/* input longw;sat; output azim;elev */
      re = 6378.16;
      rs = 42240.4;
      satlong = sat * Math.PI / 180.0;
/* geocentric coords of site ws;gs;ps west;greenwich;pole */
      ws = re * Math.cos(g.get_lat()) * Math.sin(longw);
      gs = re * Math.cos(g.get_lat()) * Math.cos(longw);
      ps = re * Math.sin(g.get_lat());
/* satellite coords wsat;gsat;psat */
      wsat = rs * Math.sin(satlong);
      gsat = rs * Math.cos(satlong);
      psat = 0;
      wss = wsat - ws;
      gss = gsat - gs;
      pss = psat - ps;
      west = wss * Math.cos(longw) - gss * Math.sin(longw);
      radd = gss * Math.cos(longw) + wss * Math.sin(longw);
      zen = radd * Math.cos(g.get_lat()) + pss * Math.sin(g.get_lat());
      north = pss * Math.cos(g.get_lat()) - radd * Math.sin(g.get_lat());
      elev = Math.atan2(zen, Math.sqrt(north * north + west * west));
```

```
    azim = Math.atan2(-west, north);
    if (azim < 0.0)
        azim += Math.PI * 2.0;
    return azim;
}
double get_sattosky_el()
{
  return elev;
}


double get_galactic_ra(double time, double az, double el, global g, time tim)
// convert from azimuth and elavation to galactic coords, ra and dec and
// calculate velocity of the local standard of rest
{
  double north,
    west,
    zen,
    pole,
    rad,
    ha,
    gwest,
    grad,
    gpole,
    rac,
    decc,
    ra,
    dec;
  double ggwest,
    lon0,
    vsun,
    x0,
    y0,
    z0,
    sunlong,
    soulong,
    soulat,
    x,
    y,
    z,
    dp,
    rp;
    decc = -(28.0 + 56.0 / 60.0) * Math.PI / 180.0;
    rac = (17.0 + 45.5 / 60.0) * Math.PI / 12.0;
    dp = 27.1 * Math.PI / 180.0;
    rp = (12.0 + 51.4 / 60.0) * Math.PI / 12.0;
    north = Math.cos(az * Math.PI / 180.0) * Math.cos(el * Math.PI / 180.0);
    west = -Math.sin(az * Math.PI / 180.0) * Math.cos(el * Math.PI / 180.0);
    zen = Math.sin(el * Math.PI / 180.0);
    pole = north * Math.cos(g.get_lat()) + zen * Math.sin(g.get_lat());
    rad = zen * Math.cos(g.get_lat()) - north * Math.sin(g.get_lat());
    dec = Math.atan2(pole, Math.sqrt(rad * rad + west * west));
    ha = Math.atan2(west, rad);
    ra = -ha + tim.getGst() - g.get_lon();
    x0 = 20.0 * Math.cos(18.0 * Math.PI / 12.0) * Math.cos(30.0 * Math.PI / 180.0);
    y0 = 20.0 * Math.sin(18.0 * Math.PI / 12.0) * Math.cos(30.0 * Math.PI / 180.0);
    z0 = 20.0 * Math.sin(30.0 * Math.PI / 180.0); /* sun 20km/s towards ra=18h dec
        =30.0 */
    vsun = -x0 * Math.cos(ra) * Math.cos(dec) - y0 * Math.sin(ra) * Math.cos(dec)
  - z0 * Math.sin(dec);
    x0 = Math.cos(ra) * Math.cos(dec);
    y0 = Math.sin(ra) * Math.cos(dec);
    z0 = Math.sin(dec);
    x = x0;
    y = y0 * Math.cos(23.5 * Math.PI / 180.0) + z0 * Math.sin(23.5 * Math.PI
         / 180.0);
    z = z0 * Math.cos(23.5 * Math.PI / 180.0) - y0 * Math.sin(23.5 * Math.PI
         / 180.0);
    soulat = Math.atan2(z, Math.sqrt(x * x + y * y));
    soulong = Math.atan2(y, x);
    sunlong = (time * 360.0 / (365.25 * 86400.0) + 280.0) * Math.PI / 180.0; /*
        long=280 day 1 */
    vel = vsun - 30.0 * Math.cos(soulat) * Math.sin(sunlong - soulong);
  if (ra > Math.PI * 2.0)
      ra += -Math.PI * 2.0;
```

```
      gwest = Math.cos(decc) * Math.cos(rp - rac);
      grad = Math.cos(decc) * Math.sin(rp - rac);
      ggwest = gwest * Math.sin(dp) - Math.sin(decc) * Math.cos(dp);
      gpole = gwest * Math.cos(dp) + Math.sin(decc) * Math.sin(dp);
      lon0 = (Math.atan2(ggwest, grad)) * 180.0 / Math.PI;
      gwest = Math.cos(dec) * Math.cos(rp - ra);
      grad = Math.cos(dec) * Math.sin(rp - ra);
      ggwest = gwest * Math.sin(dp) - Math.sin(dec) * Math.cos(dp);
      gpole = gwest * Math.cos(dp) + Math.sin(dec) * Math.sin(dp);
      glat = (Math.atan2(gpole, Math.sqrt(ggwest * ggwest + grad * grad))) * 180.0 /
          Math.PI;
      glon = (Math.atan2(ggwest, grad)) * 180.0 / Math.PI - lon0;
    if (glon < 0.0)
        glon += 360.0;
    if (glon > 360.0)
        glon += -360.0;
      raout = ra * 12.0 / Math.PI;
      decout = dec * 180.0 / Math.PI;
    if (raout < 0.0)
        raout += 24.0;
      return raout;
  }
  double get_galactic_dec()
  {
    return decout;
  }
  double get_galactic_glat()
  {
    return glat;
  }
  double get_galactic_glon()
  {
    return glon;
  }
  double get_galactic_vel()
  {
    return vel;
  }

  double antiltel(double az, global g)
  // calculate the elevation correction
  // associated with a tilt in azimuth axis
  {
    double a,
      b;
      a = g.get_azaxis_tilt() * Math.cos((az - g.get_tilt_az()) * Math.PI / 180.0);
      b = g.get_azaxis_tilt() *
      Math.cos((g.get_azlim1_1() - g.get_tilt_az()) * Math.PI / 180.0);
      return (a - b);
  }
  double antiltaz(double az, double el, global g)
  // calculate azimuth correction associated with axis tilts
  {
    double a,
      b,
      x;
    if (el < 89.9)
    {
      a = (g.get_azaxis_tilt() * Math.sin((az - g.get_tilt_az()) * Math.PI / 180.0)
           + g.get_elaxis_tilt()) * Math.tan(el * Math.PI / 180.0) * Math.PI / 180.0;
      a = Math.atan(a);
      b = (g.get_azaxis_tilt() *
           Math.sin((g.get_azlim1_1() - g.get_tilt_az()) * Math.PI / 180.0)
           + g.get_elaxis_tilt()) * Math.tan(g.get_ellim1_1() * Math.PI / 180.0) *
              Math.PI / 180.0;
      b = Math.atan(b);
      x = a - b;
    }
    else
        x = 0.0;
    return (x * 180.0 / Math.PI);
  }

/* Approximate Precession */
  double get_precess_ra(double rain, double decin,
```

```
                        double epin, double epout)
  {
    decout = decin + 0.0000972 * Math.cos(rain) * (epout - epin);
    raout = rain +
    ((0.000223 + 0.0000972 * Math.sin(rain) * Math.tan(decin))
     * (epout - epin));
    return raout;
  }
  double get_precess_dec()
  {
    return decout;
  }

/* Calculate Sun ra and dec (approximate) */
/* see Astronomical Almanac page C24 Sun 1999 */
  double get_sunra(double time)
  {
    double n, g, lon, ecl;
      n = -365.5 + (time - tim.tosec(1999,1,0,0,0)) / 86400.0;
      g = (357.528 + 0.9856003 * n) * Math.PI / 180.0;
      lon = (280.460 + 0.9856474 * n + 1.915 * Math.sin(g) + 0.02
             * Math.sin(2*g)) * Math.PI / 180.0;
      ecl = (23.439 - 0.0000004 * n) * Math.PI / 180.0;
      ra = Math.atan2(Math.sin(lon) * Math.cos(ecl), Math.cos(lon));
      dec = Math.asin(Math.sin(lon) * Math.sin(ecl));
      return ra;
  }
  double get_sundec()
  {
    return dec;
  }

  double get_radec_az(double has, double decs, double lat)
  {
/* convert from sky to antenna coords (azel mount) */
/* input: has,decs,lat
   output: azs=azimuth of source
   elev=elevation of source
 */
    double p,
      w,
      r,
      zen,
      north,
      azs,
      elevs;
      p = Math.sin(decs);
      w = Math.sin(has) * Math.cos(decs);
      r = Math.cos(has) * Math.cos(decs);
      zen = r * Math.cos(lat) + p * Math.sin(lat);
      north = -r * Math.sin(lat) + p * Math.cos(lat);
      elev = Math.atan2(zen, Math.sqrt(north * north + w * w));
      azs = Math.atan2(-w, north);
    if (azs < 0)
        azs = azs + Math.PI * 2.0;
      return azs;
  }
  double get_radec_el()
  {
    return elev;
  }

 String get_radecp(double ra, double dec)
 // make a printable version of ra and dec
{
 int hr,min,sec,deg;
 String rahhmm,h,m,s,d;
    sec = (int)(ra * 12.0 * 3600.0 / Math.PI);
    hr = sec / 3600;
    sec -= hr * 3600;
    min = sec / 60;
    sec = sec - min * 60;
    h = "" + hr;
    if (hr < 10)
      h = "0" + h;
```

```
    m = "" + min;
    if (min < 10)
      m = "0" + m;
    s = "" + sec;
    if (sec < 10)
      s = "0" + s;
    rahhmm = h + ":" + m + ":" + s;
    sec = (int)(Math.abs(dec) * 180.0 * 3600.0 / Math.PI);
    deg = sec / 3600;
    sec -= deg * 3600;
    min = sec / 60;
    sec = sec - min * 60;
    d = "" + deg;
    if (deg < 10)
      d = "0" + d;
    if(dec<0.0)
      d = "-" + d;
    m = "" + min;
    if (min < 10)
      m = "0" + m;
    s = "" + sec;
    if (sec < 10)
      s = "0" + s;
    rahhmm += "␣" + d + ":" + m + ":" + s;
    return(rahhmm);
}
}
```

global.java

```
public class global
// class to store all items which need to be globally available
// used set_ and get_ to distinguish from java library set get
{
  private double pwr[],
    spec[],
    avspec[],
    avspecc[],
    av,
    avc,
    vlsr,
    freqsep,
    bswav,
    bswsq,
    bswlast;
  private double freqa,
    freq0,
    azoff,
    eloff,
    elcor1,
    azcor1,
    calcons,
    beamw,
    azaxis_tilt;
  private double azlim1_1,
    azlim2_1,
    ellim1_1,
    ellim2_1,
    tsys,
    tload,
    tspill,
    elaxis_tilt;
  private long tstart,filepointer;
  private double aznow1,
    elnow1,
    azcmd1,
    elcmd1,
    fcenter,
    lat,
    lon;
  private double pazoff,
    peloff,
    tilt_az;
  private int refresh,
```

```
  fstatus,
  nfreq,
  radiosim,
  azelsim,
  mainten,
  azcount1,
  elcount1,
  recmode,
  xmark,
  mancal;

private double ras[],
  decs[],
  epoc[],
  xlast[],
  ylast[],
  gxlast[];
private int soutype[];
private double gylast[],
  mx[],
  my[];
private String sounam[],
  statnam;
private String ptime = "";
private String cmdstr = "";
private String cmdfile = "srt.cmd";
private String recfile = "";
private int clr,
  key,
  port1,
  drift,
  scan,
  stow,
  atten,
  nsou,
  sourn,
  track;
private int ppos,
  click,
  nclick,
  nsoucat,
  cmdf,
  cmdfline,
  sig,
  bsw,
  calon,
  docal,
  stopproc;
private int bswcycles,
  comerr,
  comerad,
  nmess;
private int docline = 0;
private int doclin2 = 0;
{
  pwr = new double[500];
    spec = new double[500];
    avspec = new double[500];
    avspecc = new double[500];
    ras = new double[500];
    decs = new double[500];
    epoc = new double[500];
    sounam = new String[500];
    xlast = new double[500];
    ylast = new double[500];
    gxlast = new double[500];
    gylast = new double[500];
    mx = new double[20];
    my = new double[20];
    soutype = new int[500];
}
// add code for the file pointer
public void set_filepointer(long value)
{
      filepointer=value;
```

```
}
public long get_filepointer()
{
      return filepointer;
}
public void set_sounam(String name, int index)
{
  sounam[index] = name;
}
public String get_sounam(int index)
{
  return sounam[index];
}
public void set_statnam(String name)
{
  statnam = name;
}
public String get_statnam()
{
  return statnam;
}
public void set_ptime(String name)
{
  ptime = name;
}
public String get_ptime()
{
  return ptime;
}
public void set_cmdstr(String name)
{
  cmdstr = name;
}
public String get_cmdstr()
{
  return cmdstr;
}
public void set_cmdfile(String name)
{
  cmdfile = name;
}
public String get_cmdfile()
{
  return cmdfile;
}
public void set_recfile(String name)
{
  recfile = name;
}
public String get_recfile()
{
  return recfile;
}
public void set_pwr(double power, int index)
{
  pwr[index] = power;
}
public double get_pwr(int index)
{
  return pwr[index];
}
public void set_spec(double power, int index)
{
  spec[index] = power;
}
public double get_spec(int index)
{
  return spec[index];
}
public void set_avspec(double power, int index)
{
  avspec[index] = power;
}
public double get_avspec(int index)
{
```

```
  return avspec[index];
}
public void set_avspecc(double power, int index)
{
  avspecc[index] = power;
}
public double get_avspecc(int index)
{
  return avspecc[index];
}
public void set_ras(double power, int index)
{
  ras[index] = power;
}
public double get_ras(int index)
{
  return ras[index];
}
public void set_decs(double power, int index)
{
  decs[index] = power;
}
public double get_decs(int index)
{
  return decs[index];
}
public void set_epoc(double power, int index)
{
  epoc[index] = power;
}
public double get_epoc(int index)
{
  return epoc[index];
}
public void set_soutype(int i, int index)
{
  soutype[index] = i;
}
public int get_soutype(int index)
{
  return soutype[index];
}
public void set_xlast(double p, int index)
{
  xlast[index] = p;
}
public double get_xlast(int index)
{
  return xlast[index];
}
public void set_ylast(double p, int index)
{
  ylast[index] = p;
}
public double get_ylast(int index)
{
  return ylast[index];
}
public void set_gylast(double p, int index)
{
  gylast[index] = p;
}
public double get_gylast(int index)
{
  return gylast[index];
}
public void set_gxlast(double p, int index)
{
  gxlast[index] = p;
}
public double get_gxlast(int index)
{
  return gxlast[index];
}
public void set_mx(double p, int index)
```

```
{
  mx[index] = p;
}
public double get_mx(int index)
{
  return mx[index];
}
public void set_my(double p, int index)
{
  my[index] = p;
}
public double get_my(int index)
{
  return my[index];
}
public void set_av(double p)
{
  av = p;
}
public double get_av()
{
  return av;
}
public void set_bswav(double p)
{
  bswav = p;
}
public double get_bswav()
{
  return bswav;
}
public void set_bswsq(double p)
{
  bswsq = p;
}
public double get_bswsq()
{
  return bswsq;
}
public void set_bswlast(double p)
{
  bswlast = p;
}
public double get_bswlast()
{
  return bswlast;
}
public void set_vlsr(double p)
{
  vlsr = p;
}
public double get_vlsr()
{
  return vlsr;
}
public void set_avc(double p)
{
  avc = p;
}
public double get_avc()
{
  return avc;
}
public void set_freqsep(double p)
{
  freqsep = p;
}
public double get_freqsep()
{
  return freqsep;
}
public void set_freqa(double p)
{
  freqa = p;
}
```

```
public double get_freqa()
{
  return freqa;
}
public void set_freq0(double p)
{
  freq0 = p;
}
public double get_freq0()
{
  return freq0;
}
public void set_azoff(double p)
{
  azoff = p;
}
public double get_azoff()
{
  return azoff;
}
public void set_eloff(double p)
{
  eloff = p;
}
public double get_eloff()
{
  return eloff;
}
public void set_pazoff(double p)
{
  pazoff = p;
}
public double get_pazoff()
{
  return pazoff;
}
public void set_peloff(double p)
{
  peloff = p;
}
public double get_peloff()
{
  return peloff;
}
public void set_elcor1(double p)
{
  elcor1 = p;
}
public double get_elcor1()
{
  return elcor1;
}
public void set_azcor1(double p)
{
  azcor1 = p;
}
public double get_azcor1()
{
  return azcor1;
}
public void set_calcons(double p)
{
  calcons = p;
}
public double get_calcons()
{
  return calcons;
}
public void set_beamw(double p)
{
  beamw = p;
}
public double get_beamw()
{
  return beamw;
```

```
}
public void set_azaxis_tilt(double p)
{
  azaxis_tilt = p;
}
public double get_azaxis_tilt()
{
  return azaxis_tilt;
}
public void set_elaxis_tilt(double p)
{
  elaxis_tilt = p;
}
public double get_elaxis_tilt()
{
  return elaxis_tilt;
}
public void set_tilt_az(double p)
{
  tilt_az = p;
}
public double get_tilt_az()
{
  return tilt_az;
}
public void set_azlim1_1(double p)
{
  azlim1_1 = p;
}
public double get_azlim1_1()
{
  return azlim1_1;
}
public void set_azlim2_1(double p)
{
  azlim2_1 = p;
}
public double get_azlim2_1()
{
  return azlim2_1;
}
public void set_ellim2_1(double p)
{
  ellim2_1 = p;
}
public double get_ellim2_1()
{
  return ellim2_1;
}
public void set_ellim1_1(double p)
{
  ellim1_1 = p;
}
public double get_ellim1_1()
{
  return ellim1_1;
}
public void set_tsys(double p)
{
  tsys = p;
}
public double get_tsys()
{
  return tsys;
}
public void set_tload(double p)
{
  tload = p;
}
public double get_tload()
{
  return tload;
}
public void set_tspill(double p)
{
```

```
  tspill = p;
}
public double get_tspill()
{
  return tspill;
}
public void set_tstart(long p)
{
  tstart = p;
}
public long get_tstart()
{
  return tstart;
}
public void set_aznow1(double p)
{
  aznow1 = p;
}
public double get_aznow1()
{
  return aznow1;
}
public void set_elnow1(double p)
{
  elnow1 = p;
}
public double get_elnow1()
{
  return elnow1;
}
public void set_azcmd1(double p)
{
  azcmd1 = p;
}
public double get_azcmd1()
{
  return azcmd1;
}
public void set_elcmd1(double p)
{
  elcmd1 = p;
}
public double get_elcmd1()
{
  return elcmd1;
}
public void set_fcenter(double p)
{
  fcenter = p;
}
public double get_fcenter()
{
  return fcenter;
}
public void set_lat(double p)
{
  lat = p;
}
public double get_lat()
{
  return lat;
}
public void set_lon(double p)
{
  lon = p;
}
public double get_lon()
{
  return lon;
}
public void set_refresh(int p)
{
  refresh = p;
}
public int get_refresh()
```

```
{
  return refresh;
}
public void set_fstatus(int p)
{
  fstatus = p;
}
public int get_fstatus()
{
  return fstatus;
}
public void set_nfreq(int p)
{
  nfreq = p;
}
public int get_nfreq()
{
  return nfreq;
}
public void set_radiosim(int p)
{
  radiosim = p;
}
public int get_radiosim()
{
  return radiosim;
}
public void set_azelsim(int p)
{
  azelsim = p;
}
public int get_azelsim()
{
  return azelsim;
}
public void set_mainten(int p)
{
  mainten = p;
}
public int get_mainten()
{
  return mainten;
}
public void set_azcount1(int p)
{
  azcount1 = p;
}
public int get_azcount1()
{
  return azcount1;
}
public void set_elcount1(int p)
{
  elcount1 = p;
}
public int get_elcount1()
{
  return elcount1;
}
public void set_mancal(int p)
{
  mancal = p;
}
public int get_mancal()
{
  return mancal;
}
public void set_recmode(int p)
{
  recmode = p;
}
public int get_recmode()
{
  return recmode;
}
```

```
public void set_xmark(int p)
{
  xmark = p;
}
public int get_xmark()
{
  return xmark;
}
public void set_docal(int p)
{
  docal = p;
}
public int get_docal()
{
  return docal;
}
public void set_stopproc(int p)
{
  stopproc = p;
}
public int get_stopproc()
{
  return stopproc;
}
public void set_click(int p)
{
  click = p;
}
public int get_click()
{
  return click;
}
public void set_nclick(int p)
{
  nclick = p;
}
public int get_nclick()
{
  return nclick;
}
public void set_ppos(int p)
{
  ppos = p;
}
public int get_ppos()
{
  return ppos;
}
public void set_clr(int p)
{
  clr = p;
}
public int get_clr()
{
  return clr;
}
public void set_key(int p)
{
  key = p;
}
public int get_key()
{
  return key;
}
public void set_port1(int p)
{
  port1 = p;
}
public int get_port1()
{
  return port1;
}
public void set_drift(int p)
{
  drift = p;
```

```
}
public int get_drift()
{
  return drift;
}
public void set_scan(int p)
{
  scan = p;
}
public int get_scan()
{
  return scan;
}
public void set_stow(int p)
{
  stow = p;
}
public int get_stow()
{
  return stow;
}
public void set_atten(int p)
{
  atten = p;
}
public int get_atten()
{
  return atten;
}
public void set_nsou(int p)
{
  nsou = p;
}
public int get_nsou()
{
  return nsou;
}
public void set_sourn(int p)
{
  sourn = p;
}
public int get_sourn()
{
  return sourn;
}
public void set_track(int p)
{
  track = p;
}
public int get_track()
{
  return track;
}
public void set_nsoucat(int p)
{
  nsoucat = p;
}
public int get_nsoucat()
{
  return nsoucat;
}
public void set_cmdf(int p)
{
  cmdf = p;
}
public int get_cmdf()
{
  return cmdf;
}
public void set_cmdfline(int p)
{
  cmdfline = p;
}
public int get_cmdfline()
{
```

```
    return cmdfline;
  }
  public void set_sig(int p)
  {
    sig = p;
  }
  public int get_sig()
  {
    return sig;
  }
  public void set_bsw(int p)
  {
    bsw = p;
  }
  public int get_bsw()
  {
    return bsw;
  }
  public void set_bswcycles(int p)
  {
    bswcycles = p;
  }
  public int get_bswcycles()
  {
    return bswcycles;
  }
  public void set_comerr(int p)
  {
    comerr = p;
  }
  public int get_comerr()
  {
    return comerr;
  }
  public void set_comerad(int p)
  {
    comerad = p;
  }
  public int get_comerad()
  {
    return comerad;
  }
  public void set_nmess(int p)
  {
    nmess = p;
  }
  public int get_nmess()
  {
    return nmess;
  }
  public void set_docline(int p)
  {
    docline = p;
  }
  public int get_docline()
  {
    return docline;
  }
  public void set_doclin2(int p)
  {
    doclin2 = p;
  }
  public int get_doclin2()
  {
    return doclin2;
  }
  public void set_calon(int p)
  {
    calon = p;
  }
  public int get_calon()
  {
    return calon;
  }
}
```

plots.java

```
import java.awt.*;
public class plots
// class for plotting data
{
  double xlim = 640.0;
  double ylim = 415.0;
  double ylim0 = 200.0;
  void plotbox(global g, disp d, Graphics gg)
  {
    int x,
      y,
      yy;
    double x1,y1;
    String str4;
    for (x = 1; x < 36; x++)
    {
      x1 = (double)(x * xlim / 36.0);
  if(x%2 == 0)
    d.stext(x1-8, ylim+12.0, gg, Color.black, d.dc(x*10.0,3,0));
  if(x == 17)
    d.stext(x1-8, ylim+22.0, gg, Color.black,"azimuth␣(deg)");
    }
    for (y = 0; y < 2; y++)
        d.lpaint(gg, Color.black, 0.0, (double)(y * ylim),
                 xlim, (double)(y * ylim));
    for (x = 0; x < 2; x++)
        d.lpaint(gg, Color.black, (double)(x * xlim), 0.0,
                   (double)(x * xlim), ylim);
    for (x = 1; x < 36; x++)
    {
      x1 = (double)(x * xlim / 36.0);
      d.lpaint(gg, Color.black, x1, ylim, x1, ylim - 9.0);
    }
    for (y = 0; y <= 9; y++)
    {
      y1 = (double)(ylim - y * (ylim - ylim0) / 9.0);
      d.lpaint(gg, Color.black, 0.0, y1, 9.0 , y1);
      d.lpaint(gg, Color.black, xlim, y1, xlim - 9.0 , y1);
      d.stext(xlim+1.0, y1+4.0, gg, Color.black, d.dc(y*10.0,3,0));
    }
    for (y = 0; y < 9; y++)
    {
      y1 = (double)(ylim - 0.75 * (ylim - ylim0)) + y * 10.0;
      str4="E";
      if(y==1) str4="l";
      if(y==2) str4="e";
      if(y==3) str4="v";
      if(y==4) str4="a";
      if(y==5) str4="t";
      if(y==6) str4="i";
      if(y==7) str4="o";
      if(y==8) str4="n";
      d.stext(20.0, y1, gg, Color.black, str4);
    }
    d.stext(xlim*0.5-15.0, ylim-11.0, gg, Color.black, "south");
      y = (int)(ylim - g.get_ellim1_1() * (ylim - ylim0) / 90.0);
    for (x = (int)(g.get_azlim1_1() * xlim / 360.0);
         x <= (int)(g.get_azlim2_1() * xlim / 360.0); x++)
        d.paint(gg, Color.getHSBColor((float)0.0, (float)0.0, (float)x % 2),
                  (double)x, (double)y);
      y = (int)(ylim - (180.0 - g.get_ellim2_1()) * (ylim - ylim0) / 90.0);
    for (x = 40; x <= (int)((g.get_azlim2_1() - 180.0) * xlim / 360.0); x++)
    {
      d.paint(gg, Color.getHSBColor((float)0.0, (float)0.0, (float)x % 2),
               (double)x, (double)y);
    }
    for (x = (int)((g.get_azlim1_1() + 180.0) * xlim / 360.0); x <= (int)xlim; x++)
    {
      d.paint(gg, Color.getHSBColor((float)0.0, (float)0.0, (float)x % 2),
               (double)x, (double)y);
    }
    x = (int)(g.get_azlim1_1() * xlim / 360.0);
    y = (int)(ylim - g.get_ellim1_1() * (ylim - ylim0) / 90.0);
```

```
    for (yy = (int)ylim0; yy < y; yy++)
    {
      d.paint(gg, Color.getHSBColor((float)0.0, (float)0.0, (float)yy % 2),
              (double)x, (double)yy);
    }
    x = (int)((g.get_azlim1_1() + 180.0) * xlim / 360.0);
    y = (int)(ylim - (180.0 - g.get_ellim2_1()) * (ylim - ylim0) / 90.0);
    for (yy = (int)ylim0; yy < y; yy++)
    {
      d.paint(gg, Color.getHSBColor((float)0.0, (float)0.0, (float)yy % 2),
              (double)x, (double)yy);
    }
    x = (int)((g.get_azlim2_1()) * xlim / 360.0);
    y = (int)(ylim - g.get_ellim1_1() * (ylim - ylim0) / 90.0);
    for (yy = (int)ylim0; yy < y; yy++)
    {
      d.paint(gg, Color.getHSBColor((float)0.0, (float)0.0, (float)yy % 2),
              (double)x, (double)yy);
    }
    x = (int)((g.get_azlim2_1() - 180.0) * xlim / 360.0);
    y = (int)(ylim - (180.0 - g.get_ellim2_1()) * (ylim - ylim0) / 90.0);
    for (yy = (int)ylim0; yy < y; yy++)
    {
      d.paint(gg, Color.getHSBColor((float)0.0, (float)0.0, (float)yy % 2),
              (double)x, (double)yy);
    }
    d.stext(10.0,(double)y + 4.0,gg,Color.black,"limits");
    d.lpaint(gg, Color.black, 660.0, 0.0, 660.0, ylim+88.0);
    d.lpaint(gg, Color.black, 660.0, ylim+88.0, 800.0, ylim+88.0);
    d.lpaint(gg, Color.black, 660.0, 0.0, 800.0, 0.0);
    d.lpaint(gg, Color.black, 0.0, ylim+25.0, 640.0, ylim+25.0);
    d.lpaint(gg, Color.black, 320.0, ylim+25.0, 320.0, ylim+46.0);
    d.lpaint(gg, Color.black, 640.0, ylim+25.0, 640.0, ylim+46.0);
    d.lpaint(gg, Color.black, 0.0, ylim+46.0, 640.0, ylim+46.0);
    d.stext(665.0,10.0,gg,Color.black,"Antenna␣coordinates:");
    d.lpaint(gg, Color.black, 660.0, 123.0, 800.0, 123.0);
    d.stext(665.0,136.0,gg,Color.black,"Time:");
    d.lpaint(gg, Color.black, 660.0, 172.0, 800.0, 172.0);
    d.stext(665.0,184.0,gg,Color.black,"Source:");
//    d.stext(665.0,248.0,gg,Color.black,"Center frequency:");
//    d.dtext(670.0,264.0,gg,Color.black,d.dc(g.get_fcenter(),8,2) + " MHz");
//    d.dtext(665.0,280.0,gg,Color.black,"spacing: "+d.dc(g.get_freqsep(),8,2) + "
    MHz");
//    d.dtext(665.0,296.0,gg,Color.black,"number bins: "+g.get_nfreq());

        // JJ added 2/13/01
        d.lpaint(gg, Color.black, 0.0, ylim0, xlim, ylim0);  // paint a line on top
            of plot box


  }

}
```

procs.java

```
import java.awt.*;
public class procs
// class of procedures
{
  private plots p = new plots();
  private geom geom = new geom();
  private time t = new time();
  private sport s;
  private double ylim = 415.0;
  public procs(sport sp)
  {
    s = sp;
  }
  void spectra(global g, disp d, Graphics gg)
  // take a spetrum
  {
    int i;
    double freqf,
```

```
  secc,
  avp,
  pwr,
  pp,
  lst;
if (g.get_scan() >= 26)
{
  g.set_scan(0);
}
if (g.get_scan() != 0)
{
  g.set_azoff(g.get_beamw() * 0.5 *
              (double)((g.get_scan() - 1) % 5 - 2) /
              Math.cos(g.get_elcmd1() * Math.PI / 180.0));
  g.set_eloff(g.get_beamw() * 0.5 * (double)((g.get_scan() - 1) / 5 - 2));
  g.set_pwr(0.0, g.get_scan());
  g.set_scan(g.get_scan() + 1);
}
if (g.get_bsw() != 0)
{
  g.set_sig(-g.get_sig());
  if (g.get_sig() == -1)
  {
    g.set_azoff(g.get_beamw() * g.get_bsw() /
                Math.cos(g.get_elcmd1() * Math.PI / 180.0));
    g.set_bsw(-g.get_bsw());
  }
  else
    g.set_azoff(0.0);
}
else
  g.set_sig(1);
p.plotbox(g, d, gg);
geom.setsounam(g, d, gg, t);
g.set_elcor1(geom.antiltel(g.get_azcmd1(), g));
g.set_azcor1(geom.antiltaz(g.get_azcmd1(), g.get_elcmd1(), g));
s.azel(g.get_azcmd1() + g.get_azcor1(),
       g.get_elcmd1() + g.get_elcor1(), g, d, gg); // command antenna
if (g.get_track() == 0)
{
  g.set_scan(0);
  g.set_bsw(0);
}
if (g.get_clr() != 0)
{
  for (i = 0; i < g.get_nfreq(); i++)
  {
    g.set_avspec(0.0, i);
    g.set_avspecc(0.0, i);
  }
  g.set_av(0.0);
  g.set_avc(0.0);
  g.set_bswav(0.0);
  g.set_bswsq(0.0);
  g.set_bswlast(0.0);
  g.set_bswcycles(0);
  if (g.get_clr() == -1)
    d.pclear(gg, 0.0, 0.0, 640.0, 440.0);
  g.set_clr(0);
}

if (g.get_track() != 0)
{
  d.dtext(340.0, ylim+40.0, gg, Color.black, "Status:␣tracking␣␣");
  d.set_bc(Color.green, 1);
}
else
  d.set_bc(Color.black, 1);


if(g.get_comerr() > 0 || g.get_comerad() > 0)
  d.dtext(440.0, ylim+40.0, gg, Color.blue, "ERRORS:␣"+
   "gnd␣"+g.get_comerr()+"␣radio␣"+g.get_comerad());
avp = 0.0;
for (i = 0; i < g.get_nfreq(); i++)
```

```
    {
      freqf = g.get_fcenter() + (double)(i - g.get_nfreq() / 2) * g.get_freqsep();
  //    g.set_spec(s.radio(freqf, g, d, gg, o), i);  // get power from radio
        g.set_spec(0.0,0); // This is a line to try and fool the one above
      if (i == 0)
        g.set_freq0(g.get_freqa());
      if (g.get_click() != 0)
      {
        d.stext(520.0, ylim+60.0, gg, Color.blue, "waiting␣end␣of␣freq␣seq");
      }
      pwr = g.get_spec(i);
      if (pwr > 0.0) {
      avp += pwr;
      if (g.get_sig() == 1)
        g.set_avspec(g.get_avspec(i) + pwr, i);
      else
        g.set_avspecc(g.get_avspecc(i) + pwr, i);
      if (g.get_scan() != 0)
        g.set_pwr(g.get_pwr(g.get_scan() - 1) + pwr, g.get_scan() - 1);
       }
    }
    avp = avp / (double)g.get_nfreq();
    if (g.get_bswlast() > 0.0){
       if (g.get_sig() == 1)
           pp = avp - g.get_bswlast();
       else
          pp = g.get_bswlast() - avp;
      g.set_bswav(g.get_bswav() + pp);
      g.set_bswsq(g.get_bswsq() + pp*pp);
    g.set_bswcycles(g.get_bswcycles() + 1);
      }
    g.set_bswlast(avp);
    if (g.get_click() != 0)
    {
      d.stext(520.0, ylim+60.0, gg, Color.white, "waiting␣end␣of␣freq␣seq");
    }
    if (g.get_sig() == 1)
      g.set_av(g.get_av() + 1);
    else
      g.set_avc(g.get_avc() + 1);
    if (g.get_clr() != 0)
    {
      for (i = 0; i < g.get_nfreq(); i++)
      {
        g.set_avspec(0.0, i);
        g.set_avspecc(0.0, i);
      }
      g.set_av(0.0);
      g.set_avc(0.0);
      g.set_clr(0);
    }
   // p.plotspec(0, g, d, gg, o);
   // p.plotspec(1, g, d, gg, o);
    secc = (double)t.getTsec(g, d, gg);
    lst = (t.getGst() - g.get_lon()) * 12.0 / Math.PI;
    if (lst < 0.0)
      lst += 24.0;
    if (lst > 24.0)
      lst -= 24.0;
    d.dtext(670.0, 167.0, gg, Color.black, "LST␣" + d.dc(lst, 4, 1) + "␣hrs");
    d.dtext(710.0, 136.0, gg, Color.black, "UTdate␣"+ t.getMonthday()+"␣");
/*    if (g.get_tsys() > 0.0)
      d.dtext(665.0, 312.0, gg, Color.black,"tsys: "
       + d.dc(g.get_tsys(), 5, 0)+" K              ");
    else
       d.dtext(665.0, 312.0, gg, Color.black,"tsys: not yet calibrated ");*/
    if (g.get_scan() == 0 && g.get_stow() == 0 && g.get_track() == 0)
      d.dtext(340.0, ylim+40.0, gg, Color.black, "Status:␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣");
    if(g.get_track() == 0)
      d.set_bc(Color.black, 1);
//    p.plotp(secc, g, d, gg);
  }

}
```

sport.java

```
import javax.comm.*;
import java.awt.*;
import java.util.*;
import java.text.*;
import java.lang.*;
import java.io.*;

public class sport
// serial port class - talks to the motor control and radio
// STAMP microcontrollers
// must talk to one stamp at a time and always get response
// before talking again
{
  private geom geom = new geom();
  private time t = new time();
  static CommPortIdentifier portId;
  static Enumeration portList;
  InputStream inputStream1;
  static OutputStream outputStream1;
  SerialPort serialPort1;
  private double ylim = 415.0;
  private double ylim0 = 200.0;

  public sport(global g)
  {
    if (g.get_azelsim() == 0 || g.get_radiosim() == 0)
    {
      portList = CommPortIdentifier.getPortIdentifiers();
      while (portList.hasMoreElements())
      {
        portId = (CommPortIdentifier) portList.nextElement();
        if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL)
        {
          if (portId.getName().equals("COM" + g.get_port1()))
//          if (portId.getName().equals("COM3"))
          {
            try
            {
              serialPort1 = (SerialPort) portId.open("SRT1", 2000);
            }
            catch(PortInUseException e)
            {
            System.out.println(e);
            System.out.println("exit␣window␣and␣try␣again");
            System.exit(0);
            }
            try
            {
              outputStream1 = serialPort1.getOutputStream();
            }
            catch(IOException e)
            {
            System.out.println(e);
            }
            try
            {
              inputStream1 = serialPort1.getInputStream();
            }
            catch(IOException e)
            {
            System.out.println(e);
            }
            try
            {
              serialPort1.setSerialPortParams(2400,
                                              SerialPort.DATABITS_8,
                                              SerialPort.STOPBITS_1,
                                              SerialPort.PARITY_NONE);
            }
            catch(UnsupportedCommOperationException e)
            {
            System.out.println(e);
            }
```

```
        }


      }
    }
  }
}

void azel(double az1, double el1, global g, disp d, Graphics gg)
// command antenna movement
{
  int i,
    k,
    kk,
    n1,
    n2,
    mm1,
    axis,
    count1,
    rcount1,
    flip1;
  double scale,
    azz1,
    ell1,
    ra,
    dec,
    glat,
    glon,
    vel,
    sec,
    x,
    y;
  int j1;
  char m[] = new char[80];    /* avoid byte array compiler bug */
  char recv1[] = new char[80];
  String str,
    str2;
  StringTokenizer parser1;
  String str3;
 //System.out.println("WASSUPPPPPPPPPPPPPPPPPP");
    mm1  = count1  = 0;
    d.dtext(224.0, 16.0,gg,Color.blue,"Antenna␣1␣information");//Antenna 1 area
    d.lpaint(gg, Color.blue,224.0,17.0,329.0,17.0);
    //d.lpaint(gg, Color.blue,224.0,17.0,319.0,17.0);
    d.dtext(226.0, 32.0, gg, Color.black, "cmd␣"
      + d.dc(az1, 5, 1) + "␣" + d.dc(el1, 5, 1) + "␣deg"); //Antenna 1 command
  if (g.get_azelsim() == 0)
  {
    str = "antenna␣drive␣status:";
    if (g.get_comerr() > 0) {
      str += "␣comerr=␣" + g.get_comerr();
      }
  }
  else {
      str = "antenna␣drive␣simulated";
  }
  d.dtext(226.0, 64.0, gg, Color.black,str);//Antenna 1 drive status
  if ((az1 < g.get_azlim1_1() && az1 > g.get_azlim2_1() - 180.0) ||
      (az1 < g.get_azlim1_1() + 180.0 && az1 > g.get_azlim2_1()) ||
      (az1 > g.get_azlim1_1() && az1 < g.get_azlim2_1() && el1 < g.get_ellim1_1())
           ||
       ((az1 > g.get_azlim1_1() + 180.0 || az1 < g.get_azlim2_1() - 180.0)
        && el1 < 180.0 - g.get_ellim2_1()))
  {
    d.dtext(226.0, 80.0,gg, Color.red, "cmd␣out␣of␣limits"); //Antenna 1 error msg
    g.set_track(0);
    //System.out.println(g.get_azlim1_1()+" "+g.get_azlim2_1());
    return;
  }

  flip1 = 0;
  if (az1 > 270.0)
  {
    az1 -= 180.0;
    el1 = 180.0 - el1;
```

```
      flip1 = 1;
    }

    if (az1 < g.get_azlim1_1() && flip1 == 0)
    {
      az1 += 180.0;
      el1 = 180.0 - el1;
      flip1 = 1;
    }


    azz1 = az1 - g.get_azlim1_1();
    ell1 = el1 - g.get_ellim1_1();

    scale = 52.0 * 27.0 / 120.0;
/* mm=1=clockwize incr.az mm=0=ccw mm=2= down when pointed south */
    for (axis = 0; axis < 2; axis++)
    {
      if (axis == 0)
      {
        if (azz1 * scale > g.get_azcount1())
        {
          mm1 = 1;
          count1 = (int)(azz1 * scale - g.get_azcount1());
        }

        if (azz1 * scale <= g.get_azcount1())
        {
          mm1 = 0;
          count1 = (int)(g.get_azcount1() - azz1 * scale);
        }

      }
      else
      {
        if (ell1 * scale > g.get_elcount1())
        {
          mm1 = 3;
          count1 = (int)(ell1 * scale - g.get_elcount1());
        }

        if (ell1 * scale <= g.get_elcount1())
        {
          mm1 = 2;
          count1 = (int)(g.get_elcount1() - ell1 * scale);
        }

      }
      if (g.get_stow() != 0)
      {
        if (axis == 0) {
          mm1 = 0;
        }
        if (axis == 1) {
          mm1 = 2;
        }
        count1 = 5000;
        flip1 = 0;
      }
      if (count1 > 0)
      {
      if (count1 > 1){
          d.dtext(340.0, ylim+40.0, gg, Color.black, "Status:␣slewing␣␣␣");
          d.set_bc(Color.black, 1);
          d.set_bc(Color.black, 0);
        }
        x = g.get_xlast(0);
        y = g.get_ylast(0);
        // This is apparently for the antenna cross
        d.lpaint(gg, Color.white, (double)(x - 10), (double)y,
                 (double)(x + 10), (double)y);
        d.lpaint(gg, Color.white, (double)x, (double)(y - 10),
                 (double)x, (double)(y + 10));
        x = (int)(g.get_azcmd1() * 640.0 / 360.0);
        y = (int)(ylim - g.get_elcmd1() * (ylim - ylim0) / 90);
```

```
  g.set_xlast(x, 0);
  g.set_ylast(y, 0);
  d.lpaint(gg, Color.yellow, (double)(x - 10), (double)y,
           (double)(x + 10), (double)y);
  d.lpaint(gg, Color.yellow, (double)(x), (double)(y - 10),
           (double)(x), (double)(y + 10));

  str = "␣␣move␣" + mm1 + "␣" + count1 + "\n"; /* need space at start and end
      */
  n1 = 0;
  if (g.get_azelsim() != 0)
  {
    if (((axis == 0 && g.get_azcount1() >= 0)
         || (axis != 0 && g.get_elcount1() >= 0)) && count1 < 5000)
      str2 = "M␣" + count1 + "\n";
    else
      str2 = "T␣" + count1 + "\n";


    str2.getChars(0, str2.length(), recv1, 0);
    n1 = str2.length();
  }
  d.dtext(226.0, 96.0, gg, Color.black,
          "trans␣" + str.substring(0, str.length() - 1) + "␣␣␣␣␣");// Antenna
               1 transmit
  // Add another ppclear if you want to erase the last received message
  d.ppclear(gg, 226.0,112.0,180.0);
  j1 = 0;
  kk = -1;
  if (g.get_azelsim() == 0)
  {
     try
      {
        serialPort1.enableReceiveTimeout(1000);
      }
      catch(UnsupportedCommOperationException e)
      {
        System.out.println(e);
      }
    try
    {
      outputStream1.write(str.getBytes());
      System.out.print("trans->"+str);
      j1  = n1  = rcount1  = kk = 0;
      while (kk >= 0 && kk < 300)
      {
        d.ppclear(gg, 226.0, 80.0, 180.0); //Clear error msg
        if (axis == 0) {
          d.dtext(226.0, 128.0, gg, Color.black, "waiting␣on␣azimuth␣␣␣" + kk);
        }
        else {
                          d.dtext(226.0, 128.0, gg, Color.black, "waiting␣on␣
                              elevation␣" + kk);
                          }

        j1 = inputStream1.read();
//        System.out.println("j1 = "+j1+" j2 = "+j2);
        kk++;
        if (j1 >= 0)
        {
          recv1[n1] = (char)j1;
          n1++;
        }

        if (n1 > 0 && j1 == -1 )
          kk = -1;         // end of message
        t.getTsec(g, d, gg);
      }
      d.ppclear(gg, 226.0, 128.0, 180.0); //Clear wait status box
    }
    catch(IOException e)
    {
      System.out.println(e);
    }
    // no need to close
```

```
}
str3=String.copyValueOf(recv1,0,n1-1); // Print out recv array JJ 1/30/01
        System.out.println("recv->␣"+str3); // Print above line
if (kk != -1 || (recv1[0] != 'M' && recv1[0] != 'T'))
{
  d.dtext(226.0, 32.0, gg, Color.red, "comerr␣j=" + j1 + "␣n=" + n1 + "mm" +
      count1);
        // Antenna 1 communication error box
  g.set_comerr(g.get_comerr() + 1);
  if(g.get_mainten() == 0)
     g.set_stow(1);
  return;
}

if (g.get_azelsim() != 0 && g.get_azelsim() < 10)
{
  try
  {
    Thread.sleep(100);
  }
  catch(InterruptedException e)
  {
    System.out.println(e);
  }
}
str = String.copyValueOf(recv1, 0, n1 - 1);
// May need a ppclear here if received box is not being erased when you want
    it to be
d.dtext(226.0, 112.0, gg, Color.black, "recv␣" + str); // Antenna 1 received
    box
parser1 = new StringTokenizer(str);
try {
      str2 = parser1.nextToken();
    }
catch(NoSuchElementException e)
    {
    }
rcount1 = 0;
try {
str2 = parser1.nextToken();
rcount1 = Integer.valueOf(str2).intValue();
    }
catch(NoSuchElementException e)
    {
    }
if (rcount1 != count1 && g.get_stow() == 0)
{
  d.dtext(226.0, 128.0, gg, Color.red, "lost␣count␣goto␣Stow"); // wait
      status box info
  d.dtext(352.0, 128.0, gg, Color.red, "lost␣count␣goto␣Stow"); // wait
      status box info
  d.dtext(440.0, ylim+40.0, gg, Color.blue, "ERROR␣(Antenna1):␣␣");
  d.dtext(8.0, ylim+60.0, gg, Color.black, "received␣"+
  rcount1+"␣counts␣out␣of␣"+count1+"␣counts␣expected");
  if(mm1==1)
  d.dtext(8.0, ylim+76.0, gg, Color.black,
   "while␣going␣clockwise␣in␣azimuth");
  if(mm1==0)
  d.dtext(8.0, ylim+76.0, gg, Color.black,
   "while␣going␣counter-clockwise␣in␣azimuth");
  if(mm1==3)
  d.dtext(8.0, ylim+76.0, gg, Color.black,
   "while␣going␣clockwise␣in␣elevation");
  if(mm1==2)
  d.dtext(8.0, ylim+76.0, gg, Color.black,
   "while␣going␣counter-clockwise␣in␣elevation");
  d.dtext(8.0, ylim+92.0, gg, Color.black,
   "motor␣stalled␣or␣limit␣prematurely␣reached");

  if(g.get_mainten() == 0)
     g.set_stow(1);
  return;
}
if (axis == 1 && g.get_stow() != 0)
{
```

```
      g.set_elcount1(0);
      g.set_elnow1(g.get_ellim1_1());
    }
    if (axis == 0 && g.get_stow() != 0)
    {
      g.set_azcount1(0);
      g.set_aznow1(g.get_azlim1_1());
    }
    if (recv1[0] == 'T' && g.get_stow() == 0)
    {
      d.dtext(226.0, 112.0,gg, Color.black, "timeout␣from␣antenna");  //status
          box
    }
    if (recv1[0] == 'M')
    {
      if (axis == 0)
      {
        if (mm1 == 1)
          g.set_azcount1(g.get_azcount1() + count1);
        else
          g.set_azcount1(g.get_azcount1() - count1);
      }
      if (axis == 1)
      {
        if (mm1 == 3)
          g.set_elcount1(g.get_elcount1() + count1);
        else
          g.set_elcount1(g.get_elcount1() - count1);
      }
    }


    if (g.get_azelsim() == 0)
    {
      try
      {
        Thread.sleep(1000);
      }
      catch(InterruptedException e)
      {
        System.out.println(e);
      }
    }
  }

}
  g.set_aznow1(g.get_azlim1_1() - g.get_azcor1() + g.get_azcount1() / scale);
  if (flip1 != 0)
        {
          if (g.get_aznow1() >= 180.0)
            g.set_aznow1(g.get_aznow1() - 180.0);
          else
            g.set_aznow1(g.get_aznow1() + 180.0);
        }


  g.set_elnow1(g.get_ellim1_1() - g.get_elcor1() + g.get_elcount1() / scale);

  if (flip1 != 0)
    g.set_elnow1(180.0 - g.get_elnow1());


d.dtext(670.0, 40.0, gg, Color.black,
 "azel␣␣" + d.dc(g.get_aznow1(), 5, 1) + "␣" + d.dc(g.get_elnow1(), 5, 1)+
  "␣deg");
d.dtext(226.0, 48.0, gg, Color.black,
 "azel␣␣" + d.dc(g.get_aznow1(), 5, 1) + "␣" + d.dc(g.get_elnow1(), 5, 1)+
  "␣deg"); // Antenna 1 current position
// This is the antenna cross stuff
x = g.get_xlast(0);
y = g.get_ylast(0);
d.lpaint(gg, Color.white, (double)(x - 10), (double)y,
         (double)(x + 10), (double)y);
d.lpaint(gg, Color.white, (double)x, (double)(y - 10),
         (double)x, (double)(y + 10));
```

```
    x = (int)(g.get_aznow1() * 640.0 / 360.0);
    y = (int)(ylim - g.get_elnow1() * (ylim - ylim0) / 90);
    g.set_xlast(x, 0);
    g.set_ylast(y, 0);
    d.lpaint(gg, Color.red, (double)(x - 10), (double)y,
             (double)(x + 10), (double)y);
    d.lpaint(gg, Color.red, (double)x, (double)(y - 10),
             (double)x, (double)(y + 10));

    if (g.get_aznow1() == g.get_azlim1_1() && g.get_elnow1() == g.get_ellim1_1())
    {
      d.set_bc(Color.green, 0);
      d.dtext(340.0, ylim+40.0, gg, Color.black, "Status:␣at␣stow");
    }
    else
      d.set_bc(Color.black, 0);
    if (g.get_stow() != 0)
    {
      g.set_stow(0);
      g.set_track(0);
    }
    sec = (double)t.getTsec(g, d, gg);
    ra = geom.get_galactic_ra(sec, g.get_aznow1(), g.get_elnow1(), g, t);
    dec = geom.get_galactic_dec();
    glat = geom.get_galactic_glat();
    glon = geom.get_galactic_glon();
    vel = geom.get_galactic_vel();
    g.set_vlsr(vel);
    d.dtext(670.0, 119.0, gg, Color.black,
         "radec␣" + d.dc(ra, 5, 1)+"␣hrs␣" + d.dc(dec, 5, 1)+"␣deg");
    d.dtext(670.0, 104.0, gg, Color.black,"Galactic␣l␣="
            + d.dc(glon, 4, 0) + "␣b␣=" + d.dc(glat, 3, 0));
 /*   d.dtext(665.0, 426.0, gg, Color.black,
          "VLSR    " + d.dc(vel, 6, 1)+" km/s");*/
    vel = -vel -(g.get_fcenter() - 1420.406) * 299790.0 / 1420.406;
/*    d.dtext(665.0, 442.0, gg, Color.black,
          "Vcenter " + d.dc(vel, 6, 1)+" km/s");*/
    d.lpaint(gg, Color.black, 0.0, 85.0, 217.0, 85.0);
    d.dtext(16.0, 16.0, gg, Color.black,
     g.get_statnam() + "␣lat␣" + d.dc(g.get_lat() * 180.0 / Math.PI, 4, 1) +
            "␣lonw␣" + d.dc(g.get_lon() * 180.0 / Math.PI, 4, 1));
    d.lpaint(gg, Color.black, 0.0, 0.0, 217.0, 0.0);
    return;
  }
  // End of azel1() routine to move antenna 1,2

}
```

srt.java

```
import time.*;
import disp.*;
import java.awt.*;
import global.*;
import geom.*;
import cat.*;
import plots.*;
import procs.*;
import sport.*;
import checkey.*;
import java.io.*;
public class srt
// main class that starts main thread loop
// this code evolved from a C program and is not modular
// an ongoing task is to provided a more modular version
// allowing for easier addition of modules
// since this code is not modular we recommend that find or
// grep always be used when making changes to check for all
// occurances of globals
{
  public static void main(String args[]) throws IOException
  {
    time t;
    disp d;
```

```
String ydhms;
long sec;
double gst,
  secstop;
global g;
geom geom;
int i;
sport s;
procs proc;
checkey che = new checkey();
cat cat = new cat();
Graphics gg;
if (args.length < 1)
{
  System.out.println("srt␣radiosim␣azelsim");
  System.out.println("for␣example:␣srt␣0␣to␣run␣with␣receiver␣+␣antenna");
  System.out.println("␣␣␣␣␣␣␣␣␣␣␣␣␣␣srt␣0␣1␣to␣simulate␣antenna");
  System.out.println("␣␣␣␣␣␣␣␣␣␣␣␣␣␣srt␣1␣␣to␣simulate␣radio");
  System.out.println("␣␣␣␣␣␣␣␣␣␣␣␣␣␣srt␣1␣1␣to␣simulate␣both");
  System.out.println("␣␣␣␣␣␣␣␣␣␣␣␣␣␣srt␣1␣10␣to␣simulate␣and␣speed-up␣time␣by␣x␣10
      ");
  System.out.println("␣␣␣␣␣␣␣␣␣␣␣␣␣␣srt␣1␣-1␣to␣simulate␣with␣1␣hour␣advance");
  System.out.println("␣␣␣␣␣␣␣␣␣␣␣␣␣␣srt␣1␣0␣1␣for␣maintenance");
  System.out.println("radiometer␣simulation:␣assumes␣200K␣tsys␣+␣Gaussian␣noise")
      ;
  System.out.println("␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣for␣40kHz␣BW␣0.1s␣integration␣per␣
      freq");
  System.out.println("␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣1K␣for␣Moon,␣100K␣for␣Sun,␣2.6K␣for␣
      Cass");
  System.out.println("Catalog␣Keywords:␣STATION,AZLIMITS,ELLIMITS,COMM,CALCONS");
  System.out.println("␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣TLOAD,TSPILL,BEAMWIDTH,MANCAL,AXISTILT");
  System.out.println("␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣SSAT,AZEL,GALACTIC,SOU,Sun,Moon");
  System.exit(1);
}
g = new global ();
  t = new time();
  geom = new geom();
  d = new disp("BYU␣Radio␣Astronomy-Antenna␣1", g, che);
  gg = d.getGraphics();
  d.set_font(gg);
  che.set_graph(gg);
// Set the radio to simulation mode
g.set_radiosim(1);
//if (args.length >= 1)
//    g.set_radiosim(Integer.valueOf(args[0]).intValue());
//else
//    g.set_radiosim(0);
if (args.length >= 2)
    g.set_azelsim(Integer.valueOf(args[1]).intValue());
else
    g.set_azelsim(0);
if (args.length >= 3)
    g.set_mainten(Integer.valueOf(args[2]).intValue());
else
    g.set_mainten(0);
for (i = 0; i < 100; i++)
{
  g.set_xlast(1, i);
  g.set_ylast(1, i);
}
for (i = 0; i < 360; i++)
{
  g.set_gxlast(1, i);
  g.set_gylast(1, i);
}
g.set_nfreq(1);
g.set_calcons(1.0);
g.set_tload(300.0);
g.set_tspill(20.0);
g.set_beamw(5.0);
g.set_comerr(0);
g.set_comerad(0);
g.set_fcenter(1420.0);     /* default for continuum */
g.set_freqa(1420.0);
if (g.get_radiosim() != 0)  {
```

```
      g.set_fcenter(1420.4);    /* default for H-line */
       g.set_nfreq(40);
       }
     g.set_ppos(0);
     g.set_azoff(0.0);
     g.set_eloff(0.0);
     g.set_pazoff(0.0);
     g.set_peloff(0.0);
     secstop = 0.0;
     g.set_tstart(0);
     g.set_azaxis_tilt(0.0);
     g.set_tilt_az(0.0);
     g.set_elaxis_tilt(0.0);
     g.set_tsys(0.0);
     g.set_stopproc(0);
     g.set_click(0);
     g.set_atten(0);
     g.set_calon(0);
     g.set_docal(0);
     g.set_sourn(0);
     g.set_track(0);
     g.set_scan(0);
     g.set_bsw(0);
     g.set_mancal(0);
     g.set_clr(1);
     g.set_port1(1);              /* comm1 default */
     g.set_freqsep(0.04);
     g.set_fstatus(0);
     g.set_cmdf(0);
     g.set_key(0);
     g.set_filepointer(0); // default fpointer position for command file
     cat.catfile(g,d,gg);
//     System.out.println("finished cat");
     s = new sport(g);
     proc = new procs(s);
     if(g.get_mainten() == 0){  //This is for normal operation
     g.set_azcmd1(g.get_azlim1_1());
     g.set_elcmd1(g.get_ellim1_1());
     g.set_stow(1);
     }
     else {
     g.set_azcmd1(180.0);
     g.set_aznow1(180.0);
     g.set_elcmd1(45.0);
     g.set_elnow1(45.0);
     g.set_azcount1((int)((g.get_aznow1()-g.get_azlim1_1())*27.0*52.0/120.0));
     g.set_elcount1((int)((g.get_elnow1()-g.get_ellim1_1())*27.0*52.0/120.0));
     g.set_stow(0);
     }
     while (true)
     {
       if (g.get_click() != 0)
       {
         if (g.get_cmdstr().length() > 1)
           g.set_click(0);
         che.checky(g, d, gg);
         if (g.get_click() == 1)
           g.set_click(0);
       }
       proc.spectra(g, d, gg);

           //Added to reduce computational load AJP August 16, 2002
           try
           {
         Thread.sleep(2500);
       }
       catch(InterruptedException e)
       {
         System.out.println(e);
       }

       sec = t.getTsec(g, d, gg);
       if (g.get_cmdf() == 1) {
         cat.cmdfile(g,d,gg,t);
       }
```

```
      secstop = 0.0;
    }
  }
}
```

time.java

```
import java.lang.*;
import java.awt.*;
public class time
// gets the time from the PC clock
{
  private long tmil,
    i,
    year,
    yr,
    day,
    days,
    hr,
    min,
    sec;
  private String da,
    h,
    m,
    s,
    ydhms,
    ydh;
  private double gst,
    secs;
  public long getTsec(global g, disp d, Graphics gg)
  {
    tmil = System.currentTimeMillis();  // this is always UT

    /* local time set by time command - check zone
       using windows control panel */
    if (g.get_azelsim() > 1)
    {
      if (g.get_tstart() == 0.0)
        g.set_tstart(tmil);
      tmil = (g.get_tstart() + (tmil - g.get_tstart()) * g.get_azelsim());
      /* speed up */
    }
    if (g.get_azelsim() < 0)
        tmil += -(long)g.get_azelsim() * 3600000;

      day = tmil / 86400000;
      sec = tmil / 1000 - day * 86400;

      days = 365;
    for (i = 1970; day > (days - 1); i++)
    {
      days = ((i % 4 == 0 && i % 100 != 0) || i % 400 == 0) ? 366 : 365;
      day -= days;
    }
    hr = sec / 3600;
    sec -= hr * 3600;
    min = sec / 60;
    sec = sec - min * 60;
    year = i;
    day = day + 1;
    da = "" + day;
    if (day < 100)
      da = "0" + da;
    if (day < 10)
      da = "0" + da;
    h = "" + hr;
    if (hr < 10)
      h = "0" + h;
    m = "" + min;
    if (min < 10)
      m = "0" + m;
    s = "" + sec;
    if (sec < 10)
      s = "0" + s;
```

```
    ydhms = year + ":" + da + ":" + h + ":" + m + ":" + s;
    if (year < 2000)
      yr = year - 1900;
    else
      yr = year - 2000;
    ydh = yr + da + h;
    if (ydhms.indexOf(g.get_ptime()) == -1)
      d.stext(670.0, 152.0, gg, Color.white,"UT␣" + g.get_ptime());
    d.stext(670.0, 152.0, gg, Color.black, "UT␣" + ydhms);
    g.set_ptime(ydhms);
    return tmil / 1000;
  }
  public String getYdhms()
  {
    return ydhms;
  }
  public String getYdh()
  {
    return ydh;
  }
  public long get_year()
  {
    return year;
  }
  public double getGst()
  {
    secs = (1999 - 1970) * 31536000.0 + 17.0 * 3600.0 + 16.0 * 60.0 + 20.1948;
    for (i = 1970; i < 1999; i++)
    {
      if ((i % 4 == 0 && i % 100 != 0) || i % 400 == 0)
        secs += 86400.0;
    }
    return Math.IEEEremainder(((double)tmil / 1000.0 - secs), 86164.09053)
           / 86164.09053 * 2.0 * Math.PI;
/* 17 16 20.1948 UT at 0hr newyear1999 */
  }
  public String getMonthday()
  {
   int day_tab[] = {0,31,28,31,30,31,30,31,31,30,31,30,31,
                    0,31,29,31,30,31,30,31,31,30,31,30,31};
   String mon[] = {"Jan","Feb","Mar","Apr","May","Jun","Jul",
                   "Aug","Sep","Oct","Nov","Dec"};
   int k,j,leap;
   String str;
      j = (int)day;
      if((year% 4 == 0 && year % 100 != 0) || year % 400 == 0)
        leap = 1;
      else leap = 0;
      for (k = 1; j > day_tab[leap*13 + k]; k++)
       j -= day_tab[leap*13 + k];
       str = mon[k-1];
       str += "␣"+j;
       return(str);
   }

/* Convert to Seconds since New Year 1970 */
  public double tosec(int yr, int day, int hr, int min, int sec)
  {
    int i;
    double secs;
      secs = (yr - 1970) * 31536000.0 + (day - 1) * 86400.0
    + hr * 3600.0 + min * 60.0 + sec;
    for (i = 1970; i < yr; i++)
    {
      if ((i % 4 == 0 && i % 100 != 0) || i % 400 == 0)
        secs += 86400.0;
    }
    if (secs < 0.0)
        secs = 0.0;
    return secs;
  }

}
```

tracker.cpp [47]

```
#include <windows.h>
#include <ddeml.h>
#include <string.h>

long _stdcall WndProc (HWND, UINT,UINT, LONG);

HDDEDATA CALLBACK DdeCallback (UINT iType, UINT iFmt, HCONV hCconv, HSZ hsz1,  HSZ
    hsz2,
                                HDDEDATA hData, DWORD dwData1, DWORD dwData2);

HANDLE hInstance;
DWORD idInst;
HCONV    hConv;
unsigned char satname[128];
HANDLE fHandle;

//wWinMainCRTStartup
int WINAPI WinMain (HINSTANCE hInst, HINSTANCE hPrevInstance,LPSTR lpszCmdParam, int
    nCmdShow)
{

        static char szAppName[]="HelloNova";
        HWND hWnd;
        MSG msg;
        WNDCLASS wndclass;

         hInstance = hInst;
        if (!hPrevInstance)
        {
         wndclass.style = CS_HREDRAW|CS_VREDRAW;
         wndclass.lpfnWndProc   = WndProc;
         wndclass.cbClsExtra = 0;
         wndclass.cbWndExtra = 0;
         wndclass.hInstance = (HINSTANCE__ *)hInstance;
         wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
         wndclass.hCursor =LoadCursor(NULL, IDC_ARROW);
         wndclass.hbrBackground=(HBRUSH__ *)GetStockObject (WHITE_BRUSH);
         wndclass.lpszMenuName=NULL;
         wndclass.lpszClassName=szAppName;

         RegisterClass(&wndclass);
         }

         hWnd = CreateWindow(szAppName,
                                  "Interface␣between␣Nova␣and␣Positioning␣
                                      Software",
                                  WS_TILEDWINDOW, // was originally
                                      WS_OVERLAPPED|WS_SYSMENU
                                  CW_USEDEFAULT,
                                  CW_USEDEFAULT,
                                  CW_USEDEFAULT,
                                  CW_USEDEFAULT,
                                  NULL,
                                  NULL,
                                  (struct HINSTANCE__ *)hInstance,
                                  NULL);

        ShowWindow(hWnd, nCmdShow);
        UpdateWindow(hWnd);

        //Add timer to update window
        SetTimer(hWnd,1,500,NULL);

        // Open output file
        fHandle = CreateFile("antenna\\antenna.txt",
                                                        GENERIC_WRITE,
                                                        FILE_SHARE_READ,
                                                        NULL,
                                                        CREATE_ALWAYS,
                                                        FILE_ATTRIBUTE_NORMAL,
                                                        NULL);

        if (fHandle == INVALID_HANDLE_VALUE)
```

```
        {
        MessageBox(NULL,"Cannnot␣open␣file!", "File␣Create␣Error!",MB_OK|MB_ICONERROR
            );   // process error
        }


        while (GetMessage (&msg, NULL, 0,0))
        {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }

        return msg.wParam;

}

long _stdcall WndProc (HWND hWnd, UINT message,
                                                                    UINT wParam, LONG
                                                                        lParam)
{
HDC hdc;
PAINTSTRUCT ps;
RECT    rect;
FARPROC pfnDdeCallback;
HSZ     hszService,hszTopic,hszItem;
HDDEDATA hData;
DWORD dwResult;
static char onbuff[8];
UINT errornum;

switch (message)
{
        case WM_PAINT:
              hdc=BeginPaint(hWnd,&ps);

              GetClientRect(hWnd,&rect);

              DrawText (hdc,(const char *)satname,-1, &rect,
                               DT_SINGLELINE | DT_CENTER | DT_VCENTER);

              EndPaint (hWnd, &ps);
              return 0;

        case WM_TIMER:
              InvalidateRect (hWnd, NULL, FALSE) ;
              return 0;

        case WM_DESTROY:
              if (idInst)
              DdeUninitialize(idInst);
                  PostQuitMessage(0);
                  KillTimer(hWnd,1);
                  CloseHandle(fHandle);
                  return 0;
        case WM_CREATE:
              pfnDdeCallback=MakeProcInstance((FARPROC) DdeCallback,hInstance);

              if (DdeInitialize(&idInst,(PFNCALLBACK)pfnDdeCallback,APPCLASS_STANDARD
                   |APPCMD_CLIENTONLY,0L))
              {
                  MessageBox (NULL,"Couldn't␣Init␣Logwindows␣DDE␣Client","error",
                      MB_OK);
              }
              else
              {
                  hszService=DdeCreateStringHandle(idInst,"NFW32",0);
                  hszTopic = DdeCreateStringHandle(idInst,"NFW_DATA",0);
                  hszItem  = DdeCreateStringHandle(idInst,"NFW_SERVER",0);
                  hConv=DdeConnect(idInst,hszService,hszTopic, NULL);

                  if (hConv == NULL)
                  {
                      MessageBox(NULL,"Cannnot␣connect␣to␣Nova", "DDE␣Error!",MB_OK|
                          MB_ICONERROR);
                  }
```

```
                    else
                    {
                        MessageBox(NULL, "Connected to Nova", "DDE",MB_OK);
                        memset(onbuff,0,sizeof(onbuff));
                        strcpy (onbuff,"TUNE ON");

                        hData = DdeClientTransaction(
                        (unsigned char *)onbuff, /* pass data to server */
                        strlen(onbuff)+1,/* the data                    */
                        hConv,          /* conversation handle    */
                        hszItem,         /* item name              */
                        CF_TEXT,        /* clipboard format       */
                        XTYP_POKE, /* start an advise loop   */
                        1000,           /* time-out in one second */
                        &dwResult);     /* points to result flags */

                        if (hData)
                        {
                            hData = DdeClientTransaction(
                            NULL, /* pass data to server */
                            0,/* the data                  */
                            hConv,          /* conversation handle    */
                            hszItem,         /* item name              */
                            CF_TEXT,        /* clipboard format       */
                            XTYP_ADVSTART|XTYPF_ACKREQ, /* start an advise loop   */
                                  1000,           /* time-out in one second */
                            &dwResult);     /* points to result flags */
                        }

                        if (hData)
                        {
                            MessageBox(NULL,"Started DDE to Nova", "DDE",MB_OK);

                        }
                        else
                        {
                            MessageBox(NULL,"Error starting DDE to Nova", "DDE",MB_OK);
                            errornum=DdeGetLastError(idInst);
                        }
                     }
                }
                break;
    }
    return DefWindowProc (hWnd,message, wParam, lParam);
}

HDDEDATA CALLBACK DdeCallback (UINT iType, UINT iFmt, HCONV hCconv, HSZ hsz1, HSZ
    hsz2,
                                HDDEDATA hData, DWORD dwData1, DWORD dwData2)
{
        unsigned char szBuffer[128];
        DWORD bytesWritten;

        switch (iType)
        {
        case XTYP_DISCONNECT:
                MessageBox(NULL,"NOVA Interface terminated","DDE", MB_OK);
                return (HDDEDATA) TRUE;
                break;
        case XTYP_ADVDATA:
                if (iFmt != CF_TEXT)
                    return DDE_FNOTPROCESSED;
                DdeQueryString(idInst, hsz1, (LPSTR) &szBuffer, sizeof(szBuffer), 0);

                if (strstr((const char *)szBuffer,"NFW_DATA") != NULL)
                {
                    DdeGetData(hData,(unsigned char *) satname,sizeof(satname),0);
                    WriteFile(fHandle,satname,strlen((char*)satname),&bytesWritten,
                        NULL);
                }
                return (HDDEDATA) DDE_FACK;
        }
        return NULL;
}
```

A.2 Simulation Code

In order to verify proper interference cancellation performance with the LMS adaptive filter, simulations and initial tests were conducted using MATLAB (see Section 4.4). Contrasted with the difficult and cumbersome task of implementing the algorithm in the complex DSP environment, development in MATLAB is relatively simple and adaptable. In order to facilitate conversion from MATLAB to a DSP environment, the code was written with program structure matching as closely as possible the potential DSP C code, avoiding some of the built-in MATLAB functions not available in C.

This section includes the following source code files:

loop_lms_c.m

```
clear all;

%testdata contains real data from the VSA
load testdata;

lengthdata=length(a);
Numloops=lengthdata/1024;

%initialize matrix to store adapting filter states
h_block=zeros(Numloops,p);
%filter order
p=20;
%LMS adaptive constant (normalized algorithm)
mu=.001;
%initialize the adaptive filter to zero
h=zeros(1,p);
%Find power of the first p samples
M=sum((abs(x(1:p))).^2);
%create vector to hold the filter output
dhat=zeros(length(a)-p,1);
%number of output samples computed each time lmsC is called
N_block=1024;
%find first blocks of data to be sent to lmsC
x_block=[zeros(p,1); x(1:N_block)];
a_block=a(1:N_block);

for k=1:Numloops,
    [h, dhat_block, M]=lmsC(x_block,a_block,h,p,N_block,mu,M);
    h_block(k,:)=h;
    if k<Numloops, %find next blocks of data to be sent to lmsC
        x_block=x((k*N_block+1-p):((k+1)*N_block));
        a_block=a((k*N_block+1):((k+1)*N_block));
    end
    %assign output block to complete output vector
    dhat(((k-1)*N_block+1):(k*N_block))=dhat_block;
end


save results_loop_C dhat h h_block;
```

lmsC.m

```
%This function is an attempt to verify the proper
%operation of the potential DSP LMS algorithm code.
%I tried to write the code with the same structure
%as the potential DSP code. The normalized LMS adaptive
%algorithm is used.

%x=GLONASS Data {Note length(x) must be equal to p+N in order to create N output
    samples}
%a=Desired signal corrupted by GLONASS
%h=filter taps
%p=# filter taps
%N=number output samples
%dhat=filtered output
%M=Normalizing Variable to ensure convergence

function [h, dhat, M] = lmsC(x,a,h,p,N,mu,M)
dhat=zeros(N,1);
for i=1:N,
    v_hat_r=0;
    v_hat_i=0;
    for k=1:p, %After this loop executes, v_hat=v_hat(i)
        v_hat_r=v_hat_r+real(h(k))*real(x(i+k))-imag(h(k))*imag(x(i+k));
        v_hat_i=v_hat_i+real(h(k))*imag(x(i+k))+imag(h(k))*real(x(i+k));
    end
    temp_r=real(a(i))-v_hat_r;
    temp_i=imag(a(i))-v_hat_i;
    dhat(i)=temp_r+j*temp_i;
    M=M+real(x(i+p))^2+imag(x(i+p))^2-real(x(i))^2-imag(x(i))^2;
    beta=mu/M;
    for z=1:p,
        h(z)=h(z)+beta*(temp_r*real(x(i+z))+temp_i*imag(x(i+z)))+j*beta*(-temp_r*imag
            (x(i+z))+temp_i*real(x(i+z)));
    end
end
```

A.3 DSP Application Software

Basic tools needed for radio astronomy observations and the analysis and implementation of interference mitigation algorithms were implemented in the real-time programmable DSP platform (see Chapter 3). These include a PSD estimator, a beamformer, and an array signal correlator. The LMS adaptive algorithm was also implemented in the DSP (see Section 4.5). I have also developed an application which fills the DSP memory with raw time-domain complex data, then sends this data to the host PC for storage and post-processing. This section contains the source code for all of these applications.

Many of the DSP C routines required high levels of optimization in order to take advantage of the performance potential of the four TI TMS320C6701 processors. The Texas Instruments compiler is powerful, but performance can greatly vary with seemingly insignificant changes in the C code. Some of these changes included ordering of expressions, parameter designation (e.g. "volatile"), memory allocation, structure of processing loops, etc. I also obtained different results depending on the

compile options. In a few cases, I was able to create faster code by specifying lower levels of automatic compiler optimization. With so many possible permutations, my only option was to compile the code under many different circumstances. In order to determine which was the fastest compilation, I counted the clock cycles of the assembly code produced and/or ran the code while measuring unused processing capacity.

In many instances, both the source C code (extension *.c) and corresponding fastest compiled assembly code (extension *.asm) are listed in this section. In these cases, the assembly code is included in the DSP project in lieu of the C code. In some cases, a command file (extension *.cmd) is listed. This file describes the memory structure available to each DSP processor. It is also used to designate where the compiled code and data will be loaded.

A.3.1 PSD Estimator Source Code

A PSD estimator was implemented in the DSP (see Section 3.2.3). The DSP PSD estimator has two modes of operation. Either the data is written to a file for post-processing, or displayed real-time using a LabView application on the host PC. Since the DSP code for these two applications vary only slightly from one another, I have only included one of these two projects: the application set up for writing the data to the PC memory.

This section includes the following source code files:

A routine entitled apihost.c, similar to that shown on page 309, is part of a Visual C++ application enabling data transfer from the DSP platform to host PC. The following source code files are used in the PSD estimator application, but are included in Section A.3.5:

- power_fft.c—page 326
- power_fft_opt.asm—page 327
- cfftr2_e.asm [48]—page 329

Fortunately, a hand-optimized floating-point FFT algorithm developed by Texas Instruments, cfftr2.asm [48], was available for the most computationally intensive portion of PSD estimation, the DFT calculation. This routine implements the decimation-in-time radix-2 FFT algorithm. Unfortunately, the assembly code is written for a "little endian platform" (referring to byte MSB, LSB ordering), while ours is set to big endian. Ultimately, we were able to edit the code to work properly with our big endian setup. As listed above, this function, cfftr2_e.asm, is found on page 329.

4291_330e_mod.cmd

```
-x
-l dev6xsne.lib
-l c6xe.lib
-l vime.lib
-l p4291_330e.lib
-heap   0x1000
-stack  0x1000
-l os90e.lib
-l stdioe.lib
-l snioe.lib
-l swftnte.lib
-l ose.lib
-l rts6701e.lib
-l shmeme.lib
-l rtapie.lib

/*
 * Specify the sections of the Model 4291-330 memory.
 */

MEMORY
{
    PNKG  (RWIX) : org = 0x00000000,  len = 0x0000c400 /*Reserved for SwifNet*/
    GRAM  (RWIX) : org = 0x0000c400,  len = 0x001f3bff /* only if GBPR =0x40 */
    IPRAM  (RWIX) : org = 0x01400000, len = 0x00010000
    TVEC  (RWIX) : org = 0x02000000,  len = 0x00000400
    SDRAM (RWIX) : org = 0x02000800,  len = 0x00039800
    PNK   (RWIX) : org = 0x02ffa000,  len = 0x00006000 /*SwiftNet Kernel*/
                                                        /* Mirrored at
                                                           0x2f03000 */
    SBSRAM (RWIX) : org = 0x01000000, len = 0x00080000
    IDRAM  (RWIX) : org = 0x80000000,  len = 0x0000e000
}

/*
 * Specify the allocation of the sections in the Model 4290 memory.
 */

SECTIONS
{
    .buffer0    :     > IDRAM
    .globaldata :     > IDRAM
    .buffer1    :     > IDRAM align(0x8000)  /* When using double buffers,
                                                place the second buffer in
```

```
                                                    Block 1 IDRAM for optimal
                                                    performance.
                                                */
    .sdram0             :       > SDRAM
    .sbsram13           :       > SBSRAM
    .idram0             :       > IDRAM
    .text               :       > SBSRAM
    .cinit              :       > SBSRAM
    .const              :       > SBSRAM
    .switch             :       > SBSRAM
    .data               :       > IDRAM
    .bss                :       > SBSRAM
    .cio                :       > IDRAM
    .sysmem             :       > IDRAM
    .far                :       > IDRAM
    .stack              :       > IDRAM
    .xref               :       > IDRAM
    .tvt                :
                        {
                                _traptbl = .;
                        }> TVEC
        .dram0:
                        {

                        _RTAPI_MAXOUT = 0x20000;
                        _RTAPI_MAXIN = 0x0;

                                _shmem1_base = .;
                                _shmem1_top = . + 0x1f3bff;
                                _shmem2_base = 0;
                                _shmem2_top = 0;
                                _shmem3_base = 0;
                                _shmem3_top = 0;
                                _shmem4_base = 0;
                                _shmem4_top = 0;
                        } > GRAM

/* ISR Jump table used by SwiftNet resides here */
        .ivec:
                        {
                                _isr_jump_table = .;
                        }> PNK
}
```

byusync.c

```
/*****************************************************************************
*
*   File : byusync.c (PSD estimator--write spectrum to file)
*
*****************************************************************************/
#include "stdlib.h"
#include "4290.h"
#include "4290rscm.h"
#include "4290bifo.h"
#include "4290mbx.h"
#include "6216.h"
#include "math.h"
#include "c6xtimer.h"
#include "4290dma.h"
#include "c6xdma.h"
#include "dma.h"
#include "Short_to_float.h"
#include "Power_FFT.h"
#include "cfftr2_dit.h"
#include <rtapi.h>

/*  ******* GLOBAL DEFINES ********************  */

/* Number of 6216 VIM Modules being Synchronized */
#define  NUM_6216_VIM_MODULES     2             /* 1 or 2 6216's on 4290   */

/* Synchronization Master Processor - Must be Processor A or C */
#define  SYNC_MASTER              P4290_PROC_A  /* 0 or 2 for CPU's A & C  */
```

```
/* Clock Selection - P6216_INTERNAL_CLOCK or P6216_EXTERNAL_CLOCK */
#define CLOCK_SELECT    P6216_INTERNAL_CLOCK

/* The Internal Oscillator Frequency can vary based on installed options
   on the board.
*/
#define INT_OSC_STANDARD      64000000L      /* 6216 Standard       */
#define INT_OSC_OPT20         65000000L      /* 6216 with Option 20 */
#define INT_OSC_OPT21         60000000L      /* 6216 with Option 21 */

/* External Clock Rate - When the 6216 is being clocked externally or it is
                         receiving it's clock from another 6216, specify the
                         external clock frequency.
*/
#define EXTERNAL_CLOCK_RATE   64000000L

/* ********** GLOBAL VARIABLES ******************  */

/* Global Data Buffers */

#define BUFFER_SIZE   1024

/*      FFT defines */
#define N                       1024            /* Buffer and FFT size */
#define TWO_N                   2048            /* Two Times N*/
#define ETA                     10              /* N = 2^ETA */
#define PI_long                 3.141592653589793238

#pragma DATA_SECTION(outDataBlock, ".idram0");
far int outDataBlock[2*BUFFER_SIZE];

struct complex {float real; float imag;};
#pragma  DATA_ALIGN(w, 8);
#pragma  DATA_ALIGN(x, 8);
#pragma  DATA_SECTION(x, ".buffer0");
#pragma  DATA_SECTION(w, ".buffer1");
#pragma  DATA_SECTION(y, ".idram0");
struct complex   x[N];
volatile float   y[N];
struct complex   w[N/2];

/* Other global variables  -- */
int dmaDone=0;
volatile int       loopCnt=0;
volatile int       waitCnt=0;
int                bitRevIndx[N], temp[N/2];
volatile int       pingpong=0;

/* ************ Function Prototypes **************** */

void syncABD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void syncBCD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void waitForSync(unsigned int mailbox, unsigned int syncWord);
int calcBifoClock(P6216_BOARD_PARAMS *boardParams,
                  P6216_RCVR_PARAMS *rcvrParams);
void P6216SetBoardParams(P6216_BOARD_PARAMS *boardParams);
void P6216SetRcvrParams(P6216_RCVR_PARAMS *rcvrParams,
                        P6216_BOARD_PARAMS *boardParams,
                        unsigned int intClockRate);

/* Interrupt service routine for BIFIFO dma */
interrupt void ReadBifoISR() {
        /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH2))) &= 0xffbb;

        /* Set dma complete flag */
        dmaDone=1;
}  /* end of ISR */

void main()
{
    P6216_BOARD_PARAMS             p6216BoardParams;
```

```
P6216_RCVR_PARAMS               p6216RcvrParams;
P6216_REG_ADDR                  p6216Regs;
int                             i;
unsigned int                    procId = P4290GetProcId();
volatile float                  phase;
int                             bifoClock;
unsigned int                    intClockRate = INT_OSC_STANDARD;
P4290_DMA_PARAMS                dmaParams;
int                             frameIndex;
int                             indexVal;
int                             *outData0,*outData1;
volatile int                    finalwaitcnt;
int                             k,l,m, recurse_N, offset;
volatile float                  d;
short                           *outDataShort0,*outDataShort1;
short                           *SHORTIOdata[2];
int                             sendCnt=0;
int                             streamstatus=0;
int                             outputstream=0;
int                             connected=0;
int                             finished=0;
int                             DecimationRate=16;
int                             DecimatedSampleRate=64000000/DecimationRate;
/*Do not use the following six variables in the continuous while loop*/
double                          DesiredIntegrationTime;    /*in seconds*/
double                          ActualIntegrationTime;
double                          DesiredIntegrationFrames;
double                          DesiredProgramRunTime;
double                          ActualProgramRunTime;
double                          Temp1;
/*********************************************************************/
int                             NumberBlocks;
int                             ActualIntegrationFrames;
int                             TotalFrames;

DesiredIntegrationTime=5;
DesiredProgramRunTime=180;
DesiredIntegrationFrames = (double)((DecimatedSampleRate/BUFFER_SIZE)*
    DesiredIntegrationTime);
ActualIntegrationFrames = (int)(DesiredIntegrationFrames);
if ((DesiredIntegrationFrames-ActualIntegrationFrames)>=0.5)
    ActualIntegrationFrames++;
ActualIntegrationTime = ((double)(ActualIntegrationFrames*BUFFER_SIZE))/((double)
    (DecimatedSampleRate));

Temp1 = (DesiredProgramRunTime/ActualIntegrationTime);
NumberBlocks = (int)(Temp1);
if ((Temp1-NumberBlocks)>=0.5)
    NumberBlocks++;
ActualProgramRunTime = (double)(NumberBlocks*ActualIntegrationTime);
TotalFrames=NumberBlocks*ActualIntegrationFrames;

P4290_LED1_OFF;
P4290_LED2_OFF;
P4290_LED3_OFF;

outData0 = outDataBlock;
outData1 = &outDataBlock[BUFFER_SIZE];

outDataShort0=(short*)outData0;
outDataShort1=(short*)outData1;

SHORTIOdata[1]=outDataShort0;
SHORTIOdata[0]=outDataShort1;

if (procId == P4290_PROC_A) {

    outputstream = rt_open(":FFTstream:0",O_NDELAY,0);
    if (outputstream == -1 ) { exit(255); }
    while(!connected) {
            rt_ioctl(outputstream, I_CONNECT, &connected);
    }
}

/* initialize the bit reversed addressing index table */
```

```
for (i=0; i<N; i++)
        bitRevIndx[i] = i;
recurse_N = N;

for (k=0; k<ETA-1; k++){
    recurse_N = recurse_N/2;
    for (offset=0; offset < N; offset += 2*recurse_N) {
        for (i=0; i<recurse_N; i++) {
                temp[i] = bitRevIndx[offset+1+(i<<1)];
                bitRevIndx[i+offset] = bitRevIndx[offset+(i<<1)];
        }
        for (i=0; i<recurse_N; i++)
                bitRevIndx[i+offset+recurse_N] = temp[i];
    }
}

for (k=0; k<N; k++) {
    y[k]=0;
}

/* Initialize the FFT coeficent table, N/2 point bit reversed ordering */
d = 2*PI_long/N;

for (i=0; i<N/2; i++){
    w[bitRevIndx[i]>>1].real = cosf(i*d);
    w[bitRevIndx[i]>>1].imag = sinf(i*d);
}

/* Turn off timer 0 - Enabled by bootcode to toggle led */
TIMER_RESET(C6X_TIMER_0);

/* Initialize Table of 6216 Addresses */
P6216InitRegAddr(0x320000L, &p6216Regs);

/* Determine whether option is installed on this board */
if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x21))
    intClockRate = INT_OSC_OPT21;
else if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x20))
    intClockRate = INT_OSC_OPT20;

/* Set 6216 Board Parameters */
P6216SetBoardParams(&p6216BoardParams);

/* Get 6216 Receiver Parameters */
P6216SetRcvrParams(&p6216RcvrParams, &p6216BoardParams, intClockRate);

/* Set Center Frequency and Decimation  */
p6216RcvrParams.centerFreq = 6.8e6;
p6216RcvrParams.decimationRate = DecimationRate; /* Set for 16 MHz basband fs */
p6216BoardParams.gainSetting = 0;

/* Reset Board Registers */
P6216ResetRegs(&p6216Regs);

/* Initialize Board Registers */
P6216InitBoardRegs(&p6216BoardParams, &p6216Regs);

/* Initialize Receiver Registers */
P6216InitRcvrRegs(&p6216RcvrParams, &p6216Regs);

/* Calculate Slowest Bifo Clock Rate */
bifoClock = calcBifoClock(&p6216BoardParams, &p6216RcvrParams);

#ifndef OPTION_330
/* Select IO Mezzanine Bifo */
P4290BifoSelect(P4290_BIFO_IO);
#endif

/* Clear interrupt enable registers. */
*P4290_LCR_IER0 = 0x0;
*P4290_LCR_IER1 = 0x0;

 /* Reset Interrupt Mapping Reg */
*P4290_LCR_IMR0 = 0;
```

```
/* Reset Miscellaneous Control Register. */
/* Also causes interrupts to be unlatched, which is why this example
   will not run on the standard 4290. */
*P4290_LCR_MCR0 |= 0x00f0;

/* Set up Bifo almost full interrupt */
P4290SetupBifoDMAInt(P4290_BIFO_IO, P4290_INT_EXP_BIFO_IN_ALMST_FULL,
        P4290_IMR0_IO_BIFO_IN_EVENT, CPU_INT4, ISN_EXT_INT4, NULL);

/*Calculate frame index for Global Index register for
use of ping-pong buffers.  Difference between end of buffer 1 and start
of buffer 2. */
frameIndex = (int)outData1 - ((int)outData0 + (BUFFER_SIZE*4)) + 4;

/* Move frame index into upper half of index value.  Lower half is
   number of bytes in element. */
indexVal = (frameIndex <<16) | 4;

/* This version of DmaInit uses the indexVal calculated above */
P4290DmaInit((unsigned int)P4290_LCL_FIFO_IO, (unsigned int)outData0,
        DMA_ADDR_NO_MOD, DMA_ADDR_INC, DMA_INDXA, indexVal,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, BUFFER_SIZE, 2,
        DMA_CNT_RELOADA, FRAME_SYNC_ENABLE, DMA_RELOAD_NONE, DMA_RELOAD_GARB,
        SEN_EXT_INT4, SEN_NONE, 0X2088, C6X_DMA_IE,
        DMA_AUTOINIT_TRUE, &dmaParams);

/*  Setup DMA complete interrupt */
/*     Note DMA chan. 2 is hard-wired to internal interrupt CPU_INT10.
       This is the same as DMA_INT2.   */
C6xSetupInterrupt(CPU_INT10,ISN_DMA_INT2,&ReadBifoISR,0);
C6xEnableInterrupt(CPU_INT10);

/* ***************** SYNCHRONIZE THE 4 PROCESSORS *************** */

/* Disable Fifo Writes */
P6216_DISABLE_FIFO_WRITE(p6216Regs.control);

/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
if (procId == SYNC_MASTER)
{
    if (procId == P4290_PROC_A)
            syncBCD(NUM_6216_VIM_MODULES, 1, 0xabababab);
    else  /* Sync Master is C */
            syncABD(NUM_6216_VIM_MODULES, 1, 0xabababab);
}
else
{
    waitForSync(1, 0xabababab);
}

/* Flush the Bifo */
P4290FlushBifo(P4290_BIFO_IO, bifoClock, BUFFER_SIZE - 4, 128,
              BUFFER_SIZE - 4, 128);
/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
P4290_LED1_ON;
if (procId == SYNC_MASTER)
{
    if (procId == P4290_PROC_A)
        syncBCD(NUM_6216_VIM_MODULES, 1, 0xdcdcdcdc);
    else  /* Sync Master is C */
        syncABD(NUM_6216_VIM_MODULES, 1, 0xdcdcdcdc);
}
else
{
    waitForSync(1, 0xdcdcdcdc);
}

/* If Sync 1 Master, Generate Sync Pulse */
if (procId == SYNC_MASTER)
{
    P6216_SET_SYNC_GENERATE(p6216Regs.syncGen, 0);
```

```
        while ((*p6216Regs.syncGen & 0x1) != 0);
        P6216_SET_SYNC_GENERATE(p6216Regs.syncGen, 1);
    }

    /* ************* MAIN I/O TRANSFER LOOP, ON DMA INTERRUPT *** */

    P4290OurDmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_AUTO_START_VAL,
        CPU_INT4, ISN_EXT_INT4, &dmaParams);
    while (1)
    {
        /* wait for dma to complete */
        while (!dmaDone)
                ++waitCnt;
        pingpong=!pingpong;
        dmaDone = 0;
        ++loopCnt;
        if ((sendCnt == ActualIntegrationFrames) && (procId == P4290_PROC_A)) {
                sendCnt=0;
                streamstatus=-1;
                while (streamstatus==-1) {
                        streamstatus = rt_write(outputstream, (char*)y,4096);
                }
                P4290_LED2_TOGGLE;
                for (i=0;i<1024;i++) {
                        y[i]=0;
                }
        }
        if (loopCnt > TotalFrames) {
                finalwaitcnt=waitCnt;
                P4290OurDmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_PAUSE_VAL,
                    CPU_INT4, ISN_EXT_INT4, &dmaParams);
                P4290_LED3_ON;

                if (procId==P4290_PROC_A) {
                    rt_ioctl(outputstream, O_FLUSH, &finished);
                    rt_close(outputstream);
                }
        }
        else {
                Short_to_float((float *)x, SHORTIOdata[pingpong], TWO_N);
                /* Compute the complex, radix 2 FFT, output bit reversed */
                cfftr2_dit((float *)x, (float *)w,  N);
                Power_FFT ((float *)x, y, bitRevIndx, N);
        }
        waitCnt = 0;
        ++sendCnt;
    }
}
```

A.3.2 Complex Beamformer/Correlation Estimator Source Code

A three channel beamformer and correlation estimator was implemented in the DSP (see Section 3.2.3).

This section includes the following source code files (apihost.c is part of a Visual C++ application enabling data transfer from the DSP platform to host PC):

- intersend_C.c—page 203
- intersend_C_opt.asm—page 204
- rawdatasend.c—page 207
- rawdatasend_opt.asm—page 207
- compute_matrix_corner.c—page 209
- compute_matrix_corner_opt.asm—page 209
- compute_matrix_diagonal.c—page 214
- compute_matrix_diagonal_opt.asm—page 214
- apihost.c—page 218

The following source code files are also used in the DSP beamformer and correlation estimator application, but are included in Section A.3.5:

- P6216SetRcvrParams.c—page 322
- waitForSync.c—page 322
- P6216SetBoardParams.c—page 323
- calcBifoClock.c—page 324
- syncBCD.c—page 324
- syncABD.c—page 325

4291_330e_mod.cmd

```
-x
-l dev6xsne.lib
-l c6xe.lib
-l vime.lib
-l p4291_330e.lib
-heap   0x1000
-stack  0x1000
-l os90e.lib
-l stdioe.lib
-l snioe.lib
-l swftnte.lib
-l ose.lib
-l rts6701e.lib
-l shmeme.lib

/*
 * Specify the sections of the Model 4291-330 memory.
 */

MEMORY
{
    PNKG  (RWIX) : org = 0x00000000,  len = 0x0000c400 /*Reserved for SwifNet*/
    GRAM  (RWIX) : org = 0x0000c400,  len = 0x001f3bff /* only if GBPR =0x40 */
    IPRAM  (RWIX) : org = 0x01400000, len = 0x00010000
    TVEC  (RWIX) : org = 0x02000000,  len = 0x00000400
    SDRAM (RWIX) : org = 0x02000800,  len = 0x00039800

    PNK   (RWIX) : org = 0x02ffa000,  len = 0x00006000 /*SwiftNet Kernel*/
                                                        /* Mirrored at
                                                           0x2f03000 */
    SBSRAM (RWIX) : org = 0x01000000, len = 0x00080000
    IDRAM  (RWIX) : org = 0x80000000,  len = 0x0000e000
}

/*
```

```
 * Specify the allocation of the sections in the Model 4290 memory.
 */

SECTIONS
{
        .buffer0    :     > IDRAM
        .globaldata :     > IDRAM
        .buffer1    :     > IDRAM align(0x8000)  /* When using double buffers,
                                                    place the second buffer in
                                                    Block 1 IDRAM for optimal
                                                    performance.*/
        .sbsram0        :       > IDRAM
        .text           :       > IPRAM
        .cinit          :       > IDRAM
        .const          :       > IDRAM
        .switch         :       > IDRAM
        .data           :       > IDRAM
        .bss            :       > IDRAM
        .cio            :       > IDRAM
        .sysmem         :       > IDRAM
        .far            :       > IDRAM
        .stack          :       > IDRAM
        .xref           :       > IDRAM
        .tvt            :
                        {
                                _traptbl = .;
                        }> TVEC
        .dram0:
                        {
                                _shmem1_base = .;
                                _shmem1_top = . + 0x1f3bff;
                                _shmem2_base = 0;
                                _shmem2_top = 0;
                                _shmem3_base = 0;
                                _shmem3_top = 0;
                                _shmem4_base = 0;
                                _shmem4_top = 0;
                        } > GRAM

/* ISR Jump table used by SwiftNet resides here */
        .ivec:
                        {
                                _isr_jump_table = .;
                        }> PNK
}
```

ProcessorA.c

```
/*****************************************************************************
*
*   File : ProcessorA.c
*
*****************************************************************************/
#include "stdlib.h"
#include "4290.h"
#include "4290rscm.h"
#include "4290bifo.h"
#include "4290mbx.h"
#include "6216.h"
#include "math.h"
#include "c6xtimer.h"
#include "4290dma.h"
#include "c6xdma.h"
#include "dma.h"
#include "InterSend.h"
#include "RawDataSend.h"
/* Swiftnet libs */
#include "pnkcapi.h"
#include "pnkdev.h"
#include "snio.h"

#define DECIMATION_RATE  32

/* Host - target communication defines */
```

```
/* Define the API sig for host/target communication */
#define BYUAPI_SIG 1

/* Define other API comm parameters */
#define HOSTNAME "astronomy"
#define DEVICENAME "byuapi"
#define TICK_ULEN 1000
#define TIMEOUT_TICKS 10000

/*  ******* GLOBAL DEFINES ********************  */

/* Number of 6216 VIM Modules being Synchronized */
#define  NUM_6216_VIM_MODULES     2             /* 1 or 2 6216's on 4290   */

/* Synchronization Master Processor - Must be Processor A or C */
#define  SYNC_MASTER              P4290_PROC_A  /* 0 or 2 for CPU's A & C  */

/* Clock Selection - P6216_INTERNAL_CLOCK or P6216_EXTERNAL_CLOCK */
#define CLOCK_SELECT    P6216_INTERNAL_CLOCK

/* The Internal Oscillator Frequency can vary based on installed options
   on the board.*/
#define INT_OSC_STANDARD      64000000L      /* 6216 Standard       */
#define INT_OSC_OPT20         65000000L      /* 6216 with Option 20 */
#define INT_OSC_OPT21         60000000L      /* 6216 with Option 21 */

/* External Clock Rate - When the 6216 is being clocked externally or it is
                         receiving it's clock from another 6216, specify the
                         external clock frequency.
*/
#define EXTERNAL_CLOCK_RATE   64000000L


/* ***********  GLOBAL VARIABLES ******************  */

/* Global Data Buffers */

#define BUFFER_SIZE                     1024
#define TWO_BUFFER_SIZE                 2048
#define HALF_BUFFER_SIZE                512

#define NUM_LOOPS                               200

#pragma DATA_SECTION(outDataBlock, ".sbsram0");
far int outDataBlock[2*BUFFER_SIZE];
#pragma DATA_SECTION(interDataBlock, ".buffer1");
far float interDataBlock[4*BUFFER_SIZE];

/* Other global variables  -- */
int dmaDone=0;
int interDone=0;

volatile int       loopCnt=0;
volatile int       waitCnt=0;
volatile int       waitCntInter=0;
volatile int       IOpingpong=0;
volatile int       INTERpingpong=0;
/* ************ Function Prototypes **************** */

void syncABD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void syncBCD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void waitForSync(unsigned int mailbox, unsigned int syncWord);
int calcBifoClock(P6216_BOARD_PARAMS *boardParams,
                  P6216_RCVR_PARAMS *rcvrParams);
void P6216SetBoardParams(P6216_BOARD_PARAMS *boardParams);
void P6216SetRcvrParams(P6216_RCVR_PARAMS *rcvrParams,
                        P6216_BOARD_PARAMS *boardParams,
                        unsigned int intClockRate);

/* Interrupt service routine for BIFIFO dma */

interrupt void ReadBifoISR() {
        /* Reset DMA frame and block interrupt conditions on read channel */
```

```
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH2))) &= 0xffbb;

        /* Set dma complete flag */
        dmaDone=1;
}  /* end of ISR */

interrupt void ReadInterISR() {
   /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH1))) &= 0xffbb;

   interDone=1;
}  /* end of ISR */

void main()
{
    P6216_BOARD_PARAMS p6216BoardParams;
    P6216_RCVR_PARAMS  p6216RcvrParams;
    P6216_REG_ADDR     p6216Regs;
    unsigned int       i;
    unsigned int           pasti=0;
    unsigned int       procId = P4290GetProcId();
    volatile float     phase;
    int                bifoClock;
    unsigned int       intClockRate = INT_OSC_STANDARD;
        P4290_DMA_PARAMS   dmaParams;
        P4290_DMA_PARAMS   dmaParams2;
        int                        frameIndex;
        int                                indexVal;
    int                        *outData0,*outData1;
    short                      *outDataShort0,*outDataShort1;
    volatile int           finalwaitcnt;
    float                      *interData0,*interData1;
    volatile float         *outInterprocBifo;
    volatile unsigned int       *outInterprocBifoInt;
    volatile float         *inInterprocBifo;
    /* Variables for host - target communication */
    PNKCAPI_TYPE           pnkc;
    PNK_SIGNAL_TYPE    sig;
    PNK_RETURN_TYPE    ret;
    float                      *INTERdata[2];
    int                        *IOdata[2];
    short                      *SHORTIOdata[2];
    int                        waitCntArray[NUM_LOOPS];
    int                        waitCntInterArray[NUM_LOOPS];
        float                  BeamformerWeightI=1;
        float                  BeamformerWeightQ=0;

     /* Host - target initializations */
    if (pnkc_init() != OK)      /* Initialize client API */
    {
      plog("Error␣initializing␣client␣interface!");
      exit(1);
    }

     /* Open up a connection to virtual device existing on host */

        if (pnkc_open(&pnkc, HOSTNAME, DEVICENAME) != OK)
    {
      plog("error␣opening␣connection␣to␣host");
      exit(1);
    }

        plog("target␣running");

        sig.num=BYUAPI_SIG;
        sig.args.args_len = 0;
    sig.args.args_val = NULL;
    sig.timeout_ticks = TIMEOUT_TICKS;
    sig.tick_ulen = TICK_ULEN;

    /* end of host/target comm. setup routines*/

    outData0 = outDataBlock;
    outData1 = &outDataBlock[BUFFER_SIZE];
    IOdata[1]=outData0;
```

```
IOdata[0]=outData1;

outDataShort0=(short*)outData0;
outDataShort1=(short*)outData1;

SHORTIOdata[1]=outDataShort0;
SHORTIOdata[0]=outDataShort1;

interData0 = interDataBlock;
interData1 = &interDataBlock[2*BUFFER_SIZE];

    INTERdata[1]=interData0;
INTERdata[0]=interData1;

for (i=0; i<4*BUFFER_SIZE;i++) {
    interDataBlock[i]=0;
}

/*Summary of Port usage for all four processors:

Processor A:        XX Port receives data from Processor C
                        ZZ Port sends data to Processor B

Processor B:        XX Port receives data from Processor A
                        ZZ Port sends data to Processor D

Processor C:        XX Port not used
                        ZZ Port sends data to Processor A

Processor D:        XX Port receives data from Processor B
                        ZZ Port not used

Data flow for Beam Former:
            C --> A --> B --> D

Where Processors C, A and B are connected to antennas through the digital
    receiver*/

outInterprocBifo = (float *)P4290_LCL_FIFO_ZZ;
inInterprocBifo = (float *)P4290_LCL_FIFO_XX;
outInterprocBifoInt = P4290_LCL_FIFO_ZZ;

/* Turn off timer 0 - Enabled by bootcode to toggle led */
TIMER_RESET(C6X_TIMER_0);

/* Initialize Table of 6216 Addresses */
P6216InitRegAddr(0x320000L, &p6216Regs);

/* Determine whether option is installed on this board */
if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x21))
    intClockRate = INT_OSC_OPT21;
else if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x20))
    intClockRate = INT_OSC_OPT20;

/* Set 6216 Board Parameters */
P6216SetBoardParams(&p6216BoardParams);

/* Get 6216 Receiver Parameters */
P6216SetRcvrParams(&p6216RcvrParams, &p6216BoardParams, intClockRate);

    /* Set Center Frequency and Decimation  */
p6216RcvrParams.centerFreq = 16.0e6;
p6216RcvrParams.decimationRate = DECIMATION_RATE; /* Set for 1 MHz basband fs */
p6216BoardParams.gainSetting = 6;

/* Reset Board Registers */
P6216ResetRegs(&p6216Regs);

/* Initialize Board Registers */
P6216InitBoardRegs(&p6216BoardParams, &p6216Regs);

/* Initialize Receiver Registers */
P6216InitRcvrRegs(&p6216RcvrParams, &p6216Regs);

/* Calculate Slowest Bifo Clock Rate */
```

```
bifoClock = calcBifoClock(&p6216BoardParams, &p6216RcvrParams);

    #ifndef OPTION_330
/* Select IO Mezzanine Bifo */
P4290BifoSelect(P4290_BIFO_IO);
    #endif

    /* ********** SETUP INTERRUPTS AND DMA TRASFER REGISTERS ****** */

/* Clear interrupt enable registers. */
*P4290_LCR_IER0 = 0x0;
*P4290_LCR_IER1 = 0x0;

 /* Reset Interrupt Mapping Reg */
*P4290_LCR_IMR0 = 0;

/* Reset Miscellaneous Control Register. */
/* Also causes interrupts to be unlatched, which is why this example
   will not run on the standard 4290. */
*P4290_LCR_MCR0 |= 0x00f0;

    /* Set up Bifo almost full interrupt */
P4290SetupBifoDMAInt(P4290_BIFO_IO, P4290_INT_EXP_BIFO_IN_ALMST_FULL,
        P4290_IMR0_IO_BIFO_IN_EVENT, CPU_INT4, ISN_EXT_INT4, NULL);

/*Calculate frame index for Global Index register for
use of ping-pong buffers.  Difference between end of buffer 1 and start
of buffer 2. */
frameIndex = (int)outData1 - ((int)outData0 + (BUFFER_SIZE*4)) + 4;

/* Move frame index into upper half of index value.  Lower half is
   number of bytes in element. */
indexVal = (frameIndex <<16) | 4;

/* This version of DmaInit uses the indexVal calculated above */
P4290DmaInit((unsigned int)P4290_LCL_FIFO_IO, (unsigned int)outData0,
        DMA_ADDR_NO_MOD, DMA_ADDR_INC, DMA_INDXA, indexVal,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, BUFFER_SIZE, 2,
        DMA_CNT_RELOADA, FRAME_SYNC_ENABLE, DMA_RELOAD_NONE, DMA_RELOAD_GARB,
        SEN_EXT_INT4, SEN_NONE, 0X2088, C6X_DMA_IE,
        DMA_AUTOINIT_TRUE, &dmaParams);

/*  Setup DMA complete interrupt */
/*     Note DMA chan. 2 is hard-wired to internal interrupt CPU_INT10.
       This is the same as DMA_INT2.   */
C6xSetupInterrupt(CPU_INT10,ISN_DMA_INT2,&ReadBifoISR,0);
C6xEnableInterrupt(CPU_INT10);

/*********************Interprocessor Setup ************************/

    /*Processor A Flushes the BIFIFO between A and C and the BIFIFO between A and
         B*/
    P4290FlushBifo(P4290_BIFO_ZZ, P4290_BIFO_CLOCK_FREQ, 6*BUFFER_SIZE-4, 128,
                           6*BUFFER_SIZE-4,128);
    P4290FlushBifo(P4290_BIFO_XX, P4290_BIFO_CLOCK_FREQ, 4*BUFFER_SIZE-4, 128,
                           4*BUFFER_SIZE-4,128);


P4290SetupBifoDMAInt(P4290_BIFO_XX, P4290_INT_IP_BIFO_XX_IN_ALMST_FULL,
        P4290_IMR0_XX_BIFO_IN_EVENT, CPU_INT5, ISN_EXT_INT5, NULL);

/*Calculate frame index for Global Index register for
use of ping-pong buffers.  Difference between end of buffer 1 and start
of buffer 2. */
frameIndex = (int)interData1 - ((int)interData0 + (2*BUFFER_SIZE*4)) + 4;

/* Move frame index into upper half of index value.  Lower half is
   number of bytes in element. */
indexVal = (frameIndex <<16) | 4;

P4290DmaInit((unsigned int)P4290_LCL_FIFO_XX, (unsigned int)interData0,
        DMA_ADDR_NO_MOD, DMA_ADDR_INC, DMA_INDXA, indexVal,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, 2*BUFFER_SIZE, 2,
        DMA_CNT_RELOADB, FRAME_SYNC_ENABLE, DMA_RELOAD_NONE, DMA_RELOAD_GARC,
        SEN_EXT_INT5, SEN_NONE, 0X2088, C6X_DMA_IE,
```

```
        DMA_AUTOINIT_TRUE, &dmaParams2);

C6xSetupInterrupt(CPU_INT9,ISN_DMA_INT1,&ReadInterISR,0);
C6xEnableInterrupt(CPU_INT9);
/**********************End of interproc setup **********************/


    /* ***************** SYNCHRONIZE THE 4 PROCESSORS *************** */

P4290_LED1_TOGGLE;

/* Disable Fifo Writes */
P6216_DISABLE_FIFO_WRITE(p6216Regs.control);

/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
if (procId == SYNC_MASTER)
{
    if (procId == P4290_PROC_A)
            syncBCD(NUM_6216_VIM_MODULES, 1, 0xabababab);
    else  /* Sync Master is C */
            syncABD(NUM_6216_VIM_MODULES, 1, 0xabababab);
}
else
{
    waitForSync(1, 0xabababab);
}

/*Flush the IO Bifo (different Almost Full Levels for each channel)
Processor A is the second in the chain of the beamformer, therefore the
almost full level is set to 3*BUFFER_SIZE-4*/

    P4290FlushBifo(P4290_BIFO_IO, bifoClock, 3*BUFFER_SIZE-4, 128,
    3*BUFFER_SIZE-4, 128);

/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
if (procId == SYNC_MASTER)
{
    if (procId == P4290_PROC_A)
            syncBCD(NUM_6216_VIM_MODULES, 1, 0xdcdcdcdc);
        else  /* Sync Master is C */
            syncABD(NUM_6216_VIM_MODULES, 1, 0xdcdcdcdc);
    }
    else
    {
        waitForSync(1, 0xdcdcdcdc);
    }

/* If Sync 1 Master, Generate Sync Pulse */
if (procId == SYNC_MASTER)
{
    P6216_SET_SYNC_GENERATE(p6216Regs.syncGen, 0);
    while ((*p6216Regs.syncGen & 0x1) != 0);
    P6216_SET_SYNC_GENERATE(p6216Regs.syncGen, 1);
}

    /* ************* MAIN I/O TRANSFER LOOP, ON DMA INTERRUPT *** */

/* Set up for DMA BIFO read with synchronization with IO-In AF. */
P4290OurDmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_AUTO_START_VAL,
    CPU_INT4, ISN_EXT_INT4, &dmaParams);
P4290DmaTransfer(DMA_CH1, C6X_DMA_NO_WAIT, DMA_AUTO_START_VAL,
    CPU_INT5, ISN_EXT_INT5, &dmaParams2);
P4290_LED2_ON;

while (1)
{
            /* wait for dma to complete */
    while (!dmaDone)
            ++waitCnt;
    dmaDone = 0;
    IOpingpong=!IOpingpong;
```

```
        waitCntArray[loopCnt]=waitCnt;

        while (!interDone)
                ++waitCntInter;
        interDone=0;
        INTERpingpong=!INTERpingpong;
        waitCntInterArray[loopCnt]=waitCntInter;

        InterSend(outInterprocBifo,BeamformerWeightI,BeamformerWeightQ,
                INTERdata[INTERpingpong],TWO_BUFFER_SIZE,SHORTIOdata[IOpingpong]);

        RawDataSend(outInterprocBifoInt,IOdata[IOpingpong],BUFFER_SIZE);

                ++loopCnt;
                if (loopCnt == NUM_LOOPS) {
                        finalwaitcnt=waitCnt;
                    P4290OurDmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_PAUSE_VAL,
                CPU_INT4, ISN_EXT_INT4, &dmaParams);
                }
                waitCnt = 0;
        waitCntInter = 0;
    }
}
```

ProcessorB.c

```
/*************************************************************************
*
*   File : ProcessorB.c
*
*************************************************************************/
#include "stdlib.h"
#include "4290.h"
#include "4290rscm.h"
#include "4290bifo.h"
#include "4290mbx.h"
#include "6216.h"
#include "math.h"
#include "c6xtimer.h"
#include "4290dma.h"
#include "c6xdma.h"
#include "dma.h"
#include "InterSend.h"
#include "RawDataSend.h"
/* Swiftnet libs */
#include "pnkcapi.h"
#include "pnkdev.h"
#include "snio.h"

#define DECIMATION_RATE  32

/* Host - target communication defines */
/* Define the API sig for host/target communication */
#define BYUAPI_SIG 1

/* Define other API comm parameters */
#define HOSTNAME "astronomy"
#define DEVICENAME "byuapi"
#define TICK_ULEN 1000
#define TIMEOUT_TICKS 10000

/*  ******* GLOBAL DEFINES ******************  */

/* Number of 6216 VIM Modules being Synchronized */
#define  NUM_6216_VIM_MODULES     2             /* 1 or 2 6216's on 4290   */

/* Synchronization Master Processor - Must be Processor A or C */
#define  SYNC_MASTER              P4290_PROC_A  /* 0 or 2 for CPU's A & C  */

/* Clock Selection - P6216_INTERNAL_CLOCK or P6216_EXTERNAL_CLOCK */
#define CLOCK_SELECT    P6216_INTERNAL_CLOCK

/* The Internal Oscillator Frequency can vary based on installed options
   on the board.
```

```
*/
#define INT_OSC_STANDARD      64000000L      /* 6216 Standard       */
#define INT_OSC_OPT20         65000000L      /* 6216 with Option 20 */
#define INT_OSC_OPT21         60000000L      /* 6216 with Option 21 */

/* External Clock Rate - When the 6216 is being clocked externally or it is
                         receiving it's clock from another 6216, specify the
                         external clock frequency.
*/
#define EXTERNAL_CLOCK_RATE   64000000L

/* *********** GLOBAL VARIABLES ******************  */

/* Global Data Buffers */

#define BUFFER_SIZE                    1024
#define TWO_BUFFER_SIZE                2048
#define HALF_BUFFER_SIZE               512

#define NUM_LOOPS                              199

#pragma DATA_SECTION(outDataBlock, ".sbsram0");
far int outDataBlock[2*BUFFER_SIZE];
/*interDataBlock needs to contain both the beamformer data (float/sent first) and the
     raw data
 (int/sent second) from processor A.     */
#pragma DATA_SECTION(interDataBlock, ".buffer1");
far float interDataBlock[6*BUFFER_SIZE];

/* Other global variables  -- */
int dmaDone=0;
int interDone=0;
volatile int       loopCnt=0;
volatile int       waitCnt=0;
volatile int       waitCntInter=0;
volatile int       IOpingpong=0;
volatile int       INTERpingpong=0;
/* ************ Function Prototypes **************** */

void syncABD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void syncBCD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void waitForSync(unsigned int mailbox, unsigned int syncWord);
int calcBifoClock(P6216_BOARD_PARAMS *boardParams,
                  P6216_RCVR_PARAMS *rcvrParams);
void P6216SetBoardParams(P6216_BOARD_PARAMS *boardParams);
void P6216SetRcvrParams(P6216_RCVR_PARAMS *rcvrParams,
                        P6216_BOARD_PARAMS *boardParams,
                        unsigned int intClockRate);

/* Interrupt service routine for BIFIFO dma */

interrupt void ReadBifoISR() {
        /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH2))) &= 0xffbb;

        /* Set dma complete flag */
        dmaDone=1;
}  /* end of ISR */

interrupt void ReadInterISR() {
   /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH1))) &= 0xffbb;

   interDone=1;
}  /* end of ISR */

void main()
{
    P6216_BOARD_PARAMS p6216BoardParams;
    P6216_RCVR_PARAMS  p6216RcvrParams;
    P6216_REG_ADDR     p6216Regs;
    unsigned int       i;
    unsigned int           pasti=0;
```

```
unsigned int       procId = P4290GetProcId();
volatile float     phase;
int                bifoClock;
unsigned int       intClockRate = INT_OSC_STANDARD;
    P4290_DMA_PARAMS   dmaParams;
    P4290_DMA_PARAMS   dmaParams2;
    int                        frameIndex;
    int                                indexVal;
int                            *outData0,*outData1;
short                          *outDataShort0,*outDataShort1;
volatile int           finalwaitcnt;
float                          *interData0,*interData1;
int                            *interDataRaw0,*interDataRaw1;
volatile float         *outInterprocBifo;
volatile unsigned int       *outInterprocBifoInt;
volatile float         *inInterprocBifo;
/* Variables for host - target communication */
PNKCAPI_TYPE           pnkc;
PNK_SIGNAL_TYPE    sig;
PNK_RETURN_TYPE    ret;
float                          *INTERdata[2];
int                            *INTERdataRAW[2];
int                            *IOdata[2];
short                          *SHORTIOdata[2];
int                            waitCntArray[NUM_LOOPS];
int                            waitCntInterArray[NUM_LOOPS];
    float                      BeamformerWeightI=1;
    float                      BeamformerWeightQ=0;

 /* Host - target initializations */
if (pnkc_init() != OK)      /* Initialize client API */
{
  plog("Error␣initializing␣client␣interface!");
  exit(1);
}

/* Open up a connection to virtual device existing on host */

    if (pnkc_open(&pnkc, HOSTNAME, DEVICENAME) != OK)
{
  plog("error␣opening␣connection␣to␣host");
  exit(1);
}

    plog("target␣running");

    sig.num=BYUAPI_SIG;
    sig.args.args_len = 0;
sig.args.args_val = NULL;
sig.timeout_ticks = TIMEOUT_TICKS;
sig.tick_ulen = TICK_ULEN;

/* end of host/target comm. setup routines*/

outData0 = outDataBlock;
outData1 = &outDataBlock[BUFFER_SIZE];
IOdata[1]=outData0;
IOdata[0]=outData1;

outDataShort0=(short*)outData0;
outDataShort1=(short*)outData1;

SHORTIOdata[1]=outDataShort0;
SHORTIOdata[0]=outDataShort1;

interData0 = interDataBlock;
interData1 = &interDataBlock[3*BUFFER_SIZE];

    INTERdata[1]=interData0;
INTERdata[0]=interData1;

interDataRaw0 = (int *)(&interDataBlock[2*BUFFER_SIZE]);
interDataRaw1 = (int *)(&interDataBlock[5*BUFFER_SIZE]);

INTERdataRAW[1]=interDataRaw0;
```

```
INTERdataRAW[0]=interDataRaw1;

for (i=0; i<4*BUFFER_SIZE;i++) {
    interDataBlock[i]=0;
}

/*Summary of Port usage for all four processors:

Processor A:        XX Port receives data from Processor C
                        ZZ Port sends data to Processor B

Processor B:        XX Port receives data from Processor A
                        ZZ Port sends data to Processor D

Processor C:        XX Port not used
                        ZZ Port sends data to Processor A

Processor D:        XX Port receives data from Processor B
                        ZZ Port not used

Data flow for Beam Former:
            C --> A --> B --> D

Where Processors C, A and B are connected to antennas through the digital
    receiver*/

outInterprocBifo = (float *)P4290_LCL_FIFO_ZZ;
outInterprocBifoInt = P4290_LCL_FIFO_ZZ;
inInterprocBifo = (float *)P4290_LCL_FIFO_XX;

/* Turn off timer 0 - Enabled by bootcode to toggle led */
TIMER_RESET(C6X_TIMER_0);

/* Initialize Table of 6216 Addresses */
P6216InitRegAddr(0x320000L, &p6216Regs);

/* Determine whether option is installed on this board */
if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x21))
    intClockRate = INT_OSC_OPT21;
else if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x20))
    intClockRate = INT_OSC_OPT20;

/* Set 6216 Board Parameters */
P6216SetBoardParams(&p6216BoardParams);

/* Get 6216 Receiver Parameters */
P6216SetRcvrParams(&p6216RcvrParams, &p6216BoardParams, intClockRate);

    /* Set Center Frequency and Decimation  */
p6216RcvrParams.centerFreq = 16.0e6;
p6216RcvrParams.decimationRate = DECIMATION_RATE; /* Set for 1 MHz basband fs */
p6216BoardParams.gainSetting = 6;

/* Reset Board Registers */
P6216ResetRegs(&p6216Regs);

/* Initialize Board Registers */
P6216InitBoardRegs(&p6216BoardParams, &p6216Regs);

/* Initialize Receiver Registers */
P6216InitRcvrRegs(&p6216RcvrParams, &p6216Regs);

/* Calculate Slowest Bifo Clock Rate */
bifoClock = calcBifoClock(&p6216BoardParams, &p6216RcvrParams);

    #ifndef OPTION_330
/* Select IO Mezzanine Bifo */
P4290BifoSelect(P4290_BIFO_IO);
    #endif

    /* ********** SETUP INTERRUPTS AND DMA TRASFER REGISTERS ****** */
/* Disable Cache in CSR register */
/*CSR=CSR & 0xfffffff1f; */

/* Clear interrupt enable registers. */
```

```
*P4290_LCR_IER0 = 0x0;
*P4290_LCR_IER1 = 0x0;

 /* Reset Interrupt Mapping Reg */
*P4290_LCR_IMR0 = 0;

/* Reset Miscellaneous Control Register. */
/* Also causes interrupts to be unlatched, which is why this example
   will not run on the standard 4290. */
*P4290_LCR_MCR0 |= 0x00f0;

    /* Set up Bifo almost full interrupt */
P4290SetupBifoDMAInt(P4290_BIFO_IO, P4290_INT_EXP_BIFO_IN_ALMST_FULL,
        P4290_IMR0_IO_BIFO_IN_EVENT, CPU_INT4, ISN_EXT_INT4, NULL);

/*Calculate frame index for Global Index register for
use of ping-pong buffers.  Difference between end of buffer 1 and start
of buffer 2. */
frameIndex = (int)outData1 - ((int)outData0 + (BUFFER_SIZE*4)) + 4;

/* Move frame index into upper half of index value.  Lower half is
   number of bytes in element. */
indexVal = (frameIndex <<16) | 4;

/* This version of DmaInit uses the indexVal calculated above */
P4290DmaInit((unsigned int)P4290_LCL_FIFO_IO, (unsigned int)outData0,
        DMA_ADDR_NO_MOD, DMA_ADDR_INC, DMA_INDXA, indexVal,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, BUFFER_SIZE, 2,
        DMA_CNT_RELOADA, FRAME_SYNC_ENABLE, DMA_RELOAD_NONE, DMA_RELOAD_GARB,
        SEN_EXT_INT4, SEN_NONE, 0X2088, C6X_DMA_IE,
        DMA_AUTOINIT_TRUE, &dmaParams);

/*  Setup DMA complete interrupt */
/*     Note DMA chan. 2 is hard-wired to internal interrupt CPU_INT10.
       This is the same as DMA_INT2.   */
C6xSetupInterrupt(CPU_INT10,ISN_DMA_INT2,&ReadBifoISR,0);
C6xEnableInterrupt(CPU_INT10);

/*********************Interprocessor Setup ************************/
/*Processor B Flushes the BIFIFO between B and D*/
    P4290FlushBifo(P4290_BIFO_ZZ, P4290_BIFO_CLOCK_FREQ, 8*BUFFER_SIZE-4, 128,
                                                    8*BUFFER_SIZE-4,128);

P4290SetupBifoDMAInt(P4290_BIFO_XX, P4290_INT_IP_BIFO_XX_IN_ALMST_FULL,
        P4290_IMR0_XX_BIFO_IN_EVENT, CPU_INT5, ISN_EXT_INT5, NULL);

/*Calculate frame index for Global Index register for
use of ping-pong buffers.  Difference between end of buffer 1 and start
of buffer 2. */
frameIndex = (int)interData1 - ((int)interData0 + (3*BUFFER_SIZE*4)) + 4;

/* Move frame index into upper half of index value.  Lower half is
   number of bytes in element. */
indexVal = (frameIndex <<16) | 4;

P4290DmaInit((unsigned int)P4290_LCL_FIFO_XX, (unsigned int)interData0,
        DMA_ADDR_NO_MOD, DMA_ADDR_INC, DMA_INDXA, indexVal,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, 3*BUFFER_SIZE, 2,
        DMA_CNT_RELOADB, FRAME_SYNC_ENABLE, DMA_RELOAD_NONE, DMA_RELOAD_GARC,
        SEN_EXT_INT5, SEN_NONE, 0X2088, C6X_DMA_IE,
        DMA_AUTOINIT_TRUE, &dmaParams2);

C6xSetupInterrupt(CPU_INT9,ISN_DMA_INT1,&ReadInterISR,0);
C6xEnableInterrupt(CPU_INT9);
/*********************End of interproc setup *********************/


    /* ***************** SYNCHRONIZE THE 4 PROCESSORS *************** */
P4290_LED1_TOGGLE;

/* Disable Fifo Writes */
P6216_DISABLE_FIFO_WRITE(p6216Regs.control);

/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
```

```
*/
if (procId == SYNC_MASTER)
{
    if (procId == P4290_PROC_A)
            syncBCD(NUM_6216_VIM_MODULES, 1, 0xabababab);
    else  /* Sync Master is C */
            syncABD(NUM_6216_VIM_MODULES, 1, 0xabababab);
}
else
{
    waitForSync(1, 0xabababab);
}
/*Flush the IO Bifo (different Almost Full Levels for each channel)
Processor B is the third in the chain of the beamformer, therefore the
almost full level is set to 5*BUFFER_SIZE-4*/

    P4290FlushBifo(P4290_BIFO_IO, bifoClock, 5*BUFFER_SIZE-4, 128,
            5*BUFFER_SIZE-4, 128);

/* Before proceeding with the Sync Initialization, wait for all the
     other processors to be at this point in the initialization process.
*/
if (procId == SYNC_MASTER)
{
    if (procId == P4290_PROC_A)
            syncBCD(NUM_6216_VIM_MODULES, 1, 0xdcdcdcdc);
        else  /* Sync Master is C */
            syncABD(NUM_6216_VIM_MODULES, 1, 0xdcdcdcdc);
    }
    else
    {
        waitForSync(1, 0xdcdcdcdc);
    }

/* If Sync 1 Master, Generate Sync Pulse */
if (procId == SYNC_MASTER)
{
    P6216_SET_SYNC_GENERATE(p6216Regs.syncGen, 0);
    while ((*p6216Regs.syncGen & 0x1) != 0);
    P6216_SET_SYNC_GENERATE(p6216Regs.syncGen, 1);
}
    /* ************* MAIN I/O TRANSFER LOOP, ON DMA INTERRUPT *** */

/* Set up for DMA BIFO read with synchronization with IO-In AF. */
P4290OurDmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_AUTO_START_VAL,
    CPU_INT4, ISN_EXT_INT4, &dmaParams);
P4290DmaTransfer(DMA_CH1, C6X_DMA_NO_WAIT, DMA_AUTO_START_VAL,
    CPU_INT5, ISN_EXT_INT5, &dmaParams2);
P4290_LED2_ON;

while (1)
{
            /* wait for dma to complete */
    while (!dmaDone)
            ++waitCnt;
    dmaDone = 0;
    IOpingpong=!IOpingpong;
    waitCntArray[loopCnt]=waitCnt;
    while (!interDone)
            ++waitCntInter;
    interDone=0;
    INTERpingpong=!INTERpingpong;
    waitCntInterArray[loopCnt]=waitCntInter;
    InterSend(outInterprocBifo,BeamformerWeightI,BeamformerWeightQ,
            INTERdata[INTERpingpong],TWO_BUFFER_SIZE,SHORTIOdata[IOpingpong]);
    /*Send Raw Data from processor A*/
    RawDataSend(outInterprocBifoInt,INTERdataRAW[IOpingpong],BUFFER_SIZE);
    /*Send Raw Data from processor B*/
    RawDataSend(outInterprocBifoInt,IOdata[IOpingpong],BUFFER_SIZE);
            ++loopCnt;
            if (loopCnt == NUM_LOOPS) {
                    finalwaitcnt=waitCnt;
                    P4290OurDmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_PAUSE_VAL,
                    CPU_INT4, ISN_EXT_INT4, &dmaParams);
            }
```

```
                waitCnt = 0;
        waitCntInter = 0;
    }
}
```

ProcessorC.c

```
 /*************************************************************************
*
*   File : ProcessorC.c
*
*************************************************************************/
#include "stdlib.h"
#include "4290.h"
#include "4290rscm.h"
#include "4290bifo.h"
#include "4290mbx.h"
#include "6216.h"
#include "math.h"
#include "c6xtimer.h"
#include "4290dma.h"
#include "c6xdma.h"
#include "dma.h"
#include "InterSend_C.h"
#include "RawDataSend.h"
/* Swiftnet libs */
#include "pnkcapi.h"
#include "pnkdev.h"
#include "snio.h"

#define DECIMATION_RATE  32

/* Host - target communication defines */
/* Define the API sig for host/target communication */
#define BYUAPI_SIG 1

/* Define other API comm parameters */
#define HOSTNAME "astronomy"
#define DEVICENAME "byuapi"
#define TICK_ULEN 1000
#define TIMEOUT_TICKS 10000

/*  ******* GLOBAL DEFINES ********************  */

/* Number of 6216 VIM Modules being Synchronized */
#define  NUM_6216_VIM_MODULES     2             /* 1 or 2 6216's on 4290   */

/* Synchronization Master Processor - Must be Processor A or C */
#define  SYNC_MASTER              P4290_PROC_A  /* 0 or 2 for CPU's A & C  */

/* Clock Selection - P6216_INTERNAL_CLOCK or P6216_EXTERNAL_CLOCK */
#define CLOCK_SELECT    P6216_INTERNAL_CLOCK

/* The Internal Oscillator Frequency can vary based on installed options
   on the board.
*/
#define INT_OSC_STANDARD      64000000L      /* 6216 Standard       */
#define INT_OSC_OPT20         65000000L      /* 6216 with Option 20 */
#define INT_OSC_OPT21         60000000L      /* 6216 with Option 21 */

/* External Clock Rate - When the 6216 is being clocked externally or it is
                         receiving it's clock from another 6216, specify the
                         external clock frequency.
*/
#define EXTERNAL_CLOCK_RATE   64000000L

/* ********** GLOBAL VARIABLES ******************  */

/* Global Data Buffers */

#define BUFFER_SIZE                     1024
#define TWO_BUFFER_SIZE                 2048
#define HALF_BUFFER_SIZE                512
```

```
#define NUM_LOOPS                               201

#pragma DATA_SECTION(outDataBlock, ".sbsram0");
far int outDataBlock[2*BUFFER_SIZE];
#pragma DATA_SECTION(interDataBlock, ".buffer1");
far float interDataBlock[4*BUFFER_SIZE];

/* Other global variables  -- */
int dmaDone=0;
int interDone=0;
volatile int       loopCnt=0;
volatile int       waitCnt=0;
volatile int       waitCntInter=0;
volatile int       IOpingpong=0;
volatile int       INTERpingpong=0;
/* ************ Function Prototypes ***************** */

void syncABD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void syncBCD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void waitForSync(unsigned int mailbox, unsigned int syncWord);
int calcBifoClock(P6216_BOARD_PARAMS *boardParams,
                  P6216_RCVR_PARAMS *rcvrParams);
void P6216SetBoardParams(P6216_BOARD_PARAMS *boardParams);
void P6216SetRcvrParams(P6216_RCVR_PARAMS *rcvrParams,
                        P6216_BOARD_PARAMS *boardParams,
                        unsigned int intClockRate);

/* Interrupt service routine for BIFIFO dma */

interrupt void ReadBifoISR() {
        /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH2))) &= 0xffbb;

        /* Set dma complete flag */
        dmaDone=1;
}  /* end of ISR */

interrupt void ReadInterISR() {
   /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH1))) &= 0xffbb;

   interDone=1;
}  /* end of ISR */

void main()
{
    P6216_BOARD_PARAMS p6216BoardParams;
    P6216_RCVR_PARAMS  p6216RcvrParams;
    P6216_REG_ADDR     p6216Regs;
    unsigned int       i;
    unsigned int           pasti=0;
    unsigned int       procId = P4290GetProcId();
    volatile float     phase;
    int                bifoClock;
    unsigned int       intClockRate = INT_OSC_STANDARD;
        P4290_DMA_PARAMS   dmaParams;
        P4290_DMA_PARAMS   dmaParams2;
        int                        frameIndex;
        int                                indexVal;
    int                            *outData0,*outData1;
    short                          *outDataShort0,*outDataShort1;
    volatile int           finalwaitcnt;
    float                          *interData0,*interData1;
    volatile float         *outInterprocBifo;
    volatile float         *inInterprocBifo;
    volatile unsigned int       *outInterprocBifoInt;
    /* Variables for host - target communication */
    PNKCAPI_TYPE           pnkc;
    PNK_SIGNAL_TYPE    sig;
    PNK_RETURN_TYPE    ret;
    float                          *INTERdata[2];
    int                            *IOdata[2];
    short                          *SHORTIOdata[2];
```

```
int                             waitCntArray[NUM_LOOPS];
int                             waitCntInterArray[NUM_LOOPS];
int                             NumberLoops=NUM_LOOPS;
    float                       BeamformerWeightI=1;
    float                       BeamformerWeightQ=0;

 /* Host - target initializations */
if (pnkc_init() != OK)      /* Initialize client API */
{
  plog("Error␣initializing␣client␣interface!");
  exit(1);
}

 /* Open up a connection to virtual device existing on host */

    if (pnkc_open(&pnkc, HOSTNAME, DEVICENAME) != OK)
{
  plog("error␣opening␣connection␣to␣host");
  exit(1);
}

    plog("target␣running");

    sig.num=BYUAPI_SIG;
    sig.args.args_len = 0;
sig.args.args_val = NULL;
sig.timeout_ticks = TIMEOUT_TICKS;
sig.tick_ulen = TICK_ULEN;

/* end of host/target comm. setup routines*/

outData0 = outDataBlock;
outData1 = &outDataBlock[BUFFER_SIZE];
IOdata[1]=outData0;
IOdata[0]=outData1;

outDataShort0=(short*)outData0;
outDataShort1=(short*)outData1;

SHORTIOdata[1]=outDataShort0;
SHORTIOdata[0]=outDataShort1;

interData0 = interDataBlock;
interData1 = &interDataBlock[2*BUFFER_SIZE];

    INTERdata[1]=interData0;
INTERdata[0]=interData1;
for (i=0; i<4*BUFFER_SIZE;i++) {
    interDataBlock[i]=0;
}

/*Summary of Port usage for all four processors:

Processor A:        XX Port receives data from Processor C
                        ZZ Port sends data to Processor B

Processor B:        XX Port receives data from Processor A
                        ZZ Port sends data to Processor D

Processor C:        XX Port not used
                        ZZ Port sends data to Processor A

Processor D:        XX Port receives data from Processor B
                        ZZ Port not used

Data flow for Beam Former:
           C --> A --> B --> D

Where Processors C, A and B are connected to antennas through the digital
    receiver*/

outInterprocBifo = (float *)P4290_LCL_FIFO_ZZ;
inInterprocBifo = (float *)P4290_LCL_FIFO_XX;
outInterprocBifoInt = P4290_LCL_FIFO_XX;
```

```
/* Turn off timer 0 - Enabled by bootcode to toggle led */
TIMER_RESET(C6X_TIMER_0);

/* Initialize Table of 6216 Addresses */
P6216InitRegAddr(0x320000L, &p6216Regs);

/* Determine whether option is installed on this board */
if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x21))
    intClockRate = INT_OSC_OPT21;
else if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x20))
    intClockRate = INT_OSC_OPT20;

/* Set 6216 Board Parameters */
P6216SetBoardParams(&p6216BoardParams);

/* Get 6216 Receiver Parameters */
P6216SetRcvrParams(&p6216RcvrParams, &p6216BoardParams, intClockRate);

    /* Set Center Frequency and Decimation  */
p6216RcvrParams.centerFreq = 16.0e6;
p6216RcvrParams.decimationRate = DECIMATION_RATE; /* Set for 1 MHz basband fs */
p6216BoardParams.gainSetting = 6;

/* Reset Board Registers */
P6216ResetRegs(&p6216Regs);

/* Initialize Board Registers */
P6216InitBoardRegs(&p6216BoardParams, &p6216Regs);

/* Initialize Receiver Registers */
P6216InitRcvrRegs(&p6216RcvrParams, &p6216Regs);

/* Calculate Slowest Bifo Clock Rate */
bifoClock = calcBifoClock(&p6216BoardParams, &p6216RcvrParams);

    #ifndef OPTION_330
/* Select IO Mezzanine Bifo */
P4290BifoSelect(P4290_BIFO_IO);
    #endif

    /* ********** SETUP INTERRUPTS AND DMA TRASFER REGISTERS ****** */
/* Disable Cache in CSR register */
/*CSR=CSR & 0xfffffff1f; */

/* Clear interrupt enable registers. */
*P4290_LCR_IER0 = 0x0;
*P4290_LCR_IER1 = 0x0;

 /* Reset Interrupt Mapping Reg */
*P4290_LCR_IMR0 = 0;

/* Reset Miscellaneous Control Register. */
/* Also causes interrupts to be unlatched, which is why this example
   will not run on the standard 4290. */
*P4290_LCR_MCR0 |= 0x00f0;

    /* Set up Bifo almost full interrupt */
P4290SetupBifoDMAInt(P4290_BIFO_IO, P4290_INT_EXP_BIFO_IN_ALMST_FULL,
        P4290_IMR0_IO_BIFO_IN_EVENT, CPU_INT4, ISN_EXT_INT4, NULL);

/*Calculate frame index for Global Index register for
use of ping-pong buffers.  Difference between end of buffer 1 and start
of buffer 2. */
frameIndex = (int)outData1 - ((int)outData0 + (BUFFER_SIZE*4)) + 4;

/* Move frame index into upper half of index value.  Lower half is
   number of bytes in element. */
indexVal = (frameIndex <<16) | 4;

/* This version of DmaInit uses the indexVal calculated above */
P4290DmaInit((unsigned int)P4290_LCL_FIFO_IO, (unsigned int)outData0,
        DMA_ADDR_NO_MOD, DMA_ADDR_INC, DMA_INDXA, indexVal,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, BUFFER_SIZE, 2,
        DMA_CNT_RELOADA, FRAME_SYNC_ENABLE, DMA_RELOAD_NONE, DMA_RELOAD_GARB,
        SEN_EXT_INT4, SEN_NONE, 0X2088, C6X_DMA_IE,
```

```
        DMA_AUTOINIT_TRUE, &dmaParams);


/*  Setup DMA complete interrupt */
/*      Note DMA chan. 2 is hard-wired to internal interrupt CPU_INT10.
        This is the same as DMA_INT2.    */
C6xSetupInterrupt(CPU_INT10,ISN_DMA_INT2,&ReadBifoISR,0);
C6xEnableInterrupt(CPU_INT10);

/*********************Interprocessor Setup ************************/

/*Processor C Flushes the BIFIFO between C and D*/
    P4290FlushBifo(P4290_BIFO_XX, P4290_BIFO_CLOCK_FREQ, 6*BUFFER_SIZE-4, 128,
            6*BUFFER_SIZE-4,128);
/***********************End of interproc setup *********************/

    /* ***************** SYNCHRONIZE THE 4 PROCESSORS **************** */
P4290_LED1_TOGGLE;

/* Disable Fifo Writes */
P6216_DISABLE_FIFO_WRITE(p6216Regs.control);

/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
if (procId == SYNC_MASTER)
{
    if (procId == P4290_PROC_A)
            syncBCD(NUM_6216_VIM_MODULES, 1, 0xabababab);
    else  /* Sync Master is C */
            syncABD(NUM_6216_VIM_MODULES, 1, 0xabababab);
}
else
{
    waitForSync(1, 0xabababab);
}

/*Flush the IO Bifo (different Almost Full Levels for each channel)
Processor C is the first in the chain of the beamformer, therefore the
almost full level is set to BUFFER_SIZE-4*/

    P4290FlushBifo(P4290_BIFO_IO, bifoClock, BUFFER_SIZE-4, 128,
    BUFFER_SIZE-4, 128);

/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
if (procId == SYNC_MASTER)
{
    if (procId == P4290_PROC_A)
            syncBCD(NUM_6216_VIM_MODULES, 1, 0xdcdcdcdc);
        else  /* Sync Master is C */
            syncABD(NUM_6216_VIM_MODULES, 1, 0xdcdcdcdc);
    }
    else
    {
        waitForSync(1, 0xdcdcdcdc);
    }

/* If Sync 1 Master, Generate Sync Pulse */
if (procId == SYNC_MASTER)
{
    P6216_SET_SYNC_GENERATE(p6216Regs.syncGen, 0);
    while ((*p6216Regs.syncGen & 0x1) != 0);
    P6216_SET_SYNC_GENERATE(p6216Regs.syncGen, 1);
}

    /* ************* MAIN I/O TRANSFER LOOP, ON DMA INTERRUPT *** */

/* Set up for DMA BIFO read with synchronization with IO-In AF. */
P4290OurDmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_AUTO_START_VAL,
    CPU_INT4, ISN_EXT_INT4, &dmaParams);

P4290_LED2_ON;
```

```
    while (1)
    {
        while (!dmaDone)
                ++waitCnt;
        dmaDone = 0;
        IOpingpong=!IOpingpong;
        waitCntArray[loopCnt]=waitCnt;
        InterSend_C(outInterprocBifo, BeamformerWeightI, BeamformerWeightQ,
                TWO_BUFFER_SIZE, SHORTIOdata[IOpingpong]);
        RawDataSend(outInterprocBifoInt,IOdata[IOpingpong],BUFFER_SIZE);
                ++loopCnt;
                if (loopCnt == NumberLoops) {
                        finalwaitcnt=waitCnt;
                    P4290OurDmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_PAUSE_VAL,
                CPU_INT4, ISN_EXT_INT4, &dmaParams);
                }
                waitCnt = 0;
        waitCntInter = 0;
    }
}
```

ProcessorD.c

```
/*************************************************************************
*
*   File : ProcessorD.c
*
*************************************************************************/
#include "stdlib.h"
#include "4290.h"
#include "4290rscm.h"
#include "4290bifo.h"
#include "4290mbx.h"
#include "6216.h"
#include "math.h"
#include "c6xtimer.h"
#include "4290dma.h"
#include "c6xdma.h"
#include "dma.h"
#include "InterSend.h"
#include "compute_matrix_diagonal.h"
#include "compute_matrix_corner.h" beamformer and correlation estimator
/* Swiftnet libs */
#include "pnkcapi.h"
#include "pnkdev.h"
#include "snio.h"

#define DECIMATION_RATE  32

/* Host - target communication defines */
/* Define the API sig for host/target communication */
#define BYUAPI_SIG 1

/* Define other API comm parameters */
#define HOSTNAME "astronomy"
#define DEVICENAME "byuapi"
#define TICK_ULEN 1000
#define TIMEOUT_TICKS 10000

/*  ******* GLOBAL DEFINES ********************  */

/* Number of 6216 VIM Modules being Synchronized */
#define  NUM_6216_VIM_MODULES     2             /* 1 or 2 6216's on 4290   */

/* Synchronization Master Processor - Must be Processor A or C */
#define  SYNC_MASTER              P4290_PROC_A  /* 0 or 2 for CPU's A & C  */

/* Clock Selection - P6216_INTERNAL_CLOCK or P6216_EXTERNAL_CLOCK */
#define CLOCK_SELECT    P6216_INTERNAL_CLOCK

/* The Internal Oscillator Frequency can vary based on installed options
   on the board.
*/
#define INT_OSC_STANDARD      64000000L      /* 6216 Standard       */
```

```
#define INT_OSC_OPT20          65000000L      /* 6216 with Option 20 */
#define INT_OSC_OPT21          60000000L      /* 6216 with Option 21 */

/* External Clock Rate - When the 6216 is being clocked externally or it is
                         receiving it's clock from another 6216, specify the
                         external clock frequency.
*/
#define EXTERNAL_CLOCK_RATE   64000000L

/* ********** GLOBAL VARIABLES ******************  */

/* Global Data Buffers */

#define BUFFER_SIZE                     1024
#define TWO_BUFFER_SIZE                 2048
#define HALF_BUFFER_SIZE                512

#define NUM_LOOPS                               196

#pragma DATA_SECTION(outDataBlock, ".sbsram0");
far int outDataBlock[2*BUFFER_SIZE];
#pragma DATA_SECTION(interDataBlock, ".buffer1");
far float interDataBlock[8*BUFFER_SIZE];

/* Other global variables  -- */
int dmaDone=0;
int interDone=0;
int CdataDone=0;
volatile int       loopCnt=0;
volatile int       waitCnt=0;
volatile int       waitCntInter=0;
volatile int       IOpingpong=0;
volatile int       INTERpingpong=0;
/* ************ Function Prototypes **************** */

void syncABD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void syncBCD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void waitForSync(unsigned int mailbox, unsigned int syncWord);
int calcBifoClock(P6216_BOARD_PARAMS *boardParams,
                  P6216_RCVR_PARAMS *rcvrParams);
void P6216SetBoardParams(P6216_BOARD_PARAMS *boardParams);
void P6216SetRcvrParams(P6216_RCVR_PARAMS *rcvrParams,
                        P6216_BOARD_PARAMS *boardParams,
                        unsigned int intClockRate);

/* Interrupt service routine for BIFIFO dma */

interrupt void ReadBifoISR() {
        /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH2))) &= 0xffbb;

        /* Set dma complete flag */
        dmaDone=1;
}  /* end of ISR */

interrupt void ReadInterISR() {
   /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH1))) &= 0xffbb;

   interDone=1;
}  /* end of ISR */

interrupt void ReadCdataISR() {

   /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH2))) &= 0xffbb;

   CdataDone=1;
}  /* end of ISR */

void main()
{
    P6216_BOARD_PARAMS             p6216BoardParams;
```

```
P6216_RCVR_PARAMS               p6216RcvrParams;
P6216_REG_ADDR                  p6216Regs;
unsigned int                    i;
unsigned int                    pasti=0;
unsigned int                    procId = P4290GetProcId();
volatile float                  phase;
int                             bifoClock;
unsigned int                    intClockRate = INT_OSC_STANDARD;
P4290_DMA_PARAMS                dmaParams;
P4290_DMA_PARAMS                dmaParams2;
int                             frameIndex;
int                             indexVal;
int                             *outData0,*outData1;
short                           *outDataShort0,*outDataShort1;
volatile int                    finalwaitcnt;
float                           *interData0,*interData1;
int                             *interDataRaw0,*interDataRaw1;
volatile float                  *outInterprocBifo;
volatile float                  *inInterprocBifo;
/* Variables for host - target communication */
PNKCAPI_TYPE                    pnkc;
PNK_SIGNAL_TYPE                 sig;
PNK_RETURN_TYPE                 ret;
float                           *INTERdata[2];
int                             *IOdata[2];
short                           *SHORTIOdata[2];
int                             *INTERdataRAW[2];
int                             waitCntArray[NUM_LOOPS];
int                             waitCntInterArray[NUM_LOOPS];
float                           *corr_matrix, *corr_matrix_cum;
int                             N=0;

    corr_matrix = (float *)malloc(9*sizeof(float));
    for (i=0;i<9;i++)  {
    corr_matrix[i]=0;
}
corr_matrix_cum = (float *)malloc(9*sizeof(float));
    for (i=0;i<9;i++)  {
    corr_matrix_cum[i]=0;
}

 /* Host - target initializations */
if (pnkc_init() != OK)      /* Initialize client API */
{
  plog("Error␣initializing␣client␣interface!");
  exit(1);
}

 /* Open up a connection to virtual device existing on host */

    if (pnkc_open(&pnkc, HOSTNAME, DEVICENAME) != OK)
{
  plog("error␣opening␣connection␣to␣host");
  exit(1);
}

    plog("target␣running");

    sig.num=BYUAPI_SIG;
    sig.args.args_len = 0;
sig.args.args_val = NULL;
sig.timeout_ticks = TIMEOUT_TICKS;
sig.tick_ulen = TICK_ULEN;

/* end of host/target comm. setup routines*/

outData0 = outDataBlock;
outData1 = &outDataBlock[BUFFER_SIZE];
IOdata[1]=outData0;
IOdata[0]=outData1;

outDataShort0=(short*)outData0;
outDataShort1=(short*)outData1;

SHORTIOdata[1]=outDataShort0;
```

```
SHORTIOdata[0]=outDataShort1;

interData0 = interDataBlock;
interData1 = &interDataBlock[4*BUFFER_SIZE];

    INTERdata[1]=interData0;
INTERdata[0]=interData1;

interDataRaw0 = (int *)(&interDataBlock[2*BUFFER_SIZE]);
interDataRaw1 = (int *)(&interDataBlock[6*BUFFER_SIZE]);

INTERdataRAW[1]=interDataRaw0;
INTERdataRAW[0]=interDataRaw1;

for (i=0; i<4*BUFFER_SIZE;i++) {
    interDataBlock[i]=0;
}

/*Summary of Port usage for all four processors:

Processor A:        XX Port receives data from Processor C
                    ZZ Port sends data to Processor B

Processor B:        XX Port receives data from Processor A
                    ZZ Port sends data to Processor D

Processor C:        XX Port not used
                    ZZ Port sends data to Processor A

Processor D:        XX Port receives data from Processor B
                    ZZ Port not used

Data flow for Beam Former:
           C --> A --> B --> D

Where Processors C, A and B are connected to antennas through the digital
    receiver*/

outInterprocBifo = (float *)P4290_LCL_FIFO_ZZ;
inInterprocBifo = (float *)P4290_LCL_FIFO_XX;

/* Turn off timer 0 - Enabled by bootcode to toggle led */
TIMER_RESET(C6X_TIMER_0);

/* Initialize Table of 6216 Addresses */
P6216InitRegAddr(0x320000L, &p6216Regs);

/* Determine whether option is installed on this board */
if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x21))
    intClockRate = INT_OSC_OPT21;
else if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x20))
    intClockRate = INT_OSC_OPT20;

/* Set 6216 Board Parameters */
P6216SetBoardParams(&p6216BoardParams);

/* Get 6216 Receiver Parameters */
P6216SetRcvrParams(&p6216RcvrParams, &p6216BoardParams, intClockRate);

    /* Set Center Frequency and Decimation  */
p6216RcvrParams.centerFreq = 16.0e6;
p6216RcvrParams.decimationRate = DECIMATION_RATE; /* Set for 1 MHz basband fs */
p6216BoardParams.gainSetting = 6;

/* Reset Board Registers */
P6216ResetRegs(&p6216Regs);

/* Initialize Board Registers */
P6216InitBoardRegs(&p6216BoardParams, &p6216Regs);

/* Initialize Receiver Registers */
P6216InitRcvrRegs(&p6216RcvrParams, &p6216Regs);

/* Calculate Slowest Bifo Clock Rate */
bifoClock = calcBifoClock(&p6216BoardParams, &p6216RcvrParams);
```

```
    #ifndef OPTION_330
/* Select IO Mezzanine Bifo */
P4290BifoSelect(P4290_BIFO_IO);
    #endif

    /* ********** SETUP INTERRUPTS AND DMA TRASFER REGISTERS ****** */

/* Clear interrupt enable registers. */
*P4290_LCR_IER0 = 0x0;
*P4290_LCR_IER1 = 0x0;

 /* Reset Interrupt Mapping Reg */
*P4290_LCR_IMR0 = 0;

/* Reset Miscellaneous Control Register. */
/* Also causes interrupts to be unlatched, which is why this example
   will not run on the standard 4290. */
*P4290_LCR_MCR0 |= 0x00f0;

/*********************Interprocessor Setup ************************/
    /*Processor D doesn't need to flush any of the interprocessor BIFIFOs*/

P4290SetupBifoDMAInt(P4290_BIFO_XX, P4290_INT_IP_BIFO_XX_IN_ALMST_FULL,
        P4290_IMR0_XX_BIFO_IN_EVENT, CPU_INT5, ISN_EXT_INT5, NULL);

/*Calculate frame index for Global Index register for
use of ping-pong buffers.  Difference between end of buffer 1 and start
of buffer 2. */
frameIndex = (int)interData1 - ((int)interData0 + (4*BUFFER_SIZE*4)) + 4;

/* Move frame index into upper half of index value.  Lower half is
   number of bytes in element. */
indexVal = (frameIndex <<16) | 4;

P4290DmaInit((unsigned int)P4290_LCL_FIFO_XX, (unsigned int)interData0,
        DMA_ADDR_NO_MOD, DMA_ADDR_INC, DMA_INDXA, indexVal,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, 4*BUFFER_SIZE, 2,
        DMA_CNT_RELOADB, FRAME_SYNC_ENABLE, DMA_RELOAD_NONE, DMA_RELOAD_GARC,
        SEN_EXT_INT5, SEN_NONE, 0X2088, C6X_DMA_IE,
        DMA_AUTOINIT_TRUE, &dmaParams2);

C6xSetupInterrupt(CPU_INT9,ISN_DMA_INT1,&ReadInterISR,0);
C6xEnableInterrupt(CPU_INT9);
/**********************End of interproc setup **********************/

/********DMA transfer between proc. C and D for corr. matrix********/

P4290SetupBifoDMAInt(P4290_BIFO_ZZ, P4290_INT_IP_BIFO_ZZ_IN_ALMST_FULL,
        P4290_IMR0_ZZ_BIFO_IN_EVENT, CPU_INT4, ISN_EXT_INT4, NULL);

/*Calculate frame index for Global Index register for
use of ping-pong buffers.  Difference between end of buffer 1 and start
of buffer 2. */
frameIndex = (int)outData1 - ((int)outData0 + (BUFFER_SIZE*4)) + 4;

/* Move frame index into upper half of index value.  Lower half is
   number of bytes in element. */
indexVal = (frameIndex <<16) | 4;

/* This version of DmaInit uses the indexVal calculated above */
P4290DmaInit((unsigned int)P4290_LCL_FIFO_ZZ, (unsigned int)outData0,
        DMA_ADDR_NO_MOD, DMA_ADDR_INC, DMA_INDXA, indexVal,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, BUFFER_SIZE, 2,
        DMA_CNT_RELOADA, FRAME_SYNC_ENABLE, DMA_RELOAD_NONE, DMA_RELOAD_GARB,
        SEN_EXT_INT4, SEN_NONE, 0X2088, C6X_DMA_IE,
        DMA_AUTOINIT_TRUE, &dmaParams);

/*  Setup DMA complete interrupt */
/*      Note DMA chan. 2 is hard-wired to internal interrupt CPU_INT10.
        This is the same as DMA_INT2.   */
C6xSetupInterrupt(CPU_INT10,ISN_DMA_INT2,&ReadCdataISR,0);
C6xEnableInterrupt(CPU_INT10);
/*****************************End DMA transfer setup***************/
```

```
    /* ***************** SYNCHRONIZE THE 4 PROCESSORS *************** */
P4290_LED1_TOGGLE;

/* Disable Fifo Writes */
P6216_DISABLE_FIFO_WRITE(p6216Regs.control);

/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
if (procId == SYNC_MASTER)
{
    if (procId == P4290_PROC_A)
            syncBCD(NUM_6216_VIM_MODULES, 1, 0xabababab);
    else  /* Sync Master is C */
            syncABD(NUM_6216_VIM_MODULES, 1, 0xabababab);
}
else
{
    waitForSync(1, 0xabababab);
}


/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
if (procId == SYNC_MASTER)
{
    if (procId == P4290_PROC_A)
            syncBCD(NUM_6216_VIM_MODULES, 1, 0xdcdcdcdc);
        else  /* Sync Master is C */
            syncABD(NUM_6216_VIM_MODULES, 1, 0xdcdcdcdc);
    }
    else
    {
        waitForSync(1, 0xdcdcdcdc);
    }

/* If Sync 1 Master, Generate Sync Pulse */
if (procId == SYNC_MASTER)
{
    P6216_SET_SYNC_GENERATE(p6216Regs.syncGen, 0);
    while ((*p6216Regs.syncGen & 0x1) != 0);
    P6216_SET_SYNC_GENERATE(p6216Regs.syncGen, 1);
}

    /* ************* MAIN I/O TRANSFER LOOP, ON DMA INTERRUPT *** */
P4290DmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_AUTO_START_VAL,
    CPU_INT4, ISN_EXT_INT4, &dmaParams);
P4290DmaTransfer(DMA_CH1, C6X_DMA_NO_WAIT, DMA_AUTO_START_VAL,
    CPU_INT5, ISN_EXT_INT5, &dmaParams2);
P4290_LED2_ON;

while (1)
{
    while (!interDone)
            ++waitCntInter;
    interDone=0;
    INTERpingpong=!INTERpingpong;
    waitCntInterArray[loopCnt]=waitCntInter;
    while (!CdataDone) {
            ++waitCnt;
    }
    CdataDone=0;
    IOpingpong=!IOpingpong;
    waitCntArray[loopCnt]=waitCnt;

    compute_matrix_diagonal(corr_matrix,(short*)&INTERdata[INTERpingpong][2048],
            (short*)&INTERdata[INTERpingpong][3072], (short*)IOdata[IOpingpong],
                TWO_BUFFER_SIZE);
    compute_matrix_corner(corr_matrix,(short*)&INTERdata[INTERpingpong][2048],
            (short*)&INTERdata[INTERpingpong][3072], (short*)IOdata[IOpingpong],
                TWO_BUFFER_SIZE);

    if (loopCnt>0) {
            for (i=0;i<9;i++)  {
```

```
                        corr_matrix_cum[i]=corr_matrix[i]+corr_matrix_cum[i];
                }
                N++;
        }

                ++loopCnt;
                if (loopCnt == NUM_LOOPS) {
                                finalwaitcnt=waitCnt;

                                P4290OurDmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT,
                                    DMA_PAUSE_VAL,
                        CPU_INT4, ISN_EXT_INT4, &dmaParams);

                P4290OurDmaTransfer(DMA_CH1, C6X_DMA_NO_WAIT, DMA_PAUSE_VAL,
                        CPU_INT5, ISN_EXT_INT5, &dmaParams2);

                /*Send Beamformer data to host*/
                sig.data.data_val=(u_int_32 *)interData0;
                        sig.data.data_len=1024;
                pnkc_send(&pnkc,0,&sig,&ret);
                sig.data.data_val=(u_int_32 *)(&interData0[1024]);
                        sig.data.data_len=1024;
                pnkc_send(&pnkc,0,&sig,&ret);
                        sig.data.data_val=(u_int_32 *)interData1;
                        sig.data.data_len=1024;
                        pnkc_send(&pnkc,0,&sig,&ret);
                        sig.data.data_val=(u_int_32 *)(&interData1[1024]);
                        sig.data.data_len=1024;
                pnkc_send(&pnkc,0,&sig,&ret);

                /*Send Raw Data from Processor A to host*/
                sig.data.data_val=(u_int_32 *)interDataRaw0;
                        sig.data.data_len=1024;
                pnkc_send(&pnkc,0,&sig,&ret);
                sig.data.data_val=(u_int_32 *)interDataRaw1;
                        sig.data.data_len=1024;
                pnkc_send(&pnkc,0,&sig,&ret);

                /*Send Raw Data from Processor B to host*/
                sig.data.data_val=(u_int_32 *)(&interDataRaw0[1024]);
                        sig.data.data_len=1024;
                pnkc_send(&pnkc,0,&sig,&ret);
                sig.data.data_val=(u_int_32 *)(&interDataRaw1[1024]);
                        sig.data.data_len=1024;
                pnkc_send(&pnkc,0,&sig,&ret);

                /*Send Raw Data from Processor C to host*/
                sig.data.data_val=(u_int_32 *)outData0;
                        sig.data.data_len=1024;
                pnkc_send(&pnkc,0,&sig,&ret);
                sig.data.data_val=(u_int_32 *)outData1;
                        sig.data.data_len=1024;
                pnkc_send(&pnkc,0,&sig,&ret);
                }
                waitCnt = 0;
        waitCntInter = 0;
    }
}
```

intersend.c

```
/*
Compute beamforming data in Processors A and B.
Send beamforming data from A to B or B to D without using DMA.

INPUT PARAMETERS:
        unsigned int* outInterprocBifo
        float BeamformerWeight
        short* IOdata
        float* INTERdata
        int BufferSize

RETURN PARAMETER:
        float*
```

```
*/

void InterSend (volatile float* outInterprocBifo, float BeamformerWeightI,
                                float BeamformerWeightQ, float* INTERdata, int
                                    LengthOfBuffers,
                                short* SHORTIOdata)
{
    unsigned int i;
    float tempI,tempQ;

        for (i=0;i<LengthOfBuffers;i+=4) {
                tempI=(float)SHORTIOdata[i];
                tempQ=(float)SHORTIOdata[i+1];
                *outInterprocBifo=BeamformerWeightI*tempI-BeamformerWeightQ*tempQ+
                    INTERdata[i];
        *outInterprocBifo=BeamformerWeightQ*tempI+BeamformerWeightI*tempQ+INTERdata[i
            +1];
        tempI=(float)SHORTIOdata[i+2];
                tempQ=(float)SHORTIOdata[i+3];
        *outInterprocBifo=BeamformerWeightI*tempI-BeamformerWeightQ*tempQ+INTERdata[i
            +2];
        *outInterprocBifo=BeamformerWeightQ*tempI+BeamformerWeightI*tempQ+INTERdata[i
            +3];

        }
}
```

intersend_opt.asm

```
;******************************************************************************
;* TMS320C6x ANSI C Codegen                                      Version 4.00 *
;* Date/Time created: Thu Aug 02 14:43:09 2001                                *
;******************************************************************************

;******************************************************************************
;* GLOBAL FILE PARAMETERS                                                     *
;*                                                                            *
;*   Architecture      : TMS320C670x                                          *
;*   Optimization      : Enabled at level 3                                   *
;*   Optimizing for    : Speed                                                *
;*                       Based on options: -o3, no -ms                        *
;*   Endian            : Big                                                  *
;*   Interrupt Thrshld : Disabled                                             *
;*   Memory Model      : Large                                                *
;*   Calls to RTS      : Far                                                  *
;*   Pipelining        : Enabled                                              *
;*   Speculative Load  : Enabled (Threshold = 32                             ) *
;*   Memory Aliases    : Presume are aliases (pessimistic)                    *
;*   Debug Info        : Debug                                                *
;*                                                                            *
;******************************************************************************

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 C:\TEMP\TI194_2 C:\TEMP\TI194_4 -w C:\ti\myprojects\
    astronomy
        .sect   ".text"
        .global _InterSend
        .sym    _InterSend,_InterSend, 32, 2, 0
        .func   41

;******************************************************************************
;* FUNCTION NAME: _InterSend                                                  *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Local Frame Size  : 0 Args + 0 Auto + 44 Save = 44 byte                  *
```

```
;*****************************************************************************
_InterSend:
;** ---------------------------------------------------------------------------*
        .sym    _outInterprocBifo,4, 22, 17, 32
        .sym    _BeamformerWeightI,20, 6, 17, 32
        .sym    _BeamformerWeightQ,6, 6, 17, 32
        .sym    _INTERdata,22, 22, 17, 32
        .sym    _LengthOfBuffers,8, 4, 17, 32
        .sym    _SHORTIOdata,24, 19, 17, 32
        .sym    C$1,0, 3, 4, 16
        .sym    C$2,0, 3, 4, 16
        .sym    C$3,0, 3, 4, 16
        .sym    C$4,0, 3, 4, 16
        .sym    U$12,14, 19, 4, 32
        .sym    U$28,17, 22, 4, 32
        .sym    L$1,0, 4, 4, 32
        .sym    _SHORTIOdata,24, 19, 4, 32
        .sym    _LengthOfBuffers,1, 4, 4, 32
        .sym    _INTERdata,22, 22, 4, 32
        .sym    _BeamformerWeightQ,6, 6, 4, 32
        .sym    _BeamformerWeightI,12, 6, 4, 32
        .sym    _outInterprocBifo,4, 22, 4, 32
        .sym    _tempI,0, 6, 4, 32
        .sym    _tempI,0, 6, 4, 32
        .sym    _tempQ,0, 6, 4, 32
        .sym    _tempQ,0, 6, 4, 32
           STW      .D2T2   B13,*SP--(48)      ; |41|
           STW      .D2T1   A10,*+SP(8)        ; |41|
           STW      .D2T1   A11,*+SP(12)       ; |41|
           STW      .D2T1   A12,*+SP(16)       ; |41|
           STW      .D2T1   A13,*+SP(20)       ; |41|
           STW      .D2T1   A14,*+SP(24)       ; |41|
           STW      .D2T2   B3,*+SP(28)        ; |41|
           STW      .D2T2   B10,*+SP(32)       ; |41|
           STW      .D2T2   B11,*+SP(36)       ; |41|
           STW      .D2T2   B12,*+SP(40)       ; |41|
           STW      .D2T1   A15,*+SP(44)       ; |41|

           MV       .L1     A8,A1              ;
||         MV       .S1X    B4,A12             ;

   [!A1]   B        .S1     L4                 ; |48|
           SUB      .L1X    B8,8,A14           ; |48|
           SUB      .D2     B6,16,B1           ; |48|
           NOP              3
           ; BRANCH OCCURS                     ; |48|
;** ---------------------------------------------------------------------------*
           LDH      .D1T1   *++A14(8),A3       ; (P) |49|
           LDH      .D1T1   *+A14(2),A7        ; (P) |50|
           LDH      .D1T1   *+A14(4),A0        ; (P) |53|
           LDH      .D1T1   *+A14(6),A9        ; (P) |54|
           MV       .S1     A6,A11
           MV       .L2X    A4,B10
           INTSP    .L1     A3,A8              ; (P) |49|
           INTSP    .L1     A3,A3              ; (P) |51|
           INTSP    .L1     A7,A5              ; (P) |51|

           INTSP    .L2X    A0,B4              ; (P) |55|
||         INTSP    .L1     A0,A0              ; (P) |53|

           LDH      .D1T1   *++A14(8),A4       ; (P) @|49|
||         MPYSP    .M1     A11,A8,A10         ;

           INTSP    .L1     A7,A6              ; (P) |50|
||         MPYSP    .M1     A12,A3,A3          ; (P) |51|
||         LDH      .D1T1   *+A14(2),A15       ; (P) @|50|

           MVK      .S1     0x1,A2             ; init prolog collapse predicate
||         ADD      .D1     3,A1,A5            ; |49|
||         MPYSP    .M1     A11,A5,A13         ; (P) |51|
||         INTSP    .L1     A9,A8              ; (P) |54|

           SHRU     .S1     A5,2,A0            ; |49|
||         MV       .L1     A0,A5              ; (P) |53|
||         MPYSP    .M2X    A12,B4,B11         ; (P) |55|
```

```
           ADD     .S1     1,A0,A1           ;
||         MPYSP   .M1     A11,A5,A0         ; (P) |56|
||         LDH     .D1T1   *+A14(4),A5       ; (P) @|53|
||         INTSP   .L2X    A9,B9             ; (P) |55|

;*----------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Known Minimum Trip Count         : 1
;*      Known Maximum Trip Count         : 1073741823
;*      Known Max Trip Count Factor      : 1
;*      Loop Carried Dependency Bound(^) : 4
;*      Unpartitioned Resource Bound     : 8
;*      Partitioned Resource Bound(*)    : 9
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                     8        8
;*      .S units                     1        0
;*      .D units                     4        8
;*      .M units                     5        3
;*      .X cross paths               0        9*
;*      .T address paths             4        8
;*      Long read paths              0        4
;*      Long write paths             0        0
;*      Logical  ops (.LS)           0        2     (.L or .S unit)
;*      Addition ops (.LSD)          4        0     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             5        5
;*      Bound(.L .S .D .LS .LSD)     6        6
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 9  Cannot allocate machine registers
;*                   Regs Live Always   :  4/2  (A/B-side)
;*                   Max Regs Live      : 15/14
;*                   Max Cond Regs Live :  1/0
;*         ii = 9  Register is live too long
;*         ii = 9  Cannot allocate machine registers
;*                   Regs Live Always   :  4/2  (A/B-side)
;*                   Max Regs Live      : 16/13
;*                   Max Cond Regs Live :  1/0
;*         ii = 10 Schedule found with 4 iterations in parallel
;*      done
;*
;*      Loop is interruptible
;*      Collapsed epilog stages     : 3
;*      Prolog not entirely removed
;*      Collapsed prolog stages     : 1
;*
;*      Minimum required memory pad : 24 bytes
;*
;*      Minimum safe trip count     : 1
;*----------------------------------------------------------------------------*
L1:    ; PIPED LOOP PROLOG
;** ---------------------------------------------------------------------------*
L2:    ; PIPED LOOP KERNEL

           MV      .S2X    A6,B0             ; @|50|
||         INTSP   .L1     A4,A6             ; @@|49|

           ADDSP   .L2     B4,B7,B3          ; |52|
||         MPYSP   .M2X    A12,B0,B4         ; @|52|
||         SUBSP   .L1     A3,A13,A13        ; @|51|
||         LDH     .D1T1   *+A14(6),A9       ; @@|54|

   [!A2]   STW     .D2T2   B5,*B10           ;  ^ |51|
||         ADDSP   .L2X    B8,A7,B0          ; |56|
||         MV      .S1     A8,A4             ; @|54|
||         INTSP   .L1     A4,A3             ; @@|51|

   [ A1]   SUB     .S1     A1,1,A1           ; |58|
||         ADDSP   .L2     B6,B13,B12        ; |55|
||         LDW     .D2T2   *++B1(16),B2      ; @|51|
||         MPYSP   .M2X    A11,B9,B13        ; @|55|
||         MPYSP   .M1     A12,A4,A7         ; @|56|
||         INTSP   .L1     A15,A8            ; @@|51|
```

```
   [ A1]   B       .S1     L2                ; |58|
||         LDW     .D2T2   *+B1(12),B8       ; @|56|
||         INTSP   .L2X    A5,B9             ; @@|55|
||         MV      .D1     A6,A4             ; @@|49|
||         INTSP   .L1     A5,A5             ; @@|53|

           LDW     .D2T2   *+B1(8),B6        ; @|55|
||         ADDSP   .L2X    B4,A10,B7         ; @|52|
||         INTSP   .L1     A15,A6            ; @@|50|
||         MPYSP   .M1     A11,A4,A10        ; @@|52|
||         LDH     .D1T1   *++A14(8),A4      ; @@@|49|

           LDW     .D2T2   *+B1(4),B4        ; @|52|
||         MV      .S2X    A13,B5            ; @Define a twin register
||         MPYSP   .M1     A12,A3,A3         ; @@|51|
||         LDH     .D1T1   *+A14(2),A15      ; @@@|50|

   [!A2]   STW     .D2T2   B3,*B10           ;  ^ |52|
||         SUBSP   .L2     B11,B13,B13       ; @|55|
||         INTSP   .L1     A9,A8             ; @@|54|
||         MPYSP   .M1     A11,A8,A13        ; @@|51|

   [!A2]   STW     .D2T2   B12,*B10          ;  ^ |55|
||         ADDSP   .L2     B2,B5,B5          ; @|51|
||         ADDSP   .L1     A7,A0,A7          ; @|56|
||         MPYSP   .M2X    A12,B9,B11        ; @@|55|
||         MV      .S1     A5,A0             ; @@|53|

   [ A2]   SUB     .L1     A2,1,A2           ;
|| [!A2]   STW     .D2T2   B0,*B10           ;  ^ |56|
||         INTSP   .L2X    A9,B9             ; @@|55|
||         MPYSP   .M1     A11,A0,A0         ; @@|56|
||         LDH     .D1T1   *+A14(4),A5       ; @@@|53|

;** --------------------------------------------------------------------------*
L3:    ; PIPED LOOP EPILOG
;** --------------------------------------------------------------------------*
L4:

           LDW     .D2T2   *+SP(28),B3       ; |59|
           LDW     .D2T2   *+SP(40),B12      ; |59|
           LDW     .D2T2   *+SP(36),B11      ; |59|
           LDW     .D2T2   *+SP(32),B10      ; |59|
           LDW     .D2T1   *+SP(24),A14      ; |59|
           LDW     .D2T1   *+SP(20),A13      ; |59|
           LDW     .D2T1   *+SP(16),A12      ; |59|
           LDW     .D2T1   *+SP(12),A11      ; |59|
           LDW     .D2T1   *+SP(8),A10       ; |59|

           B       .S2     B3                ; |59|
||         LDW     .D2T1   *+SP(44),A15      ; |59|

           LDW     .D2T2   *++SP(48),B13     ; |59|
           NOP             4
           ; BRANCH OCCURS                   ; |59|
        .endfunc        59,03c08fc00h,48
```

intersend_C.c

```
/*
Compute beamforming data in Processor C.
Send beamforming data from C to A without using DMA.

INPUT PARAMETERS:
        unsigned int* outInterprocBifo
        float BeamformerWeight
        short* IOdata
        float* INTERdata
        int BufferSize

RETURN PARAMETER:
        float*
*/
```

```
void InterSend_C (volatile float* outInterprocBifo, float BeamformerWeightI,
                                float BeamformerWeightQ, int LengthOfBuffers, short*
                                    SHORTIOdata)
{
    unsigned int i;
    float tempI,tempQ;

        for (i=0;i<LengthOfBuffers;i+=4) {
                tempI=(float)SHORTIOdata[i];
                tempQ=(float)SHORTIOdata[i+1];
                *outInterprocBifo=BeamformerWeightI*tempI-BeamformerWeightQ*tempQ;
        *outInterprocBifo=BeamformerWeightQ*tempI+BeamformerWeightI*tempQ;
        tempI=(float)SHORTIOdata[i+2];
                tempQ=(float)SHORTIOdata[i+3];
        *outInterprocBifo=BeamformerWeightI*tempI-BeamformerWeightQ*tempQ;
        *outInterprocBifo=BeamformerWeightQ*tempI+BeamformerWeightI*tempQ;
        }
}
```

intersend_C_opt.asm

```
;******************************************************************************
;* TMS320C6x ANSI C Codegen                                      Version 4.00 *
;* Date/Time created: Thu Aug 02 13:59:37 2001                                *
;******************************************************************************

;******************************************************************************
;* GLOBAL FILE PARAMETERS                                                     *
;*                                                                            *
;*   Architecture      : TMS320C670x                                          *
;*   Optimization      : Enabled at level 3                                   *
;*   Optimizing for    : Speed                                                *
;*                       Based on options: -o3, no -ms                        *
;*   Endian            : Big                                                  *
;*   Interrupt Thrshld : Disabled                                             *
;*   Memory Model      : Large                                                *
;*   Calls to RTS      : Far                                                  *
;*   Pipelining        : Enabled                                              *
;*   Speculative Load  : Enabled (Threshold = 32                             ) *
;*   Memory Aliases    : Presume are aliases (pessimistic)                    *
;*   Debug Info        : Debug                                                *
;*                                                                            *
;******************************************************************************

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 C:\TEMP\TI278_2 C:\TEMP\TI278_4 -w C:\ti\myprojects\
    astronomy
        .file   "intersend_c.c"
        .sect   ".text"
        .global _InterSend_C
        .sym    _InterSend_C,_InterSend_C, 32, 2, 0
        .func   60

;******************************************************************************
;* FUNCTION NAME: _InterSend_C                                                *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Local Frame Size  : 0 Args + 0 Auto + 44 Save = 44 byte                  *
;******************************************************************************
_InterSend_C:
;** --------------------------------------------------------------------------*
        .sym    _outInterprocBifo,4, 22, 17, 32
        .sym    _BeamformerWeightI,20, 6, 17, 32
        .sym    _BeamformerWeightQ,6, 6, 17, 32
```

```
        .sym    _LengthOfBuffers,22, 4, 17, 32
        .sym    _SHORTIOdata,8, 19, 17, 32
        .sym    C$1,0, 3, 4, 16
        .sym    C$2,0, 3, 4, 16
        .sym    C$3,0, 3, 4, 16
        .sym    C$4,0, 3, 4, 16
        .sym    U$11,0, 19, 4, 32
        .sym    L$1,21, 4, 4, 32
        .sym    _SHORTIOdata,8, 19, 4, 32
        .sym    _LengthOfBuffers,16, 4, 4, 32
        .sym    _BeamformerWeightQ,26, 6, 4, 32
        .sym    _BeamformerWeightI,20, 6, 4, 32
        .sym    _outInterprocBifo,19, 22, 4, 32
        .sym    _tempI,0, 6, 4, 32
        .sym    _tempI,0, 6, 4, 32
        .sym    _tempQ,0, 6, 4, 32
        .sym    _tempQ,0, 6, 4, 32
           STW     .D2T2   B13,*SP--(48)      ; |60|
           STW     .D2T1   A10,*+SP(8)        ; |60|
           STW     .D2T1   A11,*+SP(12)       ; |60|
           STW     .D2T1   A12,*+SP(16)       ; |60|
           STW     .D2T1   A13,*+SP(20)       ; |60|
           STW     .D2T1   A14,*+SP(24)       ; |60|
           STW     .D2T2   B3,*+SP(28)        ; |60|
           STW     .D2T2   B10,*+SP(32)       ; |60|
           STW     .D2T2   B11,*+SP(36)       ; |60|
           STW     .D2T2   B12,*+SP(40)       ; |60|
           STW     .D2T1   A15,*+SP(44)       ; |60|
           MV      .L2X    A4,B3              ;

           MV      .L2     B6,B0              ;
||         MV      .S2X    A6,B10             ;

   [!B0]   B       .S1     L4                 ; |66|
           SUB     .L1     A8,8,A0            ; |66|
           NOP             4
           ; BRANCH OCCURS                    ; |66|
;** --------------------------------------------------------------------------*
           ADD     .L2     3,B0,B5            ; |67|
           LDH     .D1T1   *++A0(8),A8        ; (P) |67|
           SHRU    .S2     B5,2,B5            ; |67|

           MV      .L1X    B4,A9
||         MVK     .S1     0x3,A1             ; init prolog collapse predicate
||         ADD     .L2     2,B5,B0            ;

;*----------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Known Minimum Trip Count         : 1
;*      Known Maximum Trip Count         : 1073741823
;*      Known Max Trip Count Factor      : 1
;*      Loop Carried Dependency Bound(^) : 6
;*      Unpartitioned Resource Bound     : 6
;*      Partitioned Resource Bound(*)    : 6
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                     6*       6*
;*      .S units                     0        1
;*      .D units                     4        4
;*      .M units                     4        4
;*      .X cross paths               3        5
;*      .T address paths             5        3
;*      Long read paths              2        2
;*      Long write paths             0        0
;*      Logical  ops (.LS)           0        0     (.L or .S unit)
;*      Addition ops (.LSD)          3        2     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             3        4
;*      Bound(.L .S .D .LS .LSD)     5        5
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 6  Schedule found with 5 iterations in parallel
;*      done
;*
;*      Loop is interruptible
```

```
;*      Collapsed epilog stages     : 4
;*      Prolog not entirely removed
;*      Collapsed prolog stages     : 3
;*
;*      Minimum required memory pad : 32 bytes
;*
;*      Minimum safe trip count     : 1
;*--------------------------------------------------------------------------*
L1:    ; PIPED LOOP PROLOG
;** --------------------------------------------------------------------------*
L2:    ; PIPED LOOP KERNEL

   [ B0]   B       .S2     L2              ; |75|
||         MV      .S1     A15,A11         ; @|72|
||         MV      .D2     B5,B4           ; @|71|
||         INTSP   .L1     A5,A15          ; @@|72|
||         INTSP   .L2X    A2,B2           ; @@|69|
||         LDH     .D1T1   *+A0(2),A2      ; @@@|68|

   [!A1]   STW     .D2T2   B7,*B3          ; |69|
||         MPYSP   .M1     A9,A11,A11      ; @|74|
||         MPYSP   .M2     B10,B4,B13      ; @|74|
||         INTSP   .L1X    B9,A14          ; @@|73|
||         MV      .S1     A12,A12         ; @@|67|
||         INTSP   .L2X    A5,B4           ; @@|73|
||         LDH     .D1T1   *+A0(6),A5      ; @@@|72|

   [!A1]   STW     .D2T1   A13,*B3         ; |70|
||         ADDSP   .L1     A3,A7,A13       ; @|70|
||         MPYSP   .M1X    B10,A12,A7      ; @@|70|
||         INTSP   .L2     B9,B5           ; @@|71|
||         LDH     .D1T2   *+A0(4),B9      ; @@@|71|

   [!A1]   STW     .D2T1   A6,*B3          ; |73|
||         SUBSP   .L2     B12,B11,B7      ; @|69|
||         MV      .S1     A4,A3           ; @@|68|
||         MPYSP   .M2X    A9,B8,B12       ; @@|69|
||         INTSP   .L1     A8,A12          ; @@@|67|

            SUBSP   .L1X    A10,B1,A6       ; @|73|
||         MPYSP   .M2     B10,B2,B11      ; @@|69|
||         MPYSP   .M1     A9,A3,A3        ; @@|70|
||         INTSP   .L2X    A8,B8           ; @@@|69|
||         LDH     .D1T1   *++A0(8),A8     ; @@@@|67|

   [ A1]   SUB     .S1     A1,1,A1         ;
|| [!A1]   STW     .D2T2   B6,*B3          ; |74|
|| [ B0]   SUB     .S2     B0,1,B0         ; @|75|
||         ADDSP   .L2X    A11,B13,B6      ; @|74|
||         MPYSP   .M2     B10,B4,B1       ; @@|73|
||         MPYSP   .M1     A9,A14,A10      ; @@|73|
||         INTSP   .L1     A2,A4           ; @@@|68|

;** --------------------------------------------------------------------------*
L3:    ; PIPED LOOP EPILOG
;** --------------------------------------------------------------------------*
L4:
            LDW     .D2T2   *+SP(28),B3     ; |76|
            LDW     .D2T2   *+SP(40),B12    ; |76|
            LDW     .D2T2   *+SP(36),B11    ; |76|
            LDW     .D2T2   *+SP(32),B10    ; |76|
            LDW     .D2T1   *+SP(24),A14    ; |76|
            LDW     .D2T1   *+SP(20),A13    ; |76|
            LDW     .D2T1   *+SP(16),A12    ; |76|
            LDW     .D2T1   *+SP(12),A11    ; |76|
            LDW     .D2T1   *+SP(8),A10     ; |76|

            B       .S2     B3              ; |76|
||          LDW     .D2T1   *+SP(44),A15    ; |76|

            LDW     .D2T2   *++SP(48),B13   ; |76|
            NOP             4
            ; BRANCH OCCURS                 ; |76|
        .endfunc        76,03c08fc00h,48
```

rawdatasend.c

```
/*Send raw data between processors without using DMA*/

void RawDataSend (volatile unsigned int* outInterprocBifo, int* IOdata, int
    BufferSize)
{
    unsigned int i;
    for (i=0;i<BufferSize;i++) {
                *outInterprocBifo=IOdata[i];
                *outInterprocBifo=IOdata[i+1];
                *outInterprocBifo=IOdata[i+2];
                *outInterprocBifo=IOdata[i+3];
                *outInterprocBifo=IOdata[i+4];
                *outInterprocBifo=IOdata[i+5];
                *outInterprocBifo=IOdata[i+6];
                *outInterprocBifo=IOdata[i+7];
        }
}
```

rawdatasend_opt.asm

```
;******************************************************************************
;* TMS320C6x ANSI C Codegen                                       Version 4.00 *
;* Date/Time created: Sat Aug 25 08:59:11 2001                                 *
;******************************************************************************

;******************************************************************************
;* GLOBAL FILE PARAMETERS                                                      *
;*                                                                             *
;*   Architecture      : TMS320C670x                                           *
;*   Optimization      : Enabled at level 3                                    *
;*   Optimizing for    : Speed                                                 *
;*                       Based on options: -o3, no -ms                         *
;*   Endian            : Big                                                   *
;*   Interrupt Thrshld : Disabled                                              *
;*   Memory Model      : Large                                                 *
;*   Calls to RTS      : Far                                                   *
;*   Pipelining        : Enabled                                               *
;*   Speculative Load  : Enabled (Threshold = 64                              ) *
;*   Memory Aliases    : Presume are aliases (pessimistic)                     *
;*   Debug Info        : Debug                                                 *
;*                                                                             *
;******************************************************************************

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 C:\TEMP\TI169_2 C:\TEMP\TI169_4 -w C:\ti\myprojects\
    astronomy
        .file   "rawdatasend.c"
        .sect   ".text"
        .global _RawDataSend
        .sym    _RawDataSend,_RawDataSend, 32, 2, 0
        .func   3

;******************************************************************************
;* FUNCTION NAME: _RawDataSend                                                 *
;*                                                                             *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,B0,B1,B2,B4,B5,B6, *
;*                           B7,B8,B9,SP                                       *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,B0,B1,B2,B3,B4,B5, *
;*                           B6,B7,B8,B9,SP                                    *
;*   Local Frame Size  : 0 Args + 0 Auto + 4 Save = 4 byte                     *
;******************************************************************************
_RawDataSend:
;** --------------------------------------------------------------------------*
        .sym    _outInterprocBifo,4, 30, 17, 32
        .sym    _IOdata,20, 20, 17, 32
        .sym    _BufferSize,6, 4, 17, 32
        .sym    _outInterprocBifo,4, 30, 4, 32
```

```
        .sym    _IOdata,20, 20, 4, 32
        .sym    _BufferSize,16, 4, 4, 32
        .sym    L$1,20, 4, 4, 32
        .sym    U$9,8, 20, 4, 32
           STW      .D2T1   A10,*SP--(8)       ; |3|
           MV       .L2X    A6,B0              ;
   [!B0]   B        .S1     L4                 ; |6|
           MVK      .S1     32,A0              ; |6|
           SUB      .L1X    B4,A0,A8           ; |6|
           NOP              3
           ; BRANCH OCCURS                     ; |6|
;** --------------------------------------------------------------------------*
           ADD      .L2     7,B0,B4            ; |7|

           ADD      .L2X    4,A8,B8            ;
||         SHRU     .S2     B4,3,B4            ; |7|

           MV       .D1     A4,A9
||         MV       .L2X    A4,B9
||         MVK      .S2     0x1,B2             ; init prolog collapse predicate
||         MVK      .S1     0x2,A2             ; init prolog collapse predicate
||         ADD      .L1X    2,B4,A1            ;

;*--------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Known Minimum Trip Count         : 1
;*      Known Maximum Trip Count         : 536870911
;*      Known Max Trip Count Factor      : 1
;*      Loop Carried Dependency Bound(^) : 8
;*      Unpartitioned Resource Bound     : 8
;*      Partitioned Resource Bound(*)    : 8
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                     0        0
;*      .S units                     1        0
;*      .D units                     8*       8*
;*      .M units                     0        0
;*      .X cross paths               0        2
;*      .T address paths             8*       8*
;*      Long read paths              3        5
;*      Long write paths             0        0
;*      Logical  ops (.LS)           0        2     (.L or .S unit)
;*      Addition ops (.LSD)          1        0     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             1        1
;*      Bound(.L .S .D .LS .LSD)     4        4
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 8  Schedule found with 3 iterations in parallel
;*      done
;*
;*      Loop is interruptible
;*      Collapsed epilog stages     : 2
;*      Collapsed prolog stages     : 2
;*      Minimum required memory pad : 64 bytes
;*
;*      Minimum safe trip count     : 1
;*--------------------------------------------------------------------------*
L1:    ; PIPED LOOP PROLOG
;** --------------------------------------------------------------------------*
L2:    ; PIPED LOOP KERNEL

   [!A2]   LDW      .D1T1   *+A8(28),A5        ; |14|
|| [!A2]   STW      .D2T2   B7,*B9             ;  ^ |7|

   [ A1]   SUB      .L1     A1,1,A1            ; |15|
|| [!A2]   STW      .D2T2   B5,*B9             ;  ^ |8|
|| [!B2]   LDW      .D1T1   *++A8(32),A7       ; @|7|

   [ A1]   B        .S1     L2                 ; |15|
|| [!A2]   STW      .D2T2   B4,*B9             ;  ^ |9|
|| [!B2]   LDW      .D1T1   *+A8(16),A3        ; @|11|

           MV       .L1     A0,A6              ; Inserted to split a long life
|| [!A2]   STW      .D2T2   B0,*B9             ;  ^ |10|
```

```
|| [!B2]   LDW     .D1T1   *+A8(20),A0       ; @|12|

           MV      .L1     A4,A10            ; Inserted to split a long life
|| [!A2]   STW     .D2T2   B6,*B9            ;  ^ |11|
|| [!B2]   LDW     .D1T1   *+A8(24),A4       ; @|13|

   [!A2]   STW     .D1T1   A6,*A9            ;  ^ |12|
|| [!B2]   LDW     .D2T2   *+B8(4),B4        ; @|9|

   [ B2]   SUB     .S2     B2,1,B2           ;
|| [!A2]   STW     .D1T1   A10,*A9           ;  ^ |13|
|| [!B2]   LDW     .D2T2   *+B8(8),B0        ; @|10|
||         MV      .L2X    A7,B7             ; @Define a twin register

   [ A2]   SUB     .L1     A2,1,A2           ;
|| [!A2]   STW     .D1T1   A5,*A9            ;  ^ |14|
||         MV      .S2     B1,B5             ; @Inserted to split a long life
||         MV      .L2X    A3,B6             ; @Define a twin register
||         LDW     .D2T2   *++B8(32),B1      ; @@|8|

;** --------------------------------------------------------------------------*
L3:    ; PIPED LOOP EPILOG
;** --------------------------------------------------------------------------*
L4:
           B       .S2     B3                ; |16|
           LDW     .D2T1   *++SP(8),A10      ; |16|
           NOP             4
           ; BRANCH OCCURS                   ; |16|
        .endfunc        16,000000400h,8
```

compute_matrix_corner.c

```
/*Compute the three complex valued corner entries of the correlation matrix*/
/*Note, TwoBufferLength should be two times the length of BUFFER_SIZE*/

void compute_matrix_corner (float* matrix, short* rdA, short* rdB, short* rdC, int
    TwoBufferLength) {
        int i;
        long temp3=0,temp4=0,temp5=0,temp6=0,temp7=0,temp8=0;
        for (i=0;i<TwoBufferLength;i+=2) {
        temp3 = temp3 + (long)(rdA[i]*rdB[i] + rdA[i+1]*rdB[i+1]);
        temp4 = temp4 + (long)(rdA[i+1]*rdB[i] - rdA[i]*rdB[i+1]);
        temp5 = temp5 + (long)(rdB[i]*rdC[i] + rdB[i+1]*rdC[i+1]);
        temp6 = temp6 + (long)(rdB[i+1]*rdC[i] - rdB[i]*rdC[i+1]);
        temp7 = temp7 + (long)(rdA[i]*rdC[i] + rdA[i+1]*rdC[i+1]);
        temp8 = temp8 + (long)(rdA[i+1]*rdC[i] - rdA[i]*rdC[i+1]);
        }
        matrix[3] = (float)temp3;
    matrix[4] = (float)temp4;
    matrix[5] = (float)temp5;
    matrix[6] = (float)temp6;
    matrix[7] = (float)temp7;
    matrix[8] = (float)temp8;
}
```

compute_matrix_corner_opt.asm

```
;******************************************************************************
;* TMS320C6x ANSI C Codegen                                     Version 4.00 *
;* Date/Time created: Wed Aug 08 22:06:37 2001                               *
;******************************************************************************

;******************************************************************************
;* GLOBAL FILE PARAMETERS                                                     *
;*                                                                            *
;*   Architecture      : TMS320C670x                                          *
;*   Optimization      : Enabled at level 3                                   *
;*   Optimizing for    : Speed                                                *
;*                       Based on options: -o3, no -ms                        *
;*   Endian            : Big                                                  *
;*   Interrupt Thrshld : Disabled                                             *
;*   Memory Model      : Large                                                *
```

```
;*   Calls to RTS      : Far                                                  *
;*   Pipelining        : Enabled                                              *
;*   Speculative Load  : Enabled (Threshold = 8                              ) *
;*   Memory Aliases    : Presume are aliases (pessimistic)                    *
;*   Debug Info        : Debug                                                *
;*                                                                            *
;******************************************************************************

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 C:\TEMP\TI203_2 C:\TEMP\TI203_4 -w C:\ti\myprojects\
    correlation
        .file   "compute_matrix_corner.c"
        .sect   ".text"
        .global _compute_matrix_corner
        .sym    _compute_matrix_corner,_compute_matrix_corner, 32, 2, 0
        .func   3

;******************************************************************************
;* FUNCTION NAME: _compute_matrix_corner                                      *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Local Frame Size  : 0 Args + 16 Auto + 44 Save = 60 byte                 *
;******************************************************************************
_compute_matrix_corner:
;** --------------------------------------------------------------------------*
        .sym    _matrix,4, 22, 17, 32
        .sym    _rdA,20, 19, 17, 32
        .sym    _rdB,6, 19, 17, 32
        .sym    _rdC,22, 19, 17, 32
        .sym    _TwoBufferLength,8, 4, 17, 32
        .sym    C$1,0, 3, 4, 16
        .sym    C$2,0, 3, 4, 16
        .sym    C$3,0, 3, 4, 16
        .sym    C$4,0, 3, 4, 16
        .sym    C$5,0, 3, 4, 16
        .sym    C$6,0, 3, 4, 16
        .sym    _temp8,10, 5, 4, 40
        .sym    _temp7,14, 5, 4, 40
        .sym    _temp6,26, 5, 4, 40
        .sym    _temp5,28, 5, 4, 40
        .sym    _temp4,12, 5, 1, 40
        .sym    _temp3,24, 5, 4, 40
        .sym    _matrix,4, 22, 1, 32
        .sym    _rdA,20, 19, 4, 32
        .sym    _rdB,16, 19, 4, 32
        .sym    _rdC,22, 19, 4, 32
        .sym    _TwoBufferLength,8, 4, 4, 32
        .sym    L$1,1, 4, 4, 32
        .sym    U$35,22, 19, 4, 32
        .sym    U$19,20, 19, 4, 32
        .sym    U$17,16, 19, 4, 32
           STW      .D2T2   B13,*SP--(64)      ; |3|
           STW      .D2T1   A10,*+SP(24)       ; |3|
           STW      .D2T1   A11,*+SP(28)       ; |3|
           STW      .D2T1   A12,*+SP(32)       ; |3|
           STW      .D2T1   A13,*+SP(36)       ; |3|
           STW      .D2T1   A14,*+SP(40)       ; |3|
           STW      .D2T2   B3,*+SP(44)        ; |3|
           STW      .D2T2   B10,*+SP(48)       ; |3|
           STW      .D2T2   B11,*+SP(52)       ; |3|
           STW      .D2T2   B12,*+SP(56)       ; |3|
           STW      .D2T1   A15,*+SP(60)       ; |3|

           MV       .L2X    A6,B0              ;
||         STW      .D2T1   A4,*+SP(4)
```

```
           ZERO    .L2     B9:B8             ; |5|
           MV      .L1X    B8,A10            ;

           MV      .L1X    B9,A15            ;
||         STW     .D2T2   B9,*+SP(16)

           MV      .L1X    B9,A11            ;
||         MV      .L2     B9,B11            ;
||         STW     .D2T2   B8,*+SP(12)

           MV      .L2     B9,B13            ;
||         MV      .S2     B8,B12            ;
||         MV      .D2     B8,B10            ;
||         MV      .L1X    B8,A14            ;

           CMPGT   .L1     A8,0,A1           ; |6|
   [!A1]   B       .S1     L4                ; |6|
           NOP             5
           ; BRANCH OCCURS                   ; |6|
;** --------------------------------------------------------------------------*
           NOP             1

           MV      .L1X    B11,A5
||         MVC     .S2     CSR,B11

           MV      .L1X    B9,A7
||         LDW     .D2T2   *+SP(16),B9
||         AND     .L2     -2,B11,B2

           MV      .L1X    B8,A6
||         LDW     .D2T2   *+SP(12),B8
||         MVC     .S2     B2,CSR            ; interrupts off

           MV      .L1X    B10,A4
||         LDH     .D2T2   *+B0(2),B10       ; (P) |7|

           MV      .D1     A11,A9
||         MVK     .S1     0x2,A2            ; init prolog collapse predicate
||         MV      .L2     B4,B1
||         MV      .L1X    B6,A13

;*----------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Known Minimum Trip Count         : 1
;*      Known Maximum Trip Count         : 1073741823
;*      Known Max Trip Count Factor      : 1
;*      Loop Carried Dependency Bound(^) : 2
;*      Unpartitioned Resource Bound     : 6
;*      Partitioned Resource Bound(*)    : 6
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                     3        3
;*      .S units                     1        0
;*      .D units                     2        4
;*      .M units                     6*       6*
;*      .X cross paths               6*       4
;*      .T address paths             2        4
;*      Long read paths              3        3
;*      Long write paths             3        3
;*      Logical  ops (.LS)           2        0     (.L or .S unit)
;*      Addition ops (.LSD)          4        3     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             3        2
;*      Bound(.L .S .D .LS .LSD)     4        4
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 6  Schedule found with 3 iterations in parallel
;*      done
;*
;*      Collapsed epilog stages     : 2
;*      Prolog not entirely removed
;*      Collapsed prolog stages     : 1
;*
;*      Minimum required memory pad : 8 bytes
;*
```

```
;*      Minimum safe trip count     : 1
;*----------------------------------------------------------------------------*
L1:    ; PIPED LOOP PROLOG

           MV       .S2     B13,B5
||         MV       .L2X    A15,B7
||         ADD      .L1     1,A8,A0           ; |7|
||         MV       .S1     A10,A8
||         LDH      .D1T1   *+A13(2),A10      ; (P) |9|
||         LDH      .D2T2   *+B1(2),B3        ; (P) |7|

           MV       .S2     B12,B4
||         MV       .L2X    A14,B6
||         MV       .L1X    B11,A15           ;
||         SHR      .S1     A0,1,A1           ; |7|
||         LDH      .D2T2   *B0++(4),B11      ; (P) |7|
||         LDH      .D1T1   *A13++(4),A14     ; (P) |9|

;** --------------------------------------------------------------------------*
L2:    ; PIPED LOOP KERNEL

   [ A2]   SUB      .D1     A2,1,A2           ;
||         MPY      .M1     A0,A14,A11        ; |10|
|| [ A1]   B        .S1     L2                ; |13|
||         MV       .L1X    B11,A0            ; Define a twin register
||         MPY      .M2     B3,B11,B13        ; |8|
||         LDH      .D2T2   *B1++(4),B13      ; @|7|

           SUB      .L1     A11,A12,A12       ; |12|
||         MPY      .M1X    B13,A0,A14        ; |7|
||         MPY      .M2X    B13,A14,B13       ; |11|

   [!A2]   ADD      .L1     A12,A9:A8,A9:A8   ; |12|
||         MPY      .M1     A0,A10,A12        ; |10|
||         MPY      .M2X    B11,A14,B12       ; |9|
||         SUB      .L2     B13,B3,B11        ; |8|
||         MV       .S1X    B10,A0            ; @Define a twin register
||         LDH      .D2T2   *+B0(2),B10       ; @@|7|

           ADD      .L1     A3,A14,A3         ; |7|
|| [!A2]   ADD      .L2     B11,B9:B8,B9:B8   ; |8|
||         ADD      .S2     B12,B13,B13       ; |11|
||         MPY      .M2X    B10,A10,B2        ; @|9|
||         MPY      .M1X    B3,A0,A3          ; @|7|

   [!A2]   ADD      .L1     A3,A7:A6,A7:A6    ; |7|
||         SUB      .S1     A11,A12,A11       ; |10|
||         ADD      .S2     B2,B12,B12        ; |9|
|| [!A2]   ADD      .L2     B13,B7:B6,B7:B6   ; |11|
||         MPY      .M2X    B3,A10,B12        ; @|11|
||         MPY      .M1X    B3,A14,A11        ; @|12|
||         LDH      .D2T2   *+B1(2),B3        ; @@|7|
||         LDH      .D1T1   *+A13(2),A10      ; @@|9|

   [!A2]   ADD      .L1     A11,A5:A4,A5:A4   ; |10|
|| [!A2]   ADD      .L2     B12,B5:B4,B5:B4   ; |9|
|| [ A1]   SUB      .S1     A1,1,A1           ; @|13|
||         MPY      .M1X    B13,A10,A12       ; @|12|
||         MPY      .M2     B13,B10,B3        ; @|8|
||         LDH      .D2T2   *B0++(4),B11      ; @@|7|
||         LDH      .D1T1   *A13++(4),A14     ; @@|9|

;** --------------------------------------------------------------------------*
L3:    ; PIPED LOOP EPILOG
;** --------------------------------------------------------------------------*

           MV       .S2     B5,B13
||         MV       .S1     A9,A11
||         MV       .L1X    B7,A15
||         STW      .D2T2   B8,*+SP(12)
||         MV       .L2X    A15,B0

           MV       .S2     B4,B12
||         MV       .L2X    A7,B9
||         STW      .D2T2   B9,*+SP(16)
```

```
           MV       .L2X    A4,B10

           MV       .L2X    A6,B8
||         MV       .L1     A8,A10
||         MVC      .S2     B0,CSR          ; interrupts on

           MV       .L2X    A5,B11
||         MV       .L1X    B6,A14

;** --------------------------------------------------------------------------*
L4:
           MVKL     .S2     __fltlif,B4     ; |14|
           MVKH     .S2     __fltlif,B4     ; |14|
           B        .S2     B4              ; |14|
           MV       .L1X    B8,A4           ; |14|
           MVKL     .S2     RL0,B3          ; |14|
           MVKH     .S2     RL0,B3          ; |14|
           MV       .L1X    B9,A5           ; |14|
           NOP              1
RL0:       ; CALL OCCURS                    ; |14|
           LDW      .D2T1   *+SP(4),A0      ; |14|
           NOP              4
           STW      .D1T1   A4,*+A0(12)     ; |14|
           MVKL     .S1     __fltlif,A0     ; |15|
           MVKH     .S1     __fltlif,A0     ; |15|

           B        .S2X    A0              ; |15|
||         LDW      .D2T1   *+SP(16),A5     ; |15|

           LDW      .D2T1   *+SP(12),A4     ; |15|
           MVKL     .S2     RL1,B3          ; |15|
           MVKH     .S2     RL1,B3          ; |15|
           NOP              2
RL1:       ; CALL OCCURS                    ; |15|
           LDW      .D2T1   *+SP(4),A0      ; |15|
           NOP              4
           STW      .D1T1   A4,*+A0(16)     ; |15|
           MVKL     .S1     __fltlif,A0     ; |16|
           MVKH     .S1     __fltlif,A0     ; |16|
           B        .S2X    A0              ; |16|
           MV       .L1X    B13,A5          ; |16|
           MVKL     .S2     RL2,B3          ; |16|
           MV       .L1X    B12,A4          ; |16|
           MVKH     .S2     RL2,B3          ; |16|
           NOP              1
RL2:       ; CALL OCCURS                    ; |16|
           LDW      .D2T1   *+SP(4),A0      ; |16|
           NOP              4
           STW      .D1T1   A4,*+A0(20)     ; |16|
           MVKL     .S2     __fltlif,B4     ; |17|
           MVKH     .S2     __fltlif,B4     ; |17|
           B        .S2     B4              ; |17|
           MV       .L1X    B10,A4          ; |17|
           MVKL     .S2     RL3,B3          ; |17|
           MV       .L1X    B11,A5          ; |17|
           MVKH     .S2     RL3,B3          ; |17|
           NOP              1
RL3:       ; CALL OCCURS                    ; |17|
           LDW      .D2T1   *+SP(4),A0      ; |17|
           NOP              4
           STW      .D1T1   A4,*+A0(24)     ; |17|
           MVKL     .S1     __fltlif,A0     ; |18|
           MVKH     .S1     __fltlif,A0     ; |18|
           B        .S2X    A0              ; |18|
           MVKL     .S2     RL4,B3          ; |18|
           MVKH     .S2     RL4,B3          ; |18|
           MV       .L1     A15,A5          ; |18|
           MV       .S1     A14,A4          ; |18|
           NOP              1
RL4:       ; CALL OCCURS                    ; |18|
           LDW      .D2T1   *+SP(4),A0      ; |18|
           NOP              4
           STW      .D1T1   A4,*+A0(28)     ; |18|
           MVKL     .S2     __fltlif,B4     ; |19|
```

```
           MVKH    .S2     __fltlif,B4       ; |19|
           B       .S2     B4                ; |19|
           MVKL    .S2     RL5,B3            ; |19|
           MV      .L1     A10,A4            ; |19|
           MVKH    .S2     RL5,B3            ; |19|
           MV      .S1     A11,A5            ; |19|
           NOP             1
RL5:       ; CALL OCCURS                     ; |19|
           LDW     .D2T1   *+SP(4),A0        ; |19|
           NOP             4
           STW     .D1T1   A4,*+A0(32)       ; |19|
           LDW     .D2T1   *+SP(40),A14      ; |20|
           LDW     .D2T1   *+SP(24),A10      ; |20|
           LDW     .D2T1   *+SP(60),A15      ; |20|
           LDW     .D2T2   *+SP(44),B3       ; |20|
           LDW     .D2T1   *+SP(36),A13      ; |20|
           LDW     .D2T1   *+SP(32),A12      ; |20|
           LDW     .D2T1   *+SP(28),A11      ; |20|
           LDW     .D2T2   *+SP(48),B10      ; |20|
           LDW     .D2T2   *+SP(52),B11      ; |20|

           B       .S2     B3                ; |20|
||         LDW     .D2T2   *+SP(56),B12      ; |20|

           LDW     .D2T2   *++SP(64),B13     ; |20|
           NOP             4
           ; BRANCH OCCURS                   ; |20|
        .endfunc        20,03c08fc00h,64


;*****************************************************************************
;* UNDEFINED EXTERNAL REFERENCES                                             *
;*****************************************************************************
        .global __fltlif
```

compute_matrix_diagonal.c

```
/*Compute the three real valued diagonal entries of the correlation matrix*/
/*Note, TwoBufferLength should be two times the length of BUFFER_SIZE*/

void compute_matrix_diagonal(float* matrix, short* rdA, short* rdB, short* rdC, int
    TwoBufferLength) {
        int i;
        long temp0=0,temp1=0,temp2=0;
        for (i=0;i<TwoBufferLength;i+=2) {
                temp0 = temp0 + (long)(rdA[i]*rdA[i] + rdA[i+1]*rdA[i+1]);
                temp1 = temp1 + (long)(rdB[i]*rdB[i] + rdB[i+1]*rdB[i+1]);
                temp2 = temp2 + (long)(rdC[i]*rdC[i] + rdC[i+1]*rdC[i+1]);

        }
        matrix[0] = (float)temp0;
        matrix[1] = (float)temp1;
        matrix[2] = (float)temp2;
}
```

compute_matrix_diagonal_opt.asm

```
;*****************************************************************************
;* TMS320C6x ANSI C Codegen                                    Version 4.00 *
;* Date/Time created: Wed Aug 08 22:19:20 2001                               *
;*****************************************************************************

;*****************************************************************************
;* GLOBAL FILE PARAMETERS                                                    *
;*                                                                           *
;*   Architecture      : TMS320C670x                                         *
;*   Optimization      : Enabled at level 3                                  *
;*   Optimizing for    : Speed                                               *
;*                       Based on options: -o3, no -ms                       *
;*   Endian            : Big                                                 *
;*   Interrupt Thrshld : Disabled                                            *
;*   Memory Model      : Large                                               *
```

```
;*   Calls to RTS      : Far                                                *
;*   Pipelining        : Enabled                                            *
;*   Speculative Load  : Enabled (Threshold = 12                           ) *
;*   Memory Aliases    : Presume are aliases (pessimistic)                  *
;*   Debug Info        : Debug                                              *
;*                                                                          *
;****************************************************************************

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 C:\TEMP\TI199_2 C:\TEMP\TI199_4 -w C:\ti\myprojects\
    correlation
        .file   "compute_matrix_diagonal.c"
        .sect   ".text"
        .global _compute_matrix_diagonal
        .sym    _compute_matrix_diagonal,_compute_matrix_diagonal, 32, 2, 0
        .func   4

;****************************************************************************
;* FUNCTION NAME: _compute_matrix_diagonal                                  *
;*                                                                          *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,B0,*
;*                         B1,B2,B3,B4,B5,B6,B7,B8,B9,SP                    *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,B0,*
;*                         B1,B2,B3,B4,B5,B6,B7,B8,B9,SP                    *
;*   Local Frame Size  : 0 Args + 0 Auto + 24 Save = 24 byte                *
;****************************************************************************
_compute_matrix_diagonal:
;** --------------------------------------------------------------------------*
        .sym    _matrix,4, 22, 17, 32
        .sym    _rdA,20, 19, 17, 32
        .sym    _rdB,6, 19, 17, 32
        .sym    _rdC,22, 19, 17, 32
        .sym    _TwoBufferLength,8, 4, 17, 32
        .sym    C$1,0, 3, 4, 16
        .sym    C$2,0, 3, 4, 16
        .sym    C$3,0, 3, 4, 16
        .sym    C$4,0, 3, 4, 16
        .sym    C$5,0, 3, 4, 16
        .sym    C$6,0, 3, 4, 16
        .sym    _temp2,10, 5, 4, 40
        .sym    _temp1,12, 5, 4, 40
        .sym    _temp0,24, 5, 4, 40
        .sym    _matrix,14, 22, 4, 32
        .sym    _rdA,9, 19, 4, 32
        .sym    _rdB,6, 19, 4, 32
        .sym    _rdC,22, 19, 4, 32
        .sym    _TwoBufferLength,8, 4, 4, 32
        .sym    L$1,0, 4, 4, 32
        .sym    U$32,3, 19, 4, 32
        .sym    U$23,6, 19, 4, 32
        .sym    U$14,9, 19, 4, 32
           STW     .D2T2   B3,*SP--(24)       ; |4|
           STW     .D2T1   A10,*+SP(4)        ; |4|
           STW     .D2T1   A11,*+SP(8)        ; |4|
           STW     .D2T1   A12,*+SP(12)       ; |4|
           STW     .D2T1   A13,*+SP(16)       ; |4|
           STW     .D2T1   A14,*+SP(20)       ; |4|

           MV      .L1X    B4,A9              ;
||         MV      .S1     A4,A14             ;

           ZERO    .L2     B9:B8              ; |6|
           MV      .L1X    B8,A12             ;
           MV      .L1X    B8,A10             ;
           MV      .L1X    B9,A13             ;
           MV      .L1X    B9,A11             ;
           CMPGT   .L1     A8,0,A1            ; |7|
   [!A1]   B       .S1     L4                 ; |7|
           MV      .L1X    B6,A3              ; |7|
           NOP             4
           ; BRANCH OCCURS                    ; |7|
```

```
;** --------------------------------------------------------------------------*
           MVC     .S2     CSR,B2

           SUB     .L2X    A9,2,B7           ;
||         AND     .S2     -2,B2,B4

           MV      .L2X    A6,B6
||         MV      .L1     A3,A6
||         LDH     .D1T1   *A9++(4),A0       ; (P) |8|
||         LDH     .D2T2   *++B7(4),B4       ; (P) |8|
||         MVC     .S2     B4,CSR            ; interrupts off

;*--------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Known Minimum Trip Count         : 1
;*      Known Maximum Trip Count         : 1073741823
;*      Known Max Trip Count Factor      : 1
;*      Loop Carried Dependency Bound(^) : 0
;*      Unpartitioned Resource Bound     : 3
;*      Partitioned Resource Bound(*)    : 3
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                     2        1
;*      .S units                     0        1
;*      .D units                     3*       3*
;*      .M units                     3*       3*
;*      .X cross paths               2        1
;*      .T address paths             3*       3*
;*      Long read paths              2        1
;*      Long write paths             2        1
;*      Logical  ops (.LS)           2        1     (.L or .S unit)
;*      Addition ops (.LSD)          0        1     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             2        2
;*      Bound(.L .S .D .LS .LSD)     3*       3*
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 3  Schedule found with 4 iterations in parallel
;*      done
;*
;*      Collapsed epilog stages     : 3
;*      Prolog not removed
;*      Collapsed prolog stages     : 0
;*
;*      Minimum required memory pad : 12 bytes
;*
;*      Minimum safe trip count     : 1
;*--------------------------------------------------------------------------*
L1:    ; PIPED LOOP PROLOG

           LDH     .D1T2   *+A6(2),B4        ; (P) |10|
||         LDH     .D2T1   *+B6(2),A0        ; (P) |9|

           ADD     .L1     1,A8,A3           ; |8|
||         LDH     .D2T2   *B6++(4),B8       ; (P) |9|
||         LDH     .D1T1   *A6++(4),A7       ; (P) |10|

           MV      .L1X    B8,A4
||         SHR     .S1     A3,1,A3           ; |8|
||         LDH     .D1T1   *A9++(4),A8       ; (P) @|8|
||         LDH     .D2T2   *++B7(4),B8       ; (P) @|8|

           SUB     .L2X    A3,1,B0           ;
||         LDH     .D2T1   *+B6(2),A8        ; (P) @|9|
||         LDH     .D1T2   *+A6(2),B1        ; (P) @|10|

           LDH     .D1T1   *A6++(4),A3       ; (P) @|10|
||         MPY     .M2     B4,B4,B4          ; (P) |8|
||         LDH     .D2T2   *B6++(4),B8       ; (P) @|9|
|| [ B0]   B       .S2     L2                ; (P) |12|
||         MPY     .M1     A0,A0,A3          ; (P) |8|

           MV      .S1     A12,A0
||         MV      .L2X    A11,B5
||         MV      .L1X    B9,A5
```

```
||          LDH     .D1T1   *A9++(4),A8        ; (P) @@|8|
||          MPY     .M1     A0,A0,A3           ; (P) |9|
||          LDH     .D2T2   *++B7(4),B8        ; (P) @@|8|
||          MPY     .M2     B4,B4,B9           ; (P) |10|

            MV      .S1     A13,A1
||          MV      .L2X    A10,B4
|| [ B0]    SUB     .S2     B0,1,B0            ; (P) @|12|
||          ADD     .L1X    B4,A3,A7           ; (P) |8|
||          LDH     .D1T2   *+A6(2),B1         ; (P) @@|10|
||          MPY     .M2     B8,B8,B8           ; (P) |9|
||          LDH     .D2T1   *+B6(2),A8         ; (P) @@|9|
||          MPY     .M1     A7,A7,A7           ; (P) |10|

;** --------------------------------------------------------------------------*
L2:     ; PIPED LOOP KERNEL

            ADD     .L1     A7,A5:A4,A5:A4     ; |8|
||          MPY     .M1     A8,A8,A7           ; @|8|
||          MPY     .M2     B8,B8,B9           ; @|8|
|| [ B0]    B       .S2     L2                 ; @|12|
||          LDH     .D2T2   *B6++(4),B8        ; @@|9|
||          LDH     .D1T1   *A6++(4),A3        ; @@|10|

            ADD     .L1X    A3,B8,A8           ; |9|
||          ADD     .L2X    B9,A7,B1           ; |10|
||          MPY     .M1     A8,A8,A3           ; @|9|
||          MPY     .M2     B1,B1,B9           ; @|10|
||          LDH     .D2T2   *++B7(4),B8        ; @@@|8|
||          LDH     .D1T1   *A9++(4),A8        ; @@@|8|

            ADD     .L1     A8,A1:A0,A1:A0     ; |9|
||          ADD     .L2     B1,B5:B4,B5:B4     ; |10|
||          ADD     .S1X    B9,A7,A7           ; @|8|
||          MPY     .M2     B8,B8,B8           ; @|9|
||          MPY     .M1     A3,A3,A7           ; @|10|
|| [ B0]    SUB     .S2     B0,1,B0            ; @@|12|
||          LDH     .D2T1   *+B6(2),A8         ; @@@|9|
||          LDH     .D1T2   *+A6(2),B1         ; @@@|10|

;** --------------------------------------------------------------------------*
L3:     ; PIPED LOOP EPILOG
;** --------------------------------------------------------------------------*

            MV      .S1     A0,A12
||          MV      .D1     A1,A13
||          MV      .L2X    A5,B9
||          MV      .L1X    B4,A10

            MV      .L1X    B5,A11
            NOP             1

            MV      .L2X    A4,B8
||          MVC     .S2     B2,CSR             ; interrupts on

;** --------------------------------------------------------------------------*
L4:
            MVKL    .S2     __fltlif,B4        ; |13|
            MVKH    .S2     __fltlif,B4        ; |13|
            B       .S2     B4                 ; |13|
            MV      .L1X    B8,A4              ; |13|
            MVKL    .S2     RL0,B3             ; |13|
            MVKH    .S2     RL0,B3             ; |13|
            MV      .L1X    B9,A5              ; |13|
            NOP             1
RL0:        ; CALL OCCURS                      ; |13|
            STW     .D1T1   A4,*A14            ; |13|
            MVKL    .S1     __fltlif,A0        ; |14|
            MVKH    .S1     __fltlif,A0        ; |14|
            B       .S2X    A0                 ; |14|
            MVKL    .S2     RL1,B3             ; |14|
            MV      .L1     A12,A4             ; |14|
            MV      .S1     A13,A5             ; |14|
            MVKH    .S2     RL1,B3             ; |14|
            NOP             1
```

```
RL1:        ; CALL OCCURS                       ; |14|
            STW     .D1T1   A4,*+A14(4)         ; |14|
            MVKL    .S2     __fltlif,B4         ; |15|
            MVKH    .S2     __fltlif,B4         ; |15|
            B       .S2     B4                  ; |15|
            MVKL    .S2     RL2,B3              ; |15|
            MV      .L1     A10,A4              ; |15|
            MV      .S1     A11,A5              ; |15|
            MVKH    .S2     RL2,B3              ; |15|
            NOP             1
RL2:        ; CALL OCCURS                       ; |15|
            STW     .D1T1   A4,*+A14(8)         ; |15|
            LDW     .D2T1   *+SP(16),A13        ; |16|
            LDW     .D2T1   *+SP(12),A12        ; |16|
            LDW     .D2T1   *+SP(8),A11         ; |16|
            LDW     .D2T1   *+SP(4),A10         ; |16|
            LDW     .D2T1   *+SP(20),A14        ; |16|
            LDW     .D2T2   *++SP(24),B3        ; |16|
            NOP             4
            B       .S2     B3                  ; |16|
            NOP             5
            ; BRANCH OCCURS                     ; |16|
        .endfunc        16,000087c00h,24


;******************************************************************************
;* UNDEFINED EXTERNAL REFERENCES                                              *
;******************************************************************************
        .global __fltlif
```

apihost.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#include "pnkdev.h"
#include "pnkdapi.h"


#define BYUAPI_SIG      1               /* See pnkdev.h for range of available
                                           USER signals */

#define NODENAME        "astronomy"             /* Name of node where BOARDNAME
                                           resides */

#define BOARDNAME       "4291-1"                /* Name of board running target end
                                           code */

#define TICK_ULEN       1000            /* 1 mS ticks                    */
#define TIMEOUT_TICKS   10000           /* Total wait time =
                                           TICK_ULEN * TIMEOUT_TICKS uS */

#define BUFFER_SIZE             1024

#define ITERATIONS              10

float* mydata;
short int* shortdata;
short int* shortdataA;
short int* shortdataB;
short int* shortdataC;

int* temp;
int count=0;

int handler()
{
  PNK_SIGNAL_PTR sig;
  PNK_RETURN_TYPE ret = {0,0,NULL};
  int i;
  int temp1;
```

```
  sig = pnkd_signal();  /* Retreive signal */


  /* Copy signal to global variable. sig not valid after pnkd_return */

  temp1=BUFFER_SIZE*count;
  for (i=0;i<BUFFER_SIZE;i++) {
          temp[temp1+i]=sig->data.data_val[i];
  }

  pnkd_return(&ret);    /* Complete signal transaction */

  return(0);
}


main()
{
  PNK_DEVICE_TYPE     dev;
  FILE* fp, * fp2, * fp3, * fp4, * fp5, * fp6, * fp7, * fp8;
  int i;

  if ((fp=fopen("idata.txt","w"))==NULL) {
          printf("Couldn't open file idata.txt!\n");
          exit(1);
  }

  if ((fp2=fopen("qdata.txt","w"))==NULL) {
          printf("Couldn't open file qdata.txt!\n");
          exit(1);
  }

  if ((fp3=fopen("Aidata.txt","w"))==NULL) {
          printf("Couldn't open file Aidata.txt!\n");
          exit(1);
  }

  if ((fp4=fopen("Aqdata.txt","w"))==NULL) {
          printf("Couldn't open file Aqdata.txt!\n");
          exit(1);
  }

  if ((fp5=fopen("Bidata.txt","w"))==NULL) {
          printf("Couldn't open file Bidata.txt!\n");
          exit(1);
  }

  if ((fp6=fopen("Bqdata.txt","w"))==NULL) {
          printf("Couldn't open file Bqdata.txt!\n");
          exit(1);
  }

  if ((fp7=fopen("Cidata.txt","w"))==NULL) {
          printf("Couldn't open file Cidata.txt!\n");
          exit(1);
  }

  if ((fp8=fopen("Cqdata.txt","w"))==NULL) {
          printf("Couldn't open file Cqdata.txt!\n");
          exit(1);
  }

  if ((mydata=malloc(ITERATIONS*BUFFER_SIZE*2*sizeof(short int)))==NULL) {
                printf("Malloc failed...\n");
                exit(1);
  }

  temp=(int*)mydata;
  shortdata=(short int*)mydata;

  if (pnkd_init(1024) != OK)    /* Initialize device API, maximum signal size
                                   is 1024 long words */
    {
      printf("Error initializing device interface!\n");
      exit(1);
```

```
    }

  dev.arch = "byuapi";
  dev.parent = "";
  dev.name = "byuapi";
  dev.params.params_len = 0;
  dev.params.params_val = 0;

  /* Create virtual device (A communications end point) */

  if (pnkd_createDev(&dev) != OK)
    {
      printf("Error creating device\n");
      exit(1);
    }

  /* Connect signal handler, hander to signal numbe APISAMP_SIG */

  if (pnkd_connect(BYUAPI_SIG, (FUNCPTR)handler, NULL) != OK)
    {
      printf("unable to connect signal handler\n");
      exit(1);
    }


  printf("Host running. type 'RETURN' on a blank line to quit\n");

while(1)
   {
        for (i=0;i<ITERATIONS;i++) {
          pnkd_pause();
          count++;
        }
          printf("success!");
          printf("%i, count: %i\n",i,count);
          for (i=0;i<4*BUFFER_SIZE;i+=2) {
                  fprintf(fp,"%f\n",mydata[i]);
                  fprintf(fp2,"%f\n",mydata[i+1]);
          }
          fclose(fp);
          fclose(fp2);
          shortdataA=(short int *)(&mydata[4*BUFFER_SIZE]);
          for (i=0;i<4*BUFFER_SIZE;i+=2) {
                  fprintf(fp3,"%i\n",shortdata[8*1024+i+1]);
                  fprintf(fp4,"%i\n",shortdata[8*1024+i]);
          }
          fclose(fp3);
          fclose(fp4);
          for (i=0;i<4*BUFFER_SIZE;i+=2) {
                  fprintf(fp5,"%i\n",shortdata[12*1024+i+1]);
                  fprintf(fp6,"%i\n",shortdata[12*1024+i]);
          }
          fclose(fp5);
          fclose(fp6);
          for (i=0;i<4*BUFFER_SIZE;i+=2) {
                  fprintf(fp7,"%i\n",shortdata[16*1024+i+1]);
                  fprintf(fp8,"%i\n",shortdata[16*1024+i]);
          }
          fclose(fp7);
          fclose(fp8);
          pnkd_deleteDev(0);
          return(0);
        }
}
```

A.3.3 LMS Adaptive Filter Source Code

After completing simulations and tests using MATLAB (see Sections 4.4 and A.2), the LMS adaptive algorithm was implemented in the DSP (see Section 4.5).

This section includes the following source code files (apihost.c is part of a Visual C++ application enabling data transfer from the DSP platform to host PC):

The following source code files are also used in the DSP LMS filter application, but are included in Section A.3.5:

4291_330e_mod.cmd

```
-x
-l dev6xsne.lib
-l c6xe.lib
-l vime.lib
-l p4291_330e.lib
-heap   0x1000
-stack  0x1000
-l os90e.lib
-l stdioe.lib
-l snioe.lib
-l swftnte.lib
-l ose.lib
-l rts6701e.lib
-l shmeme.lib
-l rtapie.lib

/*
 * Specify the sections of the Model 4291-330 memory.
 */

MEMORY
{
    PNKG  (RWIX) : org = 0x00000000,  len = 0x0000c400 /*Reserved for SwifNet*/
    GRAM  (RWIX) : org = 0x0000c400,  len = 0x001f3bff /* only if GBPR =0x40 */
    IPRAM  (RWIX) : org = 0x01400000, len = 0x00010000
    TVEC  (RWIX) : org = 0x02000000,  len = 0x00000400
    SDRAM (RWIX) : org = 0x02000800,  len = 0x00039800
    PNK   (RWIX) : org = 0x02ffa000,  len = 0x00006000 /*SwiftNet Kernel*/
                                                        /* Mirrored at
                                                           0x2f03000 */
    SBSRAM (RWIX) : org = 0x01000000, len = 0x00080000
    IDRAM  (RWIX) : org = 0x80000000,  len = 0x0000e000
}

/*
 * Specify the allocation of the sections in the Model 4290 memory.
 */

SECTIONS
{
        .buffer0    :     > IDRAM
        .globaldata :     > IDRAM
        .buffer1    :     > IDRAM align(0x8000)  /* When using double buffers,
                                                    place the second buffer in
                                                    Block 1 IDRAM for optimal
                                                    performance.*/
        .sbsram0        :       > IDRAM
        .sbsram1        :       > SBSRAM
        .sdram0         :       > SDRAM
        .gram0          :       > GRAM
        .text           :       > SBSRAM
        .cinit          :       > SBSRAM
        .const          :       > SBSRAM
        .switch         :       > SBSRAM
        .data           :       > IDRAM
        .bss            :       > SBSRAM
        .cio            :       > IDRAM
        .sysmem         :       > IDRAM
        .far            :       > IDRAM
        .stack          :       > IDRAM
        .xref           :       > IDRAM
        .tvt            :
                        {
                                _traptbl = .;
                        }> TVEC
        .dram0:
                        {
                                _RTAPI_MAXOUT = 0x20080;
                        _RTAPI_MAXIN = 0x0;

                                _shmem1_base = .;
                                _shmem1_top = . + 0x1f3bff;
                                _shmem2_base = 0;
```

```
                                _shmem2_top = 0;
                                _shmem3_base = 0;
                                _shmem3_top = 0;
                                _shmem4_base = 0;
                                _shmem4_top = 0;
                        } > GRAM

/* ISR Jump table used by SwiftNet resides here */
        .ivec:
                        {
                                _isr_jump_table = .;
                        }> PNK
}
```

LMS_parameters.h

```
#define DECIMATION_RATE  64
#define CENTER_FREQUENCY 8e6
#define GVAR             6
#define INTEGRATION_TIME 2
#define PROGRAM_RUN_TIME 30

/*DELAY_A is the number of samples that Processor A will be delayed.
Processor A is the auxiliary channel (interference reference channel) x[n]*/
#define DELAY_A          0

/*DELAY_C is the number of samples that Processor C will be delayed.
Processor C is the primary channel (data plus interference)   */
#define DELAY_C          0

/*P is the number of complex filter taps used in the LMS adaptive filter lms.c*/
#define P                                12

/*Filter step size*/
#define MU                               1e-10

/*Must be 1 2 4 6 8 Or 10*/
#define MASTER_ClkDiv         1

/* Clock Selection - P6216_INTERNAL_CLOCK or P6216_EXTERNAL_CLOCK */
#define CLOCK_SELECT    P6216_EXTERNAL_CLOCK

/* External Clock Rate - When the 6216 is being clocked externally or it is
                         receiving it's clock from another 6216, specify the
                         external clock frequency.*/
#define EXTERNAL_CLOCK_RATE   64400000L

/*Do you want the spectrum flipped?
If the input is coming from the BYU receiver:               P6216_NO_FLIP_SPECTRUM
If the signal is coming directly into the DSP w/o mixing:   P6216_FLIP_SPECTRUM */
#define FLIP_SPECTRUM   P6216_NO_FLIP_SPECTRUM
```

LMSadaptiveA.c

```
/*************************************************************************
*
*   File : LMSadaptiveA.c
*
*************************************************************************/
#include "stdlib.h"
#include "4290.h"
#include "4290rscm.h"
#include "4290bifo.h"
#include "4290mbx.h"
#include "6216.h"
#include "math.h"
#include "c6xtimer.h"
#include "4290dma.h"
#include "c6xdma.h"
#include "dma.h"
#include "Short_to_float.h"
#include "cfftr2_dit.h"
```

```
#include "Power_FFT.h"
#include "MoveData.h"
#include "LMS_parameters.h"
#include <rtapi.h>

/* Host - target communication defines */
/* Define the API sig for host/target communication */
#define BYUAPI_SIG 1

/* Define other API comm parameters */
#define HOSTNAME "astronomy"
#define DEVICENAME "byuapi"
#define TICK_ULEN 1000
#define TIMEOUT_TICKS 10000

/*  ******* GLOBAL DEFINES ********************  */

/* Number of 6216 VIM Modules being Synchronized */
#define  NUM_6216_VIM_MODULES     2             /* 1 or 2 6216's on 4290   */

/* Synchronization Master Processor - Must be Processor A or C */
#define  SYNC_MASTER              P4290_PROC_A  /* 0 or 2 for CPU's A & C  */

/* The Internal Oscillator Frequency can vary based on installed options
   on the board.
*/
#define INT_OSC_STANDARD      64000000L      /* 6216 Standard       */
#define INT_OSC_OPT20         65000000L      /* 6216 with Option 20 */
#define INT_OSC_OPT21         60000000L      /* 6216 with Option 21 */

/* ************ GLOBAL VARIABLES ******************  */

/* Global Data Buffers */

#define BUFFER_SIZE                     1024
#define TWO_BUFFER_SIZE                 2048
#define HALF_BUFFER_SIZE                512
#define ETA                                              10                      /*
    BUFFER_SIZE = 2^ETA */
#define NUM_LOOPS                               3002
#define PI_long                 3.141592653589793238

#pragma DATA_SECTION(outDataBlock, ".sbsram0");
far int outDataBlock[2*BUFFER_SIZE];
#pragma DATA_SECTION(interDataBlock, ".buffer1");
far float interDataBlock[4*BUFFER_SIZE];

struct complex {float real; float imag;};
#pragma  DATA_ALIGN(w, 8);
#pragma DATA_ALIGN(x, 8);
#pragma DATA_SECTION(x, ".buffer0");
#pragma  DATA_SECTION(w, ".buffer1");

#pragma  DATA_SECTION(y, ".buffer0");
struct complex   x[BUFFER_SIZE+DELAY_A];
volatile float   y[BUFFER_SIZE+1];
struct complex   w[BUFFER_SIZE/2];

#pragma DATA_SECTION(outDataBlock, ".sbsram0");
far int outDataBlock[2*BUFFER_SIZE];
#pragma DATA_SECTION(bitRevIndx, ".sbsram0");
#pragma DATA_SECTION(tempFFT, ".sbsram1");

#pragma DATA_SECTION(waitCntArray, ".sbsram1");
far unsigned int        waitCntArray[NUM_LOOPS];

/* Other global variables  -- */
int dmaDone=0;
int interDone=0;
int                     bitRevIndx[BUFFER_SIZE], tempFFT[BUFFER_SIZE/2];
volatile int       loopCnt=0;
volatile int       waitCnt=0;
volatile int       waitCntInter=0;
volatile int       IOpingpong=0;
volatile int       INTERpingpong=0;
```

```
/* ************ Function Prototypes ***************** */

/*void syncABD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);  */
void syncBCD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void waitForSync(unsigned int mailbox, unsigned int syncWord);
int calcBifoClock(P6216_BOARD_PARAMS *boardParams,
                  P6216_RCVR_PARAMS *rcvrParams);
void P6216SetBoardParams(P6216_BOARD_PARAMS *boardParams);
void P6216SetRcvrParams(P6216_RCVR_PARAMS *rcvrParams,
                        P6216_BOARD_PARAMS *boardParams,
                        unsigned int intClockRate);

/* Interrupt service routine for BIFIFO dma */

interrupt void ReadBifoISR() {

        /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH2))) &= 0xffbb;

        /* Set dma complete flag */
        dmaDone=1;

}  /* end of ISR */

interrupt void ReadInterISR() {

   /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH1))) &= 0xffbb;

   interDone=1;
}  /* end of ISR */

void main()
{
    P6216_BOARD_PARAMS p6216BoardParams;
    P6216_RCVR_PARAMS  p6216RcvrParams;
    P6216_REG_ADDR     p6216Regs;
    unsigned int       i;
    unsigned int       procId = P4290GetProcId();
    volatile float     phase;
    int                bifoClock;
    unsigned int       intClockRate = INT_OSC_STANDARD;
        P4290_DMA_PARAMS   dmaParams;
        P4290_DMA_PARAMS   dmaParams2;
        P4290_DMA_PARAMS   dmaParams3;
        P4290_DMA_PARAMS   dmaParams4;
        P4290_DMA_PARAMS   dmaParams5;
        int                        frameIndex;
        int                                indexVal;
    int                            *outData0,*outData1;
    short                          *outDataShort0,*outDataShort1;
    volatile int           finalwaitcnt;
    float                          *interData0,*interData1;
    volatile float         *outInterprocBifo;
    volatile unsigned int       *outInterprocBifoInt;
    volatile float         *inInterprocBifo;
    float                          *INTERdata[2];
    int                            *IOdata[2];
    short                          *SHORTIOdata[2];
    int                            streamstatus=0;
    int                            outputstream=0;
    int                            connected=0;
    volatile float         *spectrum;
    float                          DecimatedSampleRate;
        unsigned int       SAMPLE_RATE;

        /*Do not use the following six variables in the continuous while loop*/
        double                     DesiredIntegrationTime;    /*in seconds*/
    double                         ActualIntegrationTime;
    double                         DesiredIntegrationFrames;
    double                         DesiredProgramRunTime;
    double                         ActualProgramRunTime;
    double                         Temp1;
```

```
/*********************************************************************/
int                             NumberBlocks;
int                             ActualIntegrationFrames;
int                             TotalFrames;

int                             k, recurse_N, offset;
volatile float          d;
int                             NumberLoops;
float                           *x_new;
float                           floattemp;
unsigned int            unsigned_int_temp;

P4290_LED0_OFF;
P4290_LED1_ON;
P4290_LED2_OFF;
P4290_LED3_OFF;

P4290_LED2_ON;

if (CLOCK_SELECT==P6216_INTERNAL_CLOCK)
    SAMPLE_RATE=(unsigned int)INT_OSC_STANDARD;
    else
            SAMPLE_RATE=(unsigned int)EXTERNAL_CLOCK_RATE;
DecimatedSampleRate=(float)(SAMPLE_RATE/DECIMATION_RATE/MASTER_ClkDiv);

    DesiredIntegrationTime=INTEGRATION_TIME;
DesiredProgramRunTime=PROGRAM_RUN_TIME;
DesiredIntegrationFrames = (double)((DecimatedSampleRate/BUFFER_SIZE)*
    DesiredIntegrationTime);
ActualIntegrationFrames = (int)(DesiredIntegrationFrames);
if ((DesiredIntegrationFrames-ActualIntegrationFrames)>=0.5)
    ActualIntegrationFrames++;
ActualIntegrationTime = ((double)(ActualIntegrationFrames*BUFFER_SIZE))/((double)
    (DecimatedSampleRate));

Temp1 = (DesiredProgramRunTime/ActualIntegrationTime);
NumberBlocks = (int)(Temp1);
if ((Temp1-NumberBlocks)>=0.5)
    NumberBlocks++;
ActualProgramRunTime = (double)(NumberBlocks*ActualIntegrationTime);
TotalFrames=NumberBlocks*(ActualIntegrationFrames+1);

NumberLoops=TotalFrames+2;

/*Create the bit reversed index table*/
for (i=0; i<BUFFER_SIZE; i++)
            bitRevIndx[i] = i;
    recurse_N = BUFFER_SIZE;
    for (k=0; k<ETA-1; k++){
    recurse_N = recurse_N/2;
    for (offset=0; offset < BUFFER_SIZE; offset += 2*recurse_N) {
            for (i=0; i<recurse_N; i++) {
                    tempFFT[i] = bitRevIndx[offset+1+(i<<1)];
                    bitRevIndx[i+offset] = bitRevIndx[offset+(i<<1)];
            }
            for (i=0; i<recurse_N; i++)
                    bitRevIndx[i+offset+recurse_N] = tempFFT[i];
    }
}

y[0]=(float)procId;
spectrum=&y[1];
for (k=0; k<BUFFER_SIZE; k++) {
    spectrum[k]=0;
}

/* Initialize the FFT coefficent table, BUFFER_SIZE/2 point bit reversed ordering
     */

    d = 2*PI_long/BUFFER_SIZE;

    for (i=0; i<BUFFER_SIZE/2; i++){
    w[bitRevIndx[i]>>1].real = cosf(i*d);
    w[bitRevIndx[i]>>1].imag = sinf(i*d);
}
```

```
outData0 = outDataBlock;
outData1 = &outDataBlock[BUFFER_SIZE];
IOdata[1]=outData0;
IOdata[0]=outData1;

outDataShort0=(short*)outData0;
outDataShort1=(short*)outData1;

SHORTIOdata[1]=outDataShort0;
SHORTIOdata[0]=outDataShort1;

interData0 = interDataBlock;
interData1 = &interDataBlock[2*BUFFER_SIZE];

    INTERdata[1]=interData0;
INTERdata[0]=interData1;

x_new=(float *)&x[DELAY_A];

for (i=0; i<4*BUFFER_SIZE;i++) {
    interDataBlock[i]=0;
}

/*Summary of Port usage for all four processors:

Processor A:        XX Port receives data from Processor C
                           ZZ Port sends data to Processor B

Processor B:        XX Port receives data from Processor A
                           ZZ Port sends data to Processor D

Processor C:        XX Port not used
                           ZZ Port sends data to Processor A

Processor D:        XX Port receives data from Processor B
                           ZZ Port not used

Data flow for Beam Former:
            C --> A --> B --> D

Where Processors C, A and B are connected to antennas through the digital
    receiver*/

outInterprocBifo = (float *)P4290_LCL_FIFO_ZZ;
inInterprocBifo = (float *)P4290_LCL_FIFO_XX;
outInterprocBifoInt = P4290_LCL_FIFO_ZZ;

/* Turn off timer 0 - Enabled by bootcode to toggle led */
TIMER_RESET(C6X_TIMER_0);

/* Initialize Table of 6216 Addresses */
P6216InitRegAddr(0x320000L, &p6216Regs);

/* Determine whether option is installed on this board */
if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x21))
    intClockRate = INT_OSC_OPT21;
else if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x20))
    intClockRate = INT_OSC_OPT20;

/* Set 6216 Board Parameters */
P6216SetBoardParams(&p6216BoardParams);

/* Get 6216 Receiver Parameters */
P6216SetRcvrParams(&p6216RcvrParams, &p6216BoardParams, intClockRate);

    /* Set Center Frequency and Decimation  */
p6216RcvrParams.centerFreq = CENTER_FREQUENCY;
p6216RcvrParams.decimationRate = DECIMATION_RATE; /* Set for 1 MHz basband fs */
p6216BoardParams.gainSetting = GVAR;

/* Reset Board Registers */
P6216ResetRegs(&p6216Regs);

/* Initialize Board Registers */
```

```
P6216InitBoardRegs(&p6216BoardParams, &p6216Regs);

/* Initialize Receiver Registers */
P6216InitRcvrRegs(&p6216RcvrParams, &p6216Regs);

/* Calculate Slowest Bifo Clock Rate */
bifoClock = calcBifoClock(&p6216BoardParams, &p6216RcvrParams);

    #ifndef OPTION_330
/* Select IO Mezzanine Bifo */
P4290BifoSelect(P4290_BIFO_IO);
    #endif

    /* ********** SETUP INTERRUPTS AND DMA TRASFER REGISTERS ****** */

/* Clear interrupt enable registers. */
*P4290_LCR_IER0 = 0x0;
*P4290_LCR_IER1 = 0x0;

 /* Reset Interrupt Mapping Reg */
*P4290_LCR_IMR0 = 0;

/* Reset Miscellaneous Control Register. */
/* Also causes interrupts to be unlatched, which is why this example
   will not run on the standard 4290. */
*P4290_LCR_MCR0 |= 0x00f0;

    /* Set up Bifo almost full interrupt */
P4290SetupBifoDMAInt(P4290_BIFO_IO, P4290_INT_EXP_BIFO_IN_ALMST_FULL,
        P4290_IMR0_IO_BIFO_IN_EVENT, CPU_INT4, ISN_EXT_INT4, NULL);

/*Calculate frame index for Global Index register for
use of ping-pong buffers.  Difference between end of buffer 1 and start
of buffer 2. */
frameIndex = (int)outData1 - ((int)outData0 + (BUFFER_SIZE*4)) + 4;

/* Move frame index into upper half of index value.  Lower half is
   number of bytes in element. */
indexVal = (frameIndex <<16) | 4;

/* This version of DmaInit uses the indexVal calculated above */
P4290DmaInit((unsigned int)P4290_LCL_FIFO_IO, (unsigned int)outData0,
        DMA_ADDR_NO_MOD, DMA_ADDR_INC, DMA_INDXA, indexVal,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, BUFFER_SIZE, 2,
        DMA_CNT_RELOADA, FRAME_SYNC_ENABLE, DMA_RELOAD_NONE, DMA_RELOAD_GARB,
        SEN_EXT_INT4, SEN_NONE, 0X2088, C6X_DMA_IE,
        DMA_AUTOINIT_TRUE, &dmaParams);

/*  Setup DMA complete interrupt */
/*     Note DMA chan. 2 is hard-wired to internal interrupt CPU_INT10.
       This is the same as DMA_INT2.   */
C6xSetupInterrupt(CPU_INT10,ISN_DMA_INT2,&ReadBifoISR,0);
C6xEnableInterrupt(CPU_INT10);

/*********************Interprocessor Setup ************************/

    /*Processor A Flushes the BIFIFO between A and C and the BIFIFO between A and
        B*/
    P4290FlushBifo(P4290_BIFO_ZZ, P4290_BIFO_CLOCK_FREQ, 8*BUFFER_SIZE-4, 128,
                           8*BUFFER_SIZE-4,128);
    P4290FlushBifo(P4290_BIFO_XX, P4290_BIFO_CLOCK_FREQ, 4*BUFFER_SIZE-4, 128,
                           4*BUFFER_SIZE-4,128);

P4290SetupBifoDMAInt(P4290_BIFO_XX, P4290_INT_IP_BIFO_XX_IN_ALMST_FULL,
        P4290_IMR0_XX_BIFO_IN_EVENT, CPU_INT5, ISN_EXT_INT5, NULL);

/*Calculate frame index for Global Index register for
use of ping-pong buffers.  Difference between end of buffer 1 and start
of buffer 2. */
frameIndex = (int)interData1 - ((int)interData0 + (2*BUFFER_SIZE*4)) + 4;

/* Move frame index into upper half of index value.  Lower half is
   number of bytes in element. */
indexVal = (frameIndex <<16) | 4;
```

```
P4290DmaInit((unsigned int)P4290_LCL_FIFO_XX, (unsigned int)interData0,
        DMA_ADDR_NO_MOD, DMA_ADDR_INC, DMA_INDXA, indexVal,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, 2*BUFFER_SIZE, 2,
        DMA_CNT_RELOADB, FRAME_SYNC_ENABLE, DMA_RELOAD_NONE, DMA_RELOAD_GARC,
        SEN_EXT_INT5, SEN_NONE, 0X2088, C6X_DMA_IE,
        DMA_AUTOINIT_TRUE, &dmaParams2);

C6xSetupInterrupt(CPU_INT9,ISN_DMA_INT1,&ReadInterISR,0);
C6xEnableInterrupt(CPU_INT9);

/**********************End of interproc setup **********************/

/**********************LMS Data DMA Setup (Processor C->A for x)**********/

/* Send x*/
P4290DmaInit((unsigned int)x, (unsigned int)P4290_LCL_FIFO_ZZ,
        DMA_ADDR_INC, DMA_ADDR_NO_MOD, DMA_INDXB, 0,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, TWO_BUFFER_SIZE, 1,
        DMA_CNT_RELOADB, FRAME_SYNC_DISABLE, DMA_RELOAD_NONE, DMA_RELOAD_NONE,
        SEN_NONE, SEN_NONE, 0, C6X_DMA_WAIT,
        DMA_AUTOINIT_FALSE, &dmaParams3);

    /**********************End of LMS Data DMA Setup ******************/

    /**********************LMS Data DMA Setup (Processor B->D)**********/
/* Send only dhat0 */
P4290DmaInit((unsigned int)interData0, (unsigned int)P4290_LCL_FIFO_ZZ,
        DMA_ADDR_INC, DMA_ADDR_NO_MOD, DMA_INDXB, 0,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, 2*BUFFER_SIZE, 1,
        DMA_CNT_RELOADA, FRAME_SYNC_DISABLE, DMA_RELOAD_NONE, DMA_RELOAD_NONE,
        SEN_NONE, SEN_NONE, 0, C6X_DMA_NO_WAIT,
        DMA_AUTOINIT_FALSE, &dmaParams4);

/* Send only dhat1 */
P4290DmaInit((unsigned int)interData1, (unsigned int)P4290_LCL_FIFO_ZZ,
        DMA_ADDR_INC, DMA_ADDR_NO_MOD, DMA_INDXB, 0,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, 2*BUFFER_SIZE, 1,
        DMA_CNT_RELOADA, FRAME_SYNC_DISABLE, DMA_RELOAD_NONE, DMA_RELOAD_NONE,
        SEN_NONE, SEN_NONE, 0, C6X_DMA_NO_WAIT,
        DMA_AUTOINIT_FALSE, &dmaParams5);
/**********************End of LMS Data DMA Setup ******************/

    /* ***************** SYNCHRONIZE THE 4 PROCESSORS *************** */
/* Disable Fifo Writes */
P6216_DISABLE_FIFO_WRITE(p6216Regs.control);

/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
syncBCD(NUM_6216_VIM_MODULES, 1, 0xabababab);

/*Flush the IO Bifo (different Almost Full Levels for each channel)
Processor A is the second in the chain of the beamformer, therefore the
almost full level is set to 3*BUFFER_SIZE-4*/

    P4290FlushBifo(P4290_BIFO_IO, bifoClock, 3*BUFFER_SIZE-4, 128,
    3*BUFFER_SIZE-4, 128);

/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
syncBCD(NUM_6216_VIM_MODULES, 1, 0xdcdcdcdc);

P6216_SET_SYNC_GENERATE(p6216Regs.syncGen, 0);
while ((*p6216Regs.syncGen & 0x1) != 0);
P6216_SET_SYNC_GENERATE(p6216Regs.syncGen, 1);

    /* ************* MAIN I/O TRANSFER LOOP, ON DMA INTERRUPT *** */

/* Set up for DMA BIFO read with synchronization with IO-In AF. */
P4290DmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_AUTO_START_VAL,
    CPU_INT4, ISN_EXT_INT4, &dmaParams);
P4290DmaTransfer(DMA_CH1, C6X_DMA_NO_WAIT, DMA_AUTO_START_VAL,
    CPU_INT5, ISN_EXT_INT5, &dmaParams2);
```

```
 /*Everything above this point is entirely setup.  The infinite
loop below constitutes the processing portion of the routine*/

while (!dmaDone)
        ++waitCnt;
dmaDone = 0;
IOpingpong=!IOpingpong;
waitCntArray[0]=waitCnt;
/*Convert the data to floating point data*/
Short_to_float(x_new, SHORTIOdata[IOpingpong], TWO_BUFFER_SIZE);

while (1)
{
        /* wait for dma to complete */
    while (!dmaDone)
        ++waitCnt;
    dmaDone = 0;
    IOpingpong=!IOpingpong;

    if (loopCnt<NUM_LOOPS)
        waitCntArray[loopCnt+1]=waitCnt;

    while (!interDone)
        ++waitCntInter;
    interDone=0;
    INTERpingpong=!INTERpingpong;

    /*Implement the variable delay for processor A*/
    if (DELAY_A>0)  {
        MoveData((float*) x, (float*)&x[BUFFER_SIZE], 2*DELAY_A);
    }

    /*Convert the data to floating point data*/
    Short_to_float(x_new, SHORTIOdata[IOpingpong], TWO_BUFFER_SIZE);

    /*Start interprocessor communication from A to B*/
    P4290DmaTransfer(DMA_CH0, C6X_DMA_WAIT, DMA_START_VAL, 0, 0, &dmaParams3);

    if (INTERpingpong)
        P4290DmaTransfer(DMA_CH0, C6X_DMA_NO_WAIT, DMA_START_VAL, 0, 0, &
            dmaParams4);
    else
        P4290DmaTransfer(DMA_CH0, C6X_DMA_NO_WAIT, DMA_START_VAL, 0, 0, &
            dmaParams5);

    /*Compute the 1024 point FFT periodogram of the reference channel x[n]*/
    cfftr2_dit((float *)x, (float *)w,  BUFFER_SIZE);
        Power_FFT ((float *)x, spectrum, bitRevIndx, BUFFER_SIZE);

        ++loopCnt;

        /*If we've reached the end of the program run time, stop the dma
            transfer
        and send the final reference spectrum to the host PC along with all
            of the
        test parameters.*/
        if (loopCnt == NumberLoops) {

                finalwaitcnt=waitCnt;

        P4290DmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_PAUSE_VAL,
            CPU_INT4, ISN_EXT_INT4, &dmaParams);
            P4290DmaTransfer(DMA_CH1, C6X_DMA_NO_WAIT, DMA_PAUSE_VAL,
                    CPU_INT5, ISN_EXT_INT5, &dmaParams2);

            for (i=0;i<200000;i++);

            outputstream = rt_open(":FFTstreamA:0",O_NDELAY,0);
            if (outputstream == -1 ) { exit(255); }
            while(!connected) {
                    rt_ioctl(outputstream, I_CONNECT, &connected);
            }
            for(i=0;i<9;i++) {
                    if (i==0)
                            unsigned_int_temp=(unsigned int)P;
```

```
                        else if (i==1)
                                unsigned_int_temp=(unsigned int)DECIMATION_RATE;
                        else if (i==2)
                                unsigned_int_temp=(unsigned int)MASTER_ClkDiv;
                        else if (i==3)
                                unsigned_int_temp=(unsigned int)
                                    ActualIntegrationFrames;
                        else if (i==4)
                                unsigned_int_temp=(unsigned int)TotalFrames;
                        else if (i==5)
                                unsigned_int_temp=(unsigned int)DELAY_A;
                        else if (i==6)
                                unsigned_int_temp=(unsigned int)DELAY_C;
                        else if (i==7)
                                unsigned_int_temp=(unsigned int)SAMPLE_RATE;
                        else
                                unsigned_int_temp=(unsigned int)NumberBlocks;

                        streamstatus=-1;
                        while (streamstatus==-1) {
                                streamstatus = rt_write(outputstream, (char*)&
                                    unsigned_int_temp,4);
                        }
                }
                for(i=0;i<8;i++) {
                        if (i==0)
                                floattemp=(float)CENTER_FREQUENCY;
                        else if (i==1)
                                floattemp=(float)GVAR;
                        else if (i==2)
                                floattemp=(float)INTEGRATION_TIME;
                        else if (i==3)
                                floattemp=(float)PROGRAM_RUN_TIME;
                        else if (i==4)
                                floattemp=(float)ActualIntegrationTime;
                        else if (i==5)
                                floattemp=(float)ActualProgramRunTime;
                        else if (i==6)
                                floattemp=(float)MU;
                        else
                                floattemp=(float)DecimatedSampleRate;
                        streamstatus=-1;
                        while (streamstatus==-1) {
                                streamstatus = rt_write(outputstream, (char*)&
                                    floattemp,4);
                        }
                }
                streamstatus=-1;
                y[0]=(float)loopCnt;
                while (streamstatus==-1) {
                        streamstatus = rt_write(outputstream, (char*)y,4100);
                }
            rt_close(outputstream);
            P4290_LED3_ON;
                }
                waitCnt = 0;
        waitCntInter = 0;
    }
}
```

LMSadaptiveB.c

```
/*************************************************************************
*
*   File : LMSadaptiveB.c
*
*************************************************************************/
#include "stdlib.h"
#include "4290.h"
#include "4290rscm.h"
#include "4290bifo.h"
#include "4290mbx.h"
#include "6216.h"
#include "math.h"
```

```
#include "c6xtimer.h"
#include "4290dma.h"
#include "c6xdma.h"
#include "dma.h"
#include "lms.h"
#include "MoveData.h"
#include "LMS_parameters.h"
/* Swiftnet libs */
#include "pnkcapi.h"
#include "pnkdev.h"
#include "snio.h"

/* Host - target communication defines */
/* Define the API sig for host/target communication */
#define BYUAPI_SIG 1

/* Define other API comm parameters */
#define HOSTNAME "astronomy"
#define DEVICENAME "byuapi"
#define TICK_ULEN 1000
#define TIMEOUT_TICKS 10000

/*  ******* GLOBAL DEFINES ********************  */

/* Number of 6216 VIM Modules being Synchronized */
#define  NUM_6216_VIM_MODULES     2             /* 1 or 2 6216's on 4290   */

/* Synchronization Master Processor - Must be Processor A or C */
#define  SYNC_MASTER              P4290_PROC_A  /* 0 or 2 for CPU's A & C  */

/* The Internal Oscillator Frequency can vary based on installed options
   on the board.
*/
#define INT_OSC_STANDARD      64000000L      /* 6216 Standard       */
#define INT_OSC_OPT20         65000000L      /* 6216 with Option 20 */
#define INT_OSC_OPT21         60000000L      /* 6216 with Option 21 */

/* ************ GLOBAL VARIABLES ******************  */

/* Global Data Buffers */

#define BUFFER_SIZE                     1024
#define TWO_BUFFER_SIZE                 2048
#define HALF_BUFFER_SIZE                512

#define NUM_LOOPS                               3001

#pragma DATA_SECTION(interDataBlock, ".buffer1");
far float interDataBlock[8*BUFFER_SIZE+4*(P-1)];
#pragma DATA_SECTION(h, ".sbsram0");
far float h[2*P];
#pragma DATA_SECTION(dhat, ".sbsram0");
far float dhat[2*TWO_BUFFER_SIZE];

#pragma DATA_SECTION(waitCntInterArray, ".sbsram1");
far unsigned int waitCntInterArray[NUM_LOOPS];

/* Other global variables  -- */
int dmaDone=0;
int interDone=0;
volatile int       waitCnt=0;
volatile int       IOpingpong=0;
volatile int       INTERpingpong=0;
/* ************ Function Prototypes **************** */

void syncABD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void syncBCD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void waitForSync(unsigned int mailbox, unsigned int syncWord);
int calcBifoClock(P6216_BOARD_PARAMS *boardParams,
                  P6216_RCVR_PARAMS *rcvrParams);
void P6216SetBoardParams(P6216_BOARD_PARAMS *boardParams);
void P6216SetRcvrParams(P6216_RCVR_PARAMS *rcvrParams,
                        P6216_BOARD_PARAMS *boardParams,
```

```
                        unsigned int intClockRate);

/* Interrupt service routine for BIFIFO dma */

interrupt void ReadBifoISR() {

        /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH2))) &= 0xffbb;

        /* Set dma complete flag */
        dmaDone=1;

}  /* end of ISR */

interrupt void ReadInterISR() {

   /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH1))) &= 0xffbb;

   interDone=1;
}  /* end of ISR */

void main()
{
    P6216_BOARD_PARAMS p6216BoardParams;
    P6216_RCVR_PARAMS  p6216RcvrParams;
    P6216_REG_ADDR     p6216Regs;
    unsigned int       i;
    unsigned int       procId = P4290GetProcId();
    volatile float     phase;
    int                bifoClock;
    unsigned int       intClockRate = INT_OSC_STANDARD;
        P4290_DMA_PARAMS   dmaParams0;
        P4290_DMA_PARAMS   dmaParams1;
        P4290_DMA_PARAMS   dmaParams2;
        P4290_DMA_PARAMS   dmaParams3;
        int                        frameIndex,frameIndex1;
        int                                indexVal;
    volatile int           finalwaitcnt;
    float                      *interData0,*interData1;
    float                      *x0,*x1,*dhat0,*dhat1;
    float                      *backupData0,*backupData1;
    volatile float         *outInterprocBifo;
    volatile unsigned int       *outInterprocBifoInt;
    volatile float         *inInterprocBifo;
    float                      *INTERdata[2];
    float                      *BACKUPdata[2];
    float                      *x[2];
    float                      *a[2];
    float                      *dhat_pp[2];
        float                  mu=(float)MU;
    unsigned int           loopCnt=0;
    unsigned int           waitCntInter=0;

    P4290_LED0_OFF;
    P4290_LED1_ON;
    P4290_LED2_OFF;
    P4290_LED3_OFF;
    waitCnt=0;
        IOpingpong=0;
        INTERpingpong=0;

    interData0 = &interDataBlock[2*(P-1)];
    interData1 = &interDataBlock[4*BUFFER_SIZE+4*(P-1)];

        INTERdata[1]=interData0;
    INTERdata[0]=interData1;

    dhat0=dhat;
    dhat1=&dhat[TWO_BUFFER_SIZE];

    dhat_pp[1]=dhat0;
    dhat_pp[0]=dhat1;

    x0 = interDataBlock;
```

```
x1 = &interDataBlock[4*BUFFER_SIZE+2*(P-1)];

x[1]=x0;
x[0]=x1;

a[1]=&interData0[2048];
a[0]=&interData1[2048];

backupData0 = (float *)(&interData0[2048-2*(P-1)]);
backupData1 = (float *)(&interData1[2048-2*(P-1)]);

BACKUPdata[1]=backupData0;
BACKUPdata[0]=backupData1;

for (i=0; i<(8*BUFFER_SIZE+4*(P-1));i++) {
    interDataBlock[i]=0.0f;
}

/*The Filter Taps MUST be initialized to zero*/
for (i=0;i<(2*P);i++) {
    h[i]=0.0f;
}
for (i=0;i<(2*BUFFER_SIZE);i++) {
    /*dhat0[i]=i;*/
    dhat0[i]=0.0f;
}
for (i=0;i<(2*BUFFER_SIZE);i++) {
    /*dhat1[i]=2047-i;*/
    dhat1[i]=0.0f;
}

/*Summary of Port usage for all four processors:

Processor A:        XX Port receives data from Processor C
                        ZZ Port sends data to Processor B

Processor B:        XX Port receives data from Processor A
                        ZZ Port sends data to Processor D

Processor C:        XX Port not used
                        ZZ Port sends data to Processor A

Processor D:        XX Port receives data from Processor B
                        ZZ Port not used

Data flow for Beam Former:
            C --> A --> B --> D

Where Processors C, A and B are connected to antennas through the digital
    receiver*/

outInterprocBifo = (float *)P4290_LCL_FIFO_ZZ;
outInterprocBifoInt = P4290_LCL_FIFO_ZZ;
inInterprocBifo = (float *)P4290_LCL_FIFO_XX;

/* Turn off timer 0 - Enabled by bootcode to toggle led */
TIMER_RESET(C6X_TIMER_0);

/* Initialize Table of 6216 Addresses */
P6216InitRegAddr(0x320000L, &p6216Regs);

/* Determine whether option is installed on this board */
if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x21))
    intClockRate = INT_OSC_OPT21;
else if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x20))
    intClockRate = INT_OSC_OPT20;

/* Set 6216 Board Parameters */
P6216SetBoardParams(&p6216BoardParams);

/* Get 6216 Receiver Parameters */
P6216SetRcvrParams(&p6216RcvrParams, &p6216BoardParams, intClockRate);

    /* Set Center Frequency and Decimation  */
p6216RcvrParams.centerFreq = CENTER_FREQUENCY;
```

```
p6216RcvrParams.decimationRate = DECIMATION_RATE; /* Set for 1 MHz basband fs */
p6216BoardParams.gainSetting = GVAR;

/* Reset Board Registers */
P6216ResetRegs(&p6216Regs);

/* Initialize Board Registers */
P6216InitBoardRegs(&p6216BoardParams, &p6216Regs);

/* Initialize Receiver Registers */
P6216InitRcvrRegs(&p6216RcvrParams, &p6216Regs);

/* Calculate Slowest Bifo Clock Rate */
bifoClock = calcBifoClock(&p6216BoardParams, &p6216RcvrParams);

    #ifndef OPTION_330
/* Select IO Mezzanine Bifo */
P4290BifoSelect(P4290_BIFO_IO);
    #endif

    /* ********** SETUP INTERRUPTS AND DMA TRASFER REGISTERS ****** */

/* Clear interrupt enable registers. */
*P4290_LCR_IER0 = 0x0;
*P4290_LCR_IER1 = 0x0;

 /* Reset Interrupt Mapping Reg */
*P4290_LCR_IMR0 = 0;

/* Reset Miscellaneous Control Register. */
*P4290_LCR_MCR0 |= 0x00f0;

/**********************LMS Data DMA Setup (Processor B->D)**********/
frameIndex1 = (unsigned int)dhat1 - ((unsigned int)dhat0 + (4*BUFFER_SIZE)*2)
     + 4;
/*NOTE: The frameIndex is not correct on most other projects!!!*/
/* Move frame index into upper half of index value.  Lower half is
   number of bytes in element. */

indexVal = (frameIndex1 <<16) | 4;

/* Send only dhat0 */
P4290DmaInit((unsigned int)dhat0, (unsigned int)P4290_LCL_FIFO_ZZ,
        DMA_ADDR_INC, DMA_ADDR_NO_MOD, DMA_INDXB, 0,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, 2*BUFFER_SIZE, 1,
        DMA_CNT_RELOADA, FRAME_SYNC_DISABLE, DMA_RELOAD_NONE, DMA_RELOAD_NONE,
        SEN_NONE, SEN_NONE, 0, C6X_DMA_NO_WAIT,
        DMA_AUTOINIT_FALSE, &dmaParams0);

P4290DmaInit((unsigned int)dhat1, (unsigned int)P4290_LCL_FIFO_ZZ,
        DMA_ADDR_INC, DMA_ADDR_NO_MOD, DMA_INDXB, 0,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, 2*BUFFER_SIZE, 1,
        DMA_CNT_RELOADA, FRAME_SYNC_DISABLE, DMA_RELOAD_NONE, DMA_RELOAD_NONE,
        SEN_NONE, SEN_NONE, 0, C6X_DMA_NO_WAIT,
        DMA_AUTOINIT_FALSE, &dmaParams1);
/**********************End of LMS Data DMA Setup ******************/

/**********************LMS Data DMA Setup (Processor B->D for h)**********/
/* Send only dhat0 */
P4290DmaInit((unsigned int)h, (unsigned int)P4290_LCL_FIFO_ZZ,
        DMA_ADDR_INC, DMA_ADDR_NO_MOD, DMA_INDXB, 0,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, 2*P, 1,
        DMA_CNT_RELOADA, FRAME_SYNC_DISABLE, DMA_RELOAD_NONE, DMA_RELOAD_NONE,
        SEN_NONE, SEN_NONE, 0, C6X_DMA_NO_WAIT,
        DMA_AUTOINIT_FALSE, &dmaParams3);
/**********************End of LMS Data DMA Setup ******************/

/*********************Interprocessor Setup ************************/
/*Processor B Flushes the BIFIFO between B and D*/
    P4290FlushBifo(P4290_BIFO_ZZ, P4290_BIFO_CLOCK_FREQ, 4*BUFFER_SIZE-4, 128,
                                                  4*BUFFER_SIZE-4,128);

P4290SetupBifoDMAInt(P4290_BIFO_XX, P4290_INT_IP_BIFO_XX_IN_ALMST_FULL,
        P4290_IMR0_XX_BIFO_IN_EVENT, CPU_INT5, ISN_EXT_INT5, NULL);
```

```
/*Calculate frame index for Global Index register for
use of ping-pong buffers.  Difference between end of buffer 1 and start
of buffer 2. */
frameIndex = (unsigned int)interData1 - ((unsigned int)interData0 + (4*
    BUFFER_SIZE)*4) + 4;
/*frameIndex = 2*P*4+4;        Either This or the above line will work fine*/
/*NOTE: The frameIndex is not correct on most other projects!!!*/
/* Move frame index into upper half of index value.  Lower half is
   number of bytes in element. */
indexVal = (frameIndex <<16) | 4;

P4290DmaInit((unsigned int)P4290_LCL_FIFO_XX, (unsigned int)interData0,
        DMA_ADDR_NO_MOD, DMA_ADDR_INDX, DMA_INDXA, indexVal,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, 4*BUFFER_SIZE, 2,
        DMA_CNT_RELOADB, FRAME_SYNC_ENABLE, DMA_RELOAD_NONE, DMA_RELOAD_GARC,
        SEN_EXT_INT5, SEN_NONE, 0X2088, C6X_DMA_IE,
        DMA_AUTOINIT_TRUE, &dmaParams2);

C6xSetupInterrupt(CPU_INT9,ISN_DMA_INT1,&ReadInterISR,0);
C6xEnableInterrupt(CPU_INT9);
interDone=0;
/**********************End of interproc setup **********************/

    /* ***************** SYNCHRONIZE THE 4 PROCESSORS ***************** */

/* Disable Fifo Writes */
P6216_DISABLE_FIFO_WRITE(p6216Regs.control);

/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
waitForSync(1, 0xabababab);
/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
waitForSync(1, 0xdcdcdcdc);

P4290DmaTransfer(DMA_CH1, C6X_DMA_NO_WAIT, DMA_AUTO_START_VAL,
    CPU_INT5, ISN_EXT_INT5, &dmaParams2);
P4290_LED2_ON;

/*Everything above this point is entirely setup.  The infinite
loop below constitutes the processing portion of the routine*/

while (1)
{
    /*Wait for the IO BIFIFO to fill up to the appropriate level*/
    while (!interDone)
                    ++waitCntInter;
    interDone=0;
    INTERpingpong=!INTERpingpong;

    /*Start interprocessor communication from B to D*/
    P4290DmaTransfer(DMA_CH0, C6X_DMA_NO_WAIT, DMA_START_VAL, 0, 0, &dmaParams3);

    if (loopCnt<NUM_LOOPS)
            waitCntInterArray[loopCnt]=waitCntInter;

    if (INTERpingpong)
            P4290DmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_START_VAL, 0, 0, &
                dmaParams1);
    else
            P4290DmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_START_VAL, 0, 0, &
                dmaParams0);

    /*Filter the data with the LMS adaptive algorithm.  There are several
        different
    filtering routines optimized for commonly used filter length.  lms() is an
        optimized
    function which will operate on any filter length*/
    if (P==3)
            lms_3(x[INTERpingpong], a[INTERpingpong], dhat_pp[INTERpingpong], h,
                mu);
    else if (P==4)
```

```
                lms_4(x[INTERpingpong], a[INTERpingpong], dhat_pp[INTERpingpong], h,
                    mu);
        else if (P==12)
                lms_12(x[INTERpingpong], a[INTERpingpong], dhat_pp[INTERpingpong], h
                    , mu);
        else if (P==2)
                lms_2(x[INTERpingpong], a[INTERpingpong], dhat_pp[INTERpingpong], h,
                    mu);
        else if (P==5)
                lms_5(x[INTERpingpong], a[INTERpingpong], dhat_pp[INTERpingpong], h,
                    mu);
        else if (P==30)
                lms_30(x[INTERpingpong], a[INTERpingpong], dhat_pp[INTERpingpong], h
                    , mu);
        else if (P==42)
                lms_42(x[INTERpingpong], a[INTERpingpong], dhat_pp[INTERpingpong], h
                    , mu);
        else
                lms(x[INTERpingpong], a[INTERpingpong], dhat_pp[INTERpingpong], h, P
                    , BUFFER_SIZE, mu);

        /*Keep the last P-1 samples from the previous block for the next block*/
        MoveData(x[!INTERpingpong], BACKUPdata[INTERpingpong], 2*(P-1));

                ++loopCnt;
        waitCntInter = 0;
    }
}
```

LMSadaptiveC.c

```
/*************************************************************************
*
*   File : LMSadaptiveC.c
*
*************************************************************************/
#include "stdlib.h"
#include "4290.h"
#include "4290rscm.h"
#include "4290bifo.h"
#include "4290mbx.h"
#include "6216.h"
#include "math.h"
#include "c6xtimer.h"
#include "4290dma.h"
#include "c6xdma.h"
#include "dma.h"
#include "Short_to_float.h"
#include "cfftr2_dit.h"
#include "Power_FFT.h"
#include "MoveData.h"
#include "LMS_parameters.h"
#include <rtapi.h>

/* Host - target communication defines */
/* Define the API sig for host/target communication */
#define BYUAPI_SIG 1

/* Define other API comm parameters */
#define HOSTNAME "astronomy"
#define DEVICENAME "byuapi"
#define TICK_ULEN 1000
#define TIMEOUT_TICKS 10000

/*  ******* GLOBAL DEFINES ********************  */

/* Number of 6216 VIM Modules being Synchronized */
#define  NUM_6216_VIM_MODULES     2             /* 1 or 2 6216's on 4290   */

/* Synchronization Master Processor - Must be Processor A or C */
#define  SYNC_MASTER              P4290_PROC_A  /* 0 or 2 for CPU's A & C  */

/* The Internal Oscillator Frequency can vary based on installed options
   on the board.
```

```
*/
/*#define INT_OSC_STANDARD      32000000L*/
#define INT_OSC_STANDARD      64000000L      /* 6216 Standard       */
#define INT_OSC_OPT20         65000000L      /* 6216 with Option 20 */
#define INT_OSC_OPT21         60000000L      /* 6216 with Option 21 */

/* *********** GLOBAL VARIABLES ******************  */

/* Global Data Buffers */

#define BUFFER_SIZE                     1024
#define TWO_BUFFER_SIZE                 2048
#define HALF_BUFFER_SIZE                512
#define ETA                                             10                      /*
    BUFFER_SIZE = 2^ETA */
#define NUM_LOOPS                               3003
/*#define NUM_LOOPS                             201  */
#define PI_long                 3.141592653589793238

struct complex {float real; float imag;};
#pragma  DATA_ALIGN(w, 8);
#pragma DATA_ALIGN(x, 8);
#pragma DATA_SECTION(x, ".buffer0");
#pragma  DATA_SECTION(w, ".buffer1");
#pragma  DATA_SECTION(y, ".buffer1");

struct complex   x[BUFFER_SIZE+DELAY_C];
volatile float   y[BUFFER_SIZE+1];
struct complex   w[BUFFER_SIZE/2];

#pragma DATA_SECTION(outDataBlock, ".sbsram0");
far int outDataBlock[2*BUFFER_SIZE];
#pragma DATA_SECTION(bitRevIndx, ".sbsram0");
#pragma DATA_SECTION(tempFFT, ".sbsram1");

#pragma DATA_SECTION(waitCntArray, ".sbsram1");
far unsigned int        waitCntArray[NUM_LOOPS];

/* Other global variables  -- */
int dmaDone=0;
int interDone=0;
int                     bitRevIndx[BUFFER_SIZE], tempFFT[BUFFER_SIZE/2];
volatile int       loopCnt=0;
volatile int       waitCnt=0;
volatile int       waitCntInter=0;
volatile int       IOpingpong=0;
volatile int       INTERpingpong=0;
/* ************ Function Prototypes *************** */

/*void syncABD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void syncBCD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);*/
void waitForSync(unsigned int mailbox, unsigned int syncWord);
int calcBifoClock(P6216_BOARD_PARAMS *boardParams,
                  P6216_RCVR_PARAMS *rcvrParams);
void P6216SetBoardParams(P6216_BOARD_PARAMS *boardParams);
void P6216SetRcvrParams(P6216_RCVR_PARAMS *rcvrParams,
                        P6216_BOARD_PARAMS *boardParams,
                        unsigned int intClockRate);

/* Interrupt service routine for BIFIFO dma */

interrupt void ReadBifoISR() {

        /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH2))) &= 0xffbb;

        /* Set dma complete flag */
        dmaDone=1;

}  /* end of ISR */

interrupt void ReadInterISR() {
```

```
   /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH1))) &= 0xffbb;

   interDone=1;
}  /* end of ISR */

void main()
{
    P6216_BOARD_PARAMS p6216BoardParams;
    P6216_RCVR_PARAMS  p6216RcvrParams;
    P6216_REG_ADDR     p6216Regs;
    unsigned int       i;
    unsigned int       procId = P4290GetProcId();
    volatile float     phase;
    int                bifoClock;
    unsigned int       intClockRate = INT_OSC_STANDARD;
        P4290_DMA_PARAMS   dmaParams;
        P4290_DMA_PARAMS   dmaParams2;
        int                        frameIndex;
        int                                indexVal;
        volatile float     *spectrum;
    int                            *outData0,*outData1;
    short                          *outDataShort0,*outDataShort1;
    volatile int           finalwaitcnt;
    volatile float         *outInterprocBifo;
    volatile float         *inInterprocBifo;
    volatile unsigned int       *outInterprocBifoInt;
    short                          *SHORTIOdata[2];
    int                            NumberLoops;
    int                            *IOdata[2];
    int                            streamstatus=0;
    int                            outputstream=0;
    int                            connected=0;
    float                          DecimatedSampleRate;
    unsigned int       SAMPLE_RATE;
        /*Do not use the following six variables in the continuous while loop*/
        double                     DesiredIntegrationTime;    /*in seconds*/
    double                         ActualIntegrationTime;
    double                         DesiredIntegrationFrames;
    double                         DesiredProgramRunTime;
    double                         ActualProgramRunTime;
    double                         Temp1;
    /*****************************************************************/
    int                            NumberBlocks;
    int                            ActualIntegrationFrames;
    int                            TotalFrames;
    int                            k, recurse_N, offset;
    volatile float         d;
    int                            streamCnt;
    float                          *x_new;

    P4290_LED0_OFF;
    P4290_LED1_ON;
    P4290_LED2_OFF;
    P4290_LED3_OFF;

    if (CLOCK_SELECT==P6216_INTERNAL_CLOCK)
        SAMPLE_RATE=(unsigned int)INT_OSC_STANDARD;
        else
                SAMPLE_RATE=(unsigned int)EXTERNAL_CLOCK_RATE;
    DecimatedSampleRate=(float)(SAMPLE_RATE/DECIMATION_RATE/MASTER_ClkDiv);

    DesiredIntegrationTime=INTEGRATION_TIME;
    DesiredProgramRunTime=PROGRAM_RUN_TIME;
    DesiredIntegrationFrames = (double)((DecimatedSampleRate/BUFFER_SIZE)*
        DesiredIntegrationTime);
    ActualIntegrationFrames = (int)(DesiredIntegrationFrames);
    if ((DesiredIntegrationFrames-ActualIntegrationFrames)>=0.5)
        ActualIntegrationFrames++;
    ActualIntegrationTime = ((double)(ActualIntegrationFrames*BUFFER_SIZE))/((double)
        (DecimatedSampleRate));

    Temp1 = (DesiredProgramRunTime/ActualIntegrationTime);
    NumberBlocks = (int)(Temp1);
    if ((Temp1-NumberBlocks)>=0.5)
```

```
    NumberBlocks++;
ActualProgramRunTime = (double)(NumberBlocks*ActualIntegrationTime);
TotalFrames=NumberBlocks*(ActualIntegrationFrames+1);

NumberLoops=TotalFrames+3;

/*Create the bit reversed index table*/
for (i=0; i<BUFFER_SIZE; i++)
            bitRevIndx[i] = i;
    recurse_N = BUFFER_SIZE;
    for (k=0; k<ETA-1; k++){
    recurse_N = recurse_N/2;
    for (offset=0; offset < BUFFER_SIZE; offset += 2*recurse_N) {
            for (i=0; i<recurse_N; i++) {
                    tempFFT[i] = bitRevIndx[offset+1+(i<<1)];
                    bitRevIndx[i+offset] = bitRevIndx[offset+(i<<1)];
            }
            for (i=0; i<recurse_N; i++)
                    bitRevIndx[i+offset+recurse_N] = tempFFT[i];
    }
}

spectrum=&y[1];
for (k=0; k<BUFFER_SIZE; k++) {
    spectrum[k]=0;
}

/* Initialize the FFT coefficent table, BUFFER_SIZE/2 point bit reversed ordering
     */

    d = 2*PI_long/BUFFER_SIZE;

    for (i=0; i<BUFFER_SIZE/2; i++){
    w[bitRevIndx[i]>>1].real = cosf(i*d);
    w[bitRevIndx[i]>>1].imag = sinf(i*d);
}

streamCnt=(int)(0.5*ActualIntegrationFrames);

outData0 = outDataBlock;
outData1 = &outDataBlock[BUFFER_SIZE];
IOdata[1]=outData0;
IOdata[0]=outData1;

outDataShort0=(short*)outData0;
outDataShort1=(short*)outData1;

SHORTIOdata[1]=outDataShort0;
SHORTIOdata[0]=outDataShort1;

x_new=(float *)&x[DELAY_C];

/*Summary of Port usage for all four processors:

Processor A:        XX Port receives data from Processor C
                            ZZ Port sends data to Processor B

Processor B:        XX Port receives data from Processor A
                            ZZ Port sends data to Processor D

Processor C:        XX Port not used
                            ZZ Port sends data to Processor A

Processor D:        XX Port receives data from Processor B
                            ZZ Port not used

Data flow for Beam Former:
            C --> A --> B --> D

Where Processors C, A and B are connected to antennas through the digital
    receiver*/

outInterprocBifo = (float *)P4290_LCL_FIFO_ZZ;
inInterprocBifo = (float *)P4290_LCL_FIFO_XX;
outInterprocBifoInt = P4290_LCL_FIFO_XX;
```

```
/* Turn off timer 0 - Enabled by bootcode to toggle led */
TIMER_RESET(C6X_TIMER_0);

/* Initialize Table of 6216 Addresses */
P6216InitRegAddr(0x320000L, &p6216Regs);

/* Determine whether option is installed on this board */
if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x21))
    intClockRate = INT_OSC_OPT21;
else if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x20))
    intClockRate = INT_OSC_OPT20;

/* Set 6216 Board Parameters */
P6216SetBoardParams(&p6216BoardParams);

/* Get 6216 Receiver Parameters */
P6216SetRcvrParams(&p6216RcvrParams, &p6216BoardParams, intClockRate);

    /* Set Center Frequency and Decimation  */
p6216RcvrParams.centerFreq = CENTER_FREQUENCY;
p6216RcvrParams.decimationRate = DECIMATION_RATE; /* Set for 1 MHz baseband fs */
p6216BoardParams.gainSetting = GVAR;

/* Reset Board Registers */
P6216ResetRegs(&p6216Regs);

/* Initialize Board Registers */
P6216InitBoardRegs(&p6216BoardParams, &p6216Regs);

/* Initialize Receiver Registers */
P6216InitRcvrRegs(&p6216RcvrParams, &p6216Regs);

/* Calculate Slowest Bifo Clock Rate */
bifoClock = calcBifoClock(&p6216BoardParams, &p6216RcvrParams);

    #ifndef OPTION_330
/* Select IO Mezzanine Bifo */
P4290BifoSelect(P4290_BIFO_IO);
    #endif

    /* ********** SETUP INTERRUPTS AND DMA TRASFER REGISTERS ****** */
/* Disable Cache in CSR register */
/*CSR=CSR & 0xfffffff1f; */

/* Clear interrupt enable registers. */
*P4290_LCR_IER0 = 0x0;
*P4290_LCR_IER1 = 0x0;

/* Reset Interrupt Mapping Reg */
*P4290_LCR_IMR0 = 0;

/* Reset Miscellaneous Control Register. */
/* Also causes interrupts to be unlatched, which is why this example
   will not run on the standard 4290. */
*P4290_LCR_MCR0 |= 0x00f0;

    /**********************LMS Data DMA Setup (Processor C->A for x)**********/
/* Send x*/
P4290DmaInit((unsigned int)x, (unsigned int)P4290_LCL_FIFO_ZZ,
        DMA_ADDR_INC, DMA_ADDR_NO_MOD, DMA_INDXB, 0,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, TWO_BUFFER_SIZE, 1,
        DMA_CNT_RELOADB, FRAME_SYNC_DISABLE, DMA_RELOAD_NONE, DMA_RELOAD_NONE,
        SEN_NONE, SEN_NONE, 0, C6X_DMA_WAIT,
        DMA_AUTOINIT_FALSE, &dmaParams2);
    /**********************End of LMS Data DMA Setup ******************/

    /*     P4290_DB_BIFO_IN_ALMOST_FULL_INTR_CLEAR;   */

    /* Set up Bifo almost full interrupt */
P4290SetupBifoDMAInt(P4290_BIFO_IO, P4290_INT_EXP_BIFO_IN_ALMST_FULL,
        P4290_IMR0_IO_BIFO_IN_EVENT, CPU_INT4, ISN_EXT_INT4, NULL);

/*Calculate frame index for Global Index register for
use of ping-pong buffers.  Difference between end of buffer 1 and start
```

```
of buffer 2. */
frameIndex = (int)outData1 - ((int)outData0 + (BUFFER_SIZE*4)) + 4;

/* Move frame index into upper half of index value.  Lower half is
   number of bytes in element. */
indexVal = (frameIndex <<16) | 4;

/* This version of DmaInit uses the indexVal calculated above */
P4290DmaInit((unsigned int)P4290_LCL_FIFO_IO, (unsigned int)outData0,
        DMA_ADDR_NO_MOD, DMA_ADDR_INC, DMA_INDXA, indexVal,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, BUFFER_SIZE, 2,
        DMA_CNT_RELOADA, FRAME_SYNC_ENABLE, DMA_RELOAD_NONE, DMA_RELOAD_GARB,
        SEN_EXT_INT4, SEN_NONE, 0X2088, C6X_DMA_IE,
        DMA_AUTOINIT_TRUE, &dmaParams);

/*  Setup DMA complete interrupt */
/*      Note DMA chan. 2 is hard-wired to internal interrupt CPU_INT10.
        This is the same as DMA_INT2.   */
C6xSetupInterrupt(CPU_INT10,ISN_DMA_INT2,&ReadBifoISR,0);
C6xEnableInterrupt(CPU_INT10);

/*********************Interprocessor Setup ************************/
/*Processor C Flushes the BIFIFO between C and D*/
    P4290FlushBifo(P4290_BIFO_XX, P4290_BIFO_CLOCK_FREQ, 6*BUFFER_SIZE-4, 128,
            6*BUFFER_SIZE-4,128);
/***********************End of interproc setup **********************/

/*Connect to streaming node*/
outputstream = rt_open(":FFTstreamC:0",O_NDELAY,0);
if (outputstream == -1 ) { exit(255); }
while(!connected) {
    rt_ioctl(outputstream, I_CONNECT, &connected);
}
P4290_LED2_ON;
/* streamFLAG=3;  */

    /* ***************** SYNCHRONIZE THE 4 PROCESSORS **************** */

/* Disable Fifo Writes */
P6216_DISABLE_FIFO_WRITE(p6216Regs.control);

waitForSync(1, 0xabababab);

/*Flush the IO Bifo (different Almost Full Levels for each channel)
Processor C is the first in the chain of the beamformer, therefore the
almost full level is set to BUFFER_SIZE-4*/

    P4290FlushBifo(P4290_BIFO_IO, bifoClock, BUFFER_SIZE-4, 128,
    BUFFER_SIZE-4, 128);

waitForSync(1, 0xdcdcdcdc);

/* Set up for DMA BIFO read with synchronization with IO-In AF. */
P4290DmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_AUTO_START_VAL,
    CPU_INT4, ISN_EXT_INT4, &dmaParams);

/*Wait for the IO BIFIFO to fill up to the appropriate level*/
while (!dmaDone)
            ++waitCnt;

/*Everything above this point is entirely setup.  The infinite
loop below constitutes the processing portion of the routine*/

dmaDone = 0;
IOpingpong=!IOpingpong;
waitCntArray[0]=waitCnt;
/*Convert the data to floating point data*/
Short_to_float(x_new, SHORTIOdata[IOpingpong], TWO_BUFFER_SIZE);

while (1)
{
    while (!dmaDone)
            ++waitCnt;
    dmaDone = 0;
    IOpingpong=!IOpingpong;
```

```
        if (loopCnt<NUM_LOOPS)
                waitCntArray[loopCnt+1]=waitCnt;

        /*Implement the variable delay for processor C*/
        if (DELAY_C>0)  {
                MoveData((float*) x, (float*)&x[BUFFER_SIZE], 2*DELAY_C);
        }

        /*Convert the data to floating point data*/
        Short_to_float(x_new, SHORTIOdata[IOpingpong], TWO_BUFFER_SIZE);

        /*Start interprocessor communication from C to A*/
                P4290DmaTransfer(DMA_CH1, C6X_DMA_WAIT, DMA_START_VAL, 0, 0, &
                    dmaParams2);

        streamCnt++;

        /*Compute the 1024 point FFT periodogram of the primary channel alpha[n]*/
        if (streamCnt!=(ActualIntegrationFrames+1)) {
                cfftr2_dit((float *)x, (float *)w,  BUFFER_SIZE);
                        Power_FFT ((float *)x, spectrum, bitRevIndx, BUFFER_SIZE);
                }

                /*If it is time to send an integrated sample to the host, do so*/
                else {
                        streamstatus=-1;
                        y[0]=(float)(loopCnt+1);
                        while (streamstatus==-1) {
                                streamstatus = rt_write(outputstream, (char*)y, 4100)
                                    ;
                        }
                        for (i=0;i<1024;i++) {
                                spectrum[i]=0;
                        }
                        streamCnt=0;
                        P4290_LED3_TOGGLE;
                }

                ++loopCnt;

                /*If we've reached the end of the program run time, stop the dma
                    transfer*/
                if (loopCnt == NumberLoops) {
                                finalwaitcnt=waitCnt;
                P4290DmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_PAUSE_VAL,
                        CPU_INT4, ISN_EXT_INT4, &dmaParams);
                        rt_close(outputstream);
                        P4290_LED3_ON;
                }
                waitCnt = 0;
        waitCntInter = 0;
    }
}
```

LMSadaptiveD.c

```
 /*************************************************************************
*
*   File : LMSadaptiveD.c
*
*************************************************************************/
#include "stdlib.h"
#include "4290.h"
#include "4290rscm.h"
#include "4290bifo.h"
#include "4290mbx.h"
#include "6216.h"
#include "math.h"
#include "c6xtimer.h"
#include "4290dma.h"
#include "c6xdma.h"
#include "dma.h"
#include "cfftr2_dit.h"
```

```
#include "Power_FFT.h"
#include "LMS_parameters.h"
#include <rtapi.h>

/* Host - target communication defines */
/* Define the API sig for host/target communication */
#define BYUAPI_SIG 1

/* Define other API comm parameters */
#define HOSTNAME "astronomy"
#define DEVICENAME "byuapi"
#define TICK_ULEN 1000
#define TIMEOUT_TICKS 10000

/*  ******* GLOBAL DEFINES ********************  */

/* Number of 6216 VIM Modules being Synchronized */
#define  NUM_6216_VIM_MODULES     2             /* 1 or 2 6216's on 4290   */

/* Synchronization Master Processor - Must be Processor A or C */
#define  SYNC_MASTER              P4290_PROC_A  /* 0 or 2 for CPU's A & C  */

/* The Internal Oscillator Frequency can vary based on installed options
   on the board.
*/
#define INT_OSC_STANDARD      64000000L
#define INT_OSC_OPT20         65000000L      /* 6216 with Option 20 */
#define INT_OSC_OPT21         60000000L      /* 6216 with Option 21 */

/* *********** GLOBAL VARIABLES ******************  */

/* Global Data Buffers */

#define BUFFER_SIZE                     1024
#define TWO_BUFFER_SIZE                 2048
#define HALF_BUFFER_SIZE                512
#define NUM_LOOPS                               3000
/*#define NUM_LOOPS                             196*/
#define ETA                                             10                      /*
    BUFFER_SIZE = 2^ETA */
#define PI_long                 3.141592653589793238
#pragma DATA_SECTION(interDataBlock, ".buffer1");
far float interDataBlock[4*BUFFER_SIZE + 4*P];
struct complex {float real; float imag;};
#pragma  DATA_ALIGN(w, 8);
#pragma DATA_ALIGN(x, 8);
#pragma DATA_SECTION(x, ".buffer0");
#pragma  DATA_SECTION(w, ".buffer1");
struct complex   x[BUFFER_SIZE];
#pragma  DATA_SECTION(y, ".buffer0");
struct complex   x[BUFFER_SIZE];
volatile float   y[BUFFER_SIZE+1];
struct complex   w[BUFFER_SIZE/2];
#pragma DATA_SECTION(bitRevIndx, ".sbsram0");
#pragma DATA_SECTION(tempFFT, ".sbsram1");
#pragma DATA_SECTION(waitCntInterArray, ".sbsram1");
far unsigned int        waitCntInterArray[NUM_LOOPS];

/* Other global variables  -- */
int dmaDone=0;
int interDone=0;
int CdataDone=0;
int                     bitRevIndx[BUFFER_SIZE], tempFFT[BUFFER_SIZE/2];
volatile int       loopCnt=0;
volatile int       waitCnt=0;
volatile int       waitCntInter=0;
volatile int       IOpingpong=0;
volatile int       INTERpingpong=0;
/* ************ Function Prototypes *************** */

void syncABD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void syncBCD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void waitForSync(unsigned int mailbox, unsigned int syncWord);
```

```
int calcBifoClock(P6216_BOARD_PARAMS *boardParams,
                  P6216_RCVR_PARAMS *rcvrParams);
void P6216SetBoardParams(P6216_BOARD_PARAMS *boardParams);
void P6216SetRcvrParams(P6216_RCVR_PARAMS *rcvrParams,
                        P6216_BOARD_PARAMS *boardParams,
                        unsigned int intClockRate);

/* Interrupt service routine for BIFIFO dma */

interrupt void ReadBifoISR() {
        /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH2))) &= 0xffbb;

        /* Set dma complete flag */
        dmaDone=1;
}  /* end of ISR */

interrupt void ReadInterISR() {
   /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH1))) &= 0xffbb;

   interDone=1;
}  /* end of ISR */

interrupt void ReadCdataISR() {
   /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH2))) &= 0xffbb;

   CdataDone=1;
}  /* end of ISR */

void main()
{
    P6216_BOARD_PARAMS p6216BoardParams;
    P6216_RCVR_PARAMS  p6216RcvrParams;
    P6216_REG_ADDR     p6216Regs;
    unsigned int       i;
    unsigned int       procId = P4290GetProcId();
    volatile float     phase;
    int                bifoClock;
    unsigned int       intClockRate = INT_OSC_STANDARD;
        P4290_DMA_PARAMS   dmaParams;
        P4290_DMA_PARAMS   dmaParams2;
        int                        frameIndex;
        int                                indexVal;
    volatile int           finalwaitcnt;
    float                          *interData0,*interData1;
    float                          *h0,*h1;
    volatile float         *outInterprocBifo;
    volatile float         *inInterprocBifo;
    float                          *INTERdata[2];
    float                          *h[2];
    float                          *corr_matrix;
    int                            streamstatus=0;
    int                            outputstream=0;
    int                            connected=0;
        volatile float     *spectrum;
        float                      DecimatedSampleRate;
        unsigned int       SAMPLE_RATE;
        /*Do not use the following six variables in the continuous while loop*/
        double                     DesiredIntegrationTime;    /*in seconds*/
    double                         ActualIntegrationTime;
    double                         DesiredIntegrationFrames;
    double                         DesiredProgramRunTime;
    double                         ActualProgramRunTime;
    double                         Temp1;
    /*****************************************************************/
    int                            NumberBlocks;
    int                            ActualIntegrationFrames;
    int                            TotalFrames;
    int                            k, recurse_N, offset;
    volatile float         d;
    int                            streamCnt=0;
    int                            NumberLoops;
```

```
/*Connect to streaming node*/
outputstream = rt_open(":FFTstreamD:0",O_NDELAY,0);
if (outputstream == -1 ) { exit(255); }
while(!connected) {
    rt_ioctl(outputstream, I_CONNECT, &connected);
}

if (CLOCK_SELECT==P6216_INTERNAL_CLOCK)
    SAMPLE_RATE=(unsigned int)INT_OSC_STANDARD;
    else
            SAMPLE_RATE=(unsigned int)EXTERNAL_CLOCK_RATE;
DecimatedSampleRate=(float)(SAMPLE_RATE/DECIMATION_RATE/MASTER_ClkDiv);

P4290_LED0_OFF;
P4290_LED1_ON;
P4290_LED2_OFF;
P4290_LED3_OFF;

    DesiredIntegrationTime=INTEGRATION_TIME;
DesiredProgramRunTime=PROGRAM_RUN_TIME;
DesiredIntegrationFrames = (double)((DecimatedSampleRate/BUFFER_SIZE)*
    DesiredIntegrationTime);
ActualIntegrationFrames = (int)(DesiredIntegrationFrames);
if ((DesiredIntegrationFrames-ActualIntegrationFrames)>=0.5)
    ActualIntegrationFrames++;
ActualIntegrationTime = ((double)(ActualIntegrationFrames*BUFFER_SIZE))/((double)
    (DecimatedSampleRate));

Temp1 = (DesiredProgramRunTime/ActualIntegrationTime);
NumberBlocks = (int)(Temp1);
if ((Temp1-NumberBlocks)>=0.5)
    NumberBlocks++;
ActualProgramRunTime = (double)(NumberBlocks*ActualIntegrationTime);
TotalFrames=NumberBlocks*(ActualIntegrationFrames+1);

NumberLoops=TotalFrames;

 /*Create the bit reversed index table*/
for (i=0; i<BUFFER_SIZE; i++)
            bitRevIndx[i] = i;
    recurse_N = BUFFER_SIZE;
    for (k=0; k<ETA-1; k++){
    recurse_N = recurse_N/2;
    for (offset=0; offset < BUFFER_SIZE; offset += 2*recurse_N) {
            for (i=0; i<recurse_N; i++) {
                    tempFFT[i] = bitRevIndx[offset+1+(i<<1)];
                    bitRevIndx[i+offset] = bitRevIndx[offset+(i<<1)];
            }
            for (i=0; i<recurse_N; i++)
                    bitRevIndx[i+offset+recurse_N] = tempFFT[i];
    }
}

spectrum=&y[1];
for (k=0; k<BUFFER_SIZE; k++) {
    spectrum[k]=0;
}

/* Initialize the FFT coefficent table, BUFFER_SIZE/2 point bit reversed ordering
     */

    d = 2*PI_long/BUFFER_SIZE;

    for (i=0; i<BUFFER_SIZE/2; i++){
    w[bitRevIndx[i]>>1].real = cosf(i*d);
    w[bitRevIndx[i]>>1].imag = sinf(i*d);
}

corr_matrix = (float *)malloc(9*sizeof(float));
    for (i=0;i<9;i++)  {
    corr_matrix[i]=0;
}

h0 = &interDataBlock[0];
h1 = &interDataBlock[2*BUFFER_SIZE+2*P];
```

```
h[1] = h0;
h[0] = h1;

interData0 = &interDataBlock[2*P];
interData1 = &interDataBlock[2*BUFFER_SIZE+4*P];

    INTERdata[1]=interData0;
INTERdata[0]=interData1;

for (i=0; i<4*BUFFER_SIZE;i++) {
    interDataBlock[i]=0;
}
for (i=0;i<NUM_LOOPS;i++) {
    waitCntInterArray[i]=0;
}

/*Summary of Port usage for all four processors:

Processor A:        XX Port receives data from Processor C
                        ZZ Port sends data to Processor B

Processor B:        XX Port receives data from Processor A
                        ZZ Port sends data to Processor D

Processor C:        XX Port not used
                        ZZ Port sends data to Processor A

Processor D:        XX Port receives data from Processor B
                        ZZ Port not used

Data flow for Beam Former:
            C --> A --> B --> D

Where Processors C, A and B are connected to antennas through the digital
    receiver*/

outInterprocBifo = (float *)P4290_LCL_FIFO_ZZ;
inInterprocBifo = (float *)P4290_LCL_FIFO_XX;

/* Turn off timer 0 - Enabled by bootcode to toggle led */
TIMER_RESET(C6X_TIMER_0);

/* Initialize Table of 6216 Addresses */
P6216InitRegAddr(0x320000L, &p6216Regs);

/* Determine whether option is installed on this board */
if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x21))
    intClockRate = INT_OSC_OPT21;
else if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x20))
    intClockRate = INT_OSC_OPT20;

/* Set 6216 Board Parameters */
P6216SetBoardParams(&p6216BoardParams);

/* Get 6216 Receiver Parameters */
P6216SetRcvrParams(&p6216RcvrParams, &p6216BoardParams, intClockRate);

    /* Set Center Frequency and Decimation  */
p6216RcvrParams.centerFreq = CENTER_FREQUENCY;
p6216RcvrParams.decimationRate = DECIMATION_RATE; /* Set for 1 MHz basband fs */
p6216BoardParams.gainSetting = GVAR;

/* Reset Board Registers */
P6216ResetRegs(&p6216Regs);

/* Initialize Board Registers */
P6216InitBoardRegs(&p6216BoardParams, &p6216Regs);

/* Initialize Receiver Registers */
P6216InitRcvrRegs(&p6216RcvrParams, &p6216Regs);

/* Calculate Slowest Bifo Clock Rate */
bifoClock = calcBifoClock(&p6216BoardParams, &p6216RcvrParams);
```

```
    #ifndef OPTION_330
/* Select IO Mezzanine Bifo */
P4290BifoSelect(P4290_BIFO_IO);
    #endif

    /* ********** SETUP INTERRUPTS AND DMA TRASFER REGISTERS ****** */
/* Disable Cache in CSR register */
/*CSR=CSR & 0xfffff1f; */

/* Clear interrupt enable registers. */
*P4290_LCR_IER0 = 0x0;
*P4290_LCR_IER1 = 0x0;

 /* Reset Interrupt Mapping Reg */
*P4290_LCR_IMR0 = 0;

/* Reset Miscellaneous Control Register. */
/* Also causes interrupts to be unlatched, which is why this example
   will not run on the standard 4290. */
*P4290_LCR_MCR0 |= 0x00f0;

/*********************Interprocessor Setup ************************/
    /*Processor D doesn't need to flush any of the interprocessor BIFIFOs*/

P4290SetupBifoDMAInt(P4290_BIFO_XX, P4290_INT_IP_BIFO_XX_IN_ALMST_FULL,
        P4290_IMR0_XX_BIFO_IN_EVENT, CPU_INT5, ISN_EXT_INT5, NULL);

/*Calculate frame index for Global Index register for
use of ping-pong buffers.  Difference between end of buffer 1 and start
of buffer 2. */
frameIndex = (int)h1 - ((int)h0 + (2*BUFFER_SIZE*4)) + 4;

/* Move frame index into upper half of index value.  Lower half is
   number of bytes in element. */
indexVal = (frameIndex <<16) | 4;

P4290DmaInit((unsigned int)P4290_LCL_FIFO_XX, (unsigned int)h0,
        DMA_ADDR_NO_MOD, DMA_ADDR_INC, DMA_INDXA, indexVal,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, 2*BUFFER_SIZE+2*P, 2,
        DMA_CNT_RELOADB, FRAME_SYNC_ENABLE, DMA_RELOAD_NONE, DMA_RELOAD_GARC,
        SEN_EXT_INT5, SEN_NONE, 0X2088, C6X_DMA_IE,
        DMA_AUTOINIT_TRUE, &dmaParams2);

C6xSetupInterrupt(CPU_INT9,ISN_DMA_INT1,&ReadInterISR,0);
C6xEnableInterrupt(CPU_INT9);

/**********************End of interproc setup **********************/

P4290_LED2_ON;
    /* ***************** SYNCHRONIZE THE 4 PROCESSORS *************** */

/* Disable Fifo Writes */
P6216_DISABLE_FIFO_WRITE(p6216Regs.control);

/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
waitForSync(1, 0xabababab);
/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
waitForSync(1, 0xdcdcdcdc);

    /* ************* MAIN I/O TRANSFER LOOP, ON DMA INTERRUPT *** */

P4290DmaTransfer(DMA_CH1, C6X_DMA_NO_WAIT, DMA_AUTO_START_VAL,
    CPU_INT5, ISN_EXT_INT5, &dmaParams2);

while (1)
{
    /* wait for dma to complete */
    while (!interDone)
            ++waitCntInter;
    interDone=0;
    INTERpingpong=!INTERpingpong;
```

```
        if (loopCnt<NUM_LOOPS)
                waitCntInterArray[loopCnt]=waitCntInter;

        /*Compute the 1024 point FFT periodogram of the reference channel x[n]*/
        streamCnt++;
        if (streamCnt!=(ActualIntegrationFrames+1)) {
                cfftr2_dit((float *)INTERdata[INTERpingpong], (float *)w,
                     BUFFER_SIZE);
                        Power_FFT ((float *)INTERdata[INTERpingpong], spectrum,
                            bitRevIndx, BUFFER_SIZE);
                }

        /*If it is time to send an integrated sample to the host, do so*/
        else {
                        streamstatus=-1;
                        y[0]=(float)(loopCnt+1);
                        while (streamstatus==-1) {
                                streamstatus = rt_write(outputstream, (char*)h[
                                    INTERpingpong], 8*P);
                        }
                        streamstatus=-1;
                        while (streamstatus==-1) {
                                streamstatus = rt_write(outputstream, (char*)y, 4100)
                                    ;
                        }
                        for (i=0;i<1024;i++) {
                                spectrum[i]=0;
                        }
                        streamCnt=0;
                        P4290_LED3_TOGGLE;
                }
                ++loopCnt;

                /*If we've reached the end of the program run time, stop the dma
                    transfer*/
                if (loopCnt == NumberLoops) {
                                P4290DmaTransfer(DMA_CH1, C6X_DMA_NO_WAIT,
                                    DMA_PAUSE_VAL,
                        CPU_INT5, ISN_EXT_INT5, &dmaParams2);
                                finalwaitcnt=waitCnt;
                                P4290_LED3_ON;
                                rt_close(outputstream);
                }
                waitCnt = 0;
        waitCntInter = 0;
    }
}
```

lms.c

```
/*lms.c executes the lms adaptive filter algorithm

x=Interferer Data {Note length(x) must be equal to p+N-1
     in order to create N output samples}
a=Desired signal corrupted by Interferer
h=filter taps (interleaved real/imaginary).
p=# filter taps (order of filter)
N=number output samples for this block of data
dhat=filtered output   */

/*;These are the compile options:
  ;-kq -me -mh64 -ms0 -ml3 -mv6700 -o3 -frC:\ti\myprojects\adaptive -iC:\ReadyFlow
      \6216_2.0\include -iC:\ReadyFlow\4290_1_2.0\include -iC:\pentek\swiftnet.4.0.1\
      c6x\include -iC:\ti\c6000\cgtools\includ*/

/*This version of lms.c is the one which compiled the most efficiently.
It produced the compiled assembly code lms_opt.asm*/

void lms(float* x, float* a, float* dhat, float* h, int p, int N, float mu)
{
        int i=0,k;
        float v_hat_r=0.0f, v_hat_i=0.0f;
        float tempX_r,tempX_i;
```

```
        float temp_r,temp_i;
        volatile float* x_vol=(volatile float*)x;
        do
        {
                k=0;
                do
                {
                    temp_r=h[k];
                    temp_i=h[k+1];
                    tempX_r=x[i+k];
                    tempX_i=x[i+k+1];
                    v_hat_r=v_hat_r+temp_r*tempX_r-temp_i*tempX_i;
                    v_hat_i=v_hat_i+temp_r*tempX_i+temp_i*tempX_r;
                    k+=2;
                } while(k<2*p);
                temp_r=a[i]-v_hat_r;
                temp_i=a[i+1]-v_hat_i;
                v_hat_r=0.0f;
                dhat[i]=temp_r;
                v_hat_i=0.0f;
                k=0;
                do
                {
                    tempX_r=x_vol[i+k];
                    tempX_i=x_vol[i+k+1];
                    h[k]=h[k]+mu*(temp_r*tempX_r+temp_i*tempX_i);
                    h[k+1]=h[k+1]+mu*(temp_i*tempX_r-temp_r*tempX_i);
                    k+=2;
                } while(k<2*p);
                dhat[i+1]=temp_i;
                i+=2;
        }while (i<2*N);
}

/*This version of lms.c performs the same task, but doesn't compile as efficiently
void lms(float* x, float* a, float* dhat, float* h, int p, int N, float mu)
{
        int i,k;
        float v_hat_r=0.0f, v_hat_i=0.0f;
        float tempX_r,tempX_i;
        float temp_r,temp_i;
        volatile float* x_vol=(volatile float*)x;

        for (i=0;i<2*N;i+=2)
              {
              for (k=0;k<2*p;k+=2)
              {
                    temp_r=h[k];
                    temp_i=h[k+1];
                    tempX_r=x[i+k];
                    tempX_i=x[i+k+1];
                    v_hat_r=v_hat_r+temp_r*tempX_r-temp_i*tempX_i;
                    v_hat_i=v_hat_i+temp_r*tempX_i+temp_i*tempX_r;
              }
              temp_r=a[i]-v_hat_r;
              temp_i=a[i+1]-v_hat_i;
              v_hat_r=0.0f;
              dhat[i]=temp_r;
              v_hat_i=0.0f;
              dhat[i+1]=temp_i;
              for (k=0;k<2*p;k+=2)
              {
                    tempX_r=x_vol[i+k];
                    tempX_i=x_vol[i+k+1];
                    h[k]=h[k]+mu*(temp_r*tempX_r+temp_i*tempX_i);
                    h[k+1]=h[k+1]+mu*(temp_i*tempX_r-temp_r*tempX_i);

              }

        }


}  */
```

lms_opt.asm

```
;******************************************************************************
;* TMS320C6x ANSI C Codegen                                      Version 4.00 *
;* Date/Time created: Wed Sep 11 17:29:10 2002                                *
;******************************************************************************

;******************************************************************************
;* GLOBAL FILE PARAMETERS                                                     *
;*                                                                            *
;*   Architecture      : TMS320C670x                                          *
;*   Optimization      : Enabled at level 3                                   *
;*   Optimizing for    : Speed 1st, size 2nd                                  *
;*                       Based on options: -o3, -ms0                          *
;*   Endian            : Big                                                  *
;*   Interrupt Thrshld : Disabled                                             *
;*   Memory Model      : Large                                                *
;*   Calls to RTS      : Far                                                  *
;*   Pipelining        : Enabled                                              *
;*   Speculative Load  : Enabled (Threshold = 64                             ) *
;*   Memory Aliases    : Presume are aliases (pessimistic)                    *
;*   Debug Info        : No Debug Info                                        *
;*                                                                            *
;******************************************************************************

;These are the compile options:
;-kq -me -mh64 -ms0 -ml3 -mv6700 -o3 -frC:\ti\myprojects\adaptive -iC:\ReadyFlow\6216
    _2.0\include -iC:\ReadyFlow\4290_1_2.0\include -iC:\pentek\swiftnet.4.0.1\c6x\
    include -iC:\ti\c6000\cgtools\includ

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 -Z0 C:\TEMP\TI224_2 C:\TEMP\TI224_4 -w C:\ti\
    myprojects\adaptive
        .sect   ".text"
        .global _lms

;******************************************************************************
;* FUNCTION NAME: _lms                                                        *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,    *
;*                       A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,       *
;*                       B13,SP                                               *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,    *
;*                       A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,       *
;*                       B13,SP                                               *
;*   Local Frame Size  : 0 Args + 12 Auto + 44 Save = 56 byte                 *
;******************************************************************************
_lms:
;** --------------------------------------------------------------------------*
           STW     .D2T2   B13,*SP--(56)     ; |56|
           STW     .D2T2   B11,*+SP(48)      ; |56|
           STW     .D2T2   B3,*+SP(40)       ; |56|
           STW     .D2T2   B10,*+SP(44)      ; |56|
           STW     .D2T1   A4,*+SP(4)
           STW     .D2T2   B12,*+SP(52)      ; |56|
           STW     .D2T1   A13,*+SP(28)      ; |56|
           STW     .D2T1   A15,*+SP(36)      ; |56|
           STW     .D2T1   A14,*+SP(32)      ; |56|

           STW     .D2T1   A12,*+SP(24)      ; |56|
||         SHL     .S2     B8,1,B5           ; |57|

           STW     .D2T1   A11,*+SP(20)      ; |56|
||         MV      .S2     B4,B11            ;
||         CMPGT   .L2     B5,0,B0           ; |59|

           STW     .D2T1   A10,*+SP(16)      ; |56|
||         ADD     .L2     1,B5,B4           ; |59|
||         MV      .S2X    A6,B3             ;
||         ZERO    .L1     A3                ;
```

```
           STW     .D2T1   A3,*+SP(12)       ; |59|
||         SHR     .S2     B4,1,B2           ; |59|
||         MV      .L2     B6,B10            ;
||         SHL     .S1     A8,1,A4           ; |57|

   [!B0]   MVK     .S2     0x1,B2            ; |59|
||         MV      .L2X    A4,B12            ; |59|
||         STW     .D2T1   A3,*+SP(8)        ; |59|
||         ZERO    .L1     A6                ; |59|
||         MV      .S1     A10,A13           ;
||         ZERO    .D1     A0                ;

;** --------------------------------------------------------------------------*
;**   BEGIN LOOP L1
;** --------------------------------------------------------------------------*
L1:
           LDW     .D2T2   *+SP(12),B5

           MV      .L2X    A0,B4             ; |59|
||         LDW     .D2T1   *+SP(4),A0

           NOP             4
           ADD     .L2X    B5,A0,B5          ; |59|

           SUB     .L1X    B10,8,A7          ;
||         SUB     .L2     B5,8,B6           ; |59|

           LDW     .D1T1   *++A7(8),A3       ; (P) |72|
||         LDW     .D2T2   *++B6(8),B7       ; (P) |72|

           MVC     .S2     CSR,B13
||         LDW     .D1T1   *+A7(4),A4        ; (P) |72|
||         LDW     .D2T2   *+B6(4),B5        ; (P) |72|

           AND     .L2     -2,B13,B8

           LDW     .D2T2   *+SP(8),B8        ; |59|
||         MVC     .S2     B8,CSR            ; interrupts off

           LDW     .D1T1   *++A7(8),A9       ; (P) @|72|
||         LDW     .D2T2   *++B6(8),B9       ; (P) @|72|

;*--------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Known Minimum Trip Count         : 1
;*      Known Max Trip Count Factor      : 1
;*      Loop Carried Dependency Bound(^) : 4
;*      Unpartitioned Resource Bound     : 3
;*      Partitioned Resource Bound(*)    : 3
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                     3*       2
;*      .S units                     0        1
;*      .D units                     2        2
;*      .M units                     2        2
;*      .X cross paths               2        2
;*      .T address paths             2        2
;*      Long read paths              0        0
;*      Long write paths             0        0
;*      Logical  ops (.LS)           0        0     (.L or .S unit)
;*      Addition ops (.LSD)          1        0     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             2        2
;*      Bound(.L .S .D .LS .LSD)     2        2
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 4  Schedule found with 5 iterations in parallel
;*      done
;*
;*      Collapsed epilog stages     : 4
;*      Prolog not entirely removed
;*      Collapsed prolog stages     : 1
;*
;*      Minimum required memory pad : 32 bytes
;*
```

```
;*      Minimum safe trip count     : 1
;*--------------------------------------------------------------------------*
L2:    ; PIPED LOOP PROLOG

           LDW     .D2T2   *+B6(4),B7        ; (P) @|72|
||         LDW     .D1T1   *+A7(4),A4        ; (P) @|72|
||         MPYSP   .M1X    B7,A3,A0          ; (P) |72|

           CMPGT   .L2     B12,0,B0          ; |59|
||         MPYSP   .M1X    B5,A4,A0          ; (P) |72|
||         MPYSP   .M2X    B7,A4,B5          ; (P) |73|

   [!B0]   MVK     .S1     0x1,A1            ; |59|
|| [ B0]   ADD     .L1X    1,B12,A0          ;
||         MPYSP   .M2X    B5,A3,B8          ; (P) |73|

           MV      .L1X    B8,A8             ; |59|
|| [ B0]   SHR     .S1     A0,1,A1           ; |59|
||         LDW     .D2T2   *++B6(8),B8       ; (P) @@|72|
||         LDW     .D1T1   *++A7(8),A9       ; (P) @@|72|

           ZERO    .S1     A3                ; |72|
||         MV      .L2X    A1,B0             ;
||         LDW     .D1T1   *+A7(4),A0        ; (P) @@|72|
||         B       .S2     L3                ; (P) |75|
||         LDW     .D2T2   *+B6(4),B7        ; (P) @@|72|
||         MPYSP   .M1X    B9,A9,A0          ; (P) @|72|
||         ADDSP   .L1     A0,A6,A6          ; (P)  ^ |72|

           MVK     .S2     0x1,B1            ; init prolog collapse predicate
||         SUB     .S1     A1,1,A2           ;
||         MPYSP   .M1X    B7,A4,A5          ; (P) @|72|
||         ADDSP   .L2     B5,B8,B5          ; (P)  ^ |73|
||         MPYSP   .M2X    B9,A4,B9          ; (P) @|73|
||         SUBSP   .L1     A3,A0,A4          ; (P) |72|

;** --------------------------------------------------------------------------*
L3:    ; PIPED LOOP KERNEL

   [!B1]   ADDSP   .L1     A4,A8,A8          ;  ^ |72|
|| [ B0]   ADDSP   .L2     B8,B4,B4          ; @ ^ |73|
||         MPYSP   .M2X    B7,A9,B8          ; @@|73|

   [ A1]   SUB     .S1     A1,1,A1           ; @|75|
||         LDW     .D2T2   *++B6(8),B8       ; @@@@|72|
||         LDW     .D1T1   *++A7(8),A9       ; @@@@|72|

   [ A1]   B       .S2     L3                ; @|75|
|| [ A2]   ADDSP   .L1     A0,A6,A6          ; @@ ^ |72|
||         MPYSP   .M1X    B8,A9,A0          ; @@@|72|
||         LDW     .D2T2   *+B6(4),B7        ; @@@@|72|
||         LDW     .D1T1   *+A7(4),A0        ; @@@@|72|

   [ B1]   SUB     .S2     B1,1,B1           ;
|| [ B0]   SUB     .D2     B0,1,B0           ;
|| [ A2]   SUB     .S1     A2,1,A2           ;
|| [ A2]   ADDSP   .L2     B9,B5,B5          ; @@ ^ |73|
||         SUBSP   .L1     A3,A5,A4          ; @@|72|
||         MPYSP   .M2X    B8,A0,B9          ; @@@|73|
||         MPYSP   .M1X    B7,A0,A5          ; @@@|72|

;*--------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Known Minimum Trip Count         : 1
;*      Known Max Trip Count Factor      : 1
;*      Loop Carried Dependency Bound(^) : 6
;*      Unpartitioned Resource Bound     : 3
;*      Partitioned Resource Bound(*)    : 6
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                     2        2
;*      .S units                     0        1
;*      .D units                     2        4
;*      .M units                     6*       0
```

```
;*      .X cross paths               0        2
;*      .T address paths             2        4
;*      Long read paths              0        2
;*      Long write paths             0        0
;*      Logical  ops (.LS)           0        0     (.L or .S unit)
;*      Addition ops (.LSD)          0        1     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             1        2
;*      Bound(.L .S .D .LS .LSD)     2        3
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 6  Iterations in parallel > min. trip count
;*         ii = 6  Iterations in parallel > min. trip count
;*         ii = 6  Schedule found with 5 iterations in parallel
;*      done
;*
;*      Loop is interruptible
;*      Collapsed epilog stages     : 4
;*      Collapsed prolog stages     : 4
;*      Minimum required memory pad : 8 bytes
;*
;*      Minimum safe trip count     : 1
;*----------------------------------------------------------------------------*
L4:    ; PIPED LOOP EPILOG AND PROLOG

           MVK     .S2     0x4,B1            ; init prolog collapse predicate
||         MVK     .S1     0x3,A2            ; init prolog collapse predicate
||         CMPGT   .L1X    B12,0,A1          ;
||         LDW     .D2T2   *B11,B7           ; |76|

           LDW     .D2T1   *+B11(4),A4       ;

           MVC     .S2     B13,CSR           ; interrupts on
||         LDW     .D2T1   *+SP(12),A3       ; |79|
||         ADDSP   .L1     A8,A6,A6          ; |76|

           SUB     .S2     B10,8,B4          ;
|| [!A1]   MVK     .S1     0x1,A5            ; |79|
|| [ A1]   ADD     .L1X    1,B12,A5          ;
||         LDW     .D2T1   *+SP(4),A0        ; |79|
||         ADDSP   .L2     B5,B4,B5          ; |77|

   [ A1]   SHR     .S1     A5,1,A5           ; |79|
           ADD     .L2X    3,A5,B0           ;
           SUBSP   .L1X    B7,A6,A10         ; |76|
           SUBSP   .L1X    A4,B5,A12         ; |77|

           SUB     .L1X    B0,3,A1           ;
||         ADD     .S1     A3,A0,A0          ; |79|

           SUB     .S1     A0,8,A11          ; |79|
           STW     .D2T1   A10,*B3           ; |79|
;** ---------------------------------------------------------------------------*
L5:    ; PIPED LOOP KERNEL

   [ B0]   B       .S2     L5                ; |89|
||         ADDSP   .L2X    A8,B5,B7          ; |86|
||         MPYSP   .M1     A13,A9,A8         ; @|86|

           ADDSP   .L2X    A3,B6,B8          ; |87|
|| [!A2]   LDW     .D2T2   *++B4(8),B5       ; @|86|
||         MPYSP   .M1     A13,A4,A3         ; @|87|

   [!A2]   LDW     .D2T2   *+B4(4),B6        ; @|87|
||         ADDSP   .L1     A15,A14,A9        ; @@|86|
||         MPYSP   .M1     A0,A10,A14        ; @@@|86|

           SUBSP   .L1     A6,A5,A4          ; @@|87|
||         MPYSP   .M1     A0,A12,A6         ; @@@|87|
|| [ A1]   LDW     .D1T1   *++A11(8),A0      ; @@@@|84|

   [ A2]   SUB     .L1     A2,1,A2           ;
|| [!B1]   STW     .D2T2   B7,*-B4(8)        ; |86|
||         MPYSP   .M1     A7,A12,A15        ; @@@|86|

   [ B1]   SUB     .L2     B1,1,B1           ;
```

```
||	[ A1]	SUB	.S1	A1,1,A1		;
||	[!B1]	STW	.D2T2	B8,*-B4(4)	; |87|
||	[ B0]	SUB	.S2	B0,1,B0		; @|89|
||		MPYSP	.M1	A7,A10,A5	; @@@|87|
||	[ A1]	LDW	.D1T1	*+A11(4),A7	; @@@@|85|

;** --------------------------------------------------------------------------*
L6:    ; PIPED LOOP EPILOG
;** --------------------------------------------------------------------------*

		ADD	.L2	8,B11,B11	; |91|
||		LDW	.D2T1	*+SP(12),A3
||		SUB	.S2	B2,1,B2		; |93|

		LDW	.D2T1	*+SP(8),A0	; |90|
||	[ B2]	B	.S2	L1		; |93|

		ADD	.L2	8,B3,B3		; |91|
||		STW	.D2T1	A12,*+B3(4)	; |90|

		NOP		2
		ADD	.L1	8,A3,A3		; |91|

		MV	.L1	A0,A6		; |90|
||		STW	.D2T1	A3,*+SP(12)	; |91|

		; BRANCH OCCURS			; |93|
;** --------------------------------------------------------------------------*
		LDW	.D2T2	*+SP(40),B3	; |94|
		LDW	.D2T2	*+SP(48),B11	; |94|
		LDW	.D2T2	*+SP(44),B10	; |94|
		LDW	.D2T1	*+SP(36),A15	; |94|
		LDW	.D2T1	*+SP(32),A14	; |94|
		LDW	.D2T1	*+SP(28),A13	; |94|
		LDW	.D2T1	*+SP(24),A12	; |94|
		LDW	.D2T1	*+SP(20),A11	; |94|
		LDW	.D2T1	*+SP(16),A10	; |94|

		B	.S2	B3		; |94|
||		LDW	.D2T2	*+SP(52),B12	; |94|

		LDW	.D2T2	*++SP(56),B13	; |94|
		NOP		4
		; BRANCH OCCURS			; |94|
```

lms_2.c

```
/*
lms_2.c executes the lms adaptive filter algorithm for p=2

x=Reference Data {Note length(x) must be equal to p+N-1 in order to create N output
    samples}
a=Desired signal corrupted by interferer (Primary Channel)
h=filter taps (interleaved real/imaginary)
dhat=filtered output

THESE ARE NOT USED IN THIS VERSION
p=# filter taps (order of filter)       FIXED TO 2
N=2*number output samples for this block of data          FIXED TO 2048
*/

void lms_2(float* x, float* a, float* dhat, float* h, float mu)
{
    unsigned int i=0;
    float v_hat_r=0.0f, v_hat_i=0.0f;
    float tempX_r,tempX_i;
    float tempX_r2,tempX_i2;
    float temp_r,temp_i;
    float temp_r1,temp_i1;
    float temp_r2,temp_i2;

    temp_r1=h[0];
    temp_i1=h[1];
    temp_r2=h[2];
```

```
    temp_i2=h[3];

    do  {
        tempX_r=x[i];
        tempX_i=x[i+1];

        v_hat_r=v_hat_r+temp_r1*tempX_r-temp_i1*tempX_i;
        v_hat_i=v_hat_i+temp_r1*tempX_i+temp_i1*tempX_r;

        tempX_r2=x[i+2];
        tempX_i2=x[i+3];

        v_hat_r=v_hat_r+temp_r2*tempX_r2-temp_i2*tempX_i2;
        v_hat_i=v_hat_i+temp_r2*tempX_i2+temp_i2*tempX_r2;

        temp_r=a[i]-v_hat_r;
        v_hat_r=0.0f;
        temp_i=a[i+1]-v_hat_i;

        temp_r1=temp_r1+mu*(temp_r*tempX_r+temp_i*tempX_i);
        temp_i1=temp_i1+mu*(temp_i*tempX_r-temp_r*tempX_i);

        v_hat_i=0.0f;

        dhat[i]=temp_r;
        dhat[i+1]=temp_i;

        temp_r2=temp_r2+mu*(temp_r*tempX_r2+temp_i*tempX_i2);
        temp_i2=temp_i2+mu*(temp_i*tempX_r2-temp_r*tempX_i2);

        i+=2;
        }while (i<2048);

    h[0]=temp_r1;
    h[1]=temp_i1;
    h[2]=temp_r2;
    h[3]=temp_i2;
}
```

lms_2_opt.asm

```
;******************************************************************************
;* TMS320C6x ANSI C Codegen                                     Version 4.00 *
;* Date/Time created: Mon Nov 18 16:40:16 2002                                *
;******************************************************************************

;******************************************************************************
;* GLOBAL FILE PARAMETERS                                                     *
;*                                                                            *
;*   Architecture      : TMS320C670x                                          *
;*   Optimization      : Enabled at level 3                                   *
;*   Optimizing for    : Speed                                                *
;*                       Based on options: -o3, no -ms                        *
;*   Endian            : Big                                                  *
;*   Interrupt Thrshld : Disabled                                             *
;*   Memory Model      : Large                                                *
;*   Calls to RTS      : Far                                                  *
;*   Pipelining        : Enabled                                              *
;*   Speculative Load  : Enabled (Threshold = 64                             ) *
;*   Memory Aliases    : Presume are aliases (pessimistic)                    *
;*   Debug Info        : No Debug Info                                        *
;*                                                                            *
;******************************************************************************

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 C:\TEMP\TI224_2 C:\TEMP\TI224_4 -w C:\ti\myprojects\
    adaptive
        .sect   ".text"
        .global _lms_2
```

```
;*****************************************************************************
;* FUNCTION NAME: _lms_2                                                     *
;*                                                                           *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,B0,B1, *
;*                       B2,B4,B5,B6,B7,B8,B9,SP                             *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,B0,B1, *
;*                       B2,B3,B4,B5,B6,B7,B8,B9,SP                          *
;*   Local Frame Size  : 0 Args + 0 Auto + 16 Save = 16 byte                 *
;*****************************************************************************
_lms_2:
;** --------------------------------------------------------------------------*
           STW     .D2T1   A13,*SP--(16)     ; |18|

           STW     .D2T1   A12,*+SP(12)      ; |18|
||         MV      .L1X    B6,A13            ;

           MV      .L1     A8,A0             ;
||         LDW     .D1T1   *+A13(12),A8      ; |36|

           LDW     .D1T1   *A13,A3           ; |33|
           LDW     .D1T1   *+A13(8),A12      ; |35|
           LDW     .D1T1   *+A13(4),A2       ; |34|
           MVK     .S2     0x400,B0          ; |36|

           ZERO    .L2     B7                ;
||         SUB     .D2     B4,8,B5           ;
||         MVC     .S2     CSR,B2

           ZERO    .L1     A7                ;
||         MV      .L2X    A0,B4
||         STW     .D2T1   A11,*+SP(8)       ; |18|
||         AND     .S2     -2,B2,B8

           MVK     .S1     0x1,A1            ; init prolog collapse predicate
||         STW     .D2T1   A10,*+SP(4)       ; |18|
||         SUB     .L2X    A6,8,B6           ;
||         MV      .L1     A4,A10            ;
||         MVC     .S2     B8,CSR            ; interrupts off
||         ZERO    .D1     A4                ;

;*---------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Known Minimum Trip Count         : 1024
;*      Known Maximum Trip Count         : 1024
;*      Known Max Trip Count Factor      : 1024
;*      Loop Carried Dependency Bound(^) : 41
;*      Unpartitioned Resource Bound     : 10
;*      Partitioned Resource Bound(*)    : 14
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                    10        8
;*      .S units                     0        1
;*      .D units                     4        4
;*      .M units                    10       10
;*      .X cross paths               7       14*
;*      .T address paths             5        3
;*      Long read paths              1        1
;*      Long write paths             0        0
;*      Logical  ops (.LS)           0        4     (.L or .S unit)
;*      Addition ops (.LSD)          2        1     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             5        7
;*      Bound(.L .S .D .LS .LSD)     6        6
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 41 Did not find schedule
;*         ii = 42 Schedule found with 2 iterations in parallel
;*      done
;*
;*      Epilog not removed
;*      Collapsed epilog stages     : 0
;*      Collapsed prolog stages     : 1
;*      Minimum required memory pad : 0 bytes
;*
;*      Minimum safe trip count     : 1
```

```
;*----------------------------------------------------------------------------*
L1:     ; PIPED LOOP PROLOG
;** ---------------------------------------------------------------------------*
L2:     ; PIPED LOOP KERNEL
   [!A1]   LDW     .D1T1   *+A10(4),A9       ; |51|

   [!A1]   LDW     .D1T1   *A10,A0           ; |50|
||         MPYSP   .M1     A2,A11,A0         ; |48|
|| [!A1]   ADDSP   .L2X    A0,B7,B7          ;  ^ |48|

           MPYSP   .M1     A2,A6,A5          ; |47|
|| [!A1]   ADDSP   .L1     A0,A4,A4          ; |47|

           NOP             2

   [!A1]   MPYSP   .M1     A12,A9,A4         ; |54|
|| [!A1]   ADDSP   .L2X    A0,B7,B7          ;  ^ |48|

   [!A1]   MPYSP   .M1     A12,A0,A4         ; |53|
||         SUBSP   .L1     A4,A5,A5          ; |47|

           NOP             2

   [!A1]   MPYSP   .M1     A8,A0,A4          ; |54|
|| [!A1]   ADDSP   .L2X    A4,B7,B7          ;  ^ |54|

           MPYSP   .M1     A8,A9,A5          ; |53|
|| [!A1]   ADDSP   .L1     A4,A5,A4          ; |53|

   [!A1]   LDW     .D2T2   *++B5(8),B7       ; |57|
   [!A1]   LDW     .D2T2   *+B5(4),B8        ; |59|
   [!A1]   ADDSP   .L1X    A4,B7,A4          ;  ^ |54|
   [!A1]   SUBSP   .L1     A4,A5,A4          ; |53|
           NOP             2
   [!A1]   SUBSP   .L1X    B8,A4,A4          ;  ^ |59|
           SUBSP   .L2X    B7,A4,B1          ; |57|
   [!A1]   MV      .L2X    A6,B7             ; Define a twin register
           MV      .L2X    A11,B8            ; Define a twin register
           MPYSP   .M2X    B8,A4,B9          ; |64|

           MV      .L2X    A4,B8             ;  ^ Define a twin register
|| [!A1]   MPYSP   .M2     B1,B8,B7          ; |63|

           MPYSP   .M2     B7,B8,B8          ;  ^ |63|

           MPYSP   .M1     A0,A4,A0          ; |69|
|| [!A1]   MPYSP   .M2X    A9,B1,B7          ; |69|

   [!A1]   MPYSP   .M2X    B1,A6,B7          ; |64|

           MPYSP   .M1X    A0,B1,A0          ; |68|
|| [!A1]   MPYSP   .M2X    A9,B8,B7          ; |68|

   [!A1]   ADDSP   .L2     B8,B7,B7          ;  ^ |63|
   [!A1]   SUBSP   .L2X    A0,B7,B7          ; |69|
   [!A1]   SUBSP   .L2     B9,B7,B7          ; |64|
   [!A1]   ADDSP   .L2X    B7,A0,B7          ; |68|
   [!A1]   MPYSP   .M2     B4,B7,B7          ;  ^ |63|

   [!A1]   STW     .D2T2   B1,*++B6(8)       ; |66|
|| [!A1]   MPYSP   .M2     B4,B7,B7          ; |69|

   [!A1]   STW     .D2T1   A4,*+B6(4)        ; |67|
|| [!A1]   MPYSP   .M2     B4,B7,B7          ; |64|

   [!A1]   MPYSP   .M2     B4,B7,B7          ; |68|
||         LDW     .D1T1   *+A10(4),A6       ; @|47|

   [ B0]   SUB     .L2     B0,1,B0           ; |73|
|| [!A1]   ADDSP   .L1X    B7,A3,A3          ;  ^ |63|
||         LDW     .D1T1   *A10++(8),A11     ; @|47|

   [!A1]   ADDSP   .L1X    B7,A8,A8          ; |69|
|| [ B0]   B       .S2     L2                ; |73|
```

```
   [!A1]   ADDSP   .L1X     B7,A2,A2          ; |64|
   [!A1]   ADDSP   .L1X     B7,A12,A12        ; |68|
           MPYSP   .M1      A3,A6,A0          ; @ ^ |48|

           MV      .S1      A7,A0             ;
||         MPYSP   .M1      A3,A11,A0         ; @|47|

   [ A1]   SUB     .S1      A1,1,A1           ;
|| [!A1]   MV      .L2X     A0,B7             ; Define a twin register
|| [!A1]   MV      .D1      A7,A4             ;

;** --------------------------------------------------------------------------*
L3:    ; PIPED LOOP EPILOG
           LDW     .D1T1    *+A10(4),A9       ; (E) @|51|

           LDW     .D1T1    *A10,A4           ; (E) @|50|
||         ADDSP   .L2X     A0,B7,B7          ; (E) @ ^ |48|
||         MPYSP   .M1      A2,A11,A0         ; (E) @|48|

           MPYSP   .M1      A2,A6,A0          ; (E) @|47|
||         ADDSP   .L1      A0,A4,A5          ; (E) @|47|

           NOP             2

           MPYSP   .M1      A12,A9,A0         ; (E) @|54|
||         ADDSP   .L2X     A0,B7,B7          ; (E) @ ^ |48|

           MPYSP   .M1      A12,A4,A0         ; (E) @|53|
||         SUBSP   .L1      A5,A0,A5          ; (E) @|47|

           NOP             2

           MPYSP   .M1      A8,A4,A0          ; (E) @|54|
||         ADDSP   .L2X     A0,B7,B5          ; (E) @ ^ |54|

           MPYSP   .M1      A8,A9,A5          ; (E) @|53|
||         ADDSP   .L1      A0,A5,A0          ; (E) @|53|

           LDW     .D2T2    *++B5(8),B5       ; (E) @|57|
           LDW     .D2T2    *+B5(4),B7        ; (E) @|59|
           ADDSP   .L1X     A0,B5,A0          ; (E) @ ^ |54|
           SUBSP   .L1      A0,A5,A0          ; (E) @|53|
           NOP             2
           SUBSP   .L1X     B7,A0,A0          ; (E) @ ^ |59|
           SUBSP   .L2X     B5,A0,B1          ; (E) @|57|
           MV      .L2X     A6,B7             ; (E) @Define a twin register
           MV      .L2X     A11,B5            ; (E) @Define a twin register
           MPYSP   .M2X     B5,A0,B9          ; (E) @|64|

           MV      .L2X     A0,B5             ; (E) @ ^ Define a twin register
||         MPYSP   .M2      B1,B5,B7          ; (E) @|63|

           MPYSP   .M2      B7,B5,B5          ; (E) @ ^ |63|

           MPYSP   .M1      A4,A0,A4          ; (E) @|69|
||         MPYSP   .M2X     A9,B1,B5          ; (E) @|69|

           MPYSP   .M2X     B1,A6,B5          ; (E) @|64|

           MPYSP   .M2X     A9,B5,B5          ; (E) @|68|
||         MPYSP   .M1X     A4,B1,A4          ; (E) @|68|

           ADDSP   .L2      B5,B7,B5          ; (E) @ ^ |63|
           SUBSP   .L2X     A4,B5,B5          ; (E) @|69|
           SUBSP   .L2      B9,B5,B5          ; (E) @|64|
           ADDSP   .L2X     B5,A4,B5          ; (E) @|68|
           MPYSP   .M2      B4,B5,B4          ; (E) @ ^ |63|

           MPYSP   .M2      B4,B5,B4          ; (E) @|69|
||         STW     .D2T2    B1,*++B6(8)       ; (E) @|66|

           MPYSP   .M2      B4,B5,B4          ; (E) @|64|
||         STW     .D2T1    A0,*+B6(4)        ; (E) @|67|

           LDW     .D2T1    *+SP(8),A11       ; |81|
```

```
||          MPYSP   .M2     B4,B5,B4          ; (E) @|68|

            LDW     .D2T1   *+SP(4),A10       ; |81|
||          ADDSP   .L1X    B4,A3,A3          ; (E) @ ^ |63|

            ADDSP   .L1X    B4,A8,A8          ; (E) @|69|
            ADDSP   .L1X    B4,A2,A2          ; (E) @|64|

            LDW     .D2T1   *+SP(12),A12      ; |81|
||          ADDSP   .L1X    B4,A12,A0         ; (E) @|68|

            STW     .D1T1   A3,*A13           ; |76|

            MVC     .S2     B2,CSR            ; interrupts on
||          STW     .D1T1   A8,*+A13(12)      ; |79|

            STW     .D1T1   A2,*+A13(4)       ; |77|

            STW     .D1T1   A0,*+A13(8)       ; |78|
||          B       .S2     B3                ; |81|

            LDW     .D2T1   *++SP(16),A13     ; |81|
            NOP             4
            ; BRANCH OCCURS                   ; |81|
```

lms_3.c

```
/*
lms_3.c executes the lms adaptive filter algorithm for p=3

x=Reference Data {Note length(x) must be equal to p+N-1 in order to create N output
    samples}
a=Desired signal corrupted by interferer (Primary Channel)
h=filter taps (interleaved real/imaginary).
dhat=filtered output

THESE ARE NOT USED IN THIS VERSION
p=# filter taps (order of filter)       FIXED TO 3
N=2*number output samples for this block of data          FIXED TO 2048
*/

void lms_3(float* x, float* a, float* dhat, float* h, float mu)
{
    unsigned int i=0;
    float v_hat_r=0.0f, v_hat_i=0.0f;
    float tempX_r,tempX_i;
    float tempX_r2,tempX_i2;
    float tempX_r3,tempX_i3;
    float temp_r,temp_i;
    float temp_r1,temp_i1;
    float temp_r2,temp_i2;
    float temp_r3,temp_i3;

    temp_r1=h[0];
    temp_i1=h[1];
    temp_r2=h[2];
    temp_i2=h[3];
    temp_r3=h[4];
    temp_i3=h[5];

    do  {
        tempX_r=x[i];
        tempX_i=x[i+1];

        tempX_r2=x[i+2];
        tempX_i2=x[i+3];

        v_hat_r=v_hat_r+temp_r1*tempX_r-temp_i1*tempX_i;
        v_hat_i=v_hat_i+temp_r1*tempX_i+temp_i1*tempX_r;

        tempX_r3=x[i+4];
        tempX_i3=x[i+5];

        v_hat_r=v_hat_r+temp_r2*tempX_r2-temp_i2*tempX_i2;
```

```
        v_hat_i=v_hat_i+temp_r2*tempX_i2+temp_i2*tempX_r2;

        v_hat_r=v_hat_r+temp_r3*tempX_r3-temp_i3*tempX_i3;
        v_hat_i=v_hat_i+temp_r3*tempX_i3+temp_i3*tempX_r3;

        temp_r=a[i]-v_hat_r;
        v_hat_r=0.0f;
        temp_i=a[i+1]-v_hat_i;
        v_hat_i=0.0f;
        dhat[i]=temp_r;
        dhat[i+1]=temp_i;

        temp_r1=temp_r1+mu*(temp_r*tempX_r+temp_i*tempX_i);
        temp_i1=temp_i1+mu*(temp_i*tempX_r-temp_r*tempX_i);

        tempX_r3=x[i+4];
        tempX_i3=x[i+5];

        temp_r2=temp_r2+mu*(temp_r*tempX_r2+temp_i*tempX_i2);
        temp_i2=temp_i2+mu*(temp_i*tempX_r2-temp_r*tempX_i2);


        temp_r3=temp_r3+mu*(temp_r*tempX_r3+temp_i*tempX_i3);
        temp_i3=temp_i3+mu*(temp_i*tempX_r3-temp_r*tempX_i3);

        i+=2;
        }while (i<2048);

    h[0]=temp_r1;
    h[1]=temp_i1;
    h[2]=temp_r2;
    h[3]=temp_i2;
    h[4]=temp_r3;
    h[5]=temp_i3;
}
```

lms_3_opt.asm

```
;******************************************************************************
;* TMS320C6x ANSI C Codegen                                     Version 4.00 *
;* Date/Time created: Mon Nov 18 15:59:17 2002                                *
;******************************************************************************

;******************************************************************************
;* GLOBAL FILE PARAMETERS                                                     *
;*                                                                            *
;*   Architecture      : TMS320C670x                                          *
;*   Optimization      : Enabled at level 3                                   *
;*   Optimizing for    : Speed                                                *
;*                       Based on options: -o3, no -ms                        *
;*   Endian            : Big                                                  *
;*   Interrupt Thrshld : Disabled                                             *
;*   Memory Model      : Large                                                *
;*   Calls to RTS      : Far                                                  *
;*   Pipelining        : Enabled                                              *
;*   Speculative Load  : Enabled (Threshold = 64                             ) *
;*   Memory Aliases    : Presume are aliases (pessimistic)                    *
;*   Debug Info        : No Debug Info                                        *
;*                                                                            *
;******************************************************************************

;-kq -me -mh64 -ml3 -mv6700 -o3 -frC:\ti\myprojects\adaptive -iC:\ReadyFlow\6216_2.0\
    include -iC:\ReadyFlow\4290_1_2.0\include -iC:\pentek\swiftnet.4.0.1\c6x\include
     -iC:\ti\c6000\cgtools\include -dO

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 C:\TEMP\TI70_2 C:\TEMP\TI70_4 -w C:\ti\myprojects\
    adaptive
        .sect   ".text"
        .global _lms_3
```

```
;******************************************************************************
;* FUNCTION NAME: _lms_3                                                      *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,SP     *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,SP     *
;*   Local Frame Size  : 0 Args + 0 Auto + 36 Save = 36 byte                  *
;******************************************************************************
_lms_3:
;** --------------------------------------------------------------------------*
           STW     .D2T2   B11,*SP--(40)     ; |257|
           STW     .D2T1   A12,*+SP(16)      ; |257|
           STW     .D2T1   A10,*+SP(8)       ; |257|
           STW     .D2T1   A11,*+SP(12)      ; |257|

           MV      .L1X    B6,A0             ;
||         STW     .D2T1   A13,*+SP(20)      ; |257|

           MV      .L2X    A8,B7             ;
||         LDW     .D1T1   *+A0(12),A8       ; |275|

           LDW     .D1T1   *+A0(16),A2       ; |276|
           LDW     .D1T1   *+A0(8),A9        ; |274|
           STW     .D2T1   A15,*+SP(28)      ; |257|
           LDW     .D1T1   *+A0(4),A5        ; |273|

           MV      .L1X    B6,A0             ;
||         LDW     .D1T1   *A0,A3            ; |272|

           LDW     .D1T1   *+A0(20),A15      ; |277|

           MVK     .S2     0x400,B0          ; |277|
||         STW     .D2T2   B3,*+SP(32)       ; |257|

           SUB     .L2X    A6,8,B5           ;
||         STW     .D2T1   A14,*+SP(24)      ; |257|
||         MVC     .S2     CSR,B11

           ZERO    .L1     A7                ;
||         ZERO    .S1     A0                ;
||         STW     .D2T2   B10,*+SP(36)      ; |257|
||         MV      .S2     B6,B10            ;
||         AND     .L2     -2,B11,B8

           MVK     .S1     0x1,A1            ; init prolog collapse predicate
||         SUB     .L1     A4,8,A10          ;
||         SUB     .D2     B4,8,B6           ;
||         ZERO    .D1     A4                ;
||         MVC     .S2     B8,CSR            ; interrupts off
||         MV      .L2X    A3,B4

;*----------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Known Minimum Trip Count         : 1024
;*      Known Maximum Trip Count         : 1024
;*      Known Max Trip Count Factor      : 1024
;*      Loop Carried Dependency Bound(^) : 49
;*      Unpartitioned Resource Bound     : 15
;*      Partitioned Resource Bound(*)    : 15
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                    15*      11
;*      .S units                     0        1
;*      .D units                     8        4
;*      .M units                    15*      15*
;*      .X cross paths               8       15*
;*      .T address paths             8        4
;*      Long read paths              0        2
;*      Long write paths             0        0
;*      Logical  ops (.LS)           4        0     (.L or .S unit)
;*      Addition ops (.LSD)          2        1     (.L or .S or .D unit)
;*      Bound(.L .S .LS)            10        6
```

```
;*      Bound(.L .S .D .LS .LSD)    10        6
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 49 Did not find schedule
;*         ii = 50 Schedule found with 2 iterations in parallel
;*      done
;*
;*      Epilog not removed
;*      Collapsed epilog stages     : 0
;*      Collapsed prolog stages     : 1
;*      Minimum required memory pad : 0 bytes
;*
;*      Minimum safe trip count     : 1
;*----------------------------------------------------------------------------*
L1:    ; PIPED LOOP PROLOG
;** ---------------------------------------------------------------------------*
L2:    ; PIPED LOOP KERNEL
   [!A1]   LDW     .D1T1   *+A10(16),A13     ; |303|

   [!A1]   LDW     .D1T1   *+A10(20),A0      ; |303|
|| [!A1]   MPYSP   .M1     A8,A4,A3          ; |298|
|| [!A1]   ADDSP   .L1     A3,A0,A0          ;  ^ |298|

   [!A1]   MPYSP   .M1     A8,A6,A3          ; |299|
||         ADDSP   .L1     A3,A0,A14         ;  ^ |299|

           NOP             2

           MPYSP   .M1     A2,A13,A14        ; |303|
|| [!A1]   SUBSP   .L1     A0,A3,A3          ;  ^ |298|

   [!A1]   MPYSP   .M1     A2,A0,A0          ; |304|
|| [!A1]   ADDSP   .L1     A3,A14,A3         ;  ^ |299|

           NOP             2

   [!A1]   MPYSP   .M1     A15,A0,A0         ; |303|
|| [!A1]   ADDSP   .L1     A14,A3,A3         ;  ^ |303|

   [!A1]   MPYSP   .M1     A15,A13,A0        ; |304|
|| [!A1]   ADDSP   .L1     A0,A3,A3          ;  ^ |304|

           NOP             1
   [!A1]   LDW     .D2T2   *++B6(8),B8       ; |307|

   [!A1]   LDW     .D2T2   *+B6(4),B8        ; |309|
|| [!A1]   SUBSP   .L1     A3,A0,A0          ;  ^ |303|

   [!A1]   ADDSP   .L1     A0,A3,A0          ;  ^ |304|
           NOP             2
           SUBSP   .L2X    B8,A0,B9          ;  ^ |307|
           SUBSP   .L2X    B8,A0,B3          ;  ^ |309|
           NOP             2
           MPYSP   .M2X    A12,B9,B1         ;  ^ |315|
           MPYSP   .M2X    A11,B3,B8         ;  ^ |315|
           NOP             2

           MPYSP   .M2X    A11,B9,B2         ; |316|
|| [!A1]   STW     .D2T2   B9,*++B5(8)       ; |311|

   [!A1]   STW     .D2T2   B3,*+B5(4)        ; |312|
||         MPYSP   .M2X    A12,B3,B1         ; |316|
||         ADDSP   .L2     B8,B1,B8          ;  ^ |315|

           MPYSP   .M2X    A4,B9,B2          ; |322|
|| [!A1]   LDW     .D1T1   *+A10(16),A0      ; |325|

           MPYSP   .M2X    A6,B9,B1          ; |321|
|| [!A1]   LDW     .D1T1   *+A10(20),A0      ; |325|

           MPYSP   .M2X    A4,B3,B8          ; |321|

           SUBSP   .L2     B1,B2,B9          ; |316|
||         MPYSP   .M2     B7,B8,B8          ;  ^ |315|
```

```
            MPYSP   .M2X    A6,B3,B8          ; |322|

   [!A1]    MPYSP   .M2X    B3,A0,B4          ; |326|
|| [!A1]    MPYSP   .M1X    B9,A0,A3          ; |325|

            ADDSP   .L2     B8,B1,B9          ; |321|
||          MPYSP   .M2X    B9,A0,B8          ; |326|
|| [!A1]    MPYSP   .M1X    B3,A0,A0          ; |325|

            MPYSP   .M2     B7,B9,B8          ; |316|
|| [!A1]    ADDSP   .L2     B8,B4,B4          ;  ^ |315|
||          LDW     .D1T1   *++A10(8),A12     ; @|283|

            LDW     .D1T1   *+A10(4),A11      ; @|284|
            SUBSP   .L2     B8,B2,B8          ; |322|

   [!A1]    ADDSP   .L1     A0,A3,A0          ; |325|
||          SUBSP   .L2     B4,B8,B8          ; |326|

            MPYSP   .M2     B7,B9,B8          ; |321|
||          ADDSP   .L2X    B8,A5,B9          ; |316|
|| [!A1]    MV      .L1X    B4,A3             ;  ^ Define a twin register

            MPYSP   .M1     A3,A12,A3         ; @ ^ |291|

            MPYSP   .M2     B7,B8,B8          ; |322|
||          MPYSP   .M1     A3,A11,A3         ; @ ^ |292|

   [!A1]    MPYSP   .M1X    B7,A0,A0          ; |325|
||          MPYSP   .M2     B7,B8,B9          ; |326|

   [!A1]    MV      .S1     A7,A4             ;
||          ADDSP   .L2X    B8,A9,B8          ; |321|
|| [!A1]    MV      .L1X    B9,A5             ; Define a twin register
||          LDW     .D1T1   *+A10(8),A6       ; @|286|

   [ B0]    SUB     .L2     B0,1,B0           ; |331|
|| [!A1]    MV      .S1     A7,A0             ;
||          LDW     .D1T1   *+A10(12),A4      ; @|287|
||          MPYSP   .M1     A5,A11,A0         ; @|291|
||          ADDSP   .L1     A3,A4,A3          ; @ ^ |291|

   [ B0]    B       .S2     L2                ; |331|
||          MPYSP   .M1     A5,A12,A0         ; @|292|
||          ADDSP   .L1     A3,A0,A3          ; @ ^ |292|

            ADDSP   .L2X    B8,A8,B8          ; |322|
|| [!A1]    ADDSP   .L1X    B9,A15,A15        ; |326|

   [!A1]    MV      .S1X    B8,A9             ; Define a twin register
|| [!A1]    ADDSP   .L1     A0,A2,A2          ; |325|

            MPYSP   .M1     A9,A6,A3          ; @|298|
||          SUBSP   .L1     A3,A0,A0          ; @ ^ |291|

            MPYSP   .M1     A9,A4,A3          ; @|299|
||          ADDSP   .L1     A0,A3,A0          ; @ ^ |292|

   [ A1]    SUB     .D1     A1,1,A1           ;
|| [!A1]    MV      .S1X    B8,A8             ; Define a twin register

;** --------------------------------------------------------------------------*
L3:    ; PIPED LOOP EPILOG

            MV      .S1X    B10,A1
||          LDW     .D1T1   *+A10(16),A13     ; (E) @|303|

            LDW     .D1T1   *+A10(20),A3      ; (E) @|303|
||          ADDSP   .L1     A3,A0,A0          ; (E) @ ^ |298|
||          MPYSP   .M1     A8,A4,A3          ; (E) @|298|

            MPYSP   .M1     A8,A6,A14         ; (E) @|299|
||          ADDSP   .L1     A3,A0,A0          ; (E) @ ^ |299|

            NOP             2
```

```
           MPYSP   .M1     A2,A13,A14        ; (E) @|303|
||         SUBSP   .L1     A0,A3,A0          ; (E) @ ^ |298|

           MPYSP   .M1     A2,A3,A0          ; (E) @|304|
||         ADDSP   .L1     A14,A0,A3         ; (E) @ ^ |299|

           NOP             2

           MPYSP   .M1     A15,A3,A3         ; (E) @|303|
||         ADDSP   .L1     A14,A0,A0         ; (E) @ ^ |303|

           MPYSP   .M1     A15,A13,A0        ; (E) @|304|
||         ADDSP   .L1     A0,A3,A3          ; (E) @ ^ |304|

           NOP             1
           LDW     .D2T2   *++B6(8),B6       ; (E) @|307|

           LDW     .D2T2   *+B6(4),B6        ; (E) @|309|
||         SUBSP   .L1     A0,A3,A0          ; (E) @ ^ |303|

           ADDSP   .L1     A0,A3,A0          ; (E) @ ^ |304|
           NOP             2
           SUBSP   .L2X    B6,A0,B8          ; (E) @ ^ |307|
           SUBSP   .L2X    B6,A0,B6          ; (E) @ ^ |309|
           NOP             2
           MPYSP   .M2X    A12,B8,B9         ; (E) @ ^ |315|
           MPYSP   .M2X    A11,B6,B0         ; (E) @ ^ |315|
           NOP             2

           STW     .D2T2   B8,*++B5(8)       ; (E) @|311|
||         MPYSP   .M2X    A11,B8,B9         ; (E) @|316|

           MPYSP   .M2X    A12,B6,B0         ; (E) @|316|
||         ADDSP   .L2     B0,B9,B5          ; (E) @ ^ |315|
||         STW     .D2T2   B6,*+B5(4)        ; (E) @|312|

           LDW     .D2T2   *+SP(32),B3       ; |340|
||         MPYSP   .M2X    A4,B8,B5          ; (E) @|322|
||         LDW     .D1T1   *+A10(16),A0      ; (E) @|325|

           LDW     .D2T2   *+SP(36),B10      ; |340|
||         MPYSP   .M2X    A6,B8,B9          ; (E) @|321|
||         LDW     .D1T1   *+A10(20),A0      ; (E) @|325|

           LDW     .D2T1   *+SP(8),A10       ; |340|
||         MPYSP   .M2X    A4,B6,B0          ; (E) @|321|

           LDW     .D2T1   *+SP(20),A13      ; |340|
||         SUBSP   .L2     B0,B9,B8          ; (E) @|316|
||         MPYSP   .M2     B7,B5,B6          ; (E) @ ^ |315|

           LDW     .D2T1   *+SP(16),A12      ; |340|
||         MPYSP   .M2X    A6,B6,B6          ; (E) @|322|

           LDW     .D2T1   *+SP(24),A14      ; |340|
||         MPYSP   .M1X    B8,A0,A0          ; (E) @|325|
||         MPYSP   .M2X    B6,A0,B4          ; (E) @|326|

           LDW     .D2T1   *+SP(12),A11      ; |340|
||         MPYSP   .M2X    B8,A0,B5          ; (E) @|326|
||         MPYSP   .M1X    B6,A0,A3          ; (E) @|325|
||         ADDSP   .L2     B0,B9,B6          ; (E) @|321|

           MPYSP   .M2     B7,B8,B5          ; (E) @|316|
||         ADDSP   .L2     B6,B4,B4          ; (E) @ ^ |315|

           NOP             1
           SUBSP   .L2     B6,B5,B4          ; (E) @|322|

           ADDSP   .L1     A3,A0,A0          ; (E) @|325|
||         SUBSP   .L2     B4,B5,B4          ; (E) @|326|

           MV      .L1X    B4,A0             ; (E) @ ^ Define a twin register
||         MPYSP   .M2     B7,B6,B4          ; (E) @|321|
```

```
||          ADDSP   .L2X    B5,A5,B5          ; (E) @|316|

            STW     .D1T1   A0,*A1            ; |334|
            MPYSP   .M2     B7,B4,B4          ; (E) @|322|

            MPYSP   .M2     B7,B4,B5          ; (E) @|326|
||          MPYSP   .M1X    B7,A0,A0          ; (E) @|325|

            MV      .S1     A7,A4             ; (E) @
||          MV      .L1X    B5,A5             ; (E) @Define a twin register
||          ADDSP   .L2X    B4,A9,B4          ; (E) @|321|

            STW     .D1T1   A5,*+A1(4)        ; |335|
||          MV      .L1     A7,A0             ; (E) @

            NOP             1

            LDW     .D2T1   *+SP(28),A15      ; |340|
||          ADDSP   .L1X    B5,A15,A0         ; (E) @|326|
||          ADDSP   .L2X    B4,A8,B4          ; (E) @|322|

            MV      .S1X    B4,A9             ; (E) @Define a twin register
||          ADDSP   .L1     A0,A2,A2          ; (E) @|325|

            STW     .D1T1   A9,*+A1(8)        ; |336|
            B       .S2     B3                ; |340|

            STW     .D1T1   A0,*+A1(20)       ; |339|
||          MV      .S1X    B4,A8             ; (E) @Define a twin register
||          LDW     .D2T2   *++SP(40),B11     ; |340|
||          MVC     .S2     B11,CSR           ; interrupts on

            STW     .D1T1   A2,*+A1(16)       ; |338|
            STW     .D1T1   A8,*+A1(12)       ; |337|
            NOP             2
            ; BRANCH OCCURS                   ; |340|
```

lms_4.c

```
/*
lms_4.c executes the lms adaptive filter algorithm for p=4

x=Reference Data {Note length(x) must be equal to p+N-1 in order to create N output
    samples}
a=Desired signal corrupted by interferer (Primary Channel)
h=filter taps (interleaved real/imaginary).
dhat=filtered output

THESE ARE NOT USED IN THIS VERSION
p=# filter taps (order of filter)       FIXED TO 4
N=2*number output samples for this block of data          FIXED TO 2048
*/

void lms_4(float* x, float* a, float* dhat, float* h, float mu)
{
    unsigned int i=0;
    float v_hat_r=0.0f, v_hat_i=0.0f;
    float tempX_r,tempX_i;
    float tempX_r2,tempX_i2;
    float tempX_r3,tempX_i3;
    float tempX_r4,tempX_i4;
    float temp_r,temp_i;
    float temp_r1,temp_i1;
    float temp_r2,temp_i2;
    float temp_r3,temp_i3;
    float temp_r4,temp_i4;

    volatile float* x_vol=(volatile float*)x;
    temp_r1=h[0];
    temp_i1=h[1];
    temp_r2=h[2];
    temp_i2=h[3];
    temp_r3=h[4];
    temp_i3=h[5];
```

```
    temp_r4=h[6];
    temp_i4=h[7];

    do {
        tempX_r=x[i];
        tempX_i=x[i+1];

        tempX_r2=x[i+2];
        tempX_i2=x[i+3];

        v_hat_r=v_hat_r+temp_r1*tempX_r-temp_i1*tempX_i;
        v_hat_i=v_hat_i+temp_r1*tempX_i+temp_i1*tempX_r;

        tempX_r3=x[i+4];
        tempX_i3=x[i+5];

        v_hat_r=v_hat_r+temp_r2*tempX_r2-temp_i2*tempX_i2;
        v_hat_i=v_hat_i+temp_r2*tempX_i2+temp_i2*tempX_r2;

        tempX_r4=x[i+6];
        tempX_i4=x[i+7];

        v_hat_r=v_hat_r+temp_r3*tempX_r3-temp_i3*tempX_i3;
        v_hat_i=v_hat_i+temp_r3*tempX_i3+temp_i3*tempX_r3;

        v_hat_r=v_hat_r+temp_r4*tempX_r4-temp_i4*tempX_i4;
        v_hat_i=v_hat_i+temp_r4*tempX_i4+temp_i4*tempX_r4;

        temp_r=a[i]-v_hat_r;
        v_hat_r=0.0f;
        temp_i=a[i+1]-v_hat_i;
        v_hat_i=0.0f;
        dhat[i]=temp_r;
        dhat[i+1]=temp_i;

        tempX_r=x[i];
        tempX_i=x[i+1];

        tempX_r2=x[i+2];
        tempX_i2=x[i+3];

        temp_r1=temp_r1+mu*(temp_r*tempX_r+temp_i*tempX_i);
        temp_i1=temp_i1+mu*(temp_i*tempX_r-temp_r*tempX_i);

        temp_r2=temp_r2+mu*(temp_r*tempX_r2+temp_i*tempX_i2);
        temp_i2=temp_i2+mu*(temp_i*tempX_r2-temp_r*tempX_i2);

        temp_r3=temp_r3+mu*(temp_r*tempX_r3+temp_i*tempX_i3);
        temp_i3=temp_i3+mu*(temp_i*tempX_r3-temp_r*tempX_i3);

        temp_r4=temp_r4+mu*(temp_r*tempX_r4+temp_i*tempX_i4);
        temp_i4=temp_i4+mu*(temp_i*tempX_r4-temp_r*tempX_i4);

        i+=2;
    }while (i<2048);

    h[0]=temp_r1;
    h[1]=temp_i1;
    h[2]=temp_r2;
    h[3]=temp_i2;
    h[4]=temp_r3;
    h[5]=temp_i3;
    h[6]=temp_r4;
    h[7]=temp_i4;
}
```

lms_4_opt.asm

```
;******************************************************************************
;* TMS320C6x ANSI C Codegen                                     Version 4.00 *
;* Date/Time created: Mon Nov 18 18:08:29 2002                               *
;******************************************************************************

;******************************************************************************
```

```
;* GLOBAL FILE PARAMETERS                                                *
;*                                                                       *
;*   Architecture      : TMS320C670x                                     *
;*   Optimization      : Enabled at level 3                              *
;*   Optimizing for    : Speed                                           *
;*                       Based on options: -o3, no -ms                   *
;*   Endian            : Big                                             *
;*   Interrupt Thrshld : Disabled                                        *
;*   Memory Model      : Large                                           *
;*   Calls to RTS      : Far                                             *
;*   Pipelining        : Enabled                                         *
;*   Speculative Load  : Enabled (Threshold = 64                        ) *
;*   Memory Aliases    : Presume are aliases (pessimistic)               *
;*   Debug Info        : No Debug Info                                   *
;*                                                                       *
;************************************************************************

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 C:\TEMP\TI243_2 C:\TEMP\TI243_4 -w C:\ti\myprojects\
    adaptive
        .sect   ".text"
        .global _lms_4

;************************************************************************
;* FUNCTION NAME: _lms_4                                                *
;*                                                                      *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                     *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                     *
;*   Local Frame Size  : 0 Args + 8 Auto + 44 Save = 52 byte              *
;************************************************************************
_lms_4:
;** --------------------------------------------------------------------*

           STW     .D2T2   B13,*SP--(56)     ; |307|
||         MVC     .S2     CSR,B5

           STW     .D2T2   B5,*+SP(8)
           STW     .D2T1   A12,*+SP(24)      ; |307|
           STW     .D2T1   A10,*+SP(16)      ; |307|
           STW     .D2T1   A11,*+SP(20)      ; |307|
           STW     .D2T1   A13,*+SP(28)      ; |307|
           STW     .D2T2   B3,*+SP(40)       ; |307|
           STW     .D2T1   A14,*+SP(32)      ; |307|
           STW     .D2T1   A15,*+SP(36)      ; |307|
           STW     .D2T2   B10,*+SP(44)      ; |307|
           STW     .D2T2   B6,*+SP(4)
           LDW     .D2T1   *+SP(4),A0
           STW     .D2T2   B12,*+SP(52)      ; |307|

           STW     .D2T2   B11,*+SP(48)      ; |307|
||         MV      .L1     A4,A14            ;

           LDW     .D1T1   *+A14(4),A7       ; (P) |342|
           LDW     .D1T1   *A14,A13          ; (P) |342|
           LDW     .D1T1   *A0,A11           ; |322|
           LDW     .D1T1   *+A0(24),A10      ; |328|
           LDW     .D1T1   *+A0(28),A2       ; |329|
           LDW     .D1T1   *+A0(20),A1       ; |327|

           AND     .L2     -2,B5,B5
||         LDW     .D1T1   *+A0(4),A9        ; |323|

           MVC     .S2     B5,CSR            ; interrupts off
||         LDW     .D1T1   *+A0(8),A5        ; |324|
||         MPYSP   .M1     A11,A13,A12       ; (P)  ^ |342|

;*--------------------------------------------------------------------*
```

```
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Known Minimum Trip Count         : 1024
;*      Known Maximum Trip Count         : 1024
;*      Known Max Trip Count Factor      : 1024
;*      Loop Carried Dependency Bound(^) : 63
;*      Unpartitioned Resource Bound     : 20
;*      Partitioned Resource Bound(*)    : 23
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                    20       14
;*      .S units                     0        1
;*      .D units                    12        4
;*      .M units                    20       20
;*      .X cross paths              11       23*
;*      .T address paths            13        3
;*      Long read paths              1        1
;*      Long write paths             0        0
;*      Logical  ops (.LS)           0        3     (.L or .S unit)
;*      Addition ops (.LSD)          2        1     (.L or .S or .D unit)
;*      Bound(.L .S .LS)            10        9
;*      Bound(.L .S .D .LS .LSD)    12        8
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 63 Did not find schedule
;*         ii = 64 Schedule found with 2 iterations in parallel
;*      done
;*
;*      Epilog not removed
;*      Collapsed epilog stages     : 0
;*
;*      Prolog not removed
;*      Collapsed prolog stages     : 0
;*
;*      Minimum required memory pad : 0 bytes
;*
;*      Minimum safe trip count     : 2
;*----------------------------------------------------------------------------*
L1:    ; PIPED LOOP PROLOG

           LDW     .D1T1   *+A0(16),A4       ; |326|
||         MPYSP   .M1     A11,A7,A0         ; (P) |343|

           LDW     .D1T1   *+A0(12),A3       ; |325|

           ZERO    .L1     A0                ;
||         LDW     .D1T1   *+A14(8),A0       ; (P) |349|

           ZERO    .S2     B5                ;
||         MV      .L2X    A8,B6             ;
||         ADDSP   .L1     A12,A0,A8         ; (P)  ^ |342|
||         LDW     .D1T1   *+A14(12),A7      ; (P) |349|
||         MPYSP   .M1     A9,A7,A7          ; (P) |342|

           MPYSP   .M1     A9,A13,A8         ; (P) |343|
||         ADDSP   .L2X    A0,B5,B7          ; (P) |343|

           SUB     .D2     B4,8,B4           ;
||         SUB     .L2X    A6,8,B5           ;
||         MVK     .S2     1023,B0           ;
||         ZERO    .L1     A6                ;

;** ---------------------------------------------------------------------------*
L2:    ; PIPED LOOP KERNEL
           NOP             1

           MPYSP   .M1     A5,A0,A8          ; |349|
||         SUBSP   .L1     A8,A7,A12         ;  ^ |342|

           MPYSP   .M1     A5,A7,A7          ; |350|
||         ADDSP   .L2X    A8,B7,B7          ; |343|

           NOP             1
           LDW     .D1T1   *+A14(16),A7      ; |346|
```

```
           LDW     .D1T1   *+A14(20),A12     ; |347|
||         MPYSP   .M1     A3,A7,A0          ; |349|
||         ADDSP   .L1     A8,A12,A8         ;  ^ |349|

           MPYSP   .M1     A3,A0,A0          ; |350|
||         ADDSP   .L2X    A7,B7,B7          ; |350|

           NOP             2

           MPYSP   .M1     A4,A7,A0          ; |355|
||         SUBSP   .L1     A8,A0,A8          ;  ^ |349|

           MPYSP   .M1     A4,A12,A0         ; |356|
||         ADDSP   .L2X    A0,B7,B7          ; |350|

           NOP             1
           LDW     .D1T1   *+A14(24),A13     ; |352|

           LDW     .D1T1   *+A14(28),A8      ; |353|
||         MPYSP   .M1     A1,A12,A0         ; |355|
||         ADDSP   .L1     A0,A8,A8          ;  ^ |355|

           MPYSP   .M1     A1,A7,A0          ; |356|
||         ADDSP   .L2X    A0,B7,B7          ; |356|

           NOP             2

           MPYSP   .M1     A10,A13,A0        ; |358|
||         SUBSP   .L1     A8,A0,A15         ;  ^ |355|

           MPYSP   .M1     A10,A8,A0         ; |359|
||         ADDSP   .L2X    A0,B7,B7          ; |356|

           NOP             2

           MPYSP   .M1     A2,A8,A15         ; |358|
||         ADDSP   .L1     A0,A15,A0         ;  ^ |358|

           MPYSP   .M1     A2,A13,A0         ; |359|
||         ADDSP   .L1X    A0,B7,A15         ; |359|

           NOP             1
           LDW     .D2T2   *++B4(8),B7       ; |362|

           LDW     .D2T2   *+B4(4),B7        ; |364|
||         SUBSP   .L1     A0,A15,A0         ;  ^ |358|

           ADDSP   .L1     A0,A15,A0         ; |359|
           NOP             2
           SUBSP   .L2X    B7,A0,B8          ;  ^ |362|
           SUBSP   .L1X    B7,A0,A0          ; |364|
           NOP             2

           MPYSP   .M2X    A7,B8,B11         ; |385|
||         STW     .D2T2   B8,*++B5(8)       ;  ^ |366|

           MPYSP   .M2X    A12,B8,B7         ; |386|
||         STW     .D2T1   A0,*+B5(4)        ;  ^ |367|

           MPYSP   .M1     A13,A0,A7         ; |389|
||         MPYSP   .M2X    A8,B8,B9          ; |389|
||         LDW     .D1T1   *A14,A12          ;  ^ |375|

           MV      .L2X    A0,B1             ; Define a twin register
||         LDW     .D1T1   *+A14(4),A7       ;  ^ |375|

           MPYSP   .M2X    A12,B1,B13        ; |385|
||         LDW     .D1T1   *+A14(12),A0      ; |381|

           MPYSP   .M2X    A7,B1,B9          ; |386|
           SUBSP   .L2X    A7,B9,B2          ; |389|
           MPYSP   .M2X    B8,A12,B3         ;  ^ |375|

           MPYSP   .M1     A8,A0,A0          ; |388|
||         LDW     .D1T1   *++A14(8),A8      ; |381|
```

```
||          MPYSP   .M2X    B1,A7,B10         ;  ^ |375|

            MPYSP   .M1X    A13,B8,A7         ; |388|
||          MV      .S2X    A0,B12            ; Define a twin register

            MPYSP   .M2X    B8,A0,B3          ; |382|
            MPYSP   .M2X    B8,A7,B8          ; |376|

            MPYSP   .M2X    B1,A12,B9         ; |376|
||          ADDSP   .L2     B10,B3,B7         ;  ^ |375|

            ADDSP   .L1     A0,A7,A0          ; |388|
||          MPYSP   .M2X    B8,A8,B7          ; |381|

            MPYSP   .M2X    B1,A8,B8          ; |382|

            SUBSP   .L2     B9,B7,B9          ; |386|
||          MPYSP   .M2     B1,B12,B8         ; |381|

            SUBSP   .L2     B9,B8,B8          ; |376|
||          MPYSP   .M2     B6,B7,B7          ;  ^ |375|

            ADDSP   .L2     B13,B11,B7        ; |385|
||          MPYSP   .M1X    B6,A0,A0          ; |388|

            MPYSP   .M2     B6,B2,B8          ; |389|
||          SUBSP   .L2     B8,B3,B7          ; |382|

            MPYSP   .M2     B6,B9,B8          ; |386|
||          ADDSP   .L2     B8,B7,B7          ; |381|
||          LDW     .D1T1   *A14,A0           ; @|342|

            MPYSP   .M2     B6,B8,B7          ; |376|
||          ADDSP   .L1X    B7,A11,A11        ;  ^ |375|
||          LDW     .D1T1   *+A14(4),A7       ; @|342|

            MPYSP   .M2     B6,B7,B7          ; |385|
||          ADDSP   .L1     A0,A10,A10        ; |388|

            ADDSP   .L1X    B8,A2,A2          ; |389|
||          MPYSP   .M2     B6,B7,B7          ; |382|

            ADDSP   .L1X    B8,A1,A1          ; |386|
||          MPYSP   .M2     B6,B7,B7          ; |381|

   [ B0]    SUB     .L2     B0,1,B0           ; |392|
||          ADDSP   .L1X    B7,A9,A9          ; |376|
||          MPYSP   .M1     A11,A0,A12        ; @ ^ |342|

   [ B0]    B       .S2     L2                ; |392|
||          ADDSP   .L1X    B7,A4,A4          ; |385|
||          MPYSP   .M1     A11,A7,A7         ; @|343|

            MV      .S1     A6,A3             ;
||          ADDSP   .L1X    B7,A3,A3          ; |382|

            MV      .S1     A6,A8             ;
||          ADDSP   .L1X    B7,A5,A5          ; |381|
||          LDW     .D1T1   *+A14(8),A0       ; @|349|

            MV      .L2X    A3,B7             ; Define a twin register
||          LDW     .D1T1   *+A14(12),A7      ; @|349|
||          MPYSP   .M1     A9,A7,A7          ; @|342|
||          ADDSP   .L1     A12,A8,A8         ; @ ^ |342|

            MPYSP   .M1     A9,A0,A8          ; @|343|
||          ADDSP   .L2X    A7,B7,B7          ; @|343|

            NOP             1
;** --------------------------------------------------------------------------*
L3:    ; PIPED LOOP EPILOG
            NOP             1

            MPYSP   .M1     A5,A0,A8          ; (E) @|349|
||          SUBSP   .L1     A8,A7,A12         ; (E) @ ^ |342|
```

```
           MPYSP   .M1     A5,A7,A7          ; (E) @|350|
||         ADDSP   .L2X    A8,B7,B7          ; (E) @|343|

           NOP             1
           LDW     .D1T1   *+A14(16),A7      ; (E) @|346|

           LDW     .D1T1   *+A14(20),A12     ; (E) @|347|
||         ADDSP   .L1     A8,A12,A8         ; (E) @ ^ |349|
||         MPYSP   .M1     A3,A7,A0          ; (E) @|349|

           MPYSP   .M1     A3,A0,A0          ; (E) @|350|
||         ADDSP   .L2X    A7,B7,B7          ; (E) @|350|

           NOP             2

           MPYSP   .M1     A4,A7,A0          ; (E) @|355|
||         SUBSP   .L1     A8,A0,A8          ; (E) @ ^ |349|

           MPYSP   .M1     A4,A12,A0         ; (E) @|356|
||         ADDSP   .L2X    A0,B7,B7          ; (E) @|350|

           NOP             1
           LDW     .D1T1   *+A14(24),A13     ; (E) @|352|

           ADDSP   .L1     A0,A8,A8          ; (E) @ ^ |355|
||         LDW     .D1T1   *+A14(28),A8      ; (E) @|353|
||         MPYSP   .M1     A1,A12,A0         ; (E) @|355|

           ADDSP   .L2X    A0,B7,B7          ; (E) @|356|
||         MPYSP   .M1     A1,A7,A0          ; (E) @|356|

           NOP             2

           SUBSP   .L1     A8,A0,A15         ; (E) @ ^ |355|
||         MPYSP   .M1     A10,A13,A0        ; (E) @|358|

           ADDSP   .L2X    A0,B7,B7          ; (E) @|356|
||         MPYSP   .M1     A10,A8,A0         ; (E) @|359|

           NOP             2

           ADDSP   .L1     A0,A15,A0         ; (E) @ ^ |358|
||         MPYSP   .M1     A2,A8,A15         ; (E) @|358|

           ADDSP   .L1X    A0,B7,A15         ; (E) @|359|
||         MPYSP   .M1     A2,A13,A0         ; (E) @|359|

           NOP             1
           LDW     .D2T2   *++B4(8),B7       ; (E) @|362|

           SUBSP   .L1     A0,A15,A0         ; (E) @ ^ |358|
||         LDW     .D2T2   *+B4(4),B7        ; (E) @|364|

           ADDSP   .L1     A0,A15,A0         ; (E) @|359|
           NOP             2
           SUBSP   .L2X    B7,A0,B8          ; (E) @ ^ |362|
           SUBSP   .L1X    B7,A0,A0          ; (E) @|364|
           NOP             2

           STW     .D2T2   B8,*++B5(8)       ; (E) @ ^ |366|
||         MPYSP   .M2X    A7,B8,B11         ; (E) @|385|

           STW     .D2T1   A0,*+B5(4)        ; (E) @ ^ |367|
||         MPYSP   .M2X    A12,B8,B7         ; (E) @|386|

           LDW     .D2T2   *+SP(8),B4
||         MPYSP   .M2X    A8,B8,B9          ; (E) @|389|
||         LDW     .D1T1   *A14,A12          ; (E) @ ^ |375|
||         MPYSP   .M1     A13,A0,A7         ; (E) @|389|

           MV      .L2X    A0,B1             ; (E) @Define a twin register
||         LDW     .D1T1   *+A14(4),A7       ; (E) @ ^ |375|

           MPYSP   .M2X    A12,B1,B13        ; (E) @|385|
```

```
||          LDW     .D1T1   *+A14(12),A0      ; (E) @|381|

            LDW     .D2T1   *+SP(36),A15      ; |403|
||          MPYSP   .M2X    A7,B1,B9          ; (E) @|386|

            SUBSP   .L2X    A7,B9,B2          ; (E) @|389|
            MPYSP   .M2X    B8,A12,B3         ; (E) @ ^ |375|

            MPYSP   .M2X    B1,A7,B10         ; (E) @ ^ |375|
||          LDW     .D1T1   *++A14(8),A8      ; (E) @|381|
||          MPYSP   .M1     A8,A0,A0          ; (E) @|388|

            LDW     .D2T1   *+SP(28),A13      ; |403|
||          MV      .S2X    A0,B12            ; (E) @Define a twin register
||          MPYSP   .M1X    A13,B8,A7         ; (E) @|388|

            LDW     .D2T1   *+SP(32),A14      ; |403|
||          MPYSP   .M2X    B8,A0,B3          ; (E) @|382|

            MPYSP   .M2X    B8,A7,B8          ; (E) @|376|

            LDW     .D2T1   *+SP(24),A12      ; |403|
||          ADDSP   .L2     B10,B3,B7         ; (E) @ ^ |375|
||          MPYSP   .M2X    B1,A12,B9         ; (E) @|376|

            LDW     .D2T2   *+SP(44),B10      ; |403|
||          MPYSP   .M2X    B8,A8,B7          ; (E) @|381|
||          ADDSP   .L1     A0,A7,A0          ; (E) @|388|

            MPYSP   .M2X    B1,A8,B8          ; (E) @|382|

            LDW     .D2T2   *+SP(52),B12      ; |403|
||          SUBSP   .L2     B9,B7,B9          ; (E) @|386|
||          MPYSP   .M2     B1,B12,B8         ; (E) @|381|

            SUBSP   .L2     B9,B8,B8          ; (E) @|376|
||          MPYSP   .M2     B6,B7,B7          ; (E) @ ^ |375|

            LDW     .D2T2   *+SP(48),B11      ; |403|
||          ADDSP   .L2     B13,B11,B7        ; (E) @|385|
||          MPYSP   .M1X    B6,A0,A0          ; (E) @|388|

            LDW     .D2T2   *+SP(40),B3       ; |403|
||          MPYSP   .M2     B6,B2,B8          ; (E) @|389|
||          SUBSP   .L2     B8,B3,B7          ; (E) @|382|

            MPYSP   .M2     B6,B9,B8          ; (E) @|386|
||          ADDSP   .L2     B8,B7,B7          ; (E) @|381|

            MPYSP   .M2     B6,B8,B7          ; (E) @|376|
||          ADDSP   .L1X    B7,A11,A11        ; (E) @ ^ |375|

            LDW     .D2T1   *+SP(4),A0
||          MPYSP   .M2     B6,B7,B7          ; (E) @|385|
||          ADDSP   .L1     A0,A10,A10        ; (E) @|388|

            ADDSP   .L1X    B8,A2,A2          ; (E) @|389|
||          MPYSP   .M2     B6,B7,B7          ; (E) @|382|

            ADDSP   .L1X    B8,A1,A1          ; (E) @|386|
||          MPYSP   .M2     B6,B7,B7          ; (E) @|381|

            ADDSP   .L1X    B7,A9,A9          ; (E) @|376|
            ADDSP   .L1X    B7,A4,A4          ; (E) @|385|

            STW     .D1T1   A2,*+A0(28)       ; |402|
||          MV      .S1     A6,A3             ; (E) @
||          ADDSP   .L1X    B7,A3,A3          ; (E) @|382|

            STW     .D1T1   A11,*A0           ; |395|
||          MV      .S1     A6,A8             ; (E) @
||          ADDSP   .L1X    B7,A5,A5          ; (E) @|381|

            STW     .D1T1   A10,*+A0(24)      ; |401|
||          MV      .L2X    A3,B7             ; (E) @Define a twin register
```

```
           LDW     .D2T1   *+SP(16),A10       ; |403|

           LDW     .D2T1   *+SP(20),A11       ; |403|
||         MVC     .S2     B4,CSR             ; interrupts on

           B       .S2     B3                 ; |403|
||         LDW     .D2T2   *++SP(56),B13      ; |403|
||         STW     .D1T1   A5,*+A0(8)         ; |397|

           STW     .D1T1   A4,*+A0(16)        ; |399|
           STW     .D1T1   A9,*+A0(4)         ; |396|
           STW     .D1T1   A1,*+A0(20)        ; |400|
           STW     .D1T1   A3,*+A0(12)        ; |398|
           NOP             1
           ; BRANCH OCCURS                    ; |403|
```

lms_5.c

```
/*
lms_5.c executes the lms adaptive filter algorithm for p=5

x=Reference Data {Note length(x) must be equal to p+N-1 in order to create N output
    samples}
a=Desired signal corrupted by interferer (Primary Channel)
h=filter taps (interleaved real/imaginary).
dhat=filtered output

THESE ARE NOT USED IN THIS VERSION
p=# filter taps (order of filter)       FIXED TO 5
N=2*number output samples for this block of data          FIXED TO 2048
*/

void lms_5(float* x, float* a, float* dhat, float* h, float mu)
{
    int i=0;
    float v_hat_r=0.0f, v_hat_i=0.0f;
    float tempX_r,tempX_i;
    float tempX_r2,tempX_i2;
    float tempX_r3,tempX_i3;
    float tempX_r4,tempX_i4;
    float tempX_r5,tempX_i5;
    float temp_r,temp_i;
    float temp_r2,temp_i2;
    float temp_r3,temp_i3;
    float temp_r4,temp_i4;
    float temp_r5,temp_i5;
    volatile float* x_vol=(volatile float*)x;
    do  {
        temp_r=h[0];
        temp_i=h[1];
        tempX_r=x_vol[i];
        tempX_i=x_vol[i+1];

        temp_r2=h[2];
        temp_i2=h[3];
        tempX_r2=x_vol[i+2];
        tempX_i2=x_vol[i+3];

        temp_r3=h[4];
        temp_i3=h[5];
        tempX_r3=x_vol[i+4];
        tempX_i3=x_vol[i+5];

        temp_r4=h[6];
        temp_i4=h[7];
        tempX_r4=x_vol[i+6];
        tempX_i4=x_vol[i+7];

        temp_r5=h[8];
        temp_i5=h[9];
        tempX_r5=x_vol[i+8];
        tempX_i5=x_vol[i+9];
```

```
        v_hat_r=v_hat_r+temp_r*tempX_r-temp_i*tempX_i;
        v_hat_i=v_hat_i+temp_r*tempX_i+temp_i*tempX_r;

        v_hat_r=v_hat_r+temp_r2*tempX_r2-temp_i2*tempX_i2;
        v_hat_i=v_hat_i+temp_r2*tempX_i2+temp_i2*tempX_r2;

        v_hat_r=v_hat_r+temp_r3*tempX_r3-temp_i3*tempX_i3;
        v_hat_i=v_hat_i+temp_r3*tempX_i3+temp_i3*tempX_r3;

        v_hat_r=v_hat_r+temp_r4*tempX_r4-temp_i4*tempX_i4;
        v_hat_i=v_hat_i+temp_r4*tempX_i4+temp_i4*tempX_r4;

        v_hat_r=v_hat_r+temp_r5*tempX_r5-temp_i5*tempX_i5;
        v_hat_i=v_hat_i+temp_r5*tempX_i5+temp_i5*tempX_r5;

        temp_r=a[i]-v_hat_r;
        temp_i=a[i+1]-v_hat_i;

        v_hat_r=0.0f;
        dhat[i]=temp_r;
        v_hat_i=0.0f;
        dhat[i+1]=temp_i;

        tempX_r=x_vol[i];
        tempX_i=x_vol[i+1];

        tempX_r2=x[i+2];
        tempX_i2=x[i+3];

        tempX_r3=x_vol[i+4];
        tempX_i3=x_vol[i+5];

        tempX_r4=x_vol[i+6];
        tempX_i4=x_vol[i+7];

        h[0]=h[0]+mu*(temp_r*tempX_r+temp_i*tempX_i);
        h[1]=h[1]+mu*(temp_i*tempX_r-temp_r*tempX_i);
        h[2]=h[2]+mu*(temp_r*tempX_r2+temp_i*tempX_i2);
        h[3]=h[3]+mu*(temp_i*tempX_r2-temp_r*tempX_i2);
        h[4]=h[4]+mu*(temp_r*tempX_r3+temp_i*tempX_i3);
        h[5]=h[5]+mu*(temp_i*tempX_r3-temp_r*tempX_i3);

        tempX_r=x[i+8];
        tempX_i=x[i+9];

        h[6]=h[6]+mu*(temp_r*tempX_r4+temp_i*tempX_i4);
        h[7]=h[7]+mu*(temp_i*tempX_r4-temp_r*tempX_i4);
        h[8]=h[8]+mu*(temp_r*tempX_r+temp_i*tempX_i);
        h[9]=h[9]+mu*(temp_i*tempX_r-temp_r*tempX_i);

        i+=2;
    }while (i<2048);
}
```

lms_5_opt.asm

```
;******************************************************************************
;* TMS320C6x ANSI C Codegen                                     Version 4.00 *
;* Date/Time created: Thu Sep 26 23:05:49 2002                                *
;******************************************************************************

;******************************************************************************
;* GLOBAL FILE PARAMETERS                                                     *
;*                                                                            *
;*   Architecture      : TMS320C670x                                          *
;*   Optimization      : Enabled at level 3                                   *
;*   Optimizing for    : Speed                                                *
;*                       Based on options: -o3, no -ms                        *
;*   Endian            : Big                                                  *
;*   Interrupt Thrshld : Disabled                                             *
;*   Memory Model      : Large                                                *
;*   Calls to RTS      : Far                                                  *
;*   Pipelining        : Enabled                                              *
;*   Speculative Load  : Enabled (Threshold = 64                             ) *
```

```
;*   Memory Aliases    : Presume are aliases (pessimistic)                   *
;*   Debug Info        : No Debug Info                                       *
;*                                                                           *
;*****************************************************************************

;-kq -me -mh64 -ml3 -mv6700 -o3 -frC:\ti\myprojects\adaptive -iC:\ReadyFlow\6216_2.0\
    include -iC:\ReadyFlow\4290_1_2.0\include -iC:\pentek\swiftnet.4.0.1\c6x\include
     -iC:\ti\c6000\cgtools\include -dO

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 C:\TEMP\TI235_2 C:\TEMP\TI235_4 -w C:\ti\myprojects\
    adaptive
        .sect   ".text"
        .global _lms_5

;*****************************************************************************
;* FUNCTION NAME: _lms_5                                                     *
;*                                                                           *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,B0,*
;*                       B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,SP           *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,B0,*
;*                       B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,SP           *
;*   Local Frame Size  : 0 Args + 0 Auto + 36 Save = 36 byte                 *
;*****************************************************************************
_lms_5:
;** --------------------------------------------------------------------------*
           STW     .D2T2   B12,*SP--(40)     ; |14|
           STW     .D2T1   A10,*+SP(8)       ; |14|
           STW     .D2T1   A13,*+SP(20)      ; |14|
           STW     .D2T1   A11,*+SP(12)      ; |14|
           STW     .D2T1   A12,*+SP(16)      ; |14|

           MV      .L2     B6,B2
||         STW     .D2T1   A14,*+SP(24)      ; |14|

           ZERO    .L2     B1                ;
||         ZERO    .S2     B9                ;
||         STW     .D2T2   B11,*+SP(36)      ; |14|

           SUB     .S1     A4,8,A7           ;
||         MV      .L1X    B6,A9
||         SUB     .L2     B4,8,B8           ;
||         MVK     .S2     0x400,B4          ; |15|
||         STW     .D2T2   B3,*+SP(28)       ; |14|

           MV      .L2X    A8,B0             ;
||         MVK     .S1     0x1,A2            ; init prolog collapse predicate
||         SUB     .D1     A6,8,A8           ;
||         STW     .D2T2   B10,*+SP(32)      ; |14|
||         ZERO    .S2     B11               ;
||         MV      .L1X    B4,A1

;*--------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Known Minimum Trip Count         : 1024
;*      Known Maximum Trip Count         : 1024
;*      Known Max Trip Count Factor      : 1024
;*      Loop Carried Dependency Bound(^) : 73
;*      Unpartitioned Resource Bound     : 27
;*      Partitioned Resource Bound(*)    : 29
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                    16       26
;*      .S units                     1        0
;*      .D units                    27       27
;*      .M units                    24       26
;*      .X cross paths              22       29*
;*      .T address paths            27       27
;*      Long read paths              5        7
;*      Long write paths             0        0
```

```
;*      Logical  ops (.LS)           4        2     (.L or .S unit)
;*      Addition ops (.LSD)          2        4     (.L or .S or .D unit)
;*      Bound(.L .S .LS)            11       14
;*      Bound(.L .S .D .LS .LSD)    17       20
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 73 Schedule found with 2 iterations in parallel
;*      done
;*
;*      Loop is interruptible
;*      Epilog not removed
;*      Collapsed epilog stages     : 0
;*      Collapsed prolog stages     : 1
;*      Minimum required memory pad : 0 bytes
;*
;*      Minimum safe trip count     : 1
;*--------------------------------------------------------------------------*
L1:    ; PIPED LOOP PROLOG
;** --------------------------------------------------------------------------*
L2:    ; PIPED LOOP KERNEL
           NOP             1

           MPYSP   .M2X    A3,B5,B5          ; |70|
||         ADDSP   .L1     A0,A5,A0          ; |70|

   [!A2]   LDW     .D2T2   *++B8(8),B6       ; |72|
           SUBSP   .L2X    B4,A4,B7          ;  ^ |69|
   [!A2]   LDW     .D2T2   *+B8(4),B4        ; |73|
           ADDSP   .L2X    B5,A0,B5          ; |70|
   [!A2]   LDW     .D1T1   *A7,A3            ; |80|

           SUBSP   .L2     B6,B7,B3          ;  ^ |72|
|| [!A2]  LDW     .D1T1   *+A7(4),A14       ; |81|

           NOP             1
           SUBSP   .L2     B4,B5,B4          ; |73|
           NOP             1

           MV      .L1X    B3,A4             ;  ^ Define a twin register
|| [!A2]  LDW     .D1T1   *+A7(16),A0       ; |86|

           MV      .S2X    A14,B5            ; Define a twin register
||         MPYSP   .M1     A3,A4,A6          ;  ^ |94|
|| [!A2]  LDW     .D1T1   *+A7(20),A5       ; |87|

           MPYSP   .M2     B5,B4,B5          ; |94|
||         MV      .L1X    B4,A10            ; Define a twin register
|| [!A2]  STW     .D1T1   A4,*++A8(8)       ; |76|

   [!A2]   MPYSP   .M2X    A3,B4,B1          ; |95|
|| [!A2]  STW     .D1T1   A10,*+A8(4)       ; |78|

   [!A2]   LDW     .D1T1   *+A7(12),A3       ; |84|
||         MPYSP   .M2X    A14,B3,B6         ; |95|

           MV      .L2X    A6,B7             ;  ^ Define a twin register
|| [!A2]  LDW     .D1T1   *+A7(8),A11       ; |83|

           ADDSP   .L2     B5,B7,B5          ;  ^ |94|
||         MPYSP   .M1     A5,A10,A12        ; |103|

           MPYSP   .M1     A0,A4,A6          ; |103|

           MPYSP   .M2X    A5,B3,B7          ; |104|
||         MPYSP   .M1     A0,A10,A0         ; |104|

           MPYSP   .M2X    A3,B4,B6          ; |98|
||         MPYSP   .M1     A3,A4,A14         ; |99|
||         SUBSP   .L2     B1,B6,B10         ; |95|

           LDW     .D1T1   *A9,A3            ; |94|
|| [!A2]  MPYSP   .M2      B0,B5,B1          ;  ^ |94|
||         MPYSP   .M1     A11,A10,A5        ; |99|

           ADDSP   .L1     A12,A6,A6         ; |103|
```

```
||          MPYSP   .M2X    A11,B3,B5         ; |98|

            SUBSP   .L1X    A0,B7,A0          ; |104|
            MPYSP   .M2     B0,B10,B7         ; |95|

   [!A2]    LDW     .D2T2   *+B2(16),B1       ; |103|
||          LDW     .D1T1   *+A9(4),A12       ; |95|
||          MV      .S1X    B1,A14            ;  ^ Define a twin register
||          SUBSP   .L1     A5,A14,A5         ; |99|

            LDW     .D2T2   *+B2(20),B5       ; |104|
||          ADDSP   .L1     A14,A3,A11        ;  ^ |94|
||          MPYSP   .M1X    B0,A6,A13         ; |103|
||          ADDSP   .L2     B6,B5,B6          ; |98|

            MPYSP   .M1X    B0,A0,A0          ; |104|

   [!A2]    LDW     .D1T1   *+A7(24),A3       ; |89|
||          MV      .S1X    B7,A14            ; Define a twin register
|| [!A2]    LDW     .D2T2   *+B2(12),B11      ; |99|

   [!A2]    LDW     .D1T1   *+A7(28),A6       ; |90|
||          LDW     .D2T2   *+B2(8),B7        ; |98|
||          MPYSP   .M1X    B0,A5,A5          ; |99|

            ADDSP   .L1     A14,A12,A14       ; |95|
|| [!A2]    STW     .D1T1   A11,*A9           ;  ^ |94|
|| [!A2]    ADDSP   .L2X    A13,B1,B1         ; |103|
||          MPYSP   .M2     B0,B6,B10         ; |98|

            ADDSP   .L2X    A0,B5,B5          ; |104|
||          LDW     .D2T2   *B2,B6            ; @ ^ |57|
||          LDW     .D1T1   *++A7(8),A13      ; @|34|

            LDW     .D1T1   *+A7(4),A0        ; @|35|
   [!A2]    ADDSP   .L2X    A5,B11,B11        ; |99|

   [!A2]    STW     .D1T1   A14,*+A9(4)       ; |95|
|| [!A2]    STW     .D2T2   B1,*+B2(16)       ; |103|
||          ADDSP   .L2     B10,B7,B10        ; |98|

   [!A2]    MPYSP   .M2X    A3,B3,B1          ; |109|
||          MPYSP   .M1     A6,A10,A5         ; |109|
||          LDW     .D2T2   *+B2(4),B7        ; @|57|

   [!A2]    STW     .D2T2   B5,*+B2(20)       ; |104|
||          LDW     .D1T1   *+A7(8),A14       ; @|39|
||          MPYSP   .M2X    A13,B6,B5         ; @ ^ |57|

   [!A2]    STW     .D2T2   B11,*+B2(12)      ; |99|
||          MPYSP   .M2X    A0,B6,B6          ; @|58|
||          LDW     .D1T1   *+A7(12),A11      ; @|40|

   [!A2]    STW     .D1T2   B10,*+A9(8)       ; |98|

   [!A2]    MV      .L2     B9,B11            ;
||          ADDSP   .L1X    A5,B1,A5          ; |109|
|| [!A2]    LDW     .D1T1   *+A7(24),A12      ; |106|
||          LDW     .D2T2   *+B2(8),B10       ; @|60|

   [!A2]    MV      .S2     B9,B1             ;
||          MPYSP   .M1     A3,A10,A0         ; |110|
||          MPYSP   .M2X    A0,B7,B5          ; @|57|
||          ADDSP   .L2     B5,B11,B12        ; @ ^ |57|

            MPYSP   .M2X    A6,B3,B6          ; |110|
|| [!A2]    LDW     .D1T1   *+A7(28),A6       ; |107|
||          MPYSP   .M1X    A13,B7,A3         ; @|58|
||          ADDSP   .L2     B6,B1,B7          ; @|58|

            LDW     .D2T2   *+B2(12),B1       ; @|60|

   [!A2]    LDW     .D2T2   *+B2(24),B11      ; |109|
||          MPYSP   .M1X    B0,A5,A4          ; |109|
||          MV      .L1     A4,A13            ; Inserted to split a long life
```

```
           MPYSP   .M1X    A11,B10,A5        ; @|61|
||         MPYSP   .M2X    A14,B10,B12       ; @|60|
||         SUBSP   .L2     B12,B5,B5         ; @ ^ |57|

           MPYSP   .M1     A12,A13,A0        ; |114|
||         SUBSP   .L2X    A0,B6,B10         ; |110|
||         MV      .S2     B3,B6             ; Inserted to split a long life
||         ADDSP   .L1X    A3,B7,A3          ; @|58|

           MPYSP   .M2X    A6,B6,B3          ; |115|
||         MPYSP   .M1     A6,A10,A6         ; |114|

           LDW     .D2T2   *+B2(16),B6       ; @|63|
||         LDW     .D1T1   *+A7(16),A10      ; @|44|
||         MPYSP   .M2X    A11,B1,B7         ; @|60|

           LDW     .D2T2   *+B2(28),B5       ; |110|
||         LDW     .D1T1   *+A7(20),A11      ; @|45|
||         ADDSP   .L2     B12,B5,B12        ; @ ^ |60|

           ADDSP   .L2X    A4,B11,B4         ; |109|
|| [!A2]   MV      .S2     B4,B1             ; Inserted to split a long life
||         MPYSP   .M2     B0,B10,B10        ; |110|
||         MPYSP   .M1X    A14,B1,A4         ; @|61|
||         ADDSP   .L1     A5,A3,A3          ; @|61|

           ADDSP   .L1     A6,A0,A0          ; |114|
|| [!A2]   MPYSP   .M2X    A12,B1,B11        ; |115|

           LDW     .D2T2   *+B2(20),B1       ; @|63|

           MPYSP   .M2X    A10,B6,B12        ; @|63|
||         SUBSP   .L2     B12,B7,B7         ; @ ^ |60|

           LDW     .D2T2   *+B2(32),B5       ; |114|
||         ADDSP   .L2     B10,B5,B6         ; |110|
||         MPYSP   .M1X    A11,B6,A3         ; @|64|
||         ADDSP   .L1     A4,A3,A4          ; @|61|

   [!A2]   STW     .D2T2   B4,*+B2(24)       ; |109|
||         SUBSP   .L2     B11,B3,B4         ; |115|
||         MPYSP   .M1X    B0,A0,A0          ; |114|

           LDW     .D2T2   *+B2(24),B3       ; @|66|
||         LDW     .D1T1   *+A7(24),A5       ; @|49|

           MPYSP   .M2X    A11,B1,B10        ; @|63|
||         ADDSP   .L2     B12,B7,B7         ; @ ^ |63|

   [!A2]   LDW     .D2T2   *+B2(36),B1       ; |115|
||         MPYSP   .M1X    A10,B1,A3         ; @|64|
||         ADDSP   .L1     A3,A4,A6          ; @|64|
||         LDW     .D1T1   *+A7(28),A4       ; @|50|

   [!A2]   STW     .D2T2   B6,*+B2(28)       ; |110|
||         MPYSP   .M2     B0,B4,B4          ; |115|
||         ADDSP   .L1X    A0,B5,A0          ; |114|

           LDW     .D2T2   *+B2(28),B5       ; @|66|

           MPYSP   .M2X    A5,B3,B7          ; @|66|
||         SUBSP   .L2     B7,B10,B6         ; @ ^ |63|

           NOP             1

   [!A2]   ADDSP   .L2     B4,B1,B1          ; |115|
|| [!A2]   STW     .D2T1   A0,*+B2(32)       ; |114|
||         MPYSP   .M1X    A4,B3,A0          ; @|67|
||         ADDSP   .L1     A3,A6,A6          ; @|64|

           LDW     .D2T2   *+B2(32),B4       ; @|69|
||         LDW     .D1T1   *+A7(32),A3       ; @|54|

           MPYSP   .M2X    A4,B5,B6          ; @|66|
```

```
||          ADDSP   .L2     B7,B6,B7          ; @ ^ |66|

            LDW     .D1T1   *++A7(36),A4      ; @|55|

   [ A1]    SUB     .S1     A1,1,A1           ; |118|
|| [!A2]    STW     .D2T2   B1,*+B2(36)       ; |115|
||          MPYSP   .M1X    A5,B5,A0          ; @|67|
||          ADDSP   .L1     A0,A6,A5          ; @|67|

   [ A1]    B       .S1     L2                ; |118|
||          LDW     .D2T2   *++B2(36),B5      ; @|69|

            MPYSP   .M2X    A3,B4,B6          ; @|69|
||          SUBSP   .L2     B7,B6,B7          ; @ ^ |66|

            NOP             1

            MPYSP   .M1X    A4,B4,A0          ; @|70|
||          ADDSP   .L1     A0,A5,A5          ; @|67|

            NOP             1

   [ A2]    SUB     .L1     A2,1,A2           ;
||          MPYSP   .M1X    A4,B5,A4          ; @|69|
||          ADDSP   .L2     B6,B7,B4          ; @ ^ |69|

;** --------------------------------------------------------------------------*
L3:    ; PIPED LOOP EPILOG

            LDW     .D2T2   *++B8(8),B7       ; (E) @|72|
||          MPYSP   .M2X    A3,B5,B5          ; (E) @|70|

            LDW     .D2T2   *+B8(4),B6        ; (E) @|73|
||          ADDSP   .L1     A0,A5,A3          ; (E) @|70|

            LDW     .D1T1   *A7,A1            ; (E) @|80|

            LDW     .D1T1   *++A7(4),A0       ; (E) @|81|
||          SUBSP   .L2X    B4,A4,B8          ; (E) @ ^ |69|

            LDW     .D1T1   *++A7(16),A2      ; (E) @|86|

            LDW     .D1T1   *++A7(20),A6      ; (E) @|87|
||          ADDSP   .L2X    B5,A3,B4          ; (E) @|70|

            NOP             1
            SUBSP   .L2     B7,B8,B1          ; (E) @ ^ |72|
            NOP             1
            SUBSP   .L2     B6,B4,B6          ; (E) @|73|
            NOP             1

            MPYSP   .M2X    A0,B1,B8          ; (E) @|95|
||          MV      .L1X    B1,A4             ;

            MV      .L1     A4,A3             ; (E) @Inserted to split a long life
||          MPYSP   .M2X    A6,B1,B5          ; (E) @|104|
||          MPYSP   .M1X    A1,B1,A10         ;
||          STW     .D1T1   A4,*++A8(8)       ; (E) @|76|

            MV      .L2     B6,B3             ; (E) @Inserted to split a long life
||          MPYSP   .M1     A2,A4,A5          ; (E) @|103|
||          MPYSP   .M2X    A0,B6,B4          ;
||          MV      .L1X    B6,A0             ;

            MPYSP   .M1     A2,A0,A11         ; (E) @|104|
||          MPYSP   .M2X    A1,B6,B7          ; (E) @|95|
||          STW     .D1T1   A0,*+A8(4)        ; (E) @|78|

            LDW     .D2T2   *+B2(16),B10      ; (E) @|103|
||          MPYSP   .M1X    A6,B6,A2          ;
||          LDW     .D1T1   *++A7(12),A1      ; (E) @|84|

            LDW     .D2T2   *+B2(20),B11      ; (E) @|104|
||          LDW     .D1T1   *++A7(8),A6       ; (E) @|83|
```

```
           LDW     .D1T1   *+A9(4),A8        ; (E) @|95|
||         ADDSP   .L2X    B4,A10,B4        ;

           LDW     .D1T1   *A9,A10          ; (E) @|94|
||         SUBSP   .L1X    A11,B5,A11       ; (E) @|104|
||         SUBSP   .L2     B7,B8,B5         ; (E) @|95|

           LDW     .D1T1   *+A7(24),A2      ; (E) @|89|
||         ADDSP   .L1     A2,A5,A5         ; (E) @|103|

           LDW     .D2T2   *+B2(12),B8      ; (E) @|99|
||         MPYSP   .M2X    A1,B6,B9         ; (E) @|98|
||         MPYSP   .M1     A1,A4,A4         ; (E) @|99|

           LDW     .D1T1   *+A7(28),A1      ; (E) @|90|
||         MPYSP   .M2X    A6,B1,B7         ; (E) @|98|
||         MPYSP   .M1     A6,A0,A6         ; (E) @|99|

           MPYSP   .M1X    B0,A11,A11       ; (E) @|104|
||         LDW     .D2T2   *+B2(8),B5       ; (E) @|98|
||         MPYSP   .M2     B0,B5,B6         ; (E) @|95|

           MPYSP   .M1X    B0,A5,A5         ; (E) @|103|
||         MPYSP   .M2     B0,B4,B12        ; (E) @ ^ |94|

           MPYSP   .M1     A2,A0,A2         ; (E) @|110|
||         MPYSP   .M2X    A2,B1,B4         ; (E) @|109|

           ADDSP   .L2     B9,B7,B7         ; (E) @|98|
||         SUBSP   .L1     A6,A4,A4         ; (E) @|99|

           MPYSP   .M1     A1,A0,A8         ; (E) @|109|
||         ADDSP   .L2X    A11,B11,B6       ; (E) @|104|
||         ADDSP   .L1X    B6,A8,A6         ;

           ADDSP   .L2X    A5,B10,B9        ; (E) @|103|
||         ADDSP   .L1X    B12,A10,A5       ;

           MPYSP   .M2X    A1,B1,B10        ; (E) @|110|

           MPYSP   .M2     B0,B7,B7         ; (E) @|98|
||         MPYSP   .M1X    B0,A4,A4         ; (E) @|99|

           ADDSP   .L1X    A8,B4,A6         ; (E) @|109|
||         STW     .D1T1   A6,*+A9(4)       ; (E) @|95|

           STW     .D1T1   A5,*A9           ; (E) @ ^ |94|

           SUBSP   .L2X    A2,B10,B4        ; (E) @|110|
||         STW     .D2T2   B9,*+B2(16)      ; (E) @|103|

           STW     .D2T2   B6,*+B2(20)      ; (E) @|104|
||         ADDSP   .L2X    A4,B8,B6         ; (E) @|99|

           MPYSP   .M1X    B0,A6,A4         ; (E) @|109|
||         ADDSP   .L2     B7,B5,B7         ; (E) @|98|

           NOP             1
           MPYSP   .M2     B0,B4,B5         ; (E) @|110|
           STW     .D2T2   B6,*+B2(12)      ; (E) @|99|
           STW     .D1T2   B7,*+A9(8)       ; (E) @|98|

           LDW     .D2T2   *+B2(24),B8      ; (E) @|109|
||         LDW     .D1T1   *+A7(36),A5      ; (E) @|107|

           LDW     .D2T2   *+B2(28),B4      ; (E) @|110|
||         LDW     .D1T1   *+A7(32),A6      ; (E) @|106|

           LDW     .D2T2   *+B2(36),B7      ; (E) @|115|
           LDW     .D2T2   *+B2(32),B6      ; (E) @|114|
           NOP             1

           MPYSP   .M1     A5,A0,A0         ; (E) @|114|
||         MPYSP   .M2X    A5,B1,B9         ;
```

```
           ADDSP   .L2     B5,B4,B4          ; (E) @|110|
||         MPYSP   .M1     A6,A3,A3          ; (E) @|114|
||         MPYSP   .M2X    A6,B3,B5          ; (E) @|115|

           ADDSP   .L2X    A4,B8,B8          ; (E) @|109|
           NOP             2

           STW     .D2T2   B4,*+B2(28)       ; (E) @|110|
||         SUBSP   .L2     B5,B9,B4          ; (E) @|115|
||         ADDSP   .L1     A0,A3,A0          ; (E) @|114|

           STW     .D2T2   B8,*+B2(24)       ; (E) @|109|
           NOP             2

           MPYSP   .M1X    B0,A0,A0          ; (E) @|114|
||         MPYSP   .M2     B0,B4,B4          ; (E) @|115|

           NOP             3

           ADDSP   .L1X    A0,B6,A0          ; (E) @|114|
||         ADDSP   .L2     B4,B7,B4          ; (E) @|115|

           NOP             3
           STW     .D2T2   B4,*+B2(36)       ; (E) @|115|
           STW     .D2T1   A0,*+B2(32)       ; (E) @|114|
           LDW     .D2T1   *+SP(8),A10       ; |119|
           LDW     .D2T2   *+SP(28),B3       ; |119|
           LDW     .D2T2   *+SP(36),B11      ; |119|
           LDW     .D2T1   *+SP(24),A14      ; |119|
           LDW     .D2T1   *+SP(16),A12      ; |119|
           LDW     .D2T1   *+SP(12),A11      ; |119|
           LDW     .D2T1   *+SP(20),A13      ; |119|

           LDW     .D2T2   *+SP(32),B10      ; |119|
||         B       .S2     B3                ; |119|

           LDW     .D2T2   *++SP(40),B12     ; |119|
           NOP             4
           ; BRANCH OCCURS                   ; |119|
```

lms_12.c

```
/*
lms_12.c executes the lms adaptive filter algorithm for p=12

x=Reference Data {Note length(x) must be equal to p+N-1 in order to create N output
    samples}
a=Desired signal corrupted by interferer (Primary Channel)
h=filter taps (interleaved real/imaginary).
dhat=filtered output

THESE ARE NOT USED IN THIS VERSION
p=# filter taps (order of filter)       FIXED TO 12
N=2*number output samples for this block of data          FIXED TO 2048
*/

void lms_12(float* x, float* a, float* dhat, float* h, float mu)
{
        int i=0,k=0;
        float v_hat_r=0.0f, v_hat_i=0.0f;
        float tempX_r,tempX_i;
        float temp_r,temp_i;
        volatile float* x_vol=(volatile float*)x;
        do
        {
                k=0;
                do
                {
                    temp_r=h[k];
                    temp_i=h[k+1];
                    tempX_r=x[i+k];
                    tempX_i=x[i+k+1];
                    v_hat_r=v_hat_r+temp_r*tempX_r-temp_i*tempX_i;
                    v_hat_i=v_hat_i+temp_r*tempX_i+temp_i*tempX_r;
```

```
                k+=2;
            } while(k<24);
            temp_r=a[i]-v_hat_r;
            temp_i=a[i+1]-v_hat_i;
            k=0;
            do
            {
                tempX_r=x_vol[i+k];
                tempX_i=x_vol[i+k+1];
                h[k]=h[k]+mu*(temp_r*tempX_r+temp_i*tempX_i);
                h[k+1]=h[k+1]+mu*(temp_i*tempX_r-temp_r*tempX_i);
                k+=2;
            } while(k<24);
            v_hat_r=0.0f;
            dhat[i]=temp_r;
            v_hat_i=0.0f;
            dhat[i+1]=temp_i;
            i+=2;
        }while (i<2048);
}
```

lms_12_opt.asm

```
;******************************************************************************
;* TMS320C6x ANSI C Codegen                                      Version 4.00 *
;* Date/Time created: Wed Oct 16 11:29:20 2002                                *
;******************************************************************************

;******************************************************************************
;* GLOBAL FILE PARAMETERS                                                     *
;*                                                                            *
;*   Architecture      : TMS320C670x                                          *
;*   Optimization      : Enabled at level 3                                   *
;*   Optimizing for    : Speed                                                *
;*                       Based on options: -o3, no -ms                        *
;*   Endian            : Big                                                  *
;*   Interrupt Thrshld : Disabled                                             *
;*   Memory Model      : Large                                                *
;*   Calls to RTS      : Far                                                  *
;*   Pipelining        : Enabled                                              *
;*   Speculative Load  : Enabled (Threshold = 64                             ) *
;*   Memory Aliases    : Presume are aliases (pessimistic)                    *
;*   Debug Info        : No Debug Info                                        *
;*                                                                            *
;******************************************************************************

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 C:\TEMP\TI209_2 C:\TEMP\TI209_4 -w C:\ti\myprojects\
    adaptive
        .sect   ".text"
        .global _lms_12

;******************************************************************************
;* FUNCTION NAME: _lms_12                                                     *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Local Frame Size  : 0 Args + 28 Auto + 44 Save = 72 byte                 *
;******************************************************************************
_lms_12:
;** --------------------------------------------------------------------------*
           STW     .D2T2   B13,*SP--(72)     ; |14|
           STW     .D2T1   A4,*+SP(16)
           STW     .D2T2   B4,*+SP(12)
           STW     .D2T1   A11,*+SP(36)      ; |14|
           STW     .D2T2   B12,*+SP(68)      ; |14|
```

```
           STW     .D2T2   B11,*+SP(64)       ; |14|
           STW     .D2T2   B10,*+SP(60)       ; |14|
           STW     .D2T2   B3,*+SP(56)        ; |14|
           STW     .D2T1   A15,*+SP(52)       ; |14|
           STW     .D2T1   A14,*+SP(48)       ; |14|
           STW     .D2T1   A13,*+SP(44)       ; |14|
           STW     .D2T1   A12,*+SP(40)       ; |14|
           STW     .D2T1   A10,*+SP(32)       ; |14|
           STW     .D2T2   B6,*+SP(4)

           STW     .D2T1   A6,*+SP(8)
||         ZERO    .L1     A3                 ;

           STW     .D2T1   A3,*+SP(28)
||         ZERO    .L1     A4                 ; |15|

           STW     .D2T1   A4,*+SP(20)
||         MVK     .S2     0x400,B4           ; |15|

           STW     .D2T2   B4,*+SP(24)
||         MV      .L1     A8,A11             ;
||         ZERO    .S1     A0                 ;
||         ZERO    .D1     A5                 ;

;** --------------------------------------------------------------------------*
;**   BEGIN LOOP L1
;** --------------------------------------------------------------------------*
L1:
           LDW     .D2T1   *+SP(16),A3
           LDW     .D2T1   *+SP(20),A6
           LDW     .D2T1   *+SP(4),A4
           NOP             3

           MVC     .S2     CSR,B11
||         ADD     .L1     A6,A3,A3           ; |17|

           LDW     .D2T2   *+SP(28),B5        ; |17|
||         AND     .L2     -2,B11,B2
||         SUB     .L1     A4,8,A3            ; |17|
||         SUB     .S2X    A3,8,B8            ; |17|

           MVC     .S2     B2,CSR             ; interrupts off
||         LDW     .D1T1   *+A3(8),A7         ; (P) |30|
||         LDW     .D2T2   *+B8(8),B3         ; (P) |30|

;*--------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Loop Unroll Multiple             : 2x
;*      Known Minimum Trip Count         : 6
;*      Known Maximum Trip Count         : 6
;*      Known Max Trip Count Factor      : 6
;*      Loop Carried Dependency Bound(^) : 4
;*      Unpartitioned Resource Bound     : 5
;*      Partitioned Resource Bound(*)    : 5
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                    5*       5*
;*      .S units                    1        0
;*      .D units                    4        4
;*      .M units                    3        5*
;*      .X cross paths              3        5*
;*      .T address paths            4        4
;*      Long read paths             0        0
;*      Long write paths            0        0
;*      Logical  ops (.LS)          0        0     (.L or .S unit)
;*      Addition ops (.LSD)         1        0     (.L or .S or .D unit)
;*      Bound(.L .S .LS)            3        3
;*      Bound(.L .S .D .LS .LSD)    4        3
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 5  Schedule found with 4 iterations in parallel
;*      done
;*
;*      Epilog not entirely removed
```

```
;*      Collapsed epilog stages     : 2
;*
;*      Prolog not entirely removed
;*      Collapsed prolog stages     : 2
;*
;*      Minimum required memory pad : 32 bytes
;*
;*      Minimum safe trip count     : 2
;*----------------------------------------------------------------------------*
L2:    ; PIPED LOOP PROLOG

           MV      .L2X    A0,B6             ; |17|
||         LDW     .D2T2   *+B8(12),B10      ; (P) |30|
||         LDW     .D1T1   *+A3(12),A0       ; (P) |30|

           ZERO    .L1     A9                ; Accumulator initialization
||         LDW     .D1T1   *++A3(16),A7      ; (P) |30|
||         LDW     .D2T2   *++B8(16),B10     ; (P) |30|

           ZERO    .L2     B9                ; Accumulator initialization
||         ZERO    .L1     A4                ; |30|
||         MVK     .S2     0x2,B0            ; init prolog collapse predicate
||         MVK     .S1     0xc,A2            ; |17|
||         LDW     .D2T2   *+B8(4),B3        ; (P) |30|
||         LDW     .D1T1   *+A3(4),A0        ; (P) |30|

           ZERO    .L2     B7                ; Accumulator initialization
||         ZERO    .L1     A6                ; Accumulator initialization
||         MVK     .S2     0x1,B1            ; init prolog collapse predicate
||         MV      .D1     A2,A1             ;
||         MV      .D2     B5,B4             ;
||         B       .S1     L3                ; (P) |33|

;** ---------------------------------------------------------------------------*
L3:    ; PIPED LOOP KERNEL

   [!B1]   ADDSP   .L2     B10,B9,B9         ; @ ^ |31|
||         SUBSP   .L1     A4,A0,A0          ; @|30|
||         MPYSP   .M2X    B3,A7,B10         ; @@|30|
||         LDW     .D2T2   *+B8(8),B3        ; @@@|30|
||         LDW     .D1T1   *+A3(8),A7        ; @@@|30|

   [!B0]   ADDSP   .L1     A6,A5,A5          ;  ^ |30|
|| [!B1]   ADDSP   .L2     B2,B4,B4          ; @ ^ |30|
||         MPYSP   .M2X    B10,A7,B10        ; @@|31|
||         MPYSP   .M1X    B10,A0,A0         ; @@|30|
||         LDW     .D2T2   *+B8(12),B10      ; @@@|30|
||         LDW     .D1T1   *+A3(12),A0       ; @@@|30|

   [!B1]   ADDSP   .L2     B2,B6,B6          ; @ ^ |31|
|| [!B1]   SUBSP   .L1     A4,A8,A6          ; @|30|
||         MPYSP   .M1X    B3,A0,A10         ; @@|31|
||         MPYSP   .M2X    B10,A7,B2         ; @@|30|
||         LDW     .D2T2   *++B8(16),B10     ; @@@|30|
||         LDW     .D1T1   *++A3(16),A7      ; @@@|30|

   [ A1]   SUB     .S1     A1,2,A1           ; @
|| [!B1]   ADDSP   .L1     A10,A9,A9         ; @ ^ |31|
|| [!B1]   ADDSP   .L2     B2,B5,B5          ; @ ^ |31|
||         MPYSP   .M2X    B10,A0,B2         ; @@|31|
||         MPYSP   .M1X    B3,A0,A8          ; @@|30|
||         LDW     .D2T2   *+B8(4),B3        ; @@@|30|
||         LDW     .D1T1   *+A3(4),A0        ; @@@|30|

   [ B1]   SUB     .S2     B1,1,B1           ;
|| [ B0]   SUB     .D2     B0,1,B0           ;
|| [ A2]   SUB     .D1     A2,2,A2           ;
|| [!B1]   ADDSP   .L1     A0,A6,A6          ; @ ^ |30|
|| [ A1]   B       .S1     L3                ; @|33|
|| [ A2]   ADDSP   .L2     B10,B7,B7         ; @@ ^ |30|
||         MPYSP   .M2X    B3,A7,B2          ; @@|31|

;*----------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
```

```
;*      Loop Unroll Multiple             : 2x
;*      Known Minimum Trip Count         : 6
;*      Known Maximum Trip Count         : 6
;*      Known Max Trip Count Factor      : 6
;*      Loop Carried Dependency Bound(^) : 0
;*      Unpartitioned Resource Bound     : 6
;*      Partitioned Resource Bound(*)    : 6
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                     4        4
;*      .S units                     1        0
;*      .D units                     6*       6*
;*      .M units                     6*       6*
;*      .X cross paths               2        6*
;*      .T address paths             6*       6*
;*      Long read paths              2        2
;*      Long write paths             0        0
;*      Logical  ops (.LS)           0        0     (.L or .S unit)
;*      Addition ops (.LSD)          1        0     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             3        2
;*      Bound(.L .S .D .LS .LSD)     4        4
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 6  Register is live too long
;*         ii = 6  Register is live too long
;*         ii = 6  Register is live too long
;*         ii = 7  Cannot allocate machine registers
;*                   Regs Live Always   :  6/3  (A/B-side)
;*                   Max Regs Live      : 18/15
;*                   Max Cond Regs Live :  1/0
;*         ii = 7  Register is live too long
;*         ii = 7  Register is live too long
;*         ii = 8  Cannot allocate machine registers
;*                   Regs Live Always   :  6/3  (A/B-side)
;*                   Max Regs Live      : 18/15
;*                   Max Cond Regs Live :  1/0
;*         ii = 8  Register is live too long
;*         ii = 8  Register is live too long
;*         ii = 9  Cannot allocate machine registers
;*                   Regs Live Always   :  6/3  (A/B-side)
;*                   Max Regs Live      : 18/14
;*                   Max Cond Regs Live :  1/0
;*         ii = 9  Register is live too long
;*         ii = 9  Register is live too long
;*         ii = 10 Cannot allocate machine registers
;*                   Regs Live Always   :  6/3  (A/B-side)
;*                   Max Regs Live      : 15/13
;*                   Max Cond Regs Live :  1/0
;*         ii = 10 Register is live too long
;*         ii = 10 Schedule found with 4 iterations in parallel
;*      done
;*
;*      Loop is interruptible
;*      Epilog not removed
;*      Collapsed epilog stages     : 0
;*
;*      Prolog not removed
;*      Collapsed prolog stages     : 0
;*
;*      Minimum required memory pad : 0 bytes
;*
;*      Minimum safe trip count     : 8
;*----------------------------------------------------------------------------*
L4:    ; PIPED LOOP EPILOG AND PROLOG

           MVK     .S1     6,A1              ;
||         MV      .D1     A11,A4            ;
||         MV      .S2X    A11,B13
||         LDW     .D2T2   *+SP(12),B8

           LDW     .D2T1   *+SP(16),A3
||         ADDSP   .L1     A6,A5,A0          ; (E) @@@ ^ |30|

           ADDSP   .L2     B9,B5,B5          ; Accumulator summation
||         LDW     .D2T1   *+SP(20),A7
```

```
           LDW     .D2T2   *+SP(4),B7        ; |35|
||         ADDSP   .L2     B7,B4,B4          ; Accumulator summation

           MVC     .S2     B11,CSR           ; interrupts on
||         ADDSP   .L2X    A9,B6,B6          ; Accumulator summation

           LDW     .D2T2   *B8,B0            ; |34|
||         ADDSP   .L1     A6,A0,A0          ; Accumulator summation

           LDW     .D2T1   *+B8(4),A5        ;
           ADD     .L1     A7,A3,A3          ; |35|

           SUB     .S2     B7,8,B7           ; |35|
||         SUB     .S1     A3,8,A7           ; |35|
||         ADDSP   .L2     B5,B6,B5          ; |35|

           LDW     .D1T2   *+A7(12),B6       ; (P) |40|
||         ADDSP   .L1X    B4,A0,A0          ;

           SUB     .L1X    B7,8,A8           ;
||         LDW     .D1T2   *+A7(8),B8        ; (P) |39|

           LDW     .D1T2   *+A7(28),B9       ; (P) @|40|

           LDW     .D1T1   *++A7(16),A9      ; (P) |39|
||         SUBSP   .L1X    A5,B5,A6          ; |35|

           LDW     .D1T2   *+A7(8),B5        ; (P) @|39|
||         SUBSP   .L1X    B0,A0,A5          ; |34|

           LDW     .D2T1   *+B7(12),A3       ; (P) |42|

           MV      .L2     B8,B4             ; (P) Inserted to split a long life
||         LDW     .D1T1   *+A7(4),A0        ; (P) |40|

           MV      .L2     B9,B3             ; (P) @Inserted to split a long life
||         LDW     .D1T2   *+A7(28),B2       ; (P) @@|40|
||         MPYSP   .M2X    B8,A6,B8          ; (P) |42|

           LDW     .D1T1   *++A7(16),A2      ; (P) @|39|
||         MPYSP   .M1     A9,A5,A12         ; (P) |41|
||         MPYSP   .M2X    B6,A5,B1          ; (P) |42|

           MV      .L2     B5,B11            ; (P) @Inserted to split a long life
||         LDW     .D1T2   *+A7(8),B0        ; (P) @@|39|
||         MPYSP   .M1     A9,A6,A10         ; (P) |42|
||         MPYSP   .M2X    B5,A6,B5          ; (P) @|42|

           LDW     .D2T1   *+B7(28),A9       ; (P) @|42|
||         MPYSP   .M2X    B9,A5,B9          ; (P) @|42|

           MPYSP   .M2X    B4,A5,B4          ; (P) |41|
||         MPYSP   .M1     A0,A6,A13         ; (P) |41|

           MPYSP   .M2X    B6,A6,B6          ;
||         MPYSP   .M1     A0,A5,A0          ; (P) |42|
||         SUBSP   .L2     B8,B1,B8          ; (P) |42|

           MV      .L2X    A6,B1             ;
           SUBSP   .L2     B5,B9,B5          ; (P) @|42|
           ADDSP   .L1     A13,A12,A12       ; (P) |41|
           MPYSP   .M2X    A11,B8,B8         ;
           NOP             3
           ADDSP   .L1X    B8,A3,A3          ; (P) |42|
           NOP             3
           STW     .D2T1   A3,*+B7(12)       ; (P) |42|

           LDW     .D1T1   *+A7(4),A15       ; (P) @|40|
||         MPYSP   .M2X    B2,A5,B9          ; (P) @@|42|
||         LDW     .D2T2   *++B7(16),B8      ; (P) |41|
||         ADDSP   .L2     B6,B4,B4          ; (P) |41|
||         SUBSP   .L1     A10,A0,A11        ; (P) |42|

           MPYSP   .M1     A2,A6,A3          ; (P) @|42|
```

```
||          MPYSP   .M2X    A4,B5,B5          ; (P) @|42|

            MPYSP   .M1     A2,A5,A0          ; (P) @|41|
||          MPYSP   .M2X    B0,A6,B10         ; (P) @@|42|

;** --------------------------------------------------------------------------*
L5:    ; PIPED LOOP KERNEL

            LDW     .D1T1   *++A8(16),A13     ; |41|
||          MPYSP   .M1     A4,A12,A10        ; |41|
||          LDW     .D2T2   *++B7(4),B12      ; |42|

            MPYSP   .M2     B13,B4,B4         ; |41|
||          MPYSP   .M1     A4,A11,A14        ; |42|

            MPYSP   .M2     B3,B1,B6          ; @|41|
||          MPYSP   .M1     A15,A6,A11        ; @|41|
||          ADDSP   .L1X    B5,A9,A12         ; @|42|
||          MV      .L2     B2,B3             ; @@Inserted to split a long life
||          LDW     .D1T2   *++A7(28),B2      ; @@@|40|

   [ A1]    SUB     .L1     A1,2,A1           ;
||          MPYSP   .M2X    B11,A5,B9         ; @|41|
||          MPYSP   .M1     A15,A5,A15        ; @|42|
||          LDW     .D1T1   *++A7(16),A2      ; @@|39|
||          SUBSP   .L2     B10,B9,B10        ; @@|42|

   [ A1]    B       .S1     L5                ; |44|
||          ADDSP   .L2X    A10,B8,B5         ; |41|
||          MV      .S2     B0,B11            ; @@Inserted to split a long life
||          LDW     .D1T2   *++A7(8),B0       ; @@@|39|

            ADDSP   .L1X    B4,A13,A10        ; |41|
||          ADDSP   .L2X    A14,B12,B12       ; |42|
||          LDW     .D2T1   *++B7(28),A9      ; @@|42|

            ADDSP   .L1     A11,A0,A12        ; @|41|
||          STW     .D2T1   A12,*++B7(12)     ; @|42|

            ADDSP   .L2     B6,B9,B4          ; @|41|
||          SUBSP   .L1     A3,A15,A11        ; @|42|
||          LDW     .D2T2   *++B7(16),B8      ; @|41|
||          LDW     .D1T1   *++A7(4),A15      ; @@|40|
||          MPYSP   .M2X    B2,A5,B9          ; @@@|42|

            STW     .D2T2   B5,*-B7(16)       ; |41|
||          MPYSP   .M1     A2,A6,A3          ; @@|42|
||          MPYSP   .M2X    A4,B10,B5         ; @@|42|

            STW     .D1T1   A10,*A8           ; |41|
||          STW     .D2T2   B12,*-B7(12)      ; |42|
||          MPYSP   .M1     A2,A5,A0          ; @@|41|
||          MPYSP   .M2X    B0,A6,B10         ; @@@|42|

;** --------------------------------------------------------------------------*
L6:    ; PIPED LOOP EPILOG
;** --------------------------------------------------------------------------*

            MPYSP   .M2X    B11,A5,B6         ; (E) @@|41|
||          LDW     .D2T2   *++B7(4),B11      ; (E) @|42|
||          MPYSP   .M1     A4,A12,A13        ; (E) @|41|
||          LDW     .D1T1   *++A8(16),A12     ; (E) @|41|

            MV      .L1     A4,A11
||          LDW     .D1T1   *++A7(16),A2      ; (E) @@@|39|
||          MPYSP   .M1     A4,A11,A14        ; (E) @|42|
||          MPYSP   .M2     B13,B4,B4         ; (E) @|41|

            LDW     .D2T1   *++B7(28),A9      ; (E) @@@|42|
||          MPYSP   .M2     B3,B1,B5          ; (E) @@|41|
||          MPYSP   .M1     A15,A6,A1         ; (E) @@|41|
||          ADDSP   .L1X    B5,A9,A10         ; (E) @@|42|

            MPYSP   .M2X    B0,A5,B9          ;
||          MPYSP   .M1     A15,A5,A15        ; (E) @@|42|
```

```
||          SUBSP   .L2     B10,B9,B10        ; (E) @@@|42|

            MPYSP   .M2     B2,B1,B8          ;
||          ADDSP   .L2X    A13,B8,B0         ; (E) @|41|

            ADDSP   .L2X    A14,B11,B4        ; (E) @|42|
||          ADDSP   .L1X    B4,A12,A12        ; (E) @|41|

            MPYSP   .M1     A2,A5,A1          ; (E) @@@|41|
||          ADDSP   .L2     B5,B6,B5          ; (E) @@|41|
||          ADDSP   .L1     A1,A0,A0          ; (E) @@|41|
||          STW     .D2T1   A10,*+B7(12)      ; (E) @@|42|

            MPYSP   .M1     A2,A6,A2          ; (E) @@@|42|
||          MPYSP   .M2X    A4,B10,B6         ; (E) @@@|42|
||          SUBSP   .L1     A3,A15,A7         ; (E) @@|42|
||          LDW     .D1T1   *+A7(4),A3        ; (E) @@@|40|

            ADDSP   .L2     B8,B9,B8          ; (E) @@@|41|
||          LDW     .D2T2   *++B7(16),B9      ; (E) @@|41|

            STW     .D2T2   B0,*-B7(16)       ; (E) @|41|

            MPYSP   .M2     B13,B5,B5         ; (E) @@|41|
||          MPYSP   .M1     A4,A0,A0          ; (E) @@|41|
||          STW     .D1T1   A12,*A8           ; (E) @|41|

            MPYSP   .M1     A4,A7,A12         ; (E) @@|42|
||          ADDSP   .L1X    B6,A9,A9          ; (E) @@@|42|
||          STW     .D2T2   B4,*-B7(12)       ; (E) @|42|

            MPYSP   .M2     B13,B8,B4         ; (E) @@@|41|
||          MPYSP   .M1     A3,A6,A10         ; (E) @@@|41|
||          LDW     .D2T2   *+B7(4),B6        ; (E) @@|42|
||          LDW     .D1T1   *++A8(16),A7      ; (E) @@|41|

            MPYSP   .M1     A3,A5,A3          ; (E) @@@|42|
            ADDSP   .L2X    A0,B9,B8          ; (E) @@|41|
            STW     .D2T1   A9,*+B7(12)       ; (E) @@@|42|

            LDW     .D2T2   *++B7(16),B9      ; (E) @@@|41|
||          ADDSP   .L1     A10,A1,A0         ; (E) @@@|41|

            ADDSP   .L2X    A12,B6,B5         ; (E) @@|42|
||          ADDSP   .L1X    B5,A7,A7          ; (E) @@|41|

            SUBSP   .L1     A2,A3,A3          ; (E) @@@|42|
||          STW     .D2T2   B8,*-B7(16)       ; (E) @@|41|

            NOP             1
            MPYSP   .M1     A4,A0,A0          ; (E) @@@|41|
            STW     .D1T1   A7,*A8            ; (E) @@|41|

            MPYSP   .M1     A4,A3,A4          ; (E) @@@|42|
||          STW     .D2T2   B5,*-B7(12)       ; (E) @@|42|

            LDW     .D2T2   *+B7(4),B5        ; (E) @@@|42|
||          LDW     .D1T1   *++A8(16),A3      ; (E) @@@|41|

            ADDSP   .L2X    A0,B9,B6          ; (E) @@@|41|
            NOP             3

            STW     .D2T2   B6,*B7            ; (E) @@@|41|
||          ADDSP   .L1X    B4,A3,A0          ; (E) @@@|41|
||          ADDSP   .L2X    A4,B5,B4          ; (E) @@@|42|

            NOP             3
            STW     .D1T1   A0,*A8            ; (E) @@@|41|
            STW     .D2T2   B4,*+B7(4)        ; (E) @@@|42|
            LDW     .D2T2   *+SP(8),B5
            LDW     .D2T2   *+SP(24),B4
            LDW     .D2T2   *+SP(12),B6
            LDW     .D2T1   *+SP(20),A3
            LDW     .D2T1   *+SP(28),A0       ; |48|
```

```
           STW     .D2T1   A5,*B5            ; |46|
||         MV      .L2     B5,B7             ; |48|

           STW     .D2T1   A6,*+B5(4)        ; |48|

           SUB     .L2     B4,1,B6           ; |50|
||         ADD     .D2     8,B6,B4           ; |49|
||         ADD     .S2     8,B7,B5           ; |49|

           STW     .D2T2   B6,*+SP(24)       ; |49|
||         ADD     .S1     8,A3,A3           ; |49|
||         MV      .L1X    B6,A1             ; |49|

           MV      .L1     A0,A5             ; |48|
||         STW     .D2T1   A3,*+SP(20)       ; |49|
|| [ A1]   B       .S2     L1                ; |50|

           STW     .D2T2   B5,*+SP(8)        ; |49|
           STW     .D2T2   B4,*+SP(12)       ; |49|
   [!A1]   LDW     .D2T2   *+SP(56),B3       ; |51|
   [!A1]   LDW     .D2T2   *+SP(64),B11      ; |51|
   [!A1]   LDW     .D2T2   *+SP(60),B10      ; |51|
           ; BRANCH OCCURS                   ; |50|
;** --------------------------------------------------------------------------*
           LDW     .D2T1   *+SP(52),A15      ; |51|
           LDW     .D2T1   *+SP(48),A14      ; |51|
           LDW     .D2T1   *+SP(44),A13      ; |51|
           LDW     .D2T1   *+SP(40),A12      ; |51|
           LDW     .D2T1   *+SP(36),A11      ; |51|
           LDW     .D2T1   *+SP(32),A10      ; |51|

           B       .S2     B3                ; |51|
||         LDW     .D2T2   *+SP(68),B12      ; |51|

           LDW     .D2T2   *++SP(72),B13     ; |51|
           NOP             4
           ; BRANCH OCCURS                   ; |51|
```

lms_30.c

```
/*
lms_30.c executes the lms adaptive filter algorithm for p=30

x=Reference Data {Note length(x) must be equal to p+N-1 in order to create N output
    samples}
a=Desired signal corrupted by interferer (Primary Channel)
h=filter taps (interleaved real/imaginary).
dhat=filtered output

THESE ARE NOT USED IN THIS VERSION
p=# filter taps (order of filter)       FIXED TO 30
N=2*number output samples for this block of data          FIXED TO 2048
*/

void lms_30(float* x, float* a, float* dhat, float* h, float mu)
{
        int i=0,k=0;
        float v_hat_r=0.0f, v_hat_i=0.0f;
        float tempX_r,tempX_i;
        float temp_r,temp_i;
        volatile float* x_vol=(volatile float*)x;
        do
        {
                k=0;
                do
                {
                    temp_r=h[k];
                    temp_i=h[k+1];
                    tempX_r=x[i+k];
                    tempX_i=x[i+k+1];
                    v_hat_r=v_hat_r+temp_r*tempX_r-temp_i*tempX_i;
                    v_hat_i=v_hat_i+temp_r*tempX_i+temp_i*tempX_r;
                    k+=2;
                } while(k<60);
```

```
                temp_r=a[i]-v_hat_r;
                temp_i=a[i+1]-v_hat_i;

                k=0;
                do
                {
                    tempX_r=x_vol[i+k];
                    tempX_i=x_vol[i+k+1];
                    h[k]=h[k]+mu*(temp_r*tempX_r+temp_i*tempX_i);
                    h[k+1]=h[k+1]+mu*(temp_i*tempX_r-temp_r*tempX_i);
                    k+=2;
                } while(k<60);
                v_hat_r=0.0f;
                dhat[i]=temp_r;
                v_hat_i=0.0f;
                dhat[i+1]=temp_i;
                i+=2;
        }while (i<2048);
}
```

lms_30_opt.asm

```
;******************************************************************************
;* TMS320C6x ANSI C Codegen                                      Version 4.00 *
;* Date/Time created: Wed Oct 16 11:55:16 2002                                *
;******************************************************************************

;******************************************************************************
;* GLOBAL FILE PARAMETERS                                                     *
;*                                                                            *
;*   Architecture      : TMS320C670x                                          *
;*   Optimization      : Enabled at level 3                                   *
;*   Optimizing for    : Speed                                                *
;*                       Based on options: -o3, no -ms                        *
;*   Endian            : Big                                                  *
;*   Interrupt Thrshld : Disabled                                             *
;*   Memory Model      : Large                                                *
;*   Calls to RTS      : Far                                                  *
;*   Pipelining        : Enabled                                              *
;*   Speculative Load  : Enabled (Threshold = 64                             ) *
;*   Memory Aliases    : Presume are aliases (pessimistic)                    *
;*   Debug Info        : No Debug Info                                        *
;*                                                                            *
;******************************************************************************

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 C:\TEMP\TI156_2 C:\TEMP\TI156_4 -w C:\ti\myprojects\
    adaptive
        .sect   ".text"
        .global _lms_30

;******************************************************************************
;* FUNCTION NAME: _lms_30                                                     *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Local Frame Size  : 0 Args + 28 Auto + 44 Save = 72 byte                 *
;******************************************************************************
_lms_30:
;** --------------------------------------------------------------------------*
           STW     .D2T2   B13,*SP--(72)     ; |15|
           STW     .D2T1   A4,*+SP(16)
           STW     .D2T2   B4,*+SP(12)
           STW     .D2T1   A14,*+SP(48)      ; |15|
           STW     .D2T2   B12,*+SP(68)      ; |15|
           STW     .D2T2   B11,*+SP(64)      ; |15|
```

```
           STW     .D2T2   B10,*+SP(60)       ; |15|
           STW     .D2T2   B3,*+SP(56)        ; |15|
           STW     .D2T1   A15,*+SP(52)       ; |15|
           STW     .D2T1   A13,*+SP(44)       ; |15|
           STW     .D2T1   A12,*+SP(40)       ; |15|
           STW     .D2T1   A11,*+SP(36)       ; |15|
           STW     .D2T1   A10,*+SP(32)       ; |15|
           STW     .D2T2   B6,*+SP(4)

           STW     .D2T1   A6,*+SP(8)
||         ZERO    .L1     A5                 ;

           STW     .D2T1   A5,*+SP(28)
||         ZERO    .L1     A4                 ; |16|

           STW     .D2T1   A4,*+SP(20)
||         MVK     .S2     0x400,B4           ; |16|

           STW     .D2T2   B4,*+SP(24)
||         MV      .L1     A8,A14             ;
||         ZERO    .S1     A0                 ;
||         ZERO    .D1     A3                 ;

;** --------------------------------------------------------------------------*
;**   BEGIN LOOP L1
;** --------------------------------------------------------------------------*
L1:
           LDW     .D2T1   *+SP(16),A5
           LDW     .D2T2   *+SP(20),B4

           MV      .L2X    A0,B5              ; |18|
||         LDW     .D2T1   *+SP(4),A0

           LDW     .D2T1   *+SP(28),A4        ; |18|
           NOP             2
           ADD     .L2X    B4,A5,B4           ; |18|

           MVC     .S2     CSR,B12
||         SUB     .D1     A0,24,A10          ; |18|
||         SUB     .D2     B4,24,B7           ; |18|

           MV      .L2X    A3,B0              ; |18|
||         AND     .S2     -2,B12,B4
||         LDW     .D1T1   *++A10(24),A3      ; (P) |31|
||         LDW     .D2T2   *++B7(24),B2       ; (P) |31|

           MVC     .S2     B4,CSR             ; interrupts off
||         LDW     .D1T1   *+A10(4),A5        ; (P) |31|
||         LDW     .D2T2   *+B7(16),B1        ; (P) |31|

;*--------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Loop Unroll Multiple             : 3x
;*      Known Minimum Trip Count         : 10
;*      Known Maximum Trip Count         : 10
;*      Known Max Trip Count Factor      : 10
;*      Loop Carried Dependency Bound(^) : 4
;*      Unpartitioned Resource Bound     : 8
;*      Partitioned Resource Bound(*)    : 8
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                     7        8*
;*      .S units                     1        0
;*      .D units                     6        6
;*      .M units                     4        8*
;*      .X cross paths               6        8*
;*      .T address paths             5        7
;*      Long read paths              0        0
;*      Long write paths             0        0
;*      Logical  ops (.LS)           0        0     (.L or .S unit)
;*      Addition ops (.LSD)          1        0     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             4        4
;*      Bound(.L .S .D .LS .LSD)     5        5
;*
```

```
;*      Searching for software pipeline schedule at ...
;*         ii = 8  Schedule found with 3 iterations in parallel
;*      done
;*
;*      Epilog not entirely removed
;*      Collapsed epilog stages     : 1
;*
;*      Prolog not entirely removed
;*      Collapsed prolog stages     : 1
;*
;*      Minimum required memory pad : 24 bytes
;*
;*      Minimum safe trip count     : 3
;*----------------------------------------------------------------------------*
L2:    ; PIPED LOOP PROLOG

           MVK     .S1     0x1,A2            ; init prolog collapse predicate
||         MV      .L2X    A4,B6             ; |18|
||         LDW     .D2T2   *+B7(4),B2        ; (P) |31|
||         LDW     .D1T1   *+A10(12),A0      ; (P) |31|

           ZERO    .L1     A7                ; |31|
||         MVK     .S1     0xa,A1            ; |18|
||         MV      .L2X    A4,B9             ; |18|
||         LDW     .D1T1   *+A10(20),A12     ; (P) |31|
||         LDW     .D2T2   *+B7(12),B2       ; (P) |31|

           MV      .L1     A4,A11            ; |18|
||         MV      .S1     A4,A8             ; |18|
||         MV      .D1     A4,A9             ; |18|
||         MV      .L2X    A4,B4             ; |18|
||         LDW     .D2T2   *+B7(20),B10      ; (P) |31|

           MV      .L1     A4,A13            ; |18|
||         MV      .S1     A4,A6             ;
||         MV      .L2X    A4,B8             ; |18|
||         LDW     .D1T1   *+A10(16),A5      ; (P) |31|
||         LDW     .D2T2   *+B7(8),B3        ; (P) |31|
||         MPYSP   .M1X    B2,A3,A3          ; (P) |31|

;** ---------------------------------------------------------------------------*
L3:    ; PIPED LOOP KERNEL

   [!A2]   SUBSP   .L1     A7,A0,A4          ; |31|
|| [!A2]   ADDSP   .L2     B8,B9,B9          ;  ^ |32|
||         MPYSP   .M2X    B2,A5,B11         ; @|32|
||         LDW     .D1T2   *+A10(8),B11      ; @|31|

   [ A1]   SUB     .S1     A1,1,A1           ; |34|
|| [!A2]   ADDSP   .L1     A8,A4,A4          ;  ^ |32|
|| [!A2]   ADDSP   .L2     B9,B0,B0          ;  ^ |31|
||         MPYSP   .M1X    B2,A3,A0          ; @|32|
||         MPYSP   .M2X    B2,A5,B5          ; @|31|

   [ A1]   B       .S1     L3                ; |34|
|| [!A2]   ADDSP   .L1X    B8,A11,A11        ;  ^ |31|
|| [!A2]   ADDSP   .L2     B0,B4,B4          ;  ^ |31|
||         MPYSP   .M2X    B2,A0,B5          ; @|31|
||         LDW     .D2T2   *++B7(24),B2      ; @@|31|
||         LDW     .D1T1   *++A10(24),A3     ; @@|31|

   [!A2]   ADDSP   .L1X    B9,A13,A13        ;  ^ |32|
|| [!A2]   ADDSP   .L2     B4,B6,B6          ;  ^ |31|
||         MPYSP   .M2X    B1,A12,B5         ; @|32|
||         LDW     .D2T2   *+B7(16),B1       ; @@|31|
||         LDW     .D1T1   *+A10(4),A5       ; @@|31|

   [!A2]   ADDSP   .L1     A4,A9,A9          ;  ^ |31|
||         ADDSP   .L2     B11,B5,B5         ; @ ^ |32|
||         MPYSP   .M1X    B10,A12,A0        ; @|31|
||         MPYSP   .M2X    B3,A0,B8          ; @|32|
||         LDW     .D2T2   *+B7(4),B2        ; @@|31|
||         LDW     .D1T1   *+A10(12),A0      ; @@|31|

           ADDSP   .L1     A0,A6,A6          ; @ ^ |32|
```

```
||         MPYSP   .M1X    B10,A5,A8         ; @|32|
||         MPYSP   .M2     B3,B11,B8         ; @|31|
||         SUBSP   .L2X    A7,B5,B9          ; @|31|
||         LDW     .D2T2   *+B7(12),B2       ; @@|31|
||         LDW     .D1T1   *+A10(20),A12     ; @@|31|

           ADDSP   .L1     A3,A8,A8          ; @ ^ |31|
||         MPYSP   .M2     B2,B11,B9         ; @|32|
||         SUBSP   .L2X    A7,B5,B0          ; @|31|
||         LDW     .D2T2   *+B7(20),B10      ; @@|31|

   [ A2]   SUB     .S1     A2,1,A2           ;
||         ADDSP   .L2     B5,B8,B8          ; @ ^ |32|
||         MPYSP   .M2X    B1,A5,B4          ; @|31|
||         MPYSP   .M1X    B2,A3,A3          ; @@|31|
||         LDW     .D1T1   *+A10(16),A5      ; @@|31|
||         LDW     .D2T2   *+B7(8),B3        ; @@|31|

;*--------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Loop Unroll Multiple             : 4x
;*      Known Minimum Trip Count         : 15
;*      Known Maximum Trip Count         : 15
;*      Known Max Trip Count Factor      : 15
;*      Loop Carried Dependency Bound(^) : 6
;*      Unpartitioned Resource Bound     : 6
;*      Partitioned Resource Bound(*)    : 6
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                     4        4
;*      .S units                     1        0
;*      .D units                     6*       6*
;*      .M units                     6*       6*
;*      .X cross paths               2        6*
;*      .T address paths             6*       6*
;*      Long read paths              2        2
;*      Long write paths             0        0
;*      Logical  ops (.LS)           0        0     (.L or .S unit)
;*      Addition ops (.LSD)          1        0     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             3        2
;*      Bound(.L .S .D .LS .LSD)     4        4
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 6  Did not find schedule
;*         ii = 7  Did not find schedule
;*         ii = 8  Cannot allocate machine registers
;*                   Regs Live Always   :  6/3  (A/B-side)
;*                   Max Regs Live      : 17/14
;*                   Max Cond Regs Live :  1/0
;*         ii = 8  Cannot allocate machine registers
;*                   Regs Live Always   :  6/3  (A/B-side)
;*                   Max Regs Live      : 17/15
;*                   Max Cond Regs Live :  1/0
;*         ii = 9  Cannot allocate machine registers
;*                   Regs Live Always   :  6/3  (A/B-side)
;*                   Max Regs Live      : 17/15
;*                   Max Cond Regs Live :  1/0
;*         ii = 10 Cannot allocate machine registers
;*                   Regs Live Always   :  6/3  (A/B-side)
;*                   Max Regs Live      : 16/11
;*                   Max Cond Regs Live :  1/0
;*         ii = 10 Cannot allocate machine registers
;*                   Regs Live Always   :  6/3  (A/B-side)
;*                   Max Regs Live      : 17/12
;*                   Max Cond Regs Live :  1/0
;*         ii = 10 Schedule found with 4 iterations in parallel
;*      done
;*
;*      Loop is interruptible
;*      Epilog not removed
;*      Collapsed epilog stages     : 0
;*
;*      Prolog not removed
;*      Collapsed prolog stages     : 0
```

```
;*
;*      Minimum required memory pad : 0 bytes
;*
;*      Minimum safe trip count     : 16
;*-----------------------------------------------------------------------------*
L4:    ; PIPED LOOP EPILOG AND PROLOG

           MVK     .S1     24,A1             ;
||         LDW     .D2T2   *+SP(12),B7
||         ADDSP   .L2     B8,B9,B0          ; (E) @@ ^ |32|
||         SUBSP   .L1     A7,A0,A3          ; (E) @@|31|

           MV      .S1     A14,A0            ;
||         LDW     .D2T1   *+SP(16),A4
||         ADDSP   .L2     B9,B0,B5          ; (E) @@ ^ |31|
||         ADDSP   .L1     A8,A4,A3          ; (E) @@ ^ |32|

           LDW     .D2T1   *+SP(20),A7
||         ADDSP   .L2     B0,B4,B1          ; (E) @@ ^ |31|
||         ADDSP   .L1X    B8,A11,A11        ; (E) @@ ^ |31|

           MV      .S2X    A14,B9
||         LDW     .D2T2   *+SP(4),B4        ; |36|
||         ADDSP   .L1X    B9,A13,A9         ; (E) @@ ^ |32|
||         ADDSP   .L2     B4,B6,B5          ; (E) @@ ^ |31|

           ADDSP   .L1     A3,A9,A4          ; (E) @@ ^ |31|
||         ADDSP   .L2X    A6,B5,B2          ;

           LDW     .D2T1   *B7,A2            ; |35|
||         ADDSP   .L1X    A8,B5,A6          ; |35|

           LDW     .D2T2   *+B7(4),B6        ; |36|

           MVC     .S2     B12,CSR           ; interrupts on
||         ADD     .S1     A7,A4,A5          ; |36|

           SUB     .S2     B4,8,B4           ; |36|
||         SUB     .S1     A5,8,A5           ; |36|
||         ADDSP   .L2     B0,B2,B0          ; |36|

           LDW     .D1T2   *+A5(12),B7       ; (P) |42|
||         ADDSP   .L1X    B1,A6,A7          ; |35|

           SUB     .L1X    B4,8,A6           ;
||         LDW     .D1T2   *+A5(8),B3        ; (P) |41|

           LDW     .D1T1   *++A5(16),A8      ; (P) |41|

           LDW     .D1T2   *+A5(12),B1       ; (P) @|42|
||         ADDSP   .L2X    A9,B0,B0          ; |36|

           LDW     .D1T2   *+A5(8),B2        ; (P) @|41|
||         ADDSP   .L1     A11,A7,A11        ; |35|

           LDW     .D2T1   *+B4(12),A10      ; (P) |44|
           LDW     .D1T1   *+A5(4),A9        ; (P) |42|

           LDW     .D1T1   *++A5(16),A7      ; (P) @|41|
||         ADDSP   .L2     B8,B0,B10         ; |36|

           LDW     .D1T2   *+A5(12),B0       ; (P) @@|42|
||         ADDSP   .L1     A4,A11,A4         ; |35|

           LDW     .D1T2   *+A5(8),B8        ; (P) @@|41|
           LDW     .D2T1   *+B4(28),A13      ; (P) @|44|
           ADDSP   .L1X    A3,B10,A3         ; |36|
           ADDSP   .L2X    B5,A4,B5          ; |35|
           NOP             2
           SUBSP   .L1X    B6,A3,A4          ; |36|
           SUBSP   .L1X    A2,B5,A3          ; |35|
           NOP             2

           MPYSP   .M1     A9,A4,A2          ; (P) |43|
||         MPYSP   .M2X    B3,A4,B11         ; (P) |44|
```

```
            MPYSP   .M1      A8,A3,A11          ; (P) |43|
||          MPYSP   .M2X     B7,A3,B6           ; (P) |44|

            MPYSP   .M1      A9,A3,A9           ; (P) |44|
||          MPYSP   .M2X     B1,A3,B5           ; (P) @|44|

            MPYSP   .M1      A8,A4,A8           ; (P) |44|
||          MPYSP   .M2X     B2,A4,B10          ; (P) @|44|

            MPYSP   .M2X     B3,A3,B3           ; (P) |43|

            ADDSP   .L1      A2,A11,A2          ; (P) |43|
||          MPYSP   .M2X     B7,A4,B6           ;
||          SUBSP   .L2      B11,B6,B7          ; (P) |44|

            MPYSP   .M2X     B2,A3,B2           ; (P) @|43|

            MPYSP   .M2X     B1,A4,B1           ;
||          SUBSP   .L2      B10,B5,B5          ; (P) @|44|

            MV      .S2X     A4,B10             ;

            ADDSP   .L2      B6,B3,B3           ; (P) |43|
||          MPYSP   .M2X     A14,B7,B6          ;

            NOP             1
            MPYSP   .M2X     A0,B5,B7           ; (P) @|44|
            NOP             1
            ADDSP   .L1X     B6,A10,A10         ; (P) |44|
            NOP             3
            STW     .D2T1    A10,*+B4(12)       ; (P) |44|

            LDW     .D2T2    *++B4(16),B5       ; (P) |43|
||          MPYSP   .M2X     B0,A3,B6           ; (P) @@|44|
||          LDW     .D1T1    *+A5(4),A14        ; (P) @|42|
||          SUBSP   .L1      A8,A9,A9           ; (P) |44|

            MPYSP   .M1      A7,A3,A10          ; (P) @|43|
||          MPYSP   .M2X     B8,A4,B13          ; (P) @@|44|

            MPYSP   .M1      A7,A4,A8           ; (P) @|44|
;** --------------------------------------------------------------------------*
L5:     ; PIPED LOOP KERNEL

            LDW     .D1T1    *++A6(16),A12      ; |43|
||          MPYSP   .M1      A0,A2,A15          ; |43|
||          LDW     .D2T2    *+B4(4),B12        ; |44|

            MPYSP   .M2      B9,B3,B11          ; |43|
||          MPYSP   .M1      A0,A9,A11          ; |44|
||          ADDSP   .L2      B1,B2,B3           ; @|43|
||          LDW     .D1T1    *++A5(16),A7       ; @@|41|

            MPYSP   .M1      A14,A4,A9          ; @|43|
||          ADDSP   .L1X     B7,A13,A2          ; @|44|
||          MPYSP   .M2      B0,B10,B1          ; @@|43|
||          SUBSP   .L2      B13,B6,B6          ; @@|44|
||          LDW     .D1T2    *+A5(12),B0        ; @@@|42|

   [ A1]    SUB     .L1      A1,2,A1            ;
||          MPYSP   .M1      A14,A3,A14         ; @|44|
||          MPYSP   .M2X     B8,A3,B2           ; @@|43|
||          LDW     .D1T2    *+A5(8),B8         ; @@@|41|

   [ A1]    B       .S1      L5                 ; |46|
||          ADDSP   .L2X     A15,B5,B13         ; |43|

            ADDSP   .L1X     B11,A12,A11        ; |43|
||          ADDSP   .L2X     A11,B12,B11        ; |44|
||          LDW     .D2T1    *+B4(28),A13       ; @@|44|

            ADDSP   .L1      A9,A10,A2          ; @|43|
||          STW     .D2T1    A2,*+B4(12)        ; @|44|
||          MPYSP   .M2X     A0,B6,B7           ; @@|44|
```

```
           SUBSP   .L1     A8,A14,A9         ; @|44|
||         LDW     .D2T2   *++B4(16),B5      ; @|43|
||         LDW     .D1T1   *+A5(4),A14       ; @@|42|
||         MPYSP   .M2X    B0,A3,B6          ; @@@|44|

           STW     .D2T2   B13,*-B4(16)      ; |43|
||         MPYSP   .M1     A7,A3,A10         ; @@|43|
||         MPYSP   .M2X    B8,A4,B13         ; @@@|44|

           STW     .D1T1   A11,*A6           ; |43|
||         STW     .D2T2   B11,*-B4(12)      ; |44|
||         MPYSP   .M1     A7,A4,A8          ; @@|44|

;** --------------------------------------------------------------------------*
L6:    ; PIPED LOOP EPILOG
;** --------------------------------------------------------------------------*

           ADDSP   .L2     B1,B2,B11         ; (E) @@|43|
||         ADDSP   .L1X    B7,A13,A1         ; (E) @@|44|
||         MPYSP   .M2     B9,B3,B7          ; (E) @|43|
||         MPYSP   .M1     A0,A2,A2          ; (E) @|43|
||         LDW     .D2T2   *+B4(4),B12       ; (E) @|44|
||         LDW     .D1T1   *++A6(16),A11     ; (E) @|43|

           MPYSP   .M2     B0,B10,B1         ; (E) @@@|43|
||         LDW     .D1T1   *++A5(16),A7      ; (E) @@@|41|
||         MPYSP   .M1     A0,A9,A9          ; (E) @|44|

           MPYSP   .M2X    B8,A3,B2          ; (E) @@@|43|
||         LDW     .D2T1   *+B4(28),A13      ; (E) @@@|44|
||         SUBSP   .L2     B13,B6,B6         ; (E) @@@|44|
||         MPYSP   .M1     A14,A4,A12        ; (E) @@|43|

           MV      .S1     A0,A14
||         MPYSP   .M1     A14,A3,A15        ; (E) @@|44|

           MPYSP   .M2     B9,B11,B5         ; (E) @@|43|
||         STW     .D2T1   A1,*+B4(12)       ; (E) @@|44|
||         ADDSP   .L2X    A2,B5,B8          ; (E) @|43|

           LDW     .D1T1   *+A5(4),A9        ; (E) @@@|42|
||         ADDSP   .L2X    A9,B12,B0         ; (E) @|44|
||         ADDSP   .L1X    B7,A11,A5         ; (E) @|43|

           ADDSP   .L2     B1,B2,B6          ; (E) @@@|43|
||         MPYSP   .M1     A7,A3,A10         ; (E) @@@|43|
||         MPYSP   .M2X    A0,B6,B7          ; (E) @@@|44|
||         LDW     .D2T2   *++B4(16),B1      ; (E) @@|43|
||         ADDSP   .L1     A12,A10,A1        ; (E) @@|43|

           MPYSP   .M1     A7,A4,A8          ; (E) @@@|44|
||         SUBSP   .L1     A8,A15,A7         ; (E) @@|44|

           STW     .D2T2   B8,*-B4(16)       ; (E) @|43|
           STW     .D1T1   A5,*A6            ; (E) @|43|

           MPYSP   .M2     B9,B6,B6          ; (E) @@@|43|
||         ADDSP   .L1X    B7,A13,A5         ; (E) @@@|44|
||         MPYSP   .M1     A0,A1,A1          ; (E) @@|43|
||         STW     .D2T2   B0,*-B4(12)       ; (E) @|44|

           MPYSP   .M1     A0,A7,A7          ; (E) @@|44|
||         LDW     .D2T2   *+B4(4),B7        ; (E) @@|44|
||         LDW     .D1T1   *++A6(16),A2      ; (E) @@|43|

           MPYSP   .M1     A9,A4,A11         ; (E) @@@|43|
           MPYSP   .M1     A9,A3,A9          ; (E) @@@|44|

           STW     .D2T1   A5,*+B4(12)       ; (E) @@@|44|
||         ADDSP   .L2X    A1,B1,B9          ; (E) @@|43|

           LDW     .D2T2   *++B4(16),B8      ; (E) @@@|43|

           ADDSP   .L2X    A7,B7,B5          ; (E) @@|44|
```

```
||          ADDSP   .L1X    B5,A2,A5          ; (E) @@|43|

            ADDSP   .L1     A11,A10,A7        ; (E) @@@|43|

            SUBSP   .L1     A8,A9,A8          ; (E) @@@|44|
||          STW     .D2T2   B9,*-B4(16)       ; (E) @@|43|

            NOP             1
            STW     .D1T1   A5,*A6            ; (E) @@|43|

            MPYSP   .M1     A0,A7,A5          ; (E) @@@|43|
||          STW     .D2T2   B5,*-B4(12)       ; (E) @@|44|

            MPYSP   .M1     A0,A8,A0          ; (E) @@@|44|
||          LDW     .D2T2   *+B4(4),B5        ; (E) @@@|44|
||          LDW     .D1T1   *++A6(16),A7      ; (E) @@@|43|

            NOP             2
            ADDSP   .L2X    A5,B8,B7          ; (E) @@@|43|
            NOP             1

            ADDSP   .L2X    A0,B5,B5          ; (E) @@@|44|
||          ADDSP   .L1X    B6,A7,A0          ; (E) @@@|43|

            NOP             1
            STW     .D2T2   B7,*B4            ; (E) @@@|43|
            NOP             1
            STW     .D1T1   A0,*A6            ; (E) @@@|43|
            STW     .D2T2   B5,*+B4(4)        ; (E) @@@|44|
            LDW     .D2T2   *+SP(12),B4
            LDW     .D2T2   *+SP(24),B6
            LDW     .D2T2   *+SP(8),B5
            LDW     .D2T1   *+SP(20),A5
            LDW     .D2T1   *+SP(28),A0       ; |50|
            ADD     .S2     8,B4,B7           ; |51|
            SUB     .L2     B6,1,B6           ; |52|

            STW     .D2T1   A3,*B5            ; |48|
||          ADD     .L2     8,B5,B4           ; |51|
||          MV      .L1X    B6,A1             ; |51|

            STW     .D2T2   B4,*+SP(8)        ; |51|
||          ADD     .L1     8,A5,A3           ; |51|
|| [ A1]   B       .S2     L1                ; |52|

            STW     .D2T1   A3,*+SP(20)       ; |51|
||          MV      .L1     A0,A3             ; |50|

            STW     .D2T1   A4,*+B5(4)        ; |50|
            STW     .D2T2   B7,*+SP(12)       ; |51|
            STW     .D2T2   B6,*+SP(24)       ; |51|
   [!A1]    LDW     .D2T2   *+SP(56),B3       ; |53|
            ; BRANCH OCCURS                   ; |52|
;** --------------------------------------------------------------------------*
            LDW     .D2T2   *+SP(64),B11      ; |53|
            LDW     .D2T2   *+SP(60),B10      ; |53|
            LDW     .D2T1   *+SP(52),A15      ; |53|
            LDW     .D2T1   *+SP(48),A14      ; |53|
            LDW     .D2T1   *+SP(44),A13      ; |53|
            LDW     .D2T1   *+SP(40),A12      ; |53|
            LDW     .D2T1   *+SP(36),A11      ; |53|
            LDW     .D2T1   *+SP(32),A10      ; |53|

            B       .S2     B3                ; |53|
||          LDW     .D2T2   *+SP(68),B12      ; |53|

            LDW     .D2T2   *++SP(72),B13     ; |53|
            NOP             4
            ; BRANCH OCCURS                   ; |53|
```

lms_42.c

```
/*
lms_42.c executes the lms adaptive filter algorithm for p=42

x=Reference Data {Note length(x) must be equal to p+N-1 in order to create N output
    samples}
a=Desired signal corrupted by interferer (Primary Channel)
h=filter taps (interleaved real/imaginary).
dhat=filtered output

THESE ARE NOT USED IN THIS VERSION
p=# filter taps (order of filter)       FIXED TO 42
N=2*number output samples for this block of data          FIXED TO 2048
*/

void lms_42(float* x, float* a, float* dhat, float* h, float mu)
{
        int i=0,k=0;
        float v_hat_r=0.0f, v_hat_i=0.0f;
        float tempX_r,tempX_i;
        float temp_r,temp_i;
        volatile float* x_vol=(volatile float*)x;
        do
        {
                k=0;
                do
                {
                    temp_r=h[k];
                    temp_i=h[k+1];
                    tempX_r=x[i+k];
                    tempX_i=x[i+k+1];
                    v_hat_r=v_hat_r+temp_r*tempX_r-temp_i*tempX_i;
                    v_hat_i=v_hat_i+temp_r*tempX_i+temp_i*tempX_r;
                    k+=2;
                } while(k<84);
                temp_r=a[i]-v_hat_r;
                temp_i=a[i+1]-v_hat_i;

                k=0;
                do
                {
                    tempX_r=x_vol[i+k];
                    tempX_i=x_vol[i+k+1];
                    h[k]=h[k]+mu*(temp_r*tempX_r+temp_i*tempX_i);
                    h[k+1]=h[k+1]+mu*(temp_i*tempX_r-temp_r*tempX_i);
                    k+=2;
                } while(k<84);
                v_hat_r=0.0f;
                dhat[i]=temp_r;
                v_hat_i=0.0f;
                dhat[i+1]=temp_i;
                i+=2;
        }while (i<2048);
}
```

lms_42_opt.asm

```
;******************************************************************************
;* TMS320C6x ANSI C Codegen                                      Version 4.00 *
;* Date/Time created: Tue Nov 19 11:59:33 2002                                *
;******************************************************************************

;******************************************************************************
;* GLOBAL FILE PARAMETERS                                                     *
;*                                                                            *
;*   Architecture      : TMS320C670x                                          *
;*   Optimization      : Enabled at level 3                                   *
;*   Optimizing for    : Speed                                                *
;*                       Based on options: -o3, no -ms                        *
;*   Endian            : Big                                                  *
;*   Interrupt Thrshld : Disabled                                             *
;*   Memory Model      : Large                                                *
```

```
;*   Calls to RTS      : Far                                                  *
;*   Pipelining        : Enabled                                              *
;*   Speculative Load  : Enabled (Threshold = 64                             ) *
;*   Memory Aliases    : Presume are aliases (pessimistic)                    *
;*   Debug Info        : No Debug Info                                        *
;*                                                                            *
;******************************************************************************

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 C:\TEMP\TI295_2 C:\TEMP\TI295_4 -w C:\ti\myprojects\
    adaptive
        .sect   ".text"
        .global _lms_42

;******************************************************************************
;* FUNCTION NAME: _lms_42                                                     *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Local Frame Size  : 0 Args + 28 Auto + 44 Save = 72 byte                 *
;******************************************************************************
_lms_42:
;** --------------------------------------------------------------------------*
           STW     .D2T2   B13,*SP--(72)     ; |15|
           STW     .D2T1   A4,*+SP(16)
           STW     .D2T2   B4,*+SP(12)
           STW     .D2T1   A14,*+SP(48)      ; |15|
           STW     .D2T2   B12,*+SP(68)      ; |15|
           STW     .D2T2   B11,*+SP(64)      ; |15|
           STW     .D2T2   B10,*+SP(60)      ; |15|
           STW     .D2T2   B3,*+SP(56)       ; |15|
           STW     .D2T1   A15,*+SP(52)      ; |15|
           STW     .D2T1   A13,*+SP(44)      ; |15|
           STW     .D2T1   A12,*+SP(40)      ; |15|
           STW     .D2T1   A11,*+SP(36)      ; |15|
           STW     .D2T1   A10,*+SP(32)      ; |15|
           STW     .D2T2   B6,*+SP(4)

           STW     .D2T1   A6,*+SP(8)
||         ZERO    .L1     A5                ;

           STW     .D2T1   A5,*+SP(28)
||         ZERO    .L1     A4                ; |16|

           STW     .D2T1   A4,*+SP(20)
||         MVK     .S2     0x400,B4          ; |16|

           STW     .D2T2   B4,*+SP(24)
||         MV      .L1     A8,A14            ;
||         ZERO    .S1     A0                ;
||         ZERO    .D1     A3                ;

;** --------------------------------------------------------------------------*
;**   BEGIN LOOP L1
;** --------------------------------------------------------------------------*
L1:
           LDW     .D2T1   *+SP(16),A5
           LDW     .D2T2   *+SP(20),B4

           MV      .L2X    A0,B5             ; |18|
||         LDW     .D2T1   *+SP(4),A0

           LDW     .D2T1   *+SP(28),A4       ; |18|
           NOP             2
           ADD     .L2X    B4,A5,B4          ; |18|

           MVC     .S2     CSR,B12
```

```
||         SUB     .D1     A0,24,A10          ; |18|
||         SUB     .D2     B4,24,B7           ; |18|

           MV      .L2X    A3,B0              ; |18|
||         AND     .S2     -2,B12,B4
||         LDW     .D1T1   *++A10(24),A3      ; (P) |31|
||         LDW     .D2T2   *++B7(24),B2       ; (P) |31|

           MVC     .S2     B4,CSR             ; interrupts off
||         LDW     .D1T1   *+A10(4),A5        ; (P) |31|
||         LDW     .D2T2   *+B7(16),B1        ; (P) |31|

;*--------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Loop Unroll Multiple             : 3x
;*      Known Minimum Trip Count         : 14
;*      Known Maximum Trip Count         : 14
;*      Known Max Trip Count Factor      : 14
;*      Loop Carried Dependency Bound(^) : 4
;*      Unpartitioned Resource Bound     : 8
;*      Partitioned Resource Bound(*)    : 8
;*      Resource Partition:
;*                                 A-side   B-side
;*      .L units                      7        8*
;*      .S units                      1        0
;*      .D units                      6        6
;*      .M units                      4        8*
;*      .X cross paths                6        8*
;*      .T address paths              5        7
;*      Long read paths               0        0
;*      Long write paths              0        0
;*      Logical  ops (.LS)            0        0     (.L or .S unit)
;*      Addition ops (.LSD)           1        0     (.L or .S or .D unit)
;*      Bound(.L .S .LS)              4        4
;*      Bound(.L .S .D .LS .LSD)      5        5
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 8  Schedule found with 3 iterations in parallel
;*      done
;*
;*      Epilog not entirely removed
;*      Collapsed epilog stages     : 1
;*
;*      Prolog not entirely removed
;*      Collapsed prolog stages     : 1
;*
;*      Minimum required memory pad : 24 bytes
;*
;*      Minimum safe trip count     : 3
;*--------------------------------------------------------------------------*
L2:    ; PIPED LOOP PROLOG

           MVK     .S1     0x1,A2             ; init prolog collapse predicate
||         MV      .L2X    A4,B6              ; |18|
||         LDW     .D2T2   *+B7(4),B2         ; (P) |31|
||         LDW     .D1T1   *+A10(12),A0       ; (P) |31|

           ZERO    .L1     A7                 ; |31|
||         MVK     .S1     0xe,A1             ; |18|
||         MV      .L2X    A4,B9              ; |18|
||         LDW     .D1T1   *+A10(20),A12      ; (P) |31|
||         LDW     .D2T2   *+B7(12),B2        ; (P) |31|

           MV      .L1     A4,A11             ; |18|
||         MV      .S1     A4,A8              ; |18|
||         MV      .D1     A4,A9              ; |18|
||         MV      .L2X    A4,B4              ; |18|
||         LDW     .D2T2   *+B7(20),B10       ; (P) |31|

           MV      .L1     A4,A13             ; |18|
||         MV      .S1     A4,A6              ;
||         MV      .L2X    A4,B8              ; |18|
||         LDW     .D1T1   *+A10(16),A5       ; (P) |31|
||         LDW     .D2T2   *+B7(8),B3         ; (P) |31|
```

```
||         MPYSP   .M1X    B2,A3,A3          ; (P) |31|

;** --------------------------------------------------------------------------*
L3:    ; PIPED LOOP KERNEL

   [!A2]   SUBSP   .L1     A7,A0,A4          ; |31|
|| [!A2]   ADDSP   .L2     B8,B9,B9          ;  ^ |32|
||         MPYSP   .M2X    B2,A5,B11         ; @|32|
||         LDW     .D1T2   *+A10(8),B11      ; @|31|

   [ A1]   SUB     .S1     A1,1,A1           ; |34|
|| [!A2]   ADDSP   .L1     A8,A4,A4          ;  ^ |32|
|| [!A2]   ADDSP   .L2     B9,B0,B0          ;  ^ |31|
||         MPYSP   .M1X    B2,A3,A0          ; @|32|
||         MPYSP   .M2X    B2,A5,B5          ; @|31|

   [ A1]   B       .S1     L3                ; |34|
|| [!A2]   ADDSP   .L1X    B8,A11,A11        ;  ^ |31|
|| [!A2]   ADDSP   .L2     B0,B4,B4          ;  ^ |31|
||         MPYSP   .M2X    B2,A0,B5          ; @|31|
||         LDW     .D2T2   *++B7(24),B2      ; @@|31|
||         LDW     .D1T1   *++A10(24),A3     ; @@|31|

   [!A2]   ADDSP   .L1X    B9,A13,A13        ;  ^ |32|
|| [!A2]   ADDSP   .L2     B4,B6,B6          ;  ^ |31|
||         MPYSP   .M2X    B1,A12,B5         ; @|32|
||         LDW     .D2T2   *+B7(16),B1       ; @@|31|
||         LDW     .D1T1   *+A10(4),A5       ; @@|31|

   [!A2]   ADDSP   .L1     A4,A9,A9          ;  ^ |31|
||         ADDSP   .L2     B11,B5,B5         ; @ ^ |32|
||         MPYSP   .M1X    B10,A12,A0        ; @|31|
||         MPYSP   .M2X    B3,A0,B8          ; @|32|
||         LDW     .D2T2   *+B7(4),B2        ; @@|31|
||         LDW     .D1T1   *+A10(12),A0      ; @@|31|

            ADDSP   .L1     A0,A6,A6          ; @ ^ |32|
||         MPYSP   .M1X    B10,A5,A8         ; @|32|
||         MPYSP   .M2     B3,B11,B8         ; @|31|
||         SUBSP   .L2X    A7,B5,B9          ; @|31|
||         LDW     .D2T2   *+B7(12),B2       ; @@|31|
||         LDW     .D1T1   *+A10(20),A12     ; @@|31|

            ADDSP   .L1     A3,A8,A8          ; @ ^ |31|
||         MPYSP   .M2     B2,B11,B9         ; @|32|
||         SUBSP   .L2X    A7,B5,B0          ; @|31|
||         LDW     .D2T2   *+B7(20),B10      ; @@|31|

   [ A2]   SUB     .S1     A2,1,A2           ;
||         ADDSP   .L2     B5,B8,B8          ; @ ^ |32|
||         MPYSP   .M2X    B1,A5,B4          ; @|31|
||         MPYSP   .M1X    B2,A3,A3          ; @@|31|
||         LDW     .D1T1   *+A10(16),A5      ; @@|31|
||         LDW     .D2T2   *+B7(8),B3        ; @@|31|

;*--------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Loop Unroll Multiple             : 4x
;*      Known Minimum Trip Count         : 21
;*      Known Maximum Trip Count         : 21
;*      Known Max Trip Count Factor      : 21
;*      Loop Carried Dependency Bound(^) : 6
;*      Unpartitioned Resource Bound     : 6
;*      Partitioned Resource Bound(*)    : 6
;*      Resource Partition:
;*                                  A-side   B-side
;*      .L units                       4        4
;*      .S units                       1        0
;*      .D units                       6*       6*
;*      .M units                       6*       6*
;*      .X cross paths                 2        6*
;*      .T address paths               6*       6*
;*      Long read paths                2        2
;*      Long write paths               0        0
```

```
;*      Logical  ops (.LS)           0        0     (.L or .S unit)
;*      Addition ops (.LSD)          1        0     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             3        2
;*      Bound(.L .S .D .LS .LSD)     4        4
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 6  Did not find schedule
;*         ii = 7  Did not find schedule
;*         ii = 8  Cannot allocate machine registers
;*                   Regs Live Always   :  6/3  (A/B-side)
;*                   Max Regs Live      : 17/14
;*                   Max Cond Regs Live :  1/0
;*         ii = 8  Cannot allocate machine registers
;*                   Regs Live Always   :  6/3  (A/B-side)
;*                   Max Regs Live      : 17/15
;*                   Max Cond Regs Live :  1/0
;*         ii = 9  Cannot allocate machine registers
;*                   Regs Live Always   :  6/3  (A/B-side)
;*                   Max Regs Live      : 17/15
;*                   Max Cond Regs Live :  1/0
;*         ii = 10 Cannot allocate machine registers
;*                   Regs Live Always   :  6/3  (A/B-side)
;*                   Max Regs Live      : 16/11
;*                   Max Cond Regs Live :  1/0
;*         ii = 10 Cannot allocate machine registers
;*                   Regs Live Always   :  6/3  (A/B-side)
;*                   Max Regs Live      : 17/12
;*                   Max Cond Regs Live :  1/0
;*         ii = 10 Schedule found with 4 iterations in parallel
;*      done
;*
;*      Loop is interruptible
;*      Epilog not removed
;*      Collapsed epilog stages     : 0
;*
;*      Prolog not removed
;*      Collapsed prolog stages     : 0
;*
;*      Minimum required memory pad : 0 bytes
;*
;*      Minimum safe trip count     : 16
;*----------------------------------------------------------------------------*
L4:    ; PIPED LOOP EPILOG AND PROLOG

           MVK     .S1     36,A1             ;
||         LDW     .D2T2   *+SP(12),B7
||         ADDSP   .L2     B8,B9,B0          ; (E) @@ ^ |32|
||         SUBSP   .L1     A7,A0,A3          ; (E) @@|31|

           MV      .S1     A14,A0            ;
||         LDW     .D2T1   *+SP(16),A4
||         ADDSP   .L2     B9,B0,B5          ; (E) @@ ^ |31|
||         ADDSP   .L1     A8,A4,A3          ; (E) @@ ^ |32|

           LDW     .D2T1   *+SP(20),A7
||         ADDSP   .L2     B0,B4,B1          ; (E) @@ ^ |31|
||         ADDSP   .L1X    B8,A11,A11        ; (E) @@ ^ |31|

           MV      .S2X    A14,B9
||         LDW     .D2T2   *+SP(4),B4        ; |36|
||         ADDSP   .L1X    B9,A13,A9         ; (E) @@ ^ |32|
||         ADDSP   .L2     B4,B6,B5          ; (E) @@ ^ |31|

           ADDSP   .L1     A3,A9,A4          ; (E) @@ ^ |31|
||         ADDSP   .L2X    A6,B5,B2          ;

           LDW     .D2T1   *B7,A2            ; |35|
||         ADDSP   .L1X    A8,B5,A6          ; |35|

           LDW     .D2T2   *+B7(4),B6        ; |36|

           MVC     .S2     B12,CSR           ; interrupts on
||         ADD     .S1     A7,A4,A5          ; |36|

           SUB     .S2     B4,8,B4           ; |36|
```

```
||          SUB     .S1     A5,8,A5           ; |36|
||          ADDSP   .L2     B0,B2,B0          ; |36|

            LDW     .D1T2   *+A5(12),B7       ; (P) |42|
||          ADDSP   .L1X    B1,A6,A7          ; |35|

            SUB     .L1X    B4,8,A6           ;
||          LDW     .D1T2   *+A5(8),B3        ; (P) |41|

            LDW     .D1T1   *++A5(16),A8      ; (P) |41|

            LDW     .D1T2   *+A5(12),B1       ; (P) @|42|
||          ADDSP   .L2X    A9,B0,B0          ; |36|

            LDW     .D1T2   *+A5(8),B2        ; (P) @|41|
||          ADDSP   .L1     A11,A7,A11        ; |35|

            LDW     .D2T1   *+B4(12),A10      ; (P) |44|
            LDW     .D1T1   *+A5(4),A9        ; (P) |42|

            LDW     .D1T1   *++A5(16),A7      ; (P) @|41|
||          ADDSP   .L2     B8,B0,B10         ; |36|

            LDW     .D1T2   *+A5(12),B0       ; (P) @@|42|
||          ADDSP   .L1     A4,A11,A4         ; |35|

            LDW     .D1T2   *+A5(8),B8        ; (P) @@|41|
            LDW     .D2T1   *+B4(28),A13      ; (P) @|44|
            ADDSP   .L1X    A3,B10,A3         ; |36|
            ADDSP   .L2X    B5,A4,B5          ; |35|
            NOP             2
            SUBSP   .L1X    B6,A3,A4          ; |36|
            SUBSP   .L1X    A2,B5,A3          ; |35|
            NOP             2

            MPYSP   .M1     A9,A4,A2          ; (P) |43|
||          MPYSP   .M2X    B3,A4,B11         ; (P) |44|

            MPYSP   .M1     A8,A3,A11         ; (P) |43|
||          MPYSP   .M2X    B7,A3,B6          ; (P) |44|

            MPYSP   .M1     A9,A3,A9          ; (P) |44|
||          MPYSP   .M2X    B1,A3,B5          ; (P) @|44|

            MPYSP   .M1     A8,A4,A8          ; (P) |44|
||          MPYSP   .M2X    B2,A4,B10         ; (P) @|44|

            MPYSP   .M2X    B3,A3,B3          ; (P) |43|

            ADDSP   .L1     A2,A11,A2         ; (P) |43|
||          MPYSP   .M2X    B7,A4,B6          ;
||          SUBSP   .L2     B11,B6,B7         ; (P) |44|

            MPYSP   .M2X    B2,A3,B2          ; (P) @|43|

            MPYSP   .M2X    B1,A4,B1          ;
||          SUBSP   .L2     B10,B5,B5         ; (P) @|44|

            MV      .S2X    A4,B10            ;

            ADDSP   .L2     B6,B3,B3          ; (P) |43|
||          MPYSP   .M2X    A14,B7,B6         ;

            NOP             1
            MPYSP   .M2X    A0,B5,B7          ; (P) @|44|
            NOP             1
            ADDSP   .L1X    B6,A10,A10        ; (P) |44|
            NOP             3
            STW     .D2T1   A10,*+B4(12)      ; (P) |44|

            LDW     .D2T2   *++B4(16),B5      ; (P) |43|
||          MPYSP   .M2X    B0,A3,B6          ; (P) @@|44|
||          LDW     .D1T1   *+A5(4),A14       ; (P) @|42|
||          SUBSP   .L1     A8,A9,A9          ; (P) |44|
```

```
           MPYSP   .M1     A7,A3,A10          ; (P) @|43|
||         MPYSP   .M2X    B8,A4,B13          ; (P) @@|44|

           MPYSP   .M1     A7,A4,A8           ; (P) @|44|
;** --------------------------------------------------------------------------*
L5:    ; PIPED LOOP KERNEL

           LDW     .D1T1   *++A6(16),A12      ; |43|
||         MPYSP   .M1     A0,A2,A15          ; |43|
||         LDW     .D2T2   *+B4(4),B12        ; |44|

           MPYSP   .M2     B9,B3,B11          ; |43|
||         MPYSP   .M1     A0,A9,A11          ; |44|
||         ADDSP   .L2     B1,B2,B3           ; @|43|
||         LDW     .D1T1   *++A5(16),A7       ; @@|41|

           MPYSP   .M1     A14,A4,A9          ; @|43|
||         ADDSP   .L1X    B7,A13,A2          ; @|44|
||         MPYSP   .M2     B0,B10,B1          ; @@|43|
||         SUBSP   .L2     B13,B6,B6          ; @@|44|
||         LDW     .D1T2   *+A5(12),B0        ; @@@|42|

   [ A1]   SUB     .L1     A1,2,A1            ;
||         MPYSP   .M1     A14,A3,A14         ; @|44|
||         MPYSP   .M2X    B8,A3,B2           ; @@|43|
||         LDW     .D1T2   *+A5(8),B8         ; @@@|41|

   [ A1]   B       .S1     L5                 ; |46|
||         ADDSP   .L2X    A15,B5,B13         ; |43|

           ADDSP   .L1X    B11,A12,A11        ; |43|
||         ADDSP   .L2X    A11,B12,B11        ; |44|
||         LDW     .D2T1   *+B4(28),A13       ; @@|44|

           ADDSP   .L1     A9,A10,A2          ; @|43|
||         STW     .D2T1   A2,*+B4(12)        ; @|44|
||         MPYSP   .M2X    A0,B6,B7           ; @@|44|

           SUBSP   .L1     A8,A14,A9          ; @|44|
||         LDW     .D2T2   *++B4(16),B5       ; @|43|
||         LDW     .D1T1   *+A5(4),A14        ; @@|42|
||         MPYSP   .M2X    B0,A3,B6           ; @@@|44|

           STW     .D2T2   B13,*-B4(16)       ; |43|
||         MPYSP   .M1     A7,A3,A10          ; @@|43|
||         MPYSP   .M2X    B8,A4,B13          ; @@@|44|

           STW     .D1T1   A11,*A6            ; |43|
||         STW     .D2T2   B11,*-B4(12)       ; |44|
||         MPYSP   .M1     A7,A4,A8           ; @@|44|

;** --------------------------------------------------------------------------*
L6:    ; PIPED LOOP EPILOG
;** --------------------------------------------------------------------------*

           ADDSP   .L2     B1,B2,B11          ; (E) @@|43|
||         ADDSP   .L1X    B7,A13,A1          ; (E) @@|44|
||         MPYSP   .M2     B9,B3,B7           ; (E) @|43|
||         MPYSP   .M1     A0,A2,A2           ; (E) @|43|
||         LDW     .D2T2   *+B4(4),B12        ; (E) @|44|
||         LDW     .D1T1   *++A6(16),A11      ; (E) @|43|

           MPYSP   .M2     B0,B10,B1          ; (E) @@@|43|
||         LDW     .D1T1   *++A5(16),A7       ; (E) @@@|41|
||         MPYSP   .M1     A0,A9,A9           ; (E) @|44|

           MPYSP   .M2X    B8,A3,B2           ; (E) @@@|43|
||         LDW     .D2T1   *+B4(28),A13       ; (E) @@@|44|
||         SUBSP   .L2     B13,B6,B6          ; (E) @@@|44|
||         MPYSP   .M1     A14,A4,A12         ; (E) @@|43|

           MV      .S1     A0,A14
||         MPYSP   .M1     A14,A3,A15         ; (E) @@|44|

           MPYSP   .M2     B9,B11,B5          ; (E) @@|43|
```

```
||          STW     .D2T1   A1,*+B4(12)        ; (E) @@|44|
||          ADDSP   .L2X    A2,B5,B8           ; (E) @|43|

            LDW     .D1T1   *+A5(4),A9         ; (E) @@@|42|
||          ADDSP   .L2X    A9,B12,B0          ; (E) @|44|
||          ADDSP   .L1X    B7,A11,A5          ; (E) @|43|

            ADDSP   .L2     B1,B2,B6           ; (E) @@@|43|
||          MPYSP   .M1     A7,A3,A10          ; (E) @@@|43|
||          MPYSP   .M2X    A0,B6,B7           ; (E) @@@|44|
||          LDW     .D2T2   *++B4(16),B1       ; (E) @@|43|
||          ADDSP   .L1     A12,A10,A1         ; (E) @@|43|

            MPYSP   .M1     A7,A4,A8           ; (E) @@@|44|
||          SUBSP   .L1     A8,A15,A7          ; (E) @@|44|

            STW     .D2T2   B8,*-B4(16)        ; (E) @|43|
            STW     .D1T1   A5,*A6             ; (E) @|43|

            MPYSP   .M2     B9,B6,B6           ; (E) @@@|43|
||          ADDSP   .L1X    B7,A13,A5          ; (E) @@@|44|
||          MPYSP   .M1     A0,A1,A1           ; (E) @@|43|
||          STW     .D2T2   B0,*-B4(12)        ; (E) @|44|

            MPYSP   .M1     A0,A7,A7           ; (E) @@|44|
||          LDW     .D2T2   *+B4(4),B7         ; (E) @@|44|
||          LDW     .D1T1   *++A6(16),A2       ; (E) @@|43|

            MPYSP   .M1     A9,A4,A11          ; (E) @@@|43|
            MPYSP   .M1     A9,A3,A9           ; (E) @@@|44|

            STW     .D2T1   A5,*+B4(12)        ; (E) @@@|44|
||          ADDSP   .L2X    A1,B1,B9           ; (E) @@|43|

            LDW     .D2T2   *++B4(16),B8       ; (E) @@@|43|

            ADDSP   .L2X    A7,B7,B5           ; (E) @@|44|
||          ADDSP   .L1X    B5,A2,A5           ; (E) @@|43|

            ADDSP   .L1     A11,A10,A7         ; (E) @@@|43|

            SUBSP   .L1     A8,A9,A8           ; (E) @@@|44|
||          STW     .D2T2   B9,*-B4(16)        ; (E) @@|43|

            NOP             1
            STW     .D1T1   A5,*A6             ; (E) @@|43|

            MPYSP   .M1     A0,A7,A5           ; (E) @@@|43|
||          STW     .D2T2   B5,*-B4(12)        ; (E) @@|44|

            MPYSP   .M1     A0,A8,A0           ; (E) @@@|44|
||          LDW     .D2T2   *+B4(4),B5         ; (E) @@@|44|
||          LDW     .D1T1   *++A6(16),A7       ; (E) @@@|43|

            NOP             2
            ADDSP   .L2X    A5,B8,B7           ; (E) @@@|43|
            NOP             1

            ADDSP   .L2X    A0,B5,B5           ; (E) @@@|44|
||          ADDSP   .L1X    B6,A7,A0           ; (E) @@@|43|

            NOP             1
            STW     .D2T2   B7,*B4             ; (E) @@@|43|
            NOP             1
            STW     .D1T1   A0,*A6             ; (E) @@@|43|
            STW     .D2T2   B5,*+B4(4)         ; (E) @@@|44|
            LDW     .D2T2   *+SP(12),B4
            LDW     .D2T2   *+SP(24),B6
            LDW     .D2T2   *+SP(8),B5
            LDW     .D2T1   *+SP(20),A5
            LDW     .D2T1   *+SP(28),A0        ; |50|
            ADD     .S2     8,B4,B7            ; |51|
            SUB     .L2     B6,1,B6            ; |52|

            STW     .D2T1   A3,*B5             ; |48|
```

```
||          ADD     .L2     8,B5,B4           ; |51|
||          MV      .L1X    B6,A1             ; |51|

            STW     .D2T2   B4,*+SP(8)        ; |51|
||          ADD     .L1     8,A5,A3           ; |51|
|| [ A1]    B       .S2     L1                ; |52|

            STW     .D2T1   A3,*+SP(20)       ; |51|
||          MV      .L1     A0,A3             ; |50|

            STW     .D2T1   A4,*+B5(4)        ; |50|
            STW     .D2T2   B7,*+SP(12)       ; |51|
            STW     .D2T2   B6,*+SP(24)       ; |51|
   [!A1]    LDW     .D2T2   *+SP(56),B3       ; |53|
            ; BRANCH OCCURS                   ; |52|
;** --------------------------------------------------------------------------*
            LDW     .D2T2   *+SP(64),B11      ; |53|
            LDW     .D2T2   *+SP(60),B10      ; |53|
            LDW     .D2T1   *+SP(52),A15      ; |53|
            LDW     .D2T1   *+SP(48),A14      ; |53|
            LDW     .D2T1   *+SP(44),A13      ; |53|
            LDW     .D2T1   *+SP(40),A12      ; |53|
            LDW     .D2T1   *+SP(36),A11      ; |53|
            LDW     .D2T1   *+SP(32),A10      ; |53|

            B       .S2     B3                ; |53|
||          LDW     .D2T2   *+SP(68),B12      ; |53|

            LDW     .D2T2   *++SP(72),B13     ; |53|
            NOP             4
            ; BRANCH OCCURS                   ; |53|
```

movedata.c

```
/*The lms() function requires the previous p complex samples to filter the data
correctly.  This moves the last p complex samples from the end of the present block
    to
the first of the next block.

x = destination
backupData = origin*/

void MoveData(volatile float* x, float* backupData, int TWO_P) {
        int i;
        for (i=0;i<TWO_P;i+=2) {
                x[i]=backupData[i];
            x[i+1]=backupData[i+1];
        }
}
```

movedata_opt.asm

```
;******************************************************************************
;* TMS320C6x ANSI C Codegen                                     Version 4.00 *
;* Date/Time created: Tue Sep 10 16:54:36 2002                                *
;******************************************************************************

;******************************************************************************
;* GLOBAL FILE PARAMETERS                                                     *
;*                                                                            *
;*   Architecture      : TMS320C670x                                          *
;*   Optimization      : Enabled at level 3                                   *
;*   Optimizing for    : Speed                                                *
;*                       Based on options: -o3, no -ms                        *
;*   Endian            : Big                                                  *
;*   Interrupt Thrshld : Disabled                                             *
;*   Memory Model      : Large                                                *
;*   Calls to RTS      : Far                                                  *
;*   Pipelining        : Enabled                                              *
;*   Speculative Load  : Enabled (Threshold = 64                             ) *
;*   Memory Aliases    : Presume are aliases (pessimistic)                    *
```

```
;*   Debug Info        : No Debug Info                                          *
;*                                                                             *
;*******************************************************************************

;/*The lms() function requires the previous p complex samples to filter the data
;correctly.  This moves the last p complex samples from the end of the present block
    to
;the first of the next block.
;
;x = destination
;backupData = origin*/
;
;void MoveData(volatile float* x, float* backupData, int TWO_P) {
;       int i;
;       for (i=0;i<TWO_P;i+=2) {
;               x[i]=backupData[i];
;           x[i+1]=backupData[i+1];
;       }
;}

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 C:\TEMP\TI163_2 C:\TEMP\TI163_4 -w C:\ti\myprojects\
    adaptive
        .sect   ".text"
        .global _MoveData

;*******************************************************************************
;* FUNCTION NAME: _MoveData                                                    *
;*                                                                             *
;*   Regs Modified     : A0,A1,A3,A4,A5,A6,B0,B1                               *
;*   Regs Used         : A0,A1,A3,A4,A5,A6,B0,B1,B3,B4                         *
;*   Local Frame Size  : 0 Args + 0 Auto + 0 Save = 0 byte                     *
;*******************************************************************************
_MoveData:
;** --------------------------------------------------------------------------*
           CMPGT   .L2X    A6,0,B1           ;
   [!B1]   B       .S2     B3                ; |15|
           ADD     .L1     1,A6,A0           ;
           SHR     .S2X    A0,1,B0           ; |12|
           SUB     .D1     A4,8,A5           ;
           SUB     .L1X    B4,8,A4           ;
   [ B1]   MVK     .S1     0x1,A1            ; init prolog collapse predicate
           ; BRANCH OCCURS                   ; |15| ; chained return
;*--------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Known Minimum Trip Count         : 1
;*      Known Maximum Trip Count         : 1073741823
;*      Known Max Trip Count Factor      : 1
;*      Loop Carried Dependency Bound(^) : 2
;*      Unpartitioned Resource Bound     : 2
;*      Partitioned Resource Bound(*)    : 4
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                     0        0
;*      .S units                     0        1
;*      .D units                     4*       0
;*      .M units                     0        0
;*      .X cross paths               0        0
;*      .T address paths             4*       0
;*      Long read paths              2        0
;*      Long write paths             0        0
;*      Logical  ops (.LS)           0        0     (.L or .S unit)
;*      Addition ops (.LSD)          0        1     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             0        1
;*      Bound(.L .S .D .LS .LSD)     2        1
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 6  Schedule found with 2 iterations in parallel
;*      done
;*
```

```
;*      Loop is interruptible
;*      Collapsed epilog stages     : 1
;*      Collapsed prolog stages     : 1
;*      Minimum required memory pad : 8 bytes
;*
;*      Minimum safe trip count     : 1
;*--------------------------------------------------------------------------*
L1:    ; PIPED LOOP PROLOG
;** --------------------------------------------------------------------------*
L2:    ; PIPED LOOP KERNEL

   [ B0]   B       .S2     L2                ; |14|
|| [!A1]   LDW     .D1T1   *+A4(4),A0        ; |13|

           NOP             2

           MV      .L1     A3,A6             ; Inserted to split a long life
||         LDW     .D1T1   *++A4(8),A3       ; @|12|

   [!A1]   STW     .D1T1   A6,*++A5(8)       ;  ^ |12|

   [ A1]   SUB     .L1     A1,1,A1           ;
|| [!A1]   STW     .D1T1   A0,*+A5(4)        ;  ^ |13|
|| [ B0]   SUB     .L2     B0,1,B0           ; @|14|

;** --------------------------------------------------------------------------*
L3:    ; PIPED LOOP EPILOG
;** --------------------------------------------------------------------------*
           B       .S2     B3                ; |15|
           NOP             5
           ; BRANCH OCCURS                   ; |15|
```

apihost.c

```
#include <stdio.h>

#include "pnk.h"
#include "rtapi.h"

/*#define READSIZE      983040*/
/*#define READSIZE      786432*/
#define READSIZE        4196
/*#define READSIZE      262144*/
/*#define READSIZE      131072*/
/*#define READSIZE      65536*/
/*#define READSIZE      8192 */

main()
{

        FILE* fp,*fp2;
        int i;
        char buffer[READSIZE];
        int size;
        fd_set fdset;
        int in;
        short temp1, temp2;
        int total=0;
        short j;
        short testI[15];
        short testQ[15];

        if ((fp=fopen("LMSdataD","wb"))==NULL) {
                printf("Couldn't open file fftdata!\n");
                exit(1);
        }

        /* Create Stream Device*/
        in = rt_creat("astronomy:FFTstreamD:0", 0);

        if (in == -1)
    {
      printf("rt_creat failed!\n");
      exit(1);
```

```
    }

        FD_ZERO(&fdset);        /* Clear file descriptor set */
        FD_SET(in,&fdset);      /* Add in to file descriptor set */

        select(FD_SETSIZE,&fdset,NULL,NULL,NULL);       /* Wait for data to become
                                                        available on "in" */
        while(1)
    {
      /* Read data from socket (and from DSP) */

                size = recv(in, buffer, READSIZE, 0);
                total=total+size;

      /* Check for error */

                if (size == -1)
                {
                        printf("[SOCKET_ERROR]\n");
                        break;
                }

                /* Check for end of file */

                if (size == 0)
                {
                        printf("[EOF]\n");
                        break;
                }

                /*Reorder bytes and write to file*/
                for (i=0;i<size;i+=4) {
                        temp1=((short)buffer[i])<<8;
                        temp1=temp1|((short)buffer[i+1]&0x00ff);
                        temp2=((short)buffer[i+2])<<8;
                        temp2=temp2|((short)buffer[i+3]&0x00ff);
                        fwrite(&temp2,sizeof(short),1,fp);
                        fwrite(&temp1,sizeof(short),1,fp);

                }

        }
        closesocket(in);

        printf("Total␣bytes␣received:␣%i\n",total);
        printf("Finished␣Writing␣to␣File\n");
        fclose(fp);
}
```

graphLMS_Thesis.m

```
clear all;

file=1;%file designates the file number to be opened
FID1=fopen(['LMSdataD',num2str(file)]);
FID2=fopen(['LMSdataA',num2str(file)]);
FID3=fopen(['LMSdataC',num2str(file)]);

load DSPCalibration;%load DSP calibration vectors (baseline smoothing)

%Load data from LMSdataA, LMSdataC and LMSdataD
[dhat,lengthdata_dhat]=fread(FID1,inf,'float');
[a,lengthdata_a]=fread(FID3,inf,'float');

%Length of adaptive filter (Number of Complex Taps)
P=fread(FID2,1,'uint32');
%DSP 6216 digital receiver signal decimation rate
DECIMATION_RATE=fread(FID2,1,'uint32');
%DSP 6216 sample clock decimation rate
MASTER_ClkDiv=fread(FID2,1,'uint32');
%Number of 1024 pt FFT frames averaged in each integration sample
ActualIntegrationFrames=fread(FID2,1,'uint32');
%Total number of integration frames
TotalFrames=fread(FID2,1,'uint32');
```

```
%Number of samples that the reference channel, x[n], (Processor A) was delayed
DELAY_A=fread(FID2,1,'uint32');
%Number of samples that the primary channel, alpha[n], (Processor C) was delayed
DELAY_C=fread(FID2,1,'uint32');
%Sample clock f_c
SAMPLE_RATE=fread(FID2,1,'uint32');
%Total number of integration blocks sent to host PC
NumberBlocks=fread(FID2,1,'uint32');

DSP_Fc=fread(FID2,1,'float');
Gvar=fread(FID2,1,'float');    %This is the variable gain setting of the DSP
%Desired disjoint integration time
INTEGRATION_TIME=fread(FID2,1,'float');
%Desired total integration time
PROGRAM_RUN_TIME=fread(FID2,1,'float');
%Actual disjoint integration time (associated with ActualIntegrationFrames)
ActualIntegrationTime_dhat_alpha=fread(FID2,1,'float');
%Actual total integration time (associated with TotalFrames)
ActualProgramRunTime=fread(FID2,1,'float');
%LMS filter adaptation constant
MU=fread(FID2,1,'float');
%Final decimated sample rate
DecimatedSampleRate=fread(FID2,1,'float');

[x,lengthdata_x]=fread(FID2,1025,'float');

BW=DecimatedSampleRate/1e6; %Bandwidth in MHz

%If making tests with the GBT, these next few lines determine the center frequency,
    Fc
DSP_GBT_MaxIF=10; %The maximum frequency in the passband coming from the Spectral
    Processor. (MHz).

if DSP_GBT_MaxIF==20,
    DSP_RF=1597; %The frequency at the feed corresponding to DSP_GBT_MaxIF. (MHz)
else
    DSP_RF=1601; %DSP_GBT_MaxIF==10
end

Fc=(DSP_GBT_MaxIF-DSP_Fc/1e6)+DSP_RF;

dBmHzFLAG=0; %If 0, then plots will be in dBm.  If 1, then plots will be in dBm/Hz
GRAPHMODE=1; %If 0, then it will replot.  If 1, it will use XOR mode in the animation
    .
same_AXES=1; %If 0, then the cumulative and instantaneous axes will be different. If
     1, then they will be the same.
SKIP=2; %Number of disjoint integrations to discard at the first of the cumulative
    integration.
BUFFER_MAX=0.5; %Spacing in dB above largest spectral value in passband. (used for
    axes calculation)
BUFFER_MIN=0.5; %Spacing in dB below smallest spectral value in passband. (used for
    axes calculation)
PAUSE=.01; %Time in seconds to pause between successive animation frames

if (dBmHzFLAG==0),
    dBmHz=0;
    yaxis='dBm';
else
    dBmHz=10*log10(BW*(10^6)/1024);
    yaxis='dBm/Hz';
end

%Frequency vector used to plot spectrums
f=linspace(Fc-.5*BW,Fc+.5*BW,1024);

IntFrames=ActualIntegrationFrames;

%Select the proper baseline smoothing vector to use
if (DECIMATION_RATE==64),
    cal=cal64;
elseif (DECIMATION_RATE==32),
    cal=cal32;
elseif (DECIMATION_RATE==16),
    cal=cal16;
elseif (DECIMATION_RATE==8),
```

```
    cal=cal8;
elseif (DECIMATION_RATE==4),
    cal=cal4;
else %DECIMATION_RATE==2
    cal=cal2;
end

%Power calibration constant for one block of data
GainConstant=GetCalConst(IntFrames,Gvar);

fclose(FID1);
fclose(FID2);
fclose(FID3);

%initialize vectors to be used for cumulative integration vectors
total_dhat=zeros(1024,1);
total_a=zeros(1024,1);
%initialize vector for adaptive filter h[n]
h_temp=zeros(2*P,1);

loopCnt_dhat=zeros(NumberBlocks,1);
loopCnt_a=zeros(NumberBlocks,1);

temp1=zeros(1024,1);
temp2=zeros(1024,1);
temp3=zeros(1024,1);
temp4=zeros(1024,1);

%Vector used for plotting adaptive filter
h_index=P-1:-1:0;

h=complex(zeros(NumberBlocks,P),zeros(NumberBlocks,P));
blocksize=1025+2*P;

%Determine the maximum and minimum filter tap values
for k=1:NumberBlocks,
    h_temp=dhat(((k-1)*blocksize+1):((k-1)*blocksize+2*P));
    max_h_temp=max(h_temp);
    min_h_temp=min(h_temp);
    h(k,:)=complex(h_temp(1:2:2*P),h_temp(2:2:2*P))';
    if (k==1),
        max_h=max_h_temp;
        min_h=min_h_temp;
    else
        if (max_h_temp>max_h),
            max_h=max_h_temp;
        end
        if (min_h_temp<min_h),
            min_h=min_h_temp;
        end
    end
end
h_limit=max(abs(max_h),abs(min_h));

%Find appropriate axes to use with the animations
z=0;
for k=1:NumberBlocks,
        tempDHAT=(fftshift(dhat(((k-1)*blocksize+2+2*P):k*blocksize)))./cal;
        tempA=(fftshift(a(((k-1)*1025+2):k*1025)))./cal;
        if ((k>SKIP)),
            z=z+1;
            GainConstant_dhat=GetCalConst(z*IntFrames,Gvar);
            GainConstant_a=GetCalConst(z*IntFrames,Gvar);
            total_dhat=total_dhat+tempDHAT;
            total_a=total_a+tempA;

            temp_total_D=10*log10(total_dhat/GainConstant_dhat);
            temp_total_A=10*log10(total_a/GainConstant_a);
            max_1_temp=max([temp_total_D(148:880); temp_total_A(148:880)]);
            min_1_temp=min([temp_total_D(148:880); temp_total_A(148:880)]);
            if (z==1),
                max_1=max_1_temp;
                min_1=min_1_temp;
            else
                if (max_1_temp>max_1),
```

```
                    max_1=max_1_temp;
                end
                if (min_1_temp<min_1),
                    min_1=min_1_temp;
                end
            end
            temp_D=10*log10(tempDHAT/GainConstant);
            temp_A=10*log10(tempA/GainConstant);
            max_2_temp=max([temp_D(148:880); temp_A(148:880)]);
            min_2_temp=min([temp_D(148:880); temp_A(148:880)]);
            if (z==1),
                max_2=max_2_temp;
                min_2=min_2_temp;
            else
                if (max_2_temp>max_2),
                    max_2=max_2_temp;
                end
                if (min_2_temp<min_2),
                    min_2=min_2_temp;
                end
            end
        end
end
total_dhat=zeros(1024,1);
total_a=zeros(1024,1);

%Set the axes to be used for animations
if same_AXES==0,
    AXES_CUM=[min(f) max(f) min_1-BUFFER_MIN-dBmHz max_1+BUFFER_MAX-dBmHz];
    AXES_INST=[min(f) max(f) min_2-BUFFER_MIN-dBmHz max_2+BUFFER_MAX-dBmHz];
else
    max_final=max([max_1 max_2]);
    min_final=min([min_1 min_2]);
    AXES_CUM=[min(f) max(f) min_final-BUFFER_MIN-dBmHz max_final+BUFFER_MAX-dBmHz];
    AXES_INST=AXES_CUM;
end

axis1=[AXES_CUM];%subplot(231)
axis2=[AXES_CUM];%subplot(234)
axis3=[AXES_INST];%subplot(232)
axis4=[AXES_INST];%subplot(235)

figure(1);clf;
%Setup of the graphs if in XOR setting
if (GRAPHMODE==1),
    subplot(231);
    h1=plot(f,temp1,'Erasemode','xor');
    xlabel('Frequency (MHz)');
    ylabel(yaxis);
    title1=title('Filtered Output (Cumulative Integration)');
    axis(axis1);
    hold on;
    grid on;

    subplot(234);
    h2=plot(f,temp2,'Erasemode','xor');
    xlabel('Frequency (MHz)');
    ylabel(yaxis);
    title('Signal + Interference (Cumulative Integration)');
    axis(axis2);
    hold on;
    grid on;

    subplot(232);
    h3=plot(f,temp3,'Erasemode','xor');
    xlabel('Frequency (MHz)');
    ylabel(yaxis);
    title('Filtered Output');
    axis(axis3);
    hold on;
    grid on;

    subplot(235);
    h4=plot(f,temp4,'Erasemode','xor');
    xlabel('Frequency (MHz)');
```

```
    ylabel(yaxis);
    title('Signal + Interference');
    axis(axis4);
    hold on;
    grid on;

    subplot(233);
    h5=stem(h_index,zeros(P,1));
    grid on;
    hold on;
    hmark5=h5(1);
    hline5=h5(2);
    set(hmark5,'Erasemode','xor');
    set(hline5,'Erasemode','xor');
    h_lines5=get(hline5,'Ydata');
    title('Real Taps of h[n]');
    xlabel('n');
    ylabel('h_{imag}[n]');
    axis([-1 P+1 -h_limit h_limit]);

    subplot(236);
    h6=stem(h_index,zeros(P,1));
    grid on;
    hold on;
    hmark6=h6(1);
    hline6=h6(2);
    set(hmark6,'Erasemode','xor');
    set(hline6,'Erasemode','xor');
    h_lines6=get(hline6,'Ydata');
    title('Imaginary Taps of h[n]');
    xlabel('n');
    ylabel('h_{real}[n]');
    axis([-1 P+1 -h_limit h_limit]);
end

z=0;

for k=1:NumberBlocks,
        %These arrays contain the LoopCnt at the time the data was sent
        loopCnt_dhat(k)=dhat((k-1)*blocksize+1+2*P);
        loopCnt_a(k)=a((k-1)*1025+1);

        temp1=(fftshift(dhat(((k-1)*blocksize+2+2*P):k*blocksize)))./cal;
        temp2=(fftshift(a(((k-1)*1025+2):k*1025)))./cal;
        if ((k>SKIP)),
            z=z+1;
            GainConstant_dhat=GetCalConst(z*IntFrames,Gvar);
            GainConstant_a=GetCalConst(z*IntFrames,Gvar);
            total_dhat=total_dhat+temp1;
            total_a=total_a+temp2;
            if (GRAPHMODE==0),
                subplot(231);
                plot(f,10*log10(total_dhat/GainConstant_dhat)-dBmHz);
                xlabel('Frequency (MHz)');
                ylabel(yaxis);
                title1=title('Filtered Output (Cumulative Integration)');
                axis(axis1);
                grid on;

                subplot(234);
                plot(f,10*log10(total_a/GainConstant_a)-dBmHz);
                xlabel('Frequency (MHz)');
                ylabel(yaxis);
                title('Signal + Interference (Cumulative Integration)');
                axis(axis2);
                grid on;
            else
                set(h1,'Ydata',10*log10(total_dhat/GainConstant_dhat)-dBmHz);
                set(h2,'Ydata',10*log10(total_a/GainConstant_a)-dBmHz);
            end
        end
        if (GRAPHMODE==0),
            subplot(232);
            plot(f,10*log10(temp1/GainConstant)-dBmHz);
            xlabel('Frequency (MHz)');
```

```
            ylabel(yaxis);
            title('Filtered Output');
            axis(axis3);
            grid on;

            subplot(235);
            plot(f,10*log10(temp2/GainConstant)-dBmHz);
            xlabel('Frequency (MHz)');
            ylabel(yaxis);
            title('Signal + Interference');
            axis(axis4);
            grid on;
        else
            set(h3,'Ydata',10*log10(temp1/GainConstant)-dBmHz);
            set(h4,'Ydata',10*log10(temp2/GainConstant)-dBmHz);
        end

        if (GRAPHMODE==0),
            subplot(233);
            stem(h_index,real(h(k,:)));
            title('Real Taps of h[n]');
            xlabel('n');
            ylabel('h_{imag}[n]');
            axis([-1 P+1 -h_limit h_limit]);

            subplot(236);
            stem(h_index,imag(h(k,:)));
            title('Imaginary Taps of h[n]');
            xlabel('n');
            ylabel('h_{real}[n]');
            axis([-1 P+1 -h_limit h_limit]);
        else
            temp=real(h(k,:));
            h_lines5(2:3:end)=temp;
            set(hmark5,'Ydata',temp);
            set(hline5,'Ydata',h_lines5);

            temp=imag(h(k,:));
            h_lines6(2:3:end)=temp;
            set(hmark6,'Ydata',temp);
            set(hline6,'Ydata',h_lines6);
        end
        pause(PAUSE);
end

%Create figure containing the input (alpha[n]), output (dhat[n]), and reference (x[n
    ]) signals
figure(2);

dhat_final=10*log10(total_dhat/GainConstant_dhat);
a_final=10*log10(total_a/GainConstant_a);
Lower_Limit=min([dhat_final(148:880);a_final(148:880)])-BUFFER_MIN-dBmHz;
Upper_Limit=max([dhat_final;a_final])+BUFFER_MAX-dBmHz;

subplot(311);
plot(f,dhat_final-dBmHz);
title(['d[n]: Integration Time =',num2str(ActualIntegrationTime_dhat_alpha*z),'
    seconds, P =',num2str(P),', \mu =',num2str(MU),', BW =',num2str(
    DecimatedSampleRate/1e6),' MHz']);
xlabel('Frequency (MHz)');
ylabel(yaxis);
axis([min(f) max(f) Lower_Limit Upper_Limit]);

subplot(312);
plot(f,a_final-dBmHz);
title(['\alpha[n]: Delay =',num2str(DELAY_C),' samples, Integration Time =',num2str
    (ActualIntegrationTime_dhat_alpha*z),' seconds']);
xlabel('Frequency (MHz)');
ylabel(yaxis);
axis([min(f) max(f) Lower_Limit Upper_Limit]);

ActualIntegrationTime_x = x(1)*1024/(DecimatedSampleRate);

subplot(313);
GainConstant_x=GetCalConst(x(1),Gvar);
```

```
x_final=10*log10(fftshift(x(2:1025)/GainConstant_x)./cal);
Lower_Limit=min(x_final(148:880))-BUFFER_MIN-dBmHz;
Upper_Limit=max(x_final)+BUFFER_MAX-dBmHz
plot(f,x_final-dBmHz);
title(['x[n]:␣Delay␣=␣',num2str(DELAY_A),'␣samples,␣Integration␣Time␣=␣',num2str(x(1)
    *1024/(BW*(10^6))),'␣seconds']);
xlabel('Frequency␣(MHz)');
ylabel(yaxis);
axis([min(f) max(f) Lower_Limit Upper_Limit]);
```

A.3.4 Store Raw Time-domain Data Source Code

I have also developed an application which fills the DSP memory with raw time-domain complex data, then sends this data to the host PC for storage and post-processing.

This section includes the following source code files:

- 4291_330e_mod.cmd—page 316
- byusync.c—page 318

A routine entitled apihost.c, similar to that shown on page 309, is part of a Visual C++ application enabling data transfer from the DSP platform to host PC. The following source code files are used in this application, but are included in Section A.3.5:

- P6216SetRcvrParams.c—page 322
- waitForSync.c—page 322
- P6216SetBoardParams.c—page 323
- calcBifoClock.c—page 324
- syncBCD.c—page 324
- syncABD.c—page 325

4291_330e_mod.cmd

```
-x
-l dev6xsne.lib
-l c6xe.lib
-l vime.lib
-l p4291_330e.lib
-heap   0x1000
-stack  0x1000
-l os90e.lib
-l stdioe.lib
-l snioe.lib
-l swftnte.lib
-l ose.lib
-l rts6701e.lib
-l shmeme.lib
-l rtapie.lib

/*
 * Specify the sections of the Model 4291-330 memory.
 */
```

```
MEMORY
{
    PNKG  (RWIX) : org = 0x00000000,  len = 0x0000c400 /*Reserved for SwifNet*/
    GRAM  (RWIX) : org = 0x0000c400,  len = 0x001f3bff /* only if GBPR =0x40 */
    IPRAM  (RWIX) : org = 0x01400000, len = 0x00010000
    TVEC  (RWIX) : org = 0x02000000,  len = 0x00000400
    SDRAM (RW) : org = 0x02000400,  len = 0x00ff9c00
    PNK   (RWIX) : org = 0x02ffa000,  len = 0x00006000 /*SwiftNet Kernel*/
                                                        /* Mirrored at
                                                           0x2f03000 */
    SBSRAM (RWIX) : org = 0x01000000, len = 0x00080000
    IDRAM  (RWIX) : org = 0x80000000,  len = 0x0000e000
}

/*
 * Specify the allocation of the sections in the Model 4290 memory.
 */

SECTIONS
{
    .buffer0    :     > IDRAM
    .globaldata :     > IDRAM
    .buffer1    :     > IDRAM align(0x8000)  /* When using double buffers,
                                                  place the second buffer in
                                                  Block 1 IDRAM for optimal
                                                  performance.
                                               */
    .sbsram0    :       > SDRAM
    .gram0              :       > GRAM
    .text               :       > IPRAM
        .cinit          :       > SBSRAM
        .const          :       > SBSRAM
        .switch         :       > SBSRAM
        .data           :       > IDRAM
        .bss            :       > IDRAM
    .cio        :   > IDRAM
    .sysmem     :   > IDRAM
    .far        :   > IDRAM
        .stack          :       > IDRAM
        .xref           :       > IDRAM

        .tvt            :
                        {
                                _traptbl = .;
                        }> TVEC
        .dram0:
                        {

                                _RTAPI_MAXOUT = 0x80000;
                        _RTAPI_MAXIN = 0x0;

                                _shmem1_base = .;
                                _shmem1_top = . + 0x1f3bff;
                                _shmem2_base = 0;
                                _shmem2_top = 0;
                                _shmem3_base = 0;
                                _shmem3_top = 0;
                                _shmem4_base = 0;
                                _shmem4_top = 0;
                        } > GRAM

/* ISR Jump table used by SwiftNet resides here */
        .ivec:
                        {
                                _isr_jump_table = .;
                        }> PNK
}
```

byusync.c

```
/*************************************************************************
*
*   File : byusync.c (Fill DSP memory and dump to host PC)
*
*************************************************************************/
#include "stdlib.h"
#include "4290.h"
#include "4290rscm.h"
#include "4290bifo.h"
#include "4290mbx.h"
#include "6216.h"
#include "math.h"
#include "c6xtimer.h"
#include "4290dma.h"
#include "c6xdma.h"
#include "dma.h"
#include <rtapi.h>

/*  ******* GLOBAL DEFINES ********************  */

/* Number of 6216 VIM Modules being Synchronized */
#define  NUM_6216_VIM_MODULES     2             /* 1 or 2 6216's on 4290   */

/* Synchronization Master Processor - Must be Processor A or C */
#define  SYNC_MASTER              P4290_PROC_A  /* 0 or 2 for CPU's A & C  */

/* Clock Selection - P6216_INTERNAL_CLOCK or P6216_EXTERNAL_CLOCK */
#define CLOCK_SELECT    P6216_INTERNAL_CLOCK

/* The Internal Oscillator Frequency can vary based on installed options
   on the board.
*/
#define INT_OSC_STANDARD      64000000L      /* 6216 Standard       */
#define INT_OSC_OPT20         65000000L      /* 6216 with Option 20 */
#define INT_OSC_OPT21         60000000L      /* 6216 with Option 21 */

/* External Clock Rate - When the 6216 is being clocked externally or it is
                         receiving it's clock from another 6216, specify the
                         external clock frequency.
*/
#define EXTERNAL_CLOCK_RATE   64000000L


/* ***********  GLOBAL VARIABLES ******************  */

/* Global Data Buffers */

#define BUFFER_SIZE   1024

#pragma DATA_SECTION(outData, ".sbsram0");
far int outData[4089*BUFFER_SIZE];
#pragma DATA_SECTION(streamFLAG, ".gram0");
far int streamFLAG;

/* Other global variables  -- */
int dmaDone=0;
volatile int       loopCnt=0;
volatile int       waitCnt=0;

/* ************ Function Prototypes *************** */

void syncABD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void syncBCD(unsigned int numVimModules, unsigned int mailbox,
             unsigned int syncWord);
void waitForSync(unsigned int mailbox, unsigned int syncWord);
int calcBifoClock(P6216_BOARD_PARAMS *boardParams,
                  P6216_RCVR_PARAMS *rcvrParams);
void P6216SetBoardParams(P6216_BOARD_PARAMS *boardParams);
void P6216SetRcvrParams(P6216_RCVR_PARAMS *rcvrParams,
                        P6216_BOARD_PARAMS *boardParams,
                        unsigned int intClockRate);
```

```
/* Interrupt service routine for BIFIFO dma */
interrupt void ReadBifoISR() {

        /* Reset DMA frame and block interrupt conditions on read channel */
        *((unsigned int *)(DMA_SECONDARY_CTRL_ADDR(DMA_CH2))) &= 0xffbb;

        /* Set dma complete flag */
        dmaDone=1;
}  /* end of ISR */

void main()
{
    P6216_BOARD_PARAMS p6216BoardParams;
    P6216_RCVR_PARAMS  p6216RcvrParams;
    P6216_REG_ADDR     p6216Regs;
    unsigned int       procId = P4290GetProcId();
    volatile float     phase;
    int                bifoClock;
    unsigned int       intClockRate = INT_OSC_STANDARD;
        P4290_DMA_PARAMS   dmaParams;
    /*streaming*/
    int                    j;
    int                            outputstream=0;
    int                            connected=0;
    int                            streamstatus=0;
    int                            STREAMerror=0;
    int                            finished=0;

    P4290_LED1_OFF;
    P4290_LED2_OFF;
    P4290_LED3_OFF;

    /* Turn off timer 0 - Enabled by bootcode to toggle led */
    TIMER_RESET(C6X_TIMER_0);

    /* Initialize Table of 6216 Addresses */
    P6216InitRegAddr(0x320000L, &p6216Regs);

    /* Determine whether option is installed on this board */
    if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x21))
        intClockRate = INT_OSC_OPT21;
    else if (VIMIsOptionInstalled(p6216Regs.eepromControl, 0x20))
        intClockRate = INT_OSC_OPT20;

    /* Set 6216 Board Parameters */
    P6216SetBoardParams(&p6216BoardParams);

    /* Get 6216 Receiver Parameters */
    P6216SetRcvrParams(&p6216RcvrParams, &p6216BoardParams, intClockRate);

        /* Set Center Frequency and Decimation  */
    p6216RcvrParams.centerFreq = 2.0e6;
    p6216RcvrParams.decimationRate = 64; /* Set for 16 MHz basband fs */
    p6216BoardParams.gainSetting = 6;

    /* Reset Board Registers */
    P6216ResetRegs(&p6216Regs);

    /* Initialize Board Registers */
    P6216InitBoardRegs(&p6216BoardParams, &p6216Regs);

    /* Initialize Receiver Registers */
    P6216InitRcvrRegs(&p6216RcvrParams, &p6216Regs);

    /* Calculate Slowest Bifo Clock Rate */
    bifoClock = calcBifoClock(&p6216BoardParams, &p6216RcvrParams);

        #ifndef OPTION_330
    /* Select IO Mezzanine Bifo */
    P4290BifoSelect(P4290_BIFO_IO);
        #endif

        /* ********** SETUP INTERRUPTS AND DMA TRASFER REGISTERS ****** */

    /* Clear interrupt enable registers. */
```

```
*P4290_LCR_IER0 = 0x0;
*P4290_LCR_IER1 = 0x0;

 /* Reset Interrupt Mapping Reg */
*P4290_LCR_IMR0 = 0;

/* Reset Miscellaneous Control Register. */
/* Also causes interrupts to be unlatched, which is why this example
   will not run on the standard 4290. */
*P4290_LCR_MCR0 |= 0x00f0;

    /* Set up Bifo almost full interrupt */
P4290SetupBifoDMAInt(P4290_BIFO_IO, P4290_INT_EXP_BIFO_IN_ALMST_FULL,
        P4290_IMR0_IO_BIFO_IN_EVENT, CPU_INT4, ISN_EXT_INT4, NULL);

P4290DmaInit((unsigned int)P4290_LCL_FIFO_IO, (unsigned int)outData,
        DMA_ADDR_NO_MOD, DMA_ADDR_INC, DMA_INDXA, 4,
        DMA_SPLIT_DIS, 0, DMA_ESIZE32, BUFFER_SIZE, 4089,
        DMA_CNT_RELOADA, FRAME_SYNC_ENABLE, DMA_RELOAD_NONE, DMA_RELOAD_NONE,
        SEN_EXT_INT4, SEN_NONE, 0X2080, C6X_DMA_IE,
        DMA_AUTOINIT_FALSE, &dmaParams);

/*  Setup DMA complete interrupt */
/*     Note DMA chan. 2 is hard-wired to internal interrupt CPU_INT10.
       This is the same as DMA_INT2.   */
C6xSetupInterrupt(CPU_INT10,ISN_DMA_INT2,&ReadBifoISR,0);
C6xEnableInterrupt(CPU_INT10);

    /* ***************** SYNCHRONIZE THE 4 PROCESSORS *************** */

/* Disable Fifo Writes */
P6216_DISABLE_FIFO_WRITE(p6216Regs.control);

/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
if (procId == SYNC_MASTER)
{
    if (procId == P4290_PROC_A)
            syncBCD(NUM_6216_VIM_MODULES, 1, 0xabababab);
    else  /* Sync Master is C */
            syncABD(NUM_6216_VIM_MODULES, 1, 0xabababab);
}
else
{
    waitForSync(1, 0xabababab);
}

P4290FlushBifo(P4290_BIFO_IO, bifoClock, 2*BUFFER_SIZE - 4, 128,
             2*BUFFER_SIZE - 4, 128);
/* Before proceeding with the Sync Initialization, wait for all the
       other processors to be at this point in the initialization process.
*/
if (procId == SYNC_MASTER)
{
    if (procId == P4290_PROC_A)
            syncBCD(NUM_6216_VIM_MODULES, 1, 0xdcdcdcdc);
    else  /* Sync Master is C */
            syncABD(NUM_6216_VIM_MODULES, 1, 0xdcdcdcdc);
    }
else
{
    waitForSync(1, 0xdcdcdcdc);
}

/* If Sync 1 Master, Generate Sync Pulse */
if (procId == SYNC_MASTER)
{
    P6216_SET_SYNC_GENERATE(p6216Regs.syncGen, 0);
    while ((*p6216Regs.syncGen & 0x1) != 0);
    P6216_SET_SYNC_GENERATE(p6216Regs.syncGen, 1);
}

    /* ************* MAIN I/O TRANSFER LOOP, ON DMA INTERRUPT *** */
```

```
    /* Set up for DMA BIFO read with synchronization with IO-In AF. */
    P4290OurDmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_AUTO_START_VAL,
        CPU_INT4, ISN_EXT_INT4, &dmaParams);

    if (procId == P4290_PROC_A)   {
        *P4290_LCR_GBPR = 0x40;
    }
    else if (procId == P4290_PROC_B)   {
        streamFLAG=0;
    }
    while (1)
    {

                /* wait for dma to complete */
        while (!dmaDone){
                ++waitCnt;
        }
        P4290OurDmaTransfer(DMA_CH2, C6X_DMA_NO_WAIT, DMA_PAUSE_VAL,
               CPU_INT4, ISN_EXT_INT4, &dmaParams);
        dmaDone = 0;
                P4290_LED1_ON;
                ++loopCnt;
                waitCnt = 0;

                if (procId != P4290_PROC_A) {
            for (j=0;j<(procId*2000);j++);
            while (streamFLAG!=procId) {
                        for (j=0;j<6000;j++);
                }
        }

        P4290_LED2_ON;
        outputstream = rt_open(":stream:0",O_NDELAY,0);
        if (outputstream == -1 ) { exit(255); }
        while(!connected) {
                rt_ioctl(outputstream, I_CONNECT, &connected);
        }
        for (j=0;j<4089;j++) {
                streamstatus=-1;
                while (streamstatus==-1) {
                        streamstatus = rt_write(outputstream, (char*)&outData[1024*j
                            ],4096);
                        if ((streamstatus != 4096)&&(streamstatus != -1))
                                STREAMerror=1;
                }
        }
        rt_ioctl(outputstream, O_FLUSH, &finished);
        P4290_LED3_ON;
        streamFLAG=procId+1;
        if (procId==P4290_PROC_D)
                rt_close(outputstream);

    }
}
```

A.3.5 Functions Common to Multiple DSP Applications

This section includes the following source code files used in multiple DSP applications:

- power_fft.c—page 326
- power_fft_opt.asm—page 327
- cfftr2_e.asm [48]—page 329

P6216SetRcvrParams.c, waitForSync.c, P6216SetBoardParams.c, calcBifoClock.c, syncBCD.c, and syncABD.c were written by Pentek for the BYU DSP platform as a part of the purchase package [30].

P6216SetRcvrParams.c

```
/*************************************************************************
 Function: P6216SetRcvrParams

 Description:  This routine initializes the P6216_RCVR_PARAMS Data Structure
               with desired settings.
 Inputs:
         rcvrParams - Address to P6216_RCVR_PARAMS Data Structure.
         boardParams - Address to P6216_BOARD_PARAMS Data Structure.
         intClockRate - Internal Clock Rate of on-board oscillator
 Outputs:
 Returns:
 Notes:
*************************************************************************/
void P6216SetRcvrParams(P6216_RCVR_PARAMS *rcvrParams,
                        P6216_BOARD_PARAMS *boardParams,
                        unsigned int intClockRate)
{
    /* Set Input Sample Frequency */
    if (boardParams->clockSelect == P6216_INTERNAL_CLOCK)
        rcvrParams->inputSmplFreq = (float) (intClockRate /
                                             boardParams->masterClkDiv);
    else if (boardParams->syncControl == P6216_SLAVE_SYNC)
        rcvrParams->inputSmplFreq = (float) EXTERNAL_CLOCK_RATE;
    else   /* External Clock and Master */
        rcvrParams->inputSmplFreq = (float) (EXTERNAL_CLOCK_RATE /
                                             boardParams->masterClkDiv);

    rcvrParams->centerFreq = 1.0e6;
    rcvrParams->decimationRate = 64;
    rcvrParams->dataMode = P6216_COMPLEX_DATA;
    rcvrParams->flipSpectrum = FLIP_SPECTRUM;
    rcvrParams->offsetSpectrum = P6216_NO_OFFSET_SPECTRUM;

    /* Set GC1012's Sync Mode to reset on System Sync. */
    rcvrParams->rcvrSyncMode = P6216_GC_AS_ON | P6216_GC_AS_MUX |
                               P6216_GC_AS_FREQ | P6216_GC_LD_FREQ;
}
```

waitForSync.c

```
/*************************************************************************
 Function: waitForSync

 Description:  This routine is called by a processor which needs to
               synchronize  with a master processor.  This function
               waits for data from a mailbox to match with a specified
               syncWord pattern. When syncWord is received. it echoes
               it back to the global bus.
 Inputs:
         mailbox - Dual Port SRAM Mailbox used to handshake(0 - 3)
         syncWord - Sync Word used to perform handshake
 Outputs:
 Returns:
 Notes:
        Swiftnet uses Dual Port SRAM Mailbox 0 on all processors.
```

```
*****************************************************************************/
void  waitForSync(unsigned int mailbox, unsigned int syncWord)
{
    unsigned int temp;

    /* Wait for a mailbox entry to contain the specified Sync Word */
    do
    {
       while (P4290_GET_DPSRAM_MBX_INTR_STATUS(P4290_LCL_DPSRAM_MAILBOX,
                                               mailbox));
       temp = P4290_READ_DPSRAM_MBX(P4290_LCL_DPSRAM_MAILBOX,mailbox);
    } while (temp != syncWord);

    /* Once Sync Word is recieved, echo it back to Global Bus to be
       read by the master processor.
    */
    P4290_WRITE_DPSRAM_MBX(P4290_LCL_DPSRAM_MAILBOX, mailbox, syncWord);
}
```

P6216SetBoardParams.c

```
/*****************************************************************************
 Function: P6216SetBoardParams

  Description: This routine initializes the P6216_BOARD_PARAMS Data Structure
               with desired settings.
 Inputs:
         boardParams - Pointer to the P6216_BOARD_PARAMS structure being
                       initialized
 Outputs:
 Returns:
 Notes:
*****************************************************************************/
void P6216SetBoardParams(P6216_BOARD_PARAMS *boardParams)
{
    unsigned int procId = P4290GetProcId();

    boardParams->lpfBypass = P6216_LPF_ENABLE;
    boardParams->ddrBypass = P6216_DDR_ENABLE;
    boardParams->dataPacking = P6216_UNPACKED_DATA;
    boardParams->gainSetting = 0.0;
    boardParams->fifoDecimation = 1;
    boardParams->masterClkDiv = MASTER_ClkDiv;
    boardParams->clockSelect = CLOCK_SELECT;
    boardParams->serialPortConn = P6216_SP_NO_CONNECT;

    /* If processor is Sync Master, then disable the sync bus termination */
    /* If processor isn't Sync Master, Then enable the termination        */
    /* This logic only works for two 6216 VIM Modules.                    */
    if (procId == SYNC_MASTER)
    {
        /* If synchronizing multiple VIM Modules, only terminate the slave
           VIM Modules.  If only one VIM Module, then enable termination.
        */
        if (NUM_6216_VIM_MODULES == 2)
            boardParams->syncBusTermination = P6216_SYNC_TERM_DISABLE;
        else
            boardParams->syncBusTermination = P6216_SYNC_TERM_ENABLE;
        boardParams->syncControl = P6216_MASTER_SYNC;
    }
    else
    {
        boardParams->syncBusTermination = P6216_SYNC_TERM_ENABLE;
        boardParams->syncControl = P6216_SLAVE_SYNC;
    }
    /* Enable System Sync */
    boardParams->syncMask = P6216_SSYNC_ENABLE;

}
```

calcBifoClock.c

```
/*************************************************************************
 Function: calcBifoClock

 Description:  This routine calculates the slowest bifo clock which
               is clocking the BIFO between the VIM Module and the processor.

 Inputs:
         boardParams - Board Paramater settings.
         rcvrParams - GC1012 Parameter settings.
 Outputs:
 Returns:
         bifoClock - Slowest Bifo Clock Setting.
 Notes:
*************************************************************************/
int calcBifoClock(P6216_BOARD_PARAMS *boardParams,
                  P6216_RCVR_PARAMS *rcvrParams)
{
    int bifoClock;

    if (boardParams->ddrBypass == P6216_DDR_BYPASS)
    {
        bifoClock = (int) rcvrParams->inputSmplFreq;
    }
    else
    {
        /* Calculate Bifo Clock Rate */
        bifoClock = (int)(rcvrParams->inputSmplFreq /
                          rcvrParams->decimationRate);
    }

    /* Set Bifo Clock Rate equal the slower of the two bifo clocks */
    if (bifoClock > P4290_BIFO_CLOCK_FREQ)
        bifoClock = P4290_BIFO_CLOCK_FREQ;

    return(bifoClock);
}
```

syncBCD.c

```
/*************************************************************************
 Function: syncBCD

 Description:  This routine is used to synchronize Processor A with Processor's
               B, C, and D.  The function uses a Dual Port SRAM Mailbox to
               perform the handshake.  This simple handshake consists of
               Processor A writing to Processor B, C, and D's mailboxes
               and waiting for a response.
 Inputs:
         numVIMModules - The number of VIM Modules being SYNC'D (1 or 2)
         mailbox - Dual Port SRAM Mailbox used to handshake(0 - 3)
         syncWord - Sync Word used to perform handshake
 Outputs:
 Returns:
 Notes:
        Swiftnet uses Dual Port SRAM Mailbox 0 on all processors.
*************************************************************************/
void syncBCD(unsigned int numVIMModules, unsigned int mailbox,
             unsigned int syncWord)
{
    volatile unsigned int *mbx;
    unsigned int          mbxAddr;
    unsigned int          i;
    unsigned int          temp;
    unsigned int          prevGBPR;

    /* Set Global Page Register to provide access to Dual Port Ram Mailbox */
    mbxAddr = P4290SetupGlobalAccess((unsigned int)P4290_GBL_DPSRAM_A_MAILBOX,
                                     P4290_DPSRAMA_MAILBOX_1, &prevGBPR);

    /* If there is another 6216, handshake with the processors which
       are connected to it(C and D).
```

```
    */
    if (numVIMModules == 2)
    {
        /* Modify mailbox address to point to Processor C's Mailbox's */
        mbx = (unsigned int *)(mbxAddr + 0x80000);
        for (i = 0; i < 2; ++i)
        {
            mbx += 0x20000 * i;

            /* Check to see if Mailbox has any data in it,
               if so clear it out
            */
            temp = P4290_GET_DPSRAM_MBX_INTR_STATUS(mbx,mailbox);
            if (!temp)
                temp = P4290_READ_DPSRAM_MBX(mbx,mailbox);

            /* Write SyncWord to Processor's Mailbox */
            P4290_WRITE_DPSRAM_MBX(mbx, mailbox, syncWord);

            /* Wait for Processor to respond back with the same sync word */
            do
            {
                while (P4290_GET_DPSRAM_MBX_INTR_STATUS(mbx,mailbox));
                temp = P4290_READ_DPSRAM_MBX(mbx,mailbox);
            } while (temp != syncWord);
        }
    }
    /* Modify mailbox address to point to Processor B's Mailbox */
    mbx = (unsigned int *)(mbxAddr + 0x180000);

    /* Check to see if Mailbox has any data in it, if so clear it out */
    temp = P4290_GET_DPSRAM_MBX_INTR_STATUS(mbx,mailbox);
    if (!temp)
        temp = P4290_READ_DPSRAM_MBX(mbx,mailbox);

    /* Write Sync Word to Processor B's Mailbox */
    P4290_WRITE_DPSRAM_MBX(mbx, mailbox, syncWord);

    /* Wait for Processor B to respond back with the same sync word */
    do
    {
        while (P4290_GET_DPSRAM_MBX_INTR_STATUS(mbx,mailbox));
        temp = P4290_READ_DPSRAM_MBX(mbx,mailbox);
    } while (temp != syncWord);

}
```

syncABD.c

```
/*****************************************************************************
 Function: syncABD

 Description:  This routine is used to synchronize Processor C with Processor's
               A, B, and D.  The function uses a Dual Port SRAM Mailbox to
               perform the handshake.  This simple handshake consists of
               Processor C writing to Processors A, B, and D's mailboxes
               and waiting for a response.
 Inputs:
         numVIMModules - The number of VIM Modules being SYNC'D (1 or 2)
         mailbox - Dual Port SRAM Mailbox used to handshake(0 - 3)
         syncWord - Sync Word used to perform handshake
 Outputs:
 Returns:
 Notes:
        Swiftnet uses Dual Port SRAM Mailbox 0 on all processors.
*****************************************************************************/
void syncABD(unsigned int numVIMModules, unsigned int mailbox,
             unsigned int syncWord)
{
    volatile unsigned int *mbx;
    unsigned int          mbxAddr;
    unsigned int          i;
    unsigned int          temp;
    unsigned int          prevGBPR;
```

```
    mbxAddr = P4290SetupGlobalAccess((unsigned int)P4290_GBL_DPSRAM_A_MAILBOX,
                                     P4290_DPSRAMA_MAILBOX_1, &prevGBPR);

    /* If there is another 6216, handshake with the processors which
       are connected to it(A and B).
    */
    if (numVIMModules == 2)
    {
        /* Modify mailbox address to point to Processor A's Mailbox's */
        mbx = (unsigned int *)mbxAddr;

        for (i = 0; i < 2; ++i)
        {
            mbx += 0x60000 * i;

            /* Check to see if Mailbox has any data in it,
               if so clear it out
            */
            temp = P4290_GET_DPSRAM_MBX_INTR_STATUS(mbx,mailbox);
            if (!temp)
                temp = P4290_READ_DPSRAM_MBX(mbx,mailbox);

            /* Write SyncWord to Processor's Mailbox */
            P4290_WRITE_DPSRAM_MBX(mbx, mailbox, syncWord);

            /* Wait for Processor to respond back with the same sync word */
            do
            {
                while (P4290_GET_DPSRAM_MBX_INTR_STATUS(mbx,mailbox));
                temp = P4290_READ_DPSRAM_MBX(mbx,mailbox);
            } while (temp != syncWord);
        }
    }

    /* Modify mailbox address to point to Processor D's Mailbox */
    mbx = (unsigned int *)(mbxAddr + 0x100000);

    /* Check to see if Mailbox has any data in it, if so clear it out */
    temp = P4290_GET_DPSRAM_MBX_INTR_STATUS(mbx,mailbox);
    if (!temp)
        temp = P4290_READ_DPSRAM_MBX(mbx,mailbox);

    /* Write Sync Word to Processor D's Mailbox */
    P4290_WRITE_DPSRAM_MBX(mbx, mailbox, syncWord);

    /* Wait for Processor D to respond back with the same sync word */
    do
    {
        while (P4290_GET_DPSRAM_MBX_INTR_STATUS(mbx,mailbox));
        temp = P4290_READ_DPSRAM_MBX(mbx,mailbox);
    } while (temp != syncWord);

}
```

power_fft.c

```
/*Power_FFT computes the magnitude squared output of each PSD bin and reorders
the data.*/

void Power_FFT (float* x, volatile float* y, int* bitRevIndx, int N)
{
    unsigned int i,k;
    float tempI1,tempQ1;

        for (i=0; i<N; i++){
        k = (bitRevIndx[i]<<1);
        tempI1=x[k];
        tempQ1=x[k+1];
        y[i] = tempI1*tempI1 + tempQ1*tempQ1 + y[i];
        }
}
```

power_fft_opt.asm

```
;******************************************************************************
;* TMS320C6x ANSI C Codegen                                      Version 4.00 *
;* Date/Time created: Thu Dec 06 09:56:42 2001                                *
;******************************************************************************

;******************************************************************************
;* GLOBAL FILE PARAMETERS                                                     *
;*                                                                            *
;*   Architecture      : TMS320C670x                                          *
;*   Optimization      : Enabled at level 3                                   *
;*   Optimizing for    : Speed                                                *
;*                       Based on options: -o3, no -ms                        *
;*   Endian            : Big                                                  *
;*   Interrupt Thrshld : Disabled                                             *
;*   Memory Model      : Large                                                *
;*   Calls to RTS      : Far                                                  *
;*   Pipelining        : Enabled                                              *
;*   Speculative Load  : Enabled (Threshold = 8                              ) *
;*   Memory Aliases    : Presume are aliases (pessimistic)                    *
;*   Debug Info        : Debug                                                *
;*                                                                            *
;******************************************************************************

;void Power_FFT (float* x, volatile float* y, int* bitRevIndx, int N)
;{
;    unsigned int i,k;
;    float tempI1,tempQ1;
;
;       for (i=0; i<N; i++){
;       k = (bitRevIndx[i]<<1);
;       tempI1=x[k];
;       tempQ1=x[k+1];
;       y[i] = tempI1*tempI1 + tempQ1*tempQ1 + y[i];
;       }
;}

FP      .set    A15
DP      .set    B14
SP      .set    B15
        .global $bss

;       opt6x -q -e -v6700 -O3 C:\TEMP\TI195_2 C:\TEMP\TI195_4 -w C:\ti\myprojects\
    lisha
        .file   "power_fft.c"
        .sect   ".text"
        .global _Power_FFT
        .sym    _Power_FFT,_Power_FFT, 32, 2, 0
        .func   1

;******************************************************************************
;* FUNCTION NAME: _Power_FFT                                                  *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,B0,B4,B5                        *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,B0,B3,B4,B5,B6                  *
;*   Local Frame Size  : 0 Args + 0 Auto + 0 Save = 0 byte                    *
;******************************************************************************
_Power_FFT:
;** --------------------------------------------------------------------------*
        .sym    _x,4, 22, 17, 32
        .sym    _y,20, 22, 17, 32
        .sym    _bitRevIndx,6, 20, 17, 32
        .sym    _N,22, 4, 17, 32
        .sym    C$1,0, 6, 4, 32
        .sym    C$2,0, 4, 4, 32
        .sym    C$3,0, 6, 4, 32
        .sym    A$4,0, 6, 4, 32
        .sym    _x,0, 22, 4, 32
        .sym    _y,20, 22, 4, 32
        .sym    _bitRevIndx,4, 20, 4, 32
        .sym    _N,1, 4, 4, 32
        .sym    L$1,21, 4, 4, 32
        .sym    U$10,4, 20, 4, 32
        .sym    U$26,20, 22, 4, 32
```

```
           MV       .L1X     B6,A1             ;
||         MV       .S1      A6,A4             ;
||         MV       .D1      A4,A0             ;

   [!A1]   B        .S1      L4                ; |6|
           NOP              5
           ; BRANCH OCCURS                     ; |6|
;** --------------------------------------------------------------------------*
           LDW      .D1T1    *A4++,A5          ; (P) |10|
           ADD      .L2X     1,A1,B0           ;
           MV       .D1      A0,A6
           MVK      .S1      0x1,A2            ; init prolog collapse predicate
           SUB      .L1X     B0,1,A1           ;
           NOP              1
;*--------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Known Minimum Trip Count         : 1
;*      Known Max Trip Count Factor      : 1
;*      Loop Carried Dependency Bound(^) : 10
;*      Unpartitioned Resource Bound     : 3
;*      Partitioned Resource Bound(*)    : 3
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                     1        1
;*      .S units                     1        1
;*      .D units                     3*       2
;*      .M units                     2        0
;*      .X cross paths               0        1
;*      .T address paths             3*       2
;*      Long read paths              0        1
;*      Long write paths             0        0
;*      Logical  ops (.LS)           0        0     (.L or .S unit)
;*      Addition ops (.LSD)          1        1     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             1        1
;*      Bound(.L .S .D .LS .LSD)     2        2
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 10 Schedule found with 3 iterations in parallel
;*      done
;*
;*      Loop is interruptible
;*      Collapsed epilog stages     : 2
;*      Prolog not entirely removed
;*      Collapsed prolog stages     : 1
;*
;*      Minimum required memory pad : 8 bytes
;*
;*      Minimum safe trip count     : 1
;*--------------------------------------------------------------------------*
L1:    ; PIPED LOOP PROLOG
;** --------------------------------------------------------------------------*
L2:    ; PIPED LOOP KERNEL

   [!A2]   LDW      .D2T2    *B4,B5            ;  ^ |10|
||         SHL      .S1      A5,1,A5           ; @|10|

           ADDSP    .L1      A3,A0,A5          ; |10|
|| [ A1]   LDW      .D1T1    *+A6[A5],A0       ; @|10|
||         ADD      .S1      1,A5,A3           ; @|10|

   [ A1]   LDW      .D1T1    *+A6[A3],A3       ; @|10|
   [ B0]   SUB      .L2      B0,1,B0           ; |11|
   [ B0]   B        .S2      L2                ; |11|

           ADDSP    .L2X     B5,A5,B5          ;  ^ |10|
||         LDW      .D1T1    *A4++,A5          ; @@|10|

           MPYSP    .M1      A0,A0,A0          ; @|10|
           MPYSP    .M1      A3,A3,A3          ; @|10|
           NOP              1

   [ A2]   SUB      .L1      A2,1,A2           ;
|| [ A1]   SUB      .S1      A1,1,A1           ;
```

```
|| [!A2]   STW     .D2T2   B5,*B4++          ;  ^ |10|

;** --------------------------------------------------------------------------*
L3:    ; PIPED LOOP EPILOG
;** --------------------------------------------------------------------------*
L4:
           B       .S2     B3                ; |12|
           NOP             5
           ; BRANCH OCCURS                   ; |12|
        .endfunc        12,000000000h,0
```

cfftr2_e.asm [48]

```
*============================================================================
*
*       TEXAS INSTRUMENTS, INC.
*
*       RADIX-2 FFT (DIT)
*
*       Revision Date: 5/21/98
*
*       USAGE
*
*               This routine is C Callable and can be called as:
*
*               void cfftr2_dit( float *x, const float *w, short N)
*
*               x       Pointer to Array of Dimension 2*N elements holding
*                       Input to and Outputs from function cfftr2_dit()
*               w       Pointer to an array holding the coefficient (Dimension
*                       N/2 complex numbers)
*               N       Number of complex points in x
*
*               If routine is not to be used as a C callable function then
*               you need to initialize values for all of the values passed
*               as these are assumed to be in registers as defined by the
*               calling convention of the compiler, (refer to the C compiler
*               reference guide).
*
*               ARGUMENTS PASSED        ->   REGISTER
*               --------------------------------
*               x                       ->   A4
*               w                       ->   B4
*               N                       ->   A6
*
*       C CODE
*
*               This is the C equivalent of the assembly code.  Note that
*               the assembly code is hand optimized and restrictions may
*               apply.
*
*               void cfftr2_dit(float* x, float* w, short n)
*               {
*                  short n2, ie, ia, i, j, k, m;
*                  float rtemp, itemp, c, s;
*
*                  n2 = n;
*                  ie = 1;
*
*                  for(k=n; k > 1; k >>= 1)
*                  {
*                     n2 >>= 1;
*                     ia = 0;
*                     for(j=0; j < ie; j++)
*                     {
*                        c = w[2*j];
*                        s = w[2*j+1];
*                        for(i=0; i < n2; i++)
*                        {
*                           m = ia + n2;
*                           rtemp     = c * x[2*m]   + s * x[2*m+1];
*                           itemp     = c * x[2*m+1] - s * x[2*m];
*                           x[2*m]    = x[2*ia]   - rtemp;
*                           x[2*m+1]  = x[2*ia+1] - itemp;
```

```
*                          x[2*ia]   = x[2*ia]   + rtemp;
*                          x[2*ia+1] = x[2*ia+1] + itemp;
*                          ia++;
*                       }
*                       ia += n2;
*                    }
*                    ie <<= 1;
*                 }
*              }
*
*
*       DESCRIPTION
*
*              This routine performs the Decimation-in-Time (DIT) Radix-2 FFT
*              of the input array x.
*              x has N complex floating point numbers arranged as successive
*              real and imaginary number pairs. Input array x contains N
*              complex points (N*2 elements). The coefficients for the
*              FFT are passed to the function in array w which contains
*              N/2 complex numbers (N elements) as successive real and
*              imaginary number pairs.
*              The FFT Coefficients w are in N/2 bit-reversed order
*              The elements of input array x are in normal order
*              The assembly routine performs 4 output samples (2 real and 2
*              imaginary) for a pass through inner loop.
*
*              Note that (bit-reversed) coefficients for higher order FFT (1024
*              point) can be used unchanged as coefficients for a lower order
*              FFT (512, 256, 128 ... ,2)
*
*              The routine can be used to implement Inverse-FFT by any ONE of
*              the following methods:
*
*              1.Inputs (x) are replaced by their Complex-conjugate values
*                Output values are divided by N
*              2.FFT Coefficients (w) are replaced by their Complex-conjugates
*                Output values are divided by N
*              3.Swap Real and Imaginary values of input
*              4.Swap Real and Imaginary values of output
*
*       TECHNIQUES
*
*              1. The inner two loops are combined into one inner loop whose
*                 loop count is N/2.
*              2. The first 4 cycles of inner loop prolog are scheduled in
*                 parallel with the outer loop.
*              3. Load counter is not used, so extreneous loads are performed
*              4. Variables n and c share the register A6 and variables w and
*                 nsave share the register B4.
*
*       ASSUMPTIONS
*
*              N is a integral power of 2 (4, 8,16,32 ...) and the FFT
*              dimension (N) must atleast be 4.
*              The FFT Coefficients w are in bit-reversed order
*              The elements of input array x are in normal order
*              The imaginary coefficients of w are negated as {cos(d*0),
*              sin(d*0), cos(d*1), sin(d*1) ...} as opposed to the normal
*              sequence of {cos(d*0), -sin(d*0), cos(d*1), -sin(d*1) ...}
*              where d = 2*PI/N.
*
*       MEMORY NOTE
*
*              Arrays x (data) and w (coefficients) must reside in
*              different memory banks to avoid memory conflicts.  If Data and
*              Coefficents do reside in the same memory bank, add (N/2)+log2(N)+1
*              cycles to the cycles equation below. The memory bank hits are due to
*              scheduling of assembly code and also due to extreneous loads
*              causing bank hits.
*
*              Data and Coefficents must be aligned on an 8 byte boundary.
*
*       CYCLES
*
*              ((2*N) + 23)*log2(N) + 6
```

```
*
*               For N=1024, Cycles = 20716
*
*       NOTATIONS
*
*               f = Function Prolog or Epilog
*               o = Outer Loop
*               p = Inner Loop Prolog
*
*============================================================================
        .global _cfftr2_dit
        .text

_cfftr2_dit:

        STW     .D2T2   B14,*B15--(48)
        STW     .D2T2   B3,*+B15(4)
        STW     .D2T1   A11,*++B15(8)    ; switched odds
        STW     .D2T1   A10,*++B15(12)   ; and evens
        STW     .D2T1   A13,*++B15(16)   ; on all
        STW     .D2T1   A12,*++B15(20)   ; of these  for
        STW     .D2T1   A15,*++B15(24)   ; to match
        STW     .D2T1   A14,*++B15(28)   ; LDDW format
        STW     .D2T2   B11,*++B15(32)   ; big endian
        STW     .D2T2   B10,*++B15(36)   ; mode
        STW     .D2T2   B13,*++B15(40)   ; *
        STW     .D2T2   B12,*++B15(44)   ; *

* Begin Benchmark Timing

        ADDAW   .D1     A4,A6,A3         ; f xx2 = x + n*4
||      MV      .L2     B4,B12           ; f wsave = w
||      SHRU    .S2X    A6,1,B4          ; f nsave = n>>1
||      MV      .L1X    B4,A5            ; f w1 = w
||      MVK     .S1     1,A14            ; f onea = 1

        MV      .S2X    A3,B8            ; f xx1 = xx2
||      MV      .L2     B4,B0            ; f i = nsave
||      SHL     .S1     A6,2,A10         ; f k1 = n<<2

        LDDW    .D2     *B8++,B7:B6      ; p @ t2_1:t2_0 = *xx1++
||      LDDW    .D1     *A5++,A7:A6      ; p @ s:c = *w1
||      MPY     .M2     B0,1,B13         ; f ireset = i*1
||      ADD     .L2X    A6,1,B9          ; f bk = n+1
||      MV      .S2     B8,B5            ; f xx3 = xx1

        MV      .L1     A4,A11           ; f xx4 = x
||      SHL     .S2X    A6,2,B1          ; f k = n * 4

        MV      .L1X    B9,A0            ; f bk1 = bk
||[B0]  SUB     .L2     B0,1,B0          ; p if (i) i = i-1
||      MV      .S1     A4,A3            ; f xx2 = x
||      MVK     .S2     1,B14            ; f oneb = 1

  [!B0] ADD     .L2     B8,B1,B8         ; p if (!i) xx1 = xx1 + k
||      MV      .S2     B13,B2           ; f t3_ctr = ireset
||      MV      .L1X    B13,A1           ; f st_ctr = ireset

oloop:

        LDDW    .D2     *B8++,B7:B6      ; p @@ t2_1:t2_0 = *xx1++
||[!B0] LDDW    .D1     *A5++,A7:A6      ; p @@ if (!i) s:c = *w1
||[!B0] MPY     .M2     B14,B13,B0       ; p if (!i) i = ireset*1

        MPYSP   .M1X    A7,B7,A13        ; p rtemp2 = c*t2_0   // switched 6 & 7
||      MPYSP   .M2X    A7,B6,B11        ; p itemp2 = c*t2_1   // for big endian

   [B0] SUB     .S2     B0,1,B0          ; p if (i) i = i-1
||      SUB     .L1X    B4,1,A2          ; f l = nsave - 1

        MPYSP   .M1X    A6,B6,A15        ; p rtemp3 = s*t2_1   // switched for b.e.
||      MPYSP   .M2X    A6,B7,B3         ; p itemp3 = s*t2_0   //
||[!B0] ADD     .L2     B8,B1,B8         ; p if (!i) xx1 = xx1 + k

        LDDW    .D2     *B8++,B7:B6      ; p @@@ t2_1:t2_0 = *xx1++
```

```
||[!B0] LDDW    .D1     *A5++,A7:A6     ; p @@@ if (!i) s:c = *w1
||[!B0] MPY     .M2     B14,B13,B0      ; p if (!i) i = ireset*1

        MPYSP   .M1X    A7,B7,A13       ; p rtemp2 = c*t2_0   // switched for b.e.
||      MPYSP   .M2X    A7,B6,B11       ; p itemp2 = c*t2_1   //

   [B0] SUB     .S2     B0,1,B0         ; p if (i) i = i-1

        LDDW    .D1     *A3++,A9:A8     ; p @ t3_1:t3_0 = *xx2++
||      MPYSP   .M1X    A6,B6,A15       ; p rtemp3 = s*t2_1   // switched for b.e.
||      MPYSP   .M2X    A6,B7,B3        ; p itemp3 = s*t2_0   //
||[!B0] ADD     .D2     B8,B1,B8        ; p if (!i) xx1 = xx1 + k
||      ADDSP   .L1     A13,A15,A12     ; p rtemp1 = rtemp2 + rtemp3
||      SUBSP   .L2     B11,B3,B10      ; p itemp1 = itemp2 - itemp3

        LDDW    .D2     *B8++,B7:B6     ; p @@@@ t2_1:t2_0 = *xx1++
||[!B0] LDDW    .D1     *A5++,A7:A6     ; p @@@@ if (!i) s:c = *w1
||[!B0] MPY     .M2     B14,B13,B0      ; p if (!i) i = ireset*1
||[B2]  SUB     .S2     B2,1,B2         ; p if (t3_ctr) t3_ctr -= 1

        MPYSP   .M1X    A7,B7,A13       ; p rtemp2 = c*t2_0   // switched for b.e.
||      MPYSP   .M2X    A7,B6,B11       ; p itemp2 = c*t2_1   //
||[!B2] ADD     .S1     A3,A10,A3       ; p if (!t3_ctr) xx2 = xx2 + k1

   [B0] SUB     .S2     B0,1,B0         ; p if (i) i = i-1
||[!B2] MPY     .M2     B14,B13,B2      ; p if (!t3_ctr) t3_ctr = ireset*1

        LDDW    .D1     *A3++,A9:A8     ; p @@ t3_1:t3_0 = *xx2++
||      MPYSP   .M1X    A6,B6,A15       ; p rtemp3 = s*t2_1   // switched for b.e.
||      MPYSP   .M2X    A6,B7,B3        ; p itemp3 = s*t2_0   //
||[!B0] ADD     .D2     B8,B1,B8        ; p if (!i) xx1 = xx1 + k
||      ADDSP   .L1     A13,A15,A12     ; p rtemp1 = rtemp2 + rtemp3
||      SUBSP   .L2     B11,B3,B10      ; p itemp1 = itemp2 - itemp3

        LDDW    .D2     *B8++,B7:B6     ; p @@@@@ t2_1:t2_0 = *xx1++
||[!B0] LDDW    .D1     *A5++,A7:A6     ; p @@@@@ if (!i) s:c = *w1
||[!B0] MPY     .M2     B14,B13,B0      ; p if (!i) i = ireset*1
||[B2]  SUB     .S2     B2,1,B2         ; p if (t3_ctr) t3_ctr -= 1
||      SUBSP   .L1     A9,A12,A15      ; p rtemp3 = t3_0 - rtemp1 // switched 8 & 9
||      SUBSP   .L2X    A8,B10,B3       ; p itemp3 = t3_1 - itemp1 // for big endian

        MPYSP   .M1X    A7,B7,A13       ; p rtemp2 = c*t2_0  // switched for b.e.
||      MPYSP   .M2X    A7,B6,B11       ; p itemp2 = c*t2_1  //
||[!B2] ADD     .S1     A3,A10,A3       ; p if (!t3_ctr) xx2 = xx2 + k1
||      B       .S2     iloop

   [B0] SUB     .S2     B0,1,B0         ; p if (i) i = i-1
||[!B2] MPY     .M2     B14,B13,B2      ; p if (!t3_ctr) t3_ctr =ireset*1
||      ADDSP   .L1     A9,A12,A15      ; p rtemp3 = t3_0 + rtemp1 //switched for b.e
    .
||      ADDSP   .L2X    A8,B10,B3       ; p itemp3 = t3_1 + itemp1 //

; Kernel Loop Begins

iloop:

        LDDW    .D1     *A3++,A9:A8     ; @@@ t3_1:t3_0 = *xx2++
||      MPYSP   .M1X    A6,B6,A15       ; rtemp3 = s*t2_1  // switched for b.e.
||      MPYSP   .M2X    A6,B7,B3        ; itemp3 = s*t2_0  //
||[!B0] ADD     .D2     B8,B1,B8        ; if (!i) xx1 = xx1 + k
||      ADDSP   .L1     A13,A15,A12     ; rtemp1 = rtemp2 + rtemp3
||      SUBSP   .L2     B11,B3,B10      ; itemp1 = itemp2 - itemp3
||[!A1] ADD     .S2     B5,B1,B5        ; if (!st_ctr) xx3 = xx3 + k
||[!A1] ADD     .S1     A11,A10,A11     ; if (!st_ctr) xx4 = xx4 + k1

        LDDW    .D2     *B8++,B7:B6     ; @@@@@@ t2_1:t2_0 = *xx1++
||[!B0] LDDW    .D1     *A5++,A7:A6     ; @@@@@@ if (!i) s:c = *w1
||[!B0] MPY     .M2     B14,B13,B0      ; if (!i) i = ireset*1
||[B2]  SUB     .S2     B2,1,B2         ; if (t3_ctr) t3_ctr -= 1
||      SUBSP   .L1     A9,A12,A15      ; rtemp3 = t3_0 - rtemp1  // switched for b.e
    .
||      SUBSP   .L2X    A8,B10,B3       ; itemp3 = t3_1 - itemp1  //
||[A2]  SUB     .S1     A2,1,A2         ; if (l) l = l-1
||[!A1] MPY     .M1X    A14,B13,A1      ; if (!st_ctr) st_ctr = ireset*1
```

```
        MPYSP   .M1X    A7,B7,A13       ; rtemp2 = c*t2_0  // switched for b.e.
||      MPYSP   .M2X    A7,B6,B11       ; itemp2 = c*t2_1  //
||[!B2] ADD     .S1     A3,A10,A3       ; if (!t3_ctr) xx2 = xx2 + k1
||[A2]  B       .S2     iloop           ; Branch iloop
||      STW     .D2T1   A15,*B5++[2]    ; *xx3++[2] = rtemp3  // switched B3 and A15
||      STW     .D1T2   B3,*+A11[A0]    ; *+xx4[bk1] = itemp3 // for big endian mode

   [B0] SUB     .S2     B0,1,B0         ; if (i) i = i-1
||[!B2] MPY     .M2     B14,B13,B2      ; if (!t3_ctr) t3_ctr = ireset*1
||      ADDSP   .L1     A9,A12,A15      ; rtemp3 = t3_0 + rtemp1  // switched for b.e
    .
||      ADDSP   .L2X    A8,B10,B3       ; itemp3 = t3_1 + itemp1  //
||      STW     .D1     A15,*A11++[2]   ; *xx4++[2] = rtemp3  // switched for b.e.
||      STW     .D2     B3,*-B5[B9]     ; *-xx3[bk] = itemp3  //
||[A1]  SUB     .S1     A1,1,A1         ; if (st_ctr) st_ctr=st_ctr-1

; Kernel Loop Ends

        MV      .S2     B13,B1          ; o k = ireset
||      MV      .S1X    B13,A2          ; o l = ireset

        SUB     .S1X    B1,1,A2         ; o l = k - 1
||      SHRU    .S2     B13,1,B0        ; o i = ireset>>1
||      ADDAW   .D1     A4,A2,A3        ; o xx2 = x + 4*l

   [A2] B       .S1     oloop
||[!A2] LDW     .D2     *+B15(4),B3     ; o if (!l) pop B3
||      MV      .S2X    A3,B8           ; o xx1 = xx2

        MV      .L1X    B12,A5          ; o w1 = wsave
||[!A2] LDDW    .D2T1   *+B15(8),A11:A10; o if (!l) pop A11:A10

   [A2] LDDW    .D2     *B8++,B7:B6     ; p @ if (l) t2_1:t2_0 = *xx1++
|| [A2] LDDW    .D1     *A5++,A7:A6     ; p @ if (l) s:c = *w1
||      SHL     .S1X    B1,2,A10        ; f k1 = k<<2
||      MPY     .M2     B0,1,B13        ; f ireset = i*1
||      ADD     .L2     B1,1,B9         ; f bk = k + 1

        MV      .L1     A4,A11          ; f xx4 = x
||      SHL     .S2     B1,2,B1         ; f k = k<<2
||      MV      .L2X    A3,B5           ; f xx3 = xx2
||[!A2] LDDW    .D2T1   *+B15(16),A13:A12; o if (!l) pop A13:A12

        MV      .L1X    B9,A0           ; f bk1 = bk
|| [B0] SUB     .L2     B0,1,B0         ; p if (i) i = i - 1
||      MV      .S1     A4,A3           ; f xx2 = x
||[!A2] LDDW    .D2T1   *+B15(24),A15:A14; o if (!l) pop A15:A14

  [!B0] ADD     .L2     B8,B1,B8        ; p if (!i) xx1 = xx1 + k
||      MV      .S2     B13,B2          ; f t3_ctr = ireset
||      MV      .L1X    B13,A1          ; f st_ctr = ireset
||[!A2] LDDW    .D2T2   *+B15(32),B11:B10; o if (!l) pop B11:B10

;-----------------------------------------------
* End Benchmark Timing

        LDDW    .D2T2   *+B15(40),B13:B12
||      B       .S2     B3

        LDW     .D2T2   *++B15(48),B14
        NOP             4
```

Appendix B

Experimental Platform

B.1 Schematics

This section contains schematics for a few subcomponents of the BYU experimental platform. Figure B.1 is a schematic of the antenna positioning hardware designed by the MIT SRT research initiative and used with the VSA (see Section 2.2) [27]. Figure B.2 is the Pentek schematic of the 4291 Quad TI TMS320C6701 Processor VME Board. Figure B.3 is the Pentek schematic of the 6216 Digital Receiver (see Section 2.4) [30]. These Pentek schematics contain more detail than Figure 2.4, a schematic illustrating our DSP platform architecture.

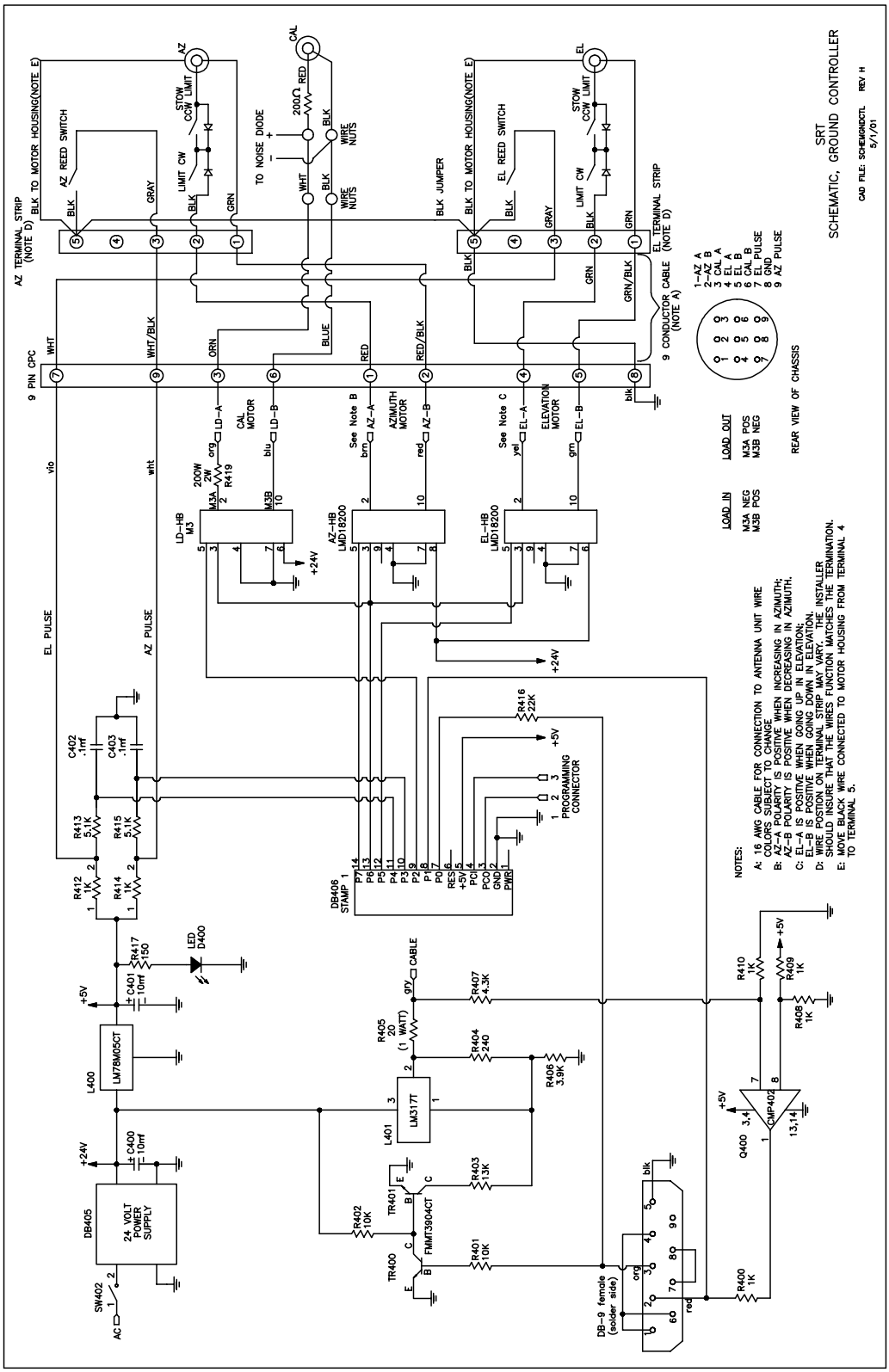


Figure B.1: Positioning hardware schematic designed by the MIT SRT research initiative [27]

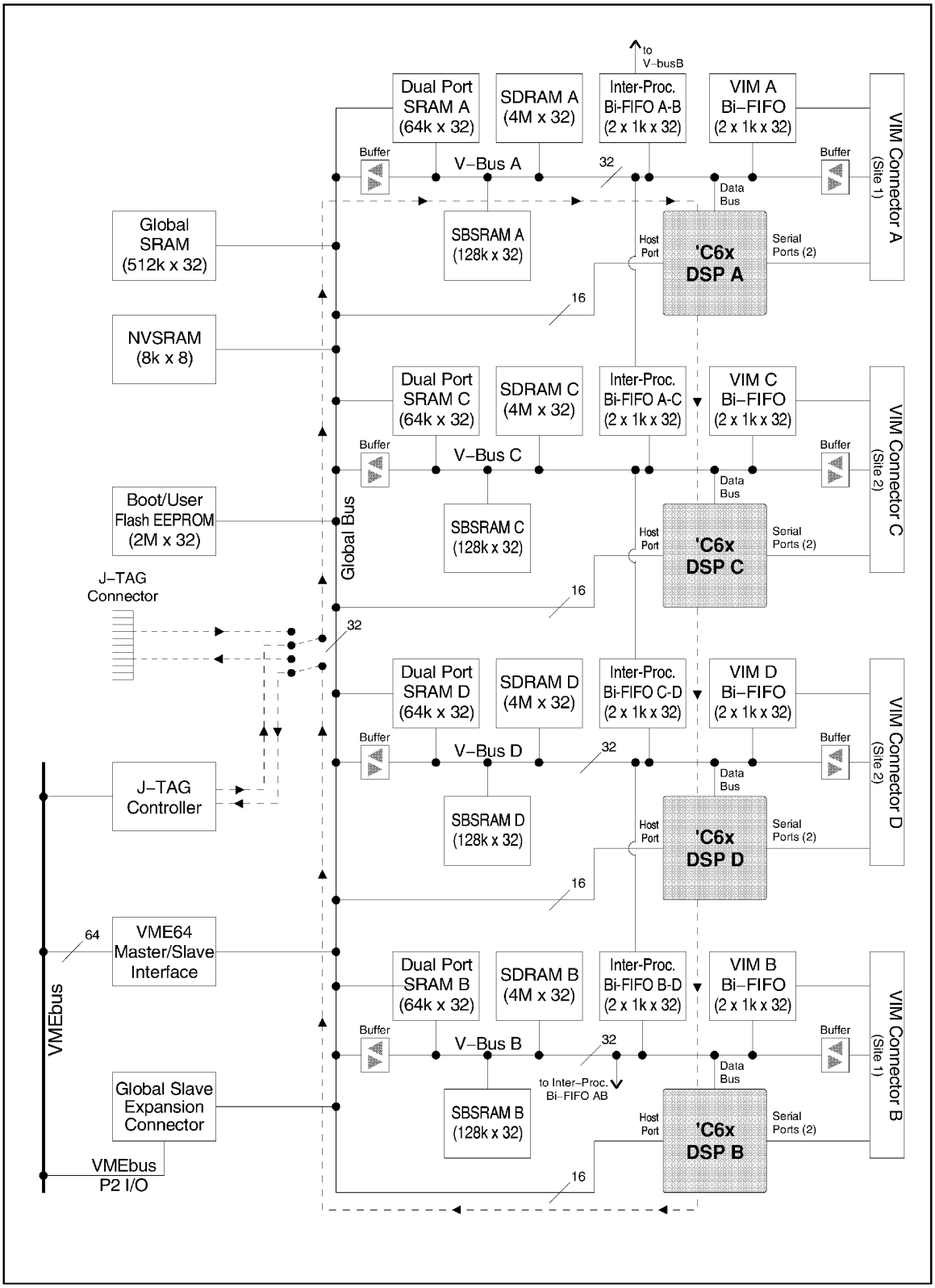


Figure B.2: Pentek schematic of the 4291 Quad TI TMS320C6701 Processor VME Board [30]

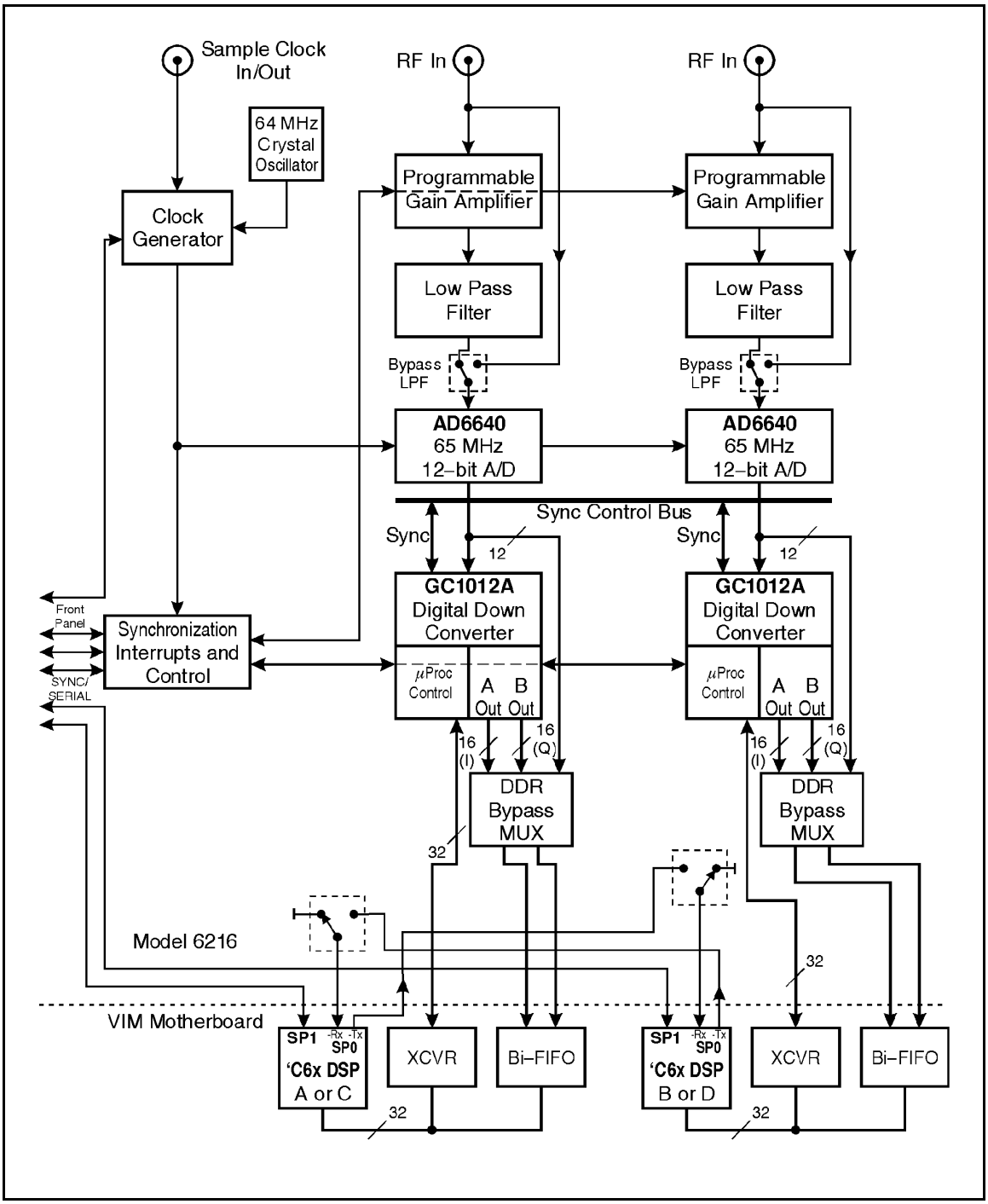


Figure B.3: Pentek schematic of the 6216 Digital Receiver [30]

B.2 Receiver LO Tests

When attempting to make 1612 MHz OH observations, we encountered interference at the output of the RF receiver due to interference both internal and external to the receiver system. With an antenna pointed to zenith, we monitored a 4 MHz bandwidth centered at 1612 MHz with many combinations of LO1, LO2, and the final DSP IF. Each spectrum represents 60 seconds of integration. Much of the interference was buried well below the observational noise floor, but became significant with integration. The results of these tests are found in Figures B.4–B.12. Each figure contains thirteen combinations of LO1 and LO2 for a fixed final DSP IF (8 MHz to 24 MHz in increments of 2 MHz). These results can be used to determine proper LO settings for clean OH maser line observations.

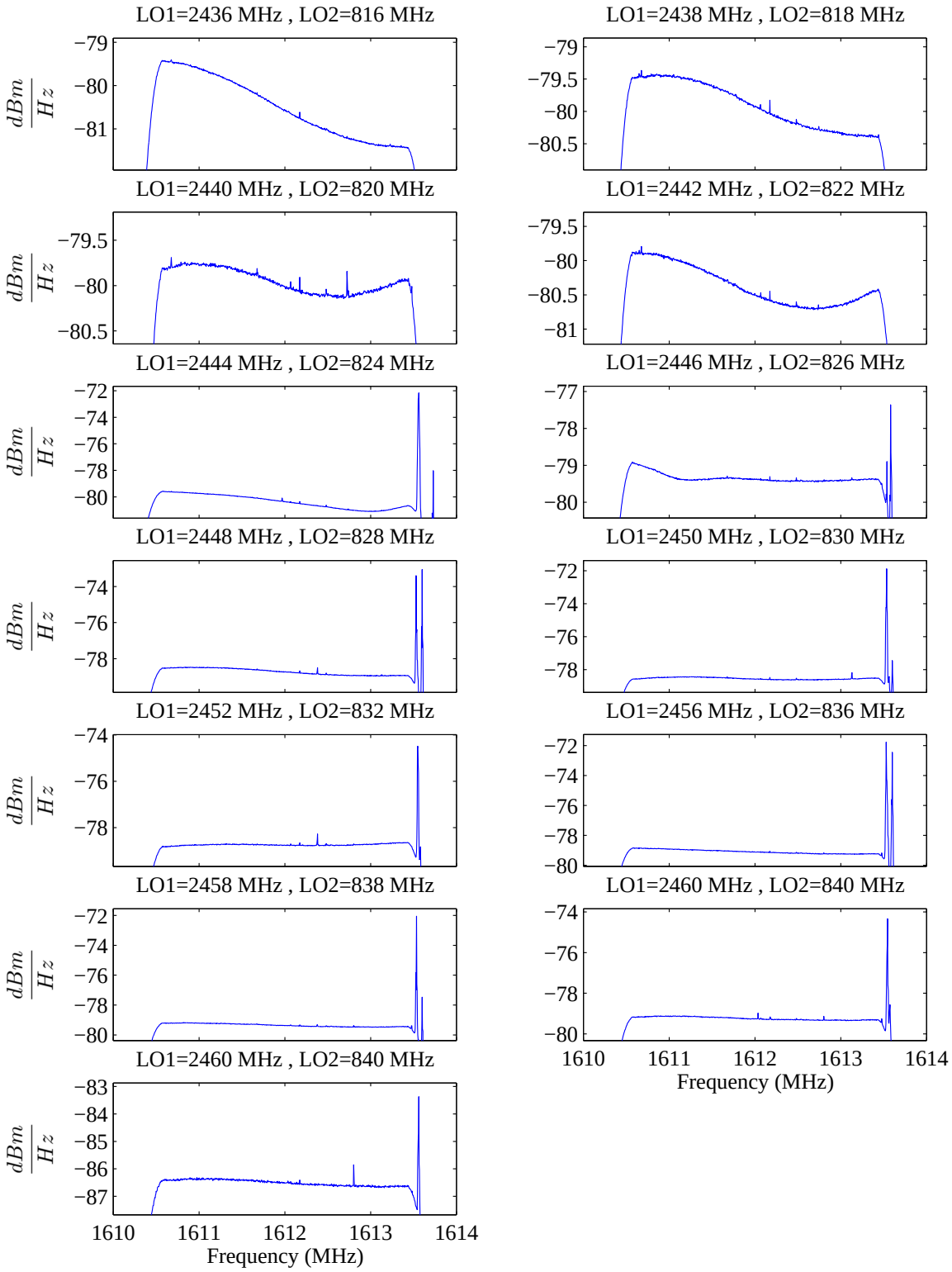


Figure B.4: Receiver LO tests for DSP IF=8 MHz

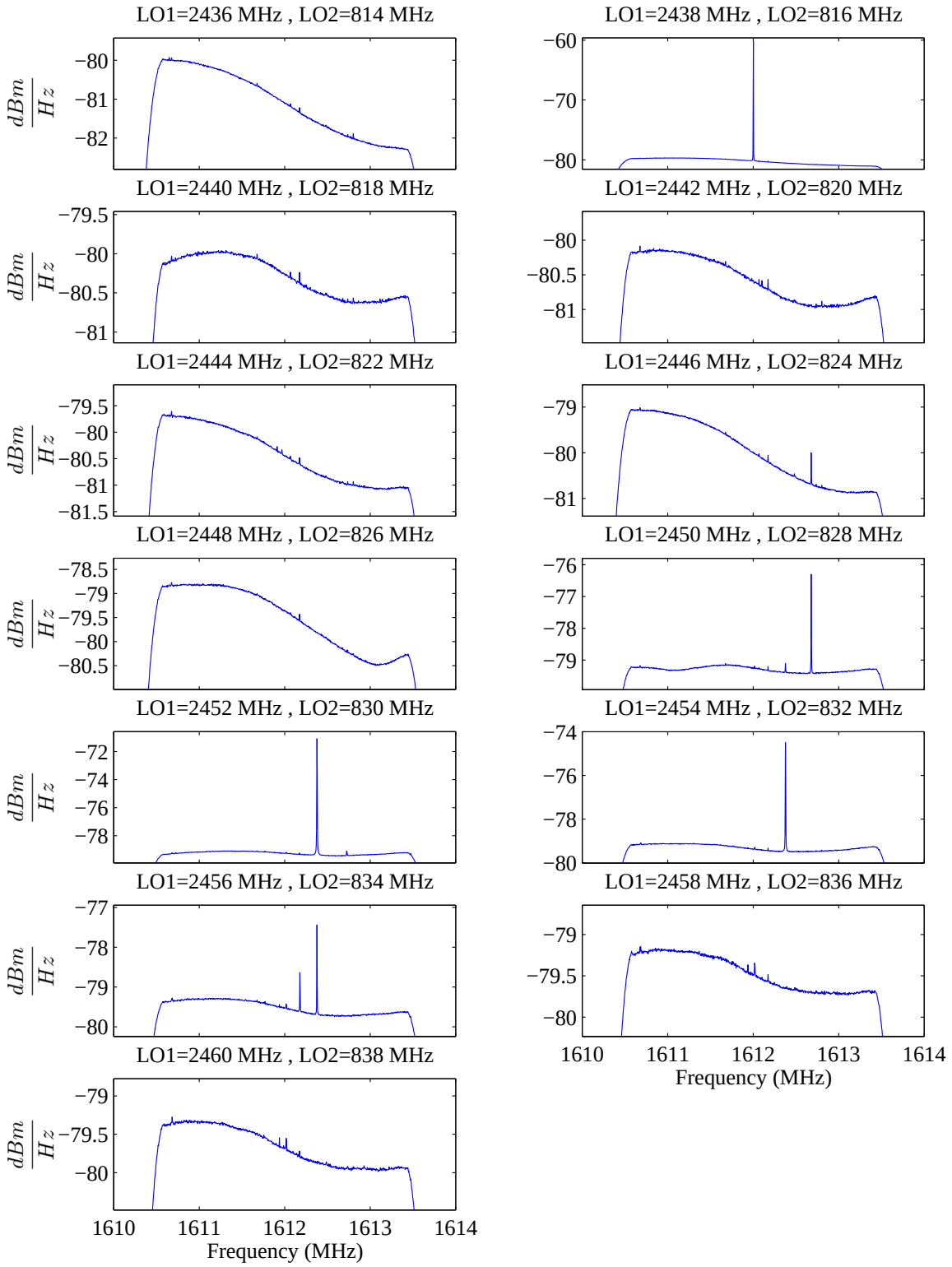


Figure B.5: Receiver LO tests for DSP IF=10 MHz

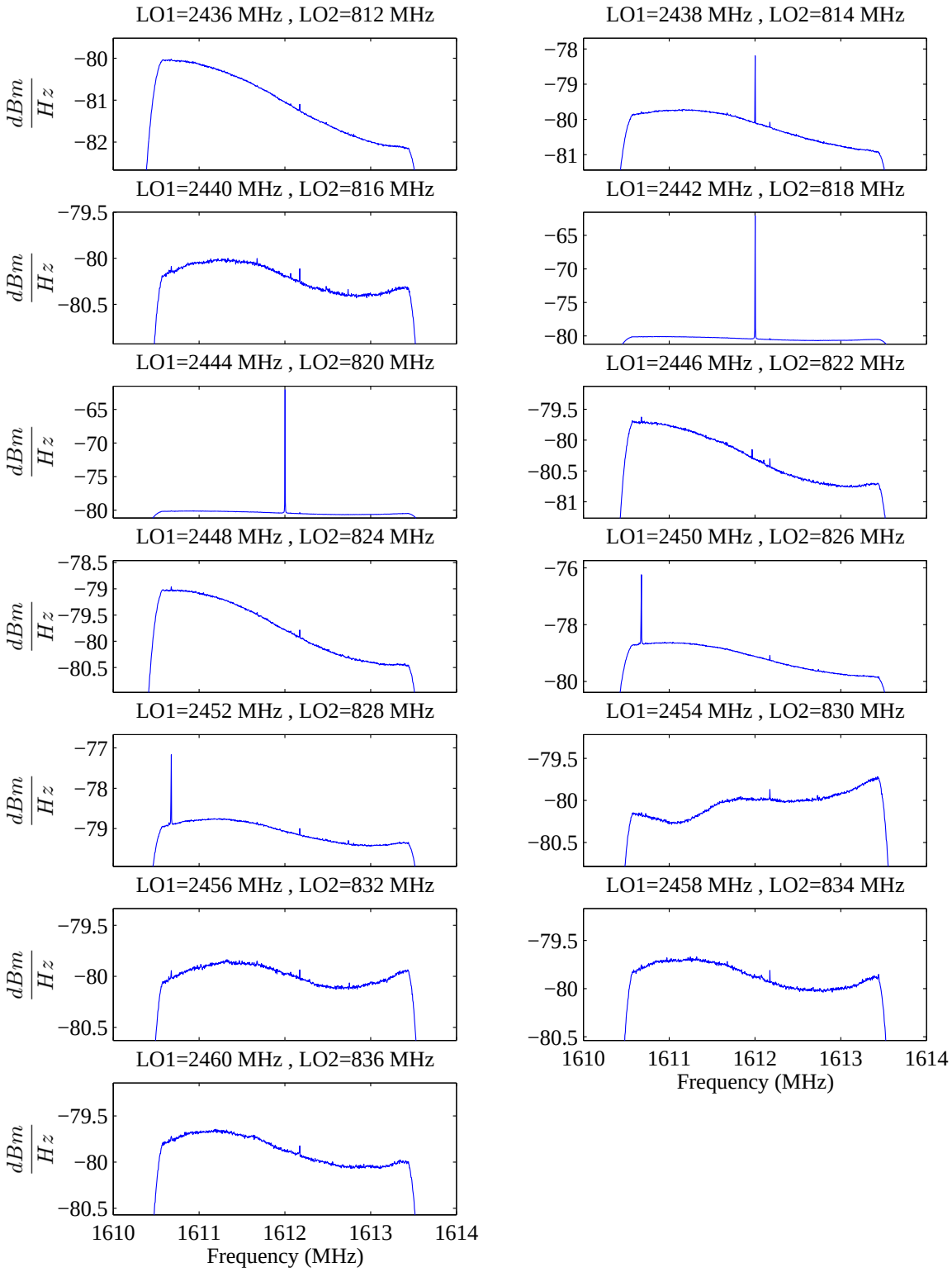


Figure B.6: Receiver LO tests for DSP IF=12 MHz

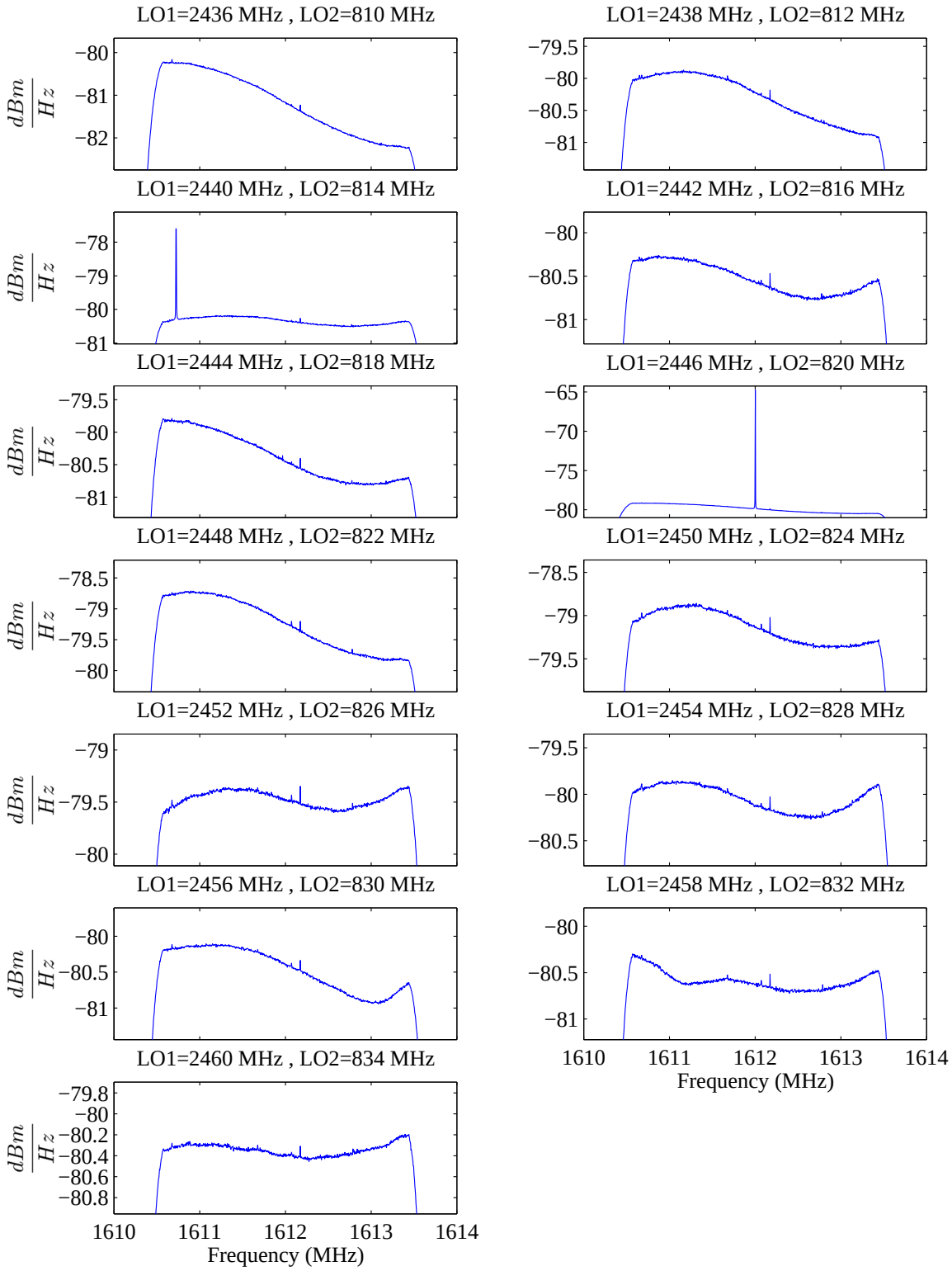


Figure B.7: Receiver LO tests for DSP IF=14 MHz

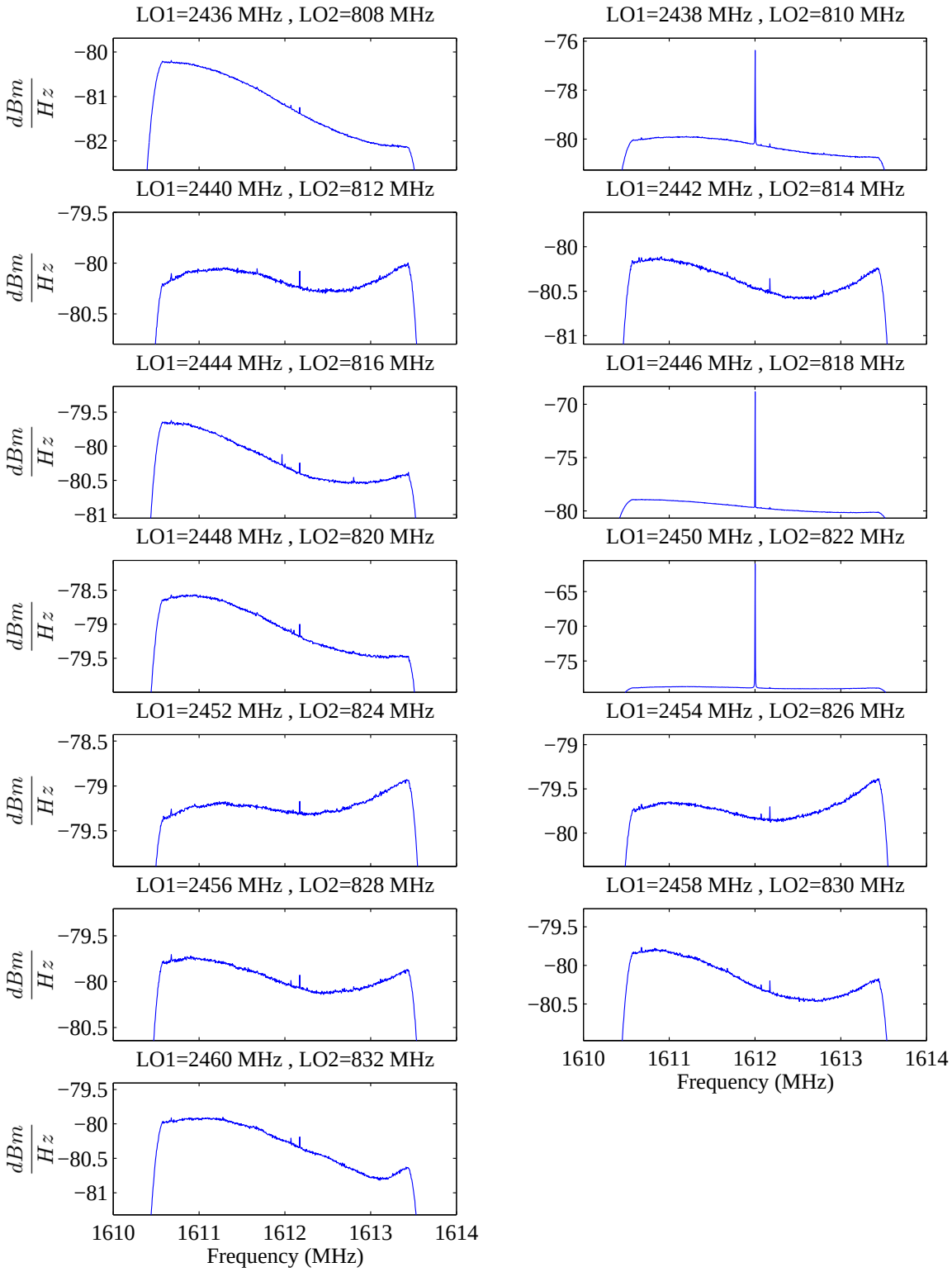


Figure B.8: Receiver LO tests for DSP IF=16 MHz

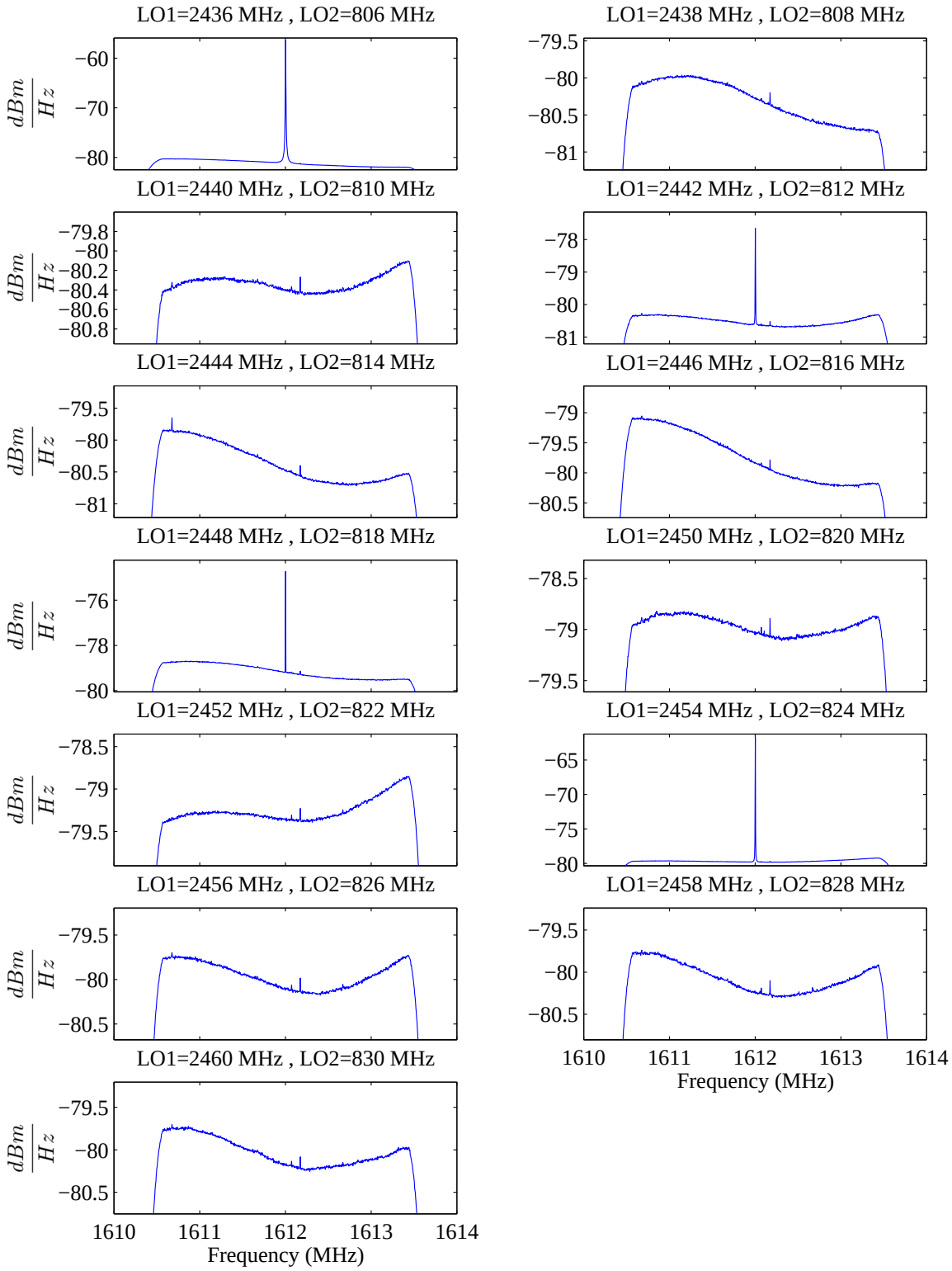


Figure B.9: Receiver LO tests for DSP IF=18 MHz

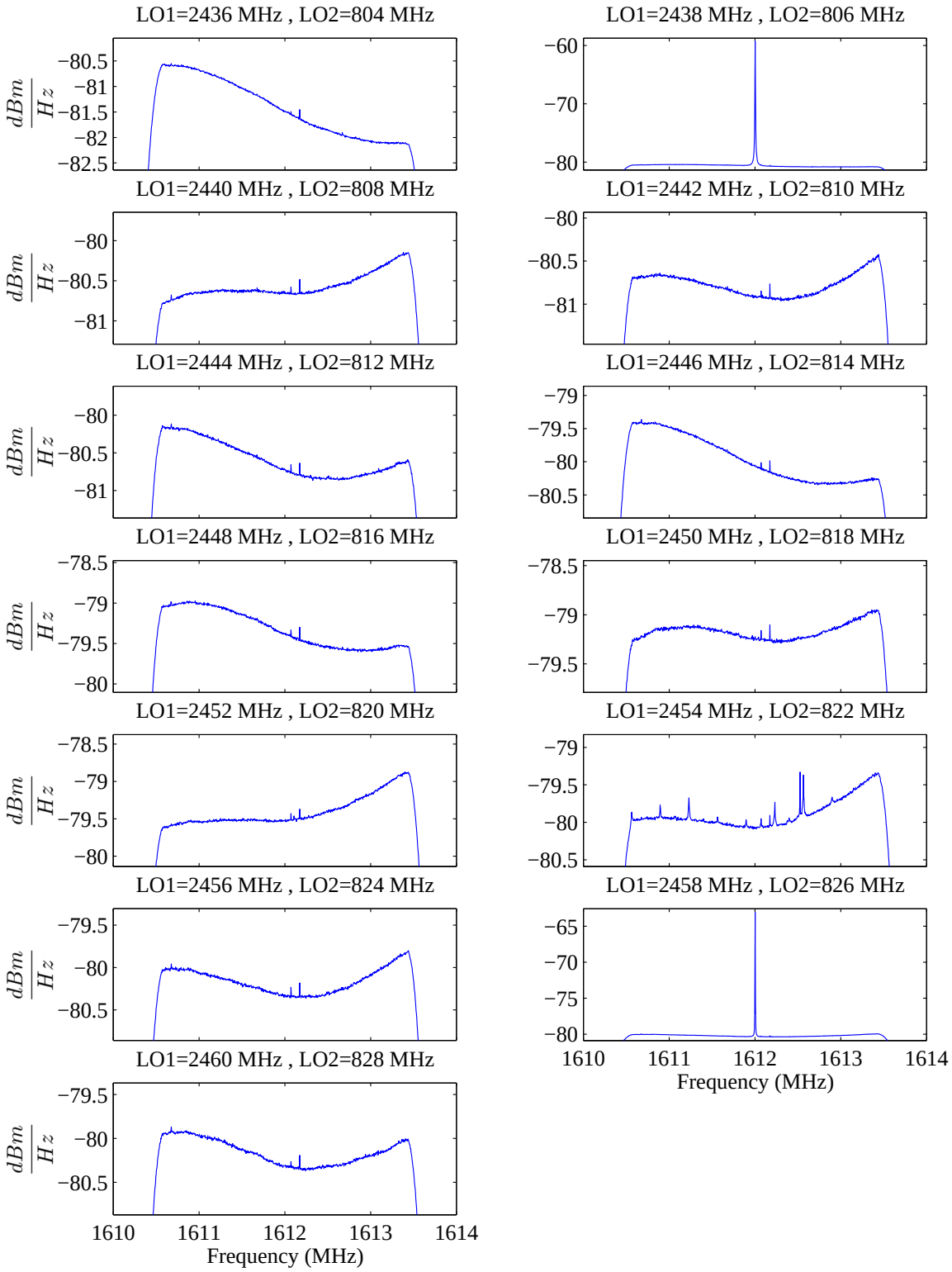


Figure B.10: Receiver LO tests for DSP IF=20 MHz

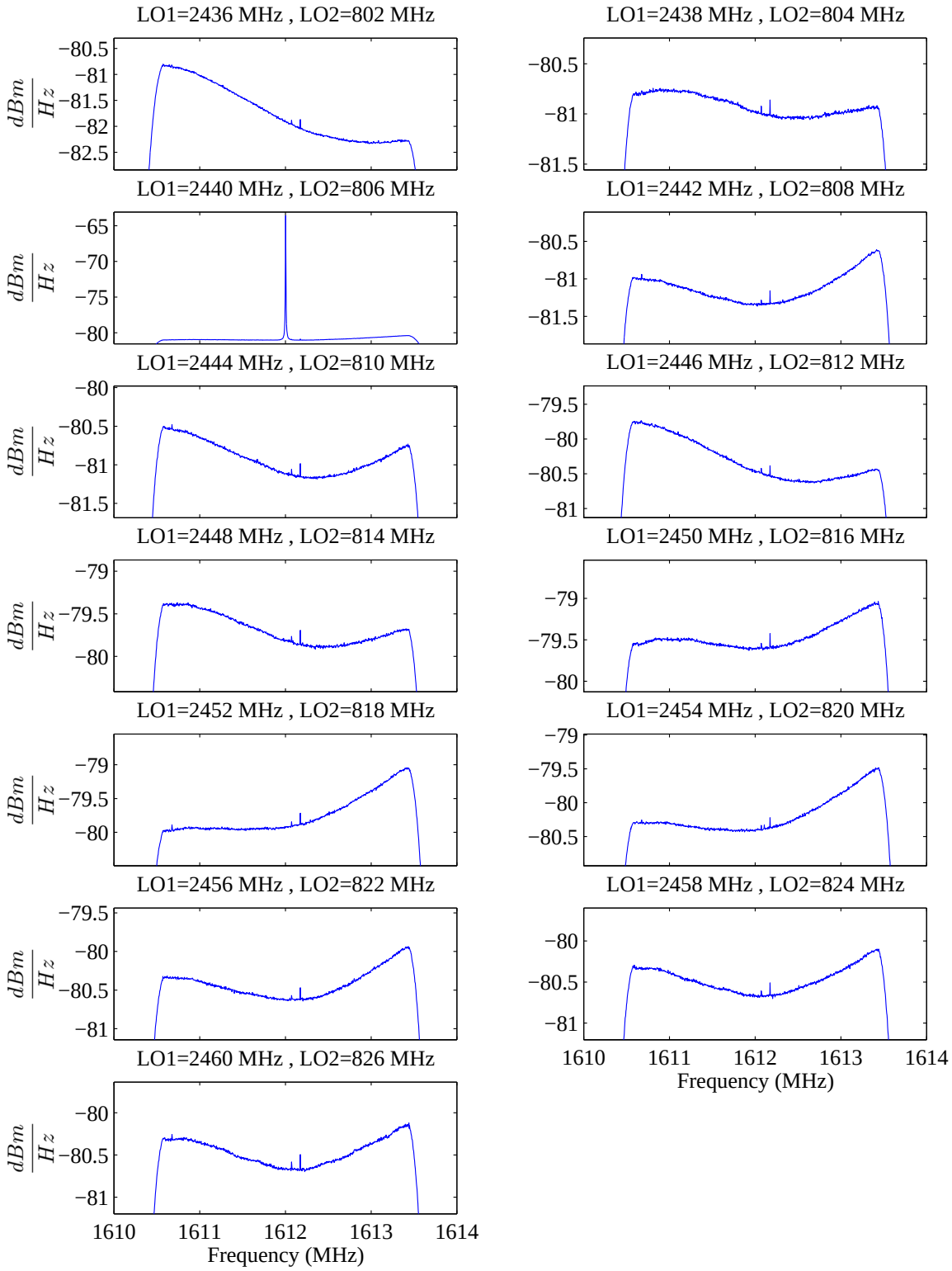


Figure B.11: Receiver LO tests for DSP IF=22 MHz

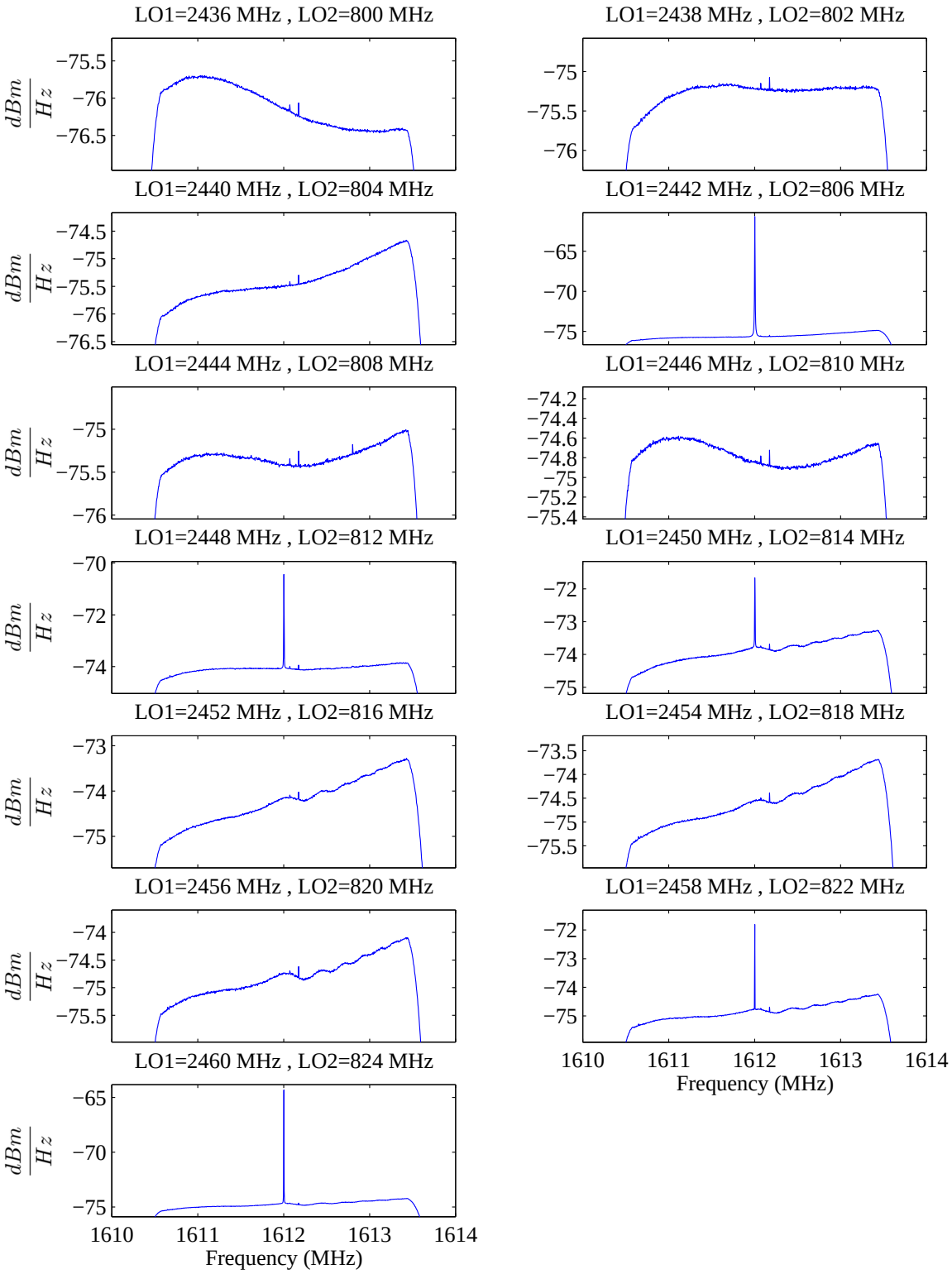


Figure B.12: Receiver LO tests for DSP IF=24 MHz

B.3 6216 Digital Receiver Decimation Anti-aliasing Filters

As a part of the decimation process in the 6216 digital receiver, the data is bandpass filtered to select the desired spectrum and to avoid aliasing. There are six digital filters, one for each of the programmable decimation rates: 2, 4, 8, 16, 32 and 64. Figures B.13–B.18 display each of these filters along with respective frequency responses, both before and after signal decimation. In order to obtain the extremely low -80 dB sidelobes characteristic of these filters with a lower filter order, a ripple of approximately 0.13 dB exists in each passband. This ripple is very evident in DSP PSD estimates at the scale needed to detect weak signals buried in noise. Because these anti-aliasing decimation bandpass filters are implemented in the digital domain, each filter's magnitude response is constant. Therefore, by computing the filter passband characteristics, the 0.13 dB ripple is removed from each DSP PSD estimate (see Section 3.1.3). The filter tap values were obtained by corresponding directly with Graychip, Inc [38].

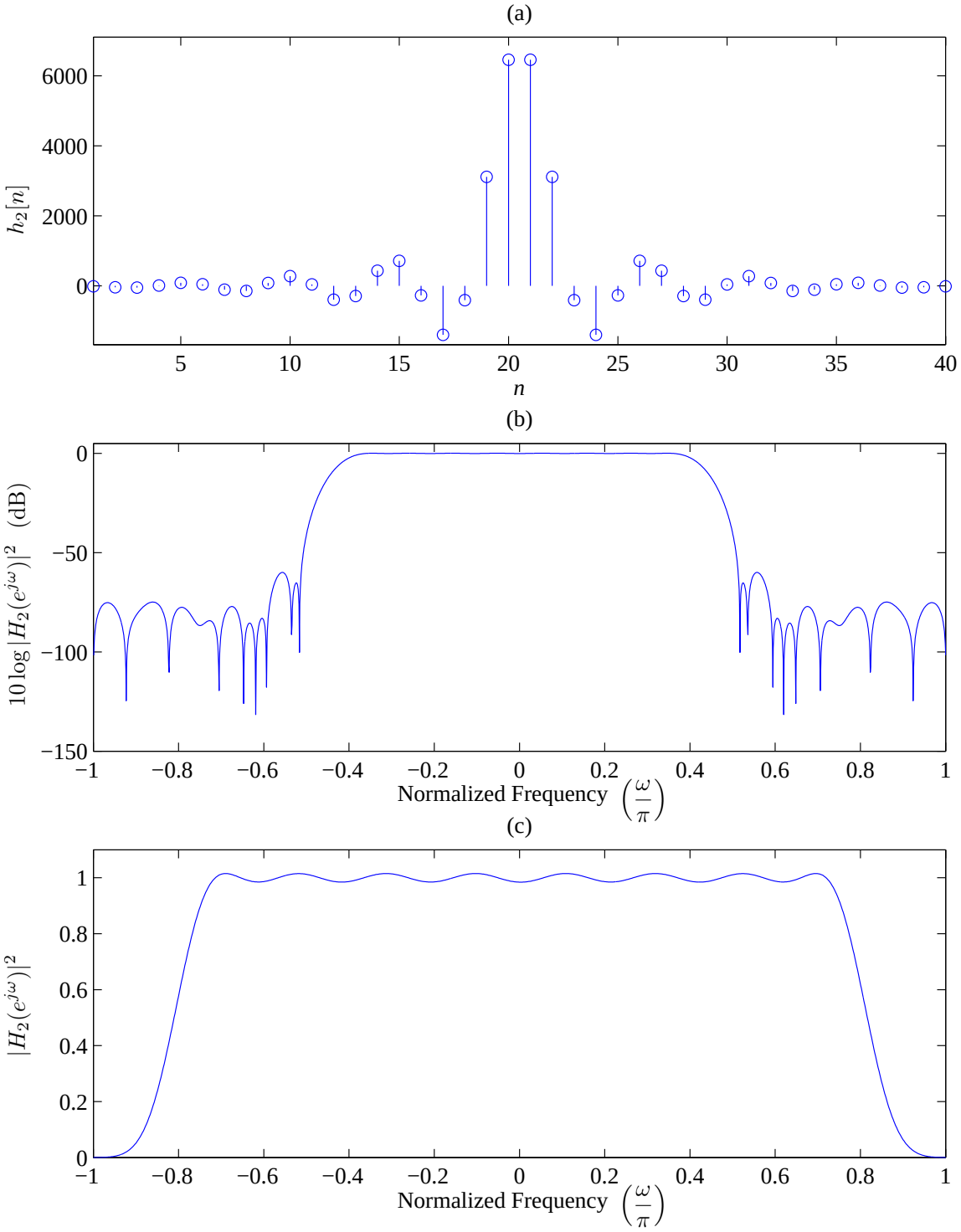


Figure B.13: (a) 40 tap decimate-by-2 anti-aliasing bandpass filter; (b) Normalized frequency response of $h_2[n]$ in dB, $10\log|H_2(e^{j\omega})|^2$; (c) Normalized frequency response of $h_2[n]$ after decimation by 2, $|H_2(e^{j\omega})|^2$. Without normalization, $h_2[n]$ has a passband gain of 84.2888 dB.

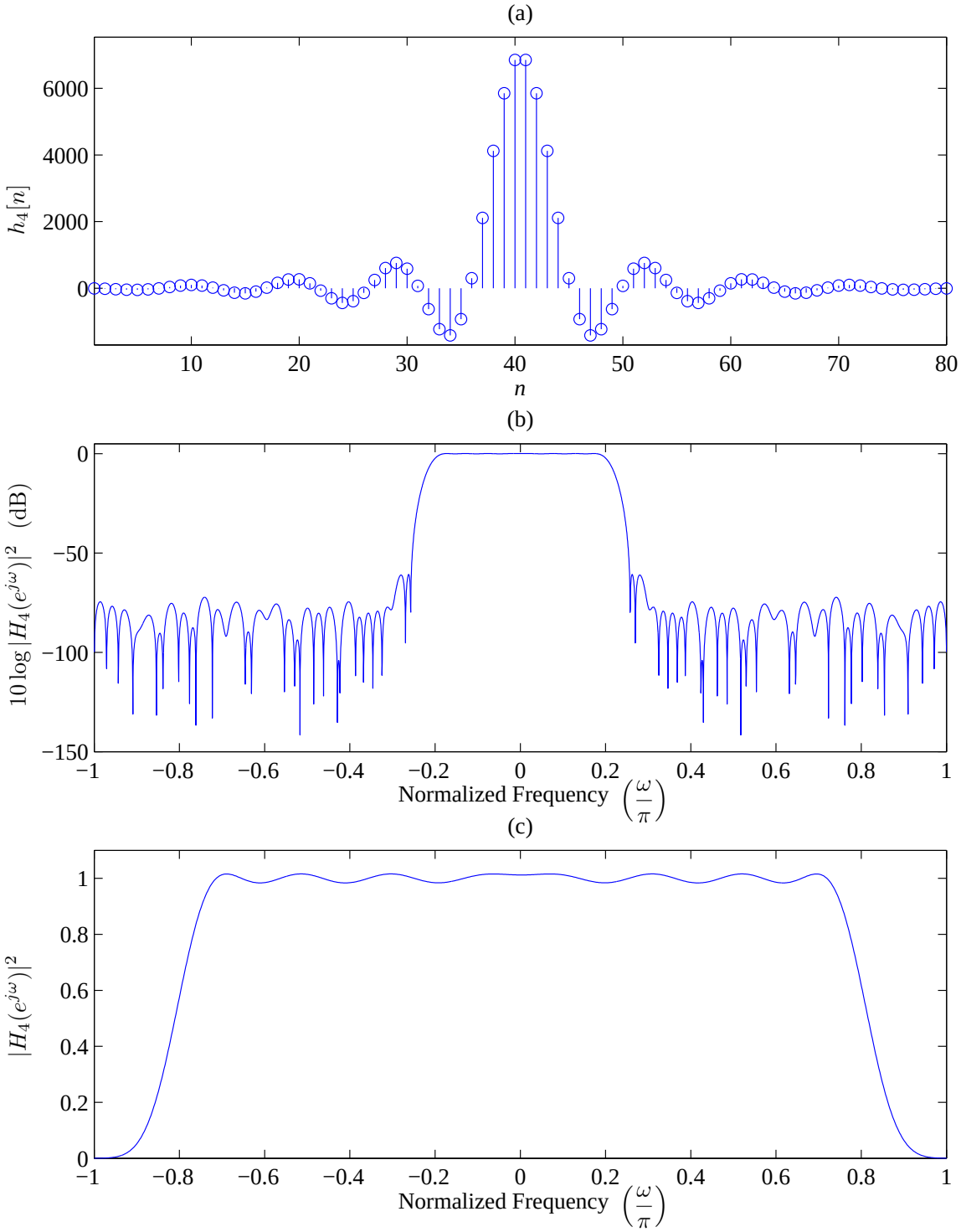


Figure B.14: (a) 80 tap decimate-by-4 anti-aliasing bandpass filter; (b) Normalized frequency response of $h_4[n]$ in dB, $10\log|H_4(e^{j\omega})|^2$; (c) Normalized frequency response of $h_4[n]$ after decimation by 4, $|H_4(e^{j\omega})|^2$. Without normalization, $h_4[n]$ has a passband gain of 90.31 dB.

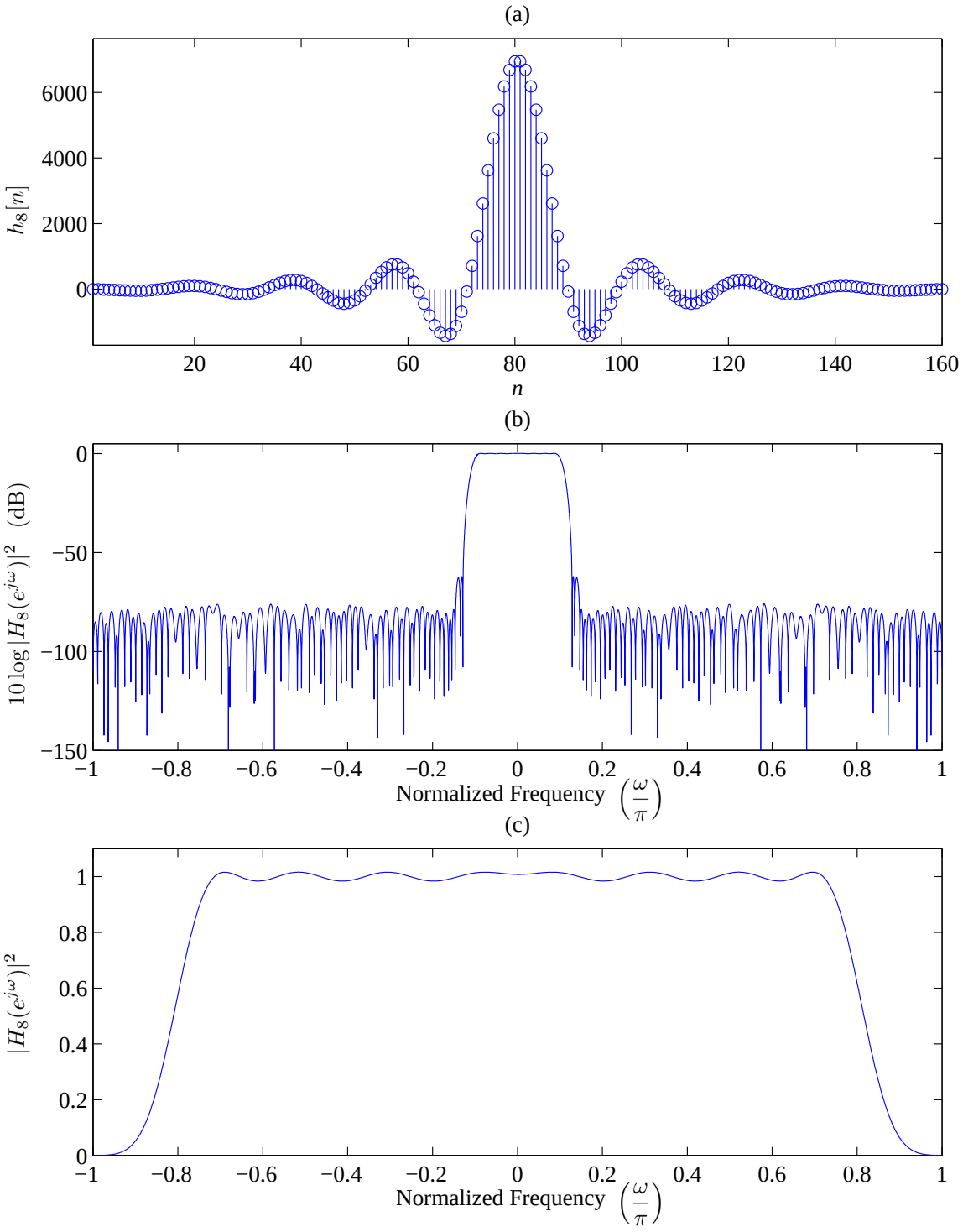


Figure B.15: (a) 160 tap decimate-by-8 anti-aliasing bandpass filter; (b) Normalized frequency response of $h_8[n]$ in dB, $10\log|H_8(e^{j\omega})|^2$; (c) Normalized frequency response of $h_8[n]$ after decimation by 8, $|H_8(e^{j\omega})|^2$. Without normalization, $h_8[n]$ has a passband gain of 96.33 dB.

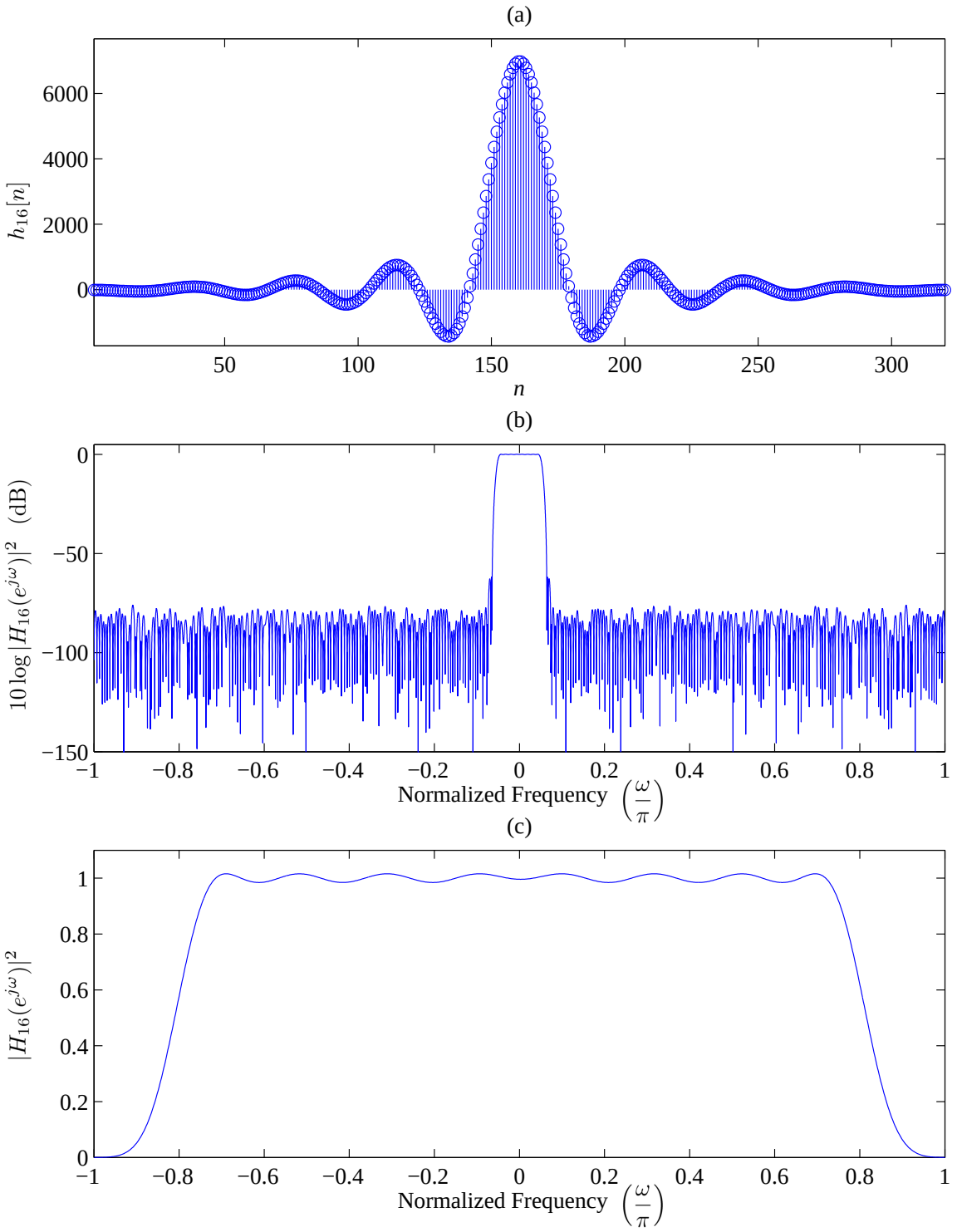


Figure B.16: (a) 320 tap decimate-by-16 anti-aliasing bandpass filter; (b) Normalized frequency response of $h_{16}[n]$ in dB, $10\log|H_{16}(e^{j\omega})|^2$; (c) Normalized frequency response of $h_{16}[n]$ after decimation by 16, $|H_{16}(e^{j\omega})|^2$. Without normalization, $h_{16}[n]$ has a passband gain of 102.35 dB.

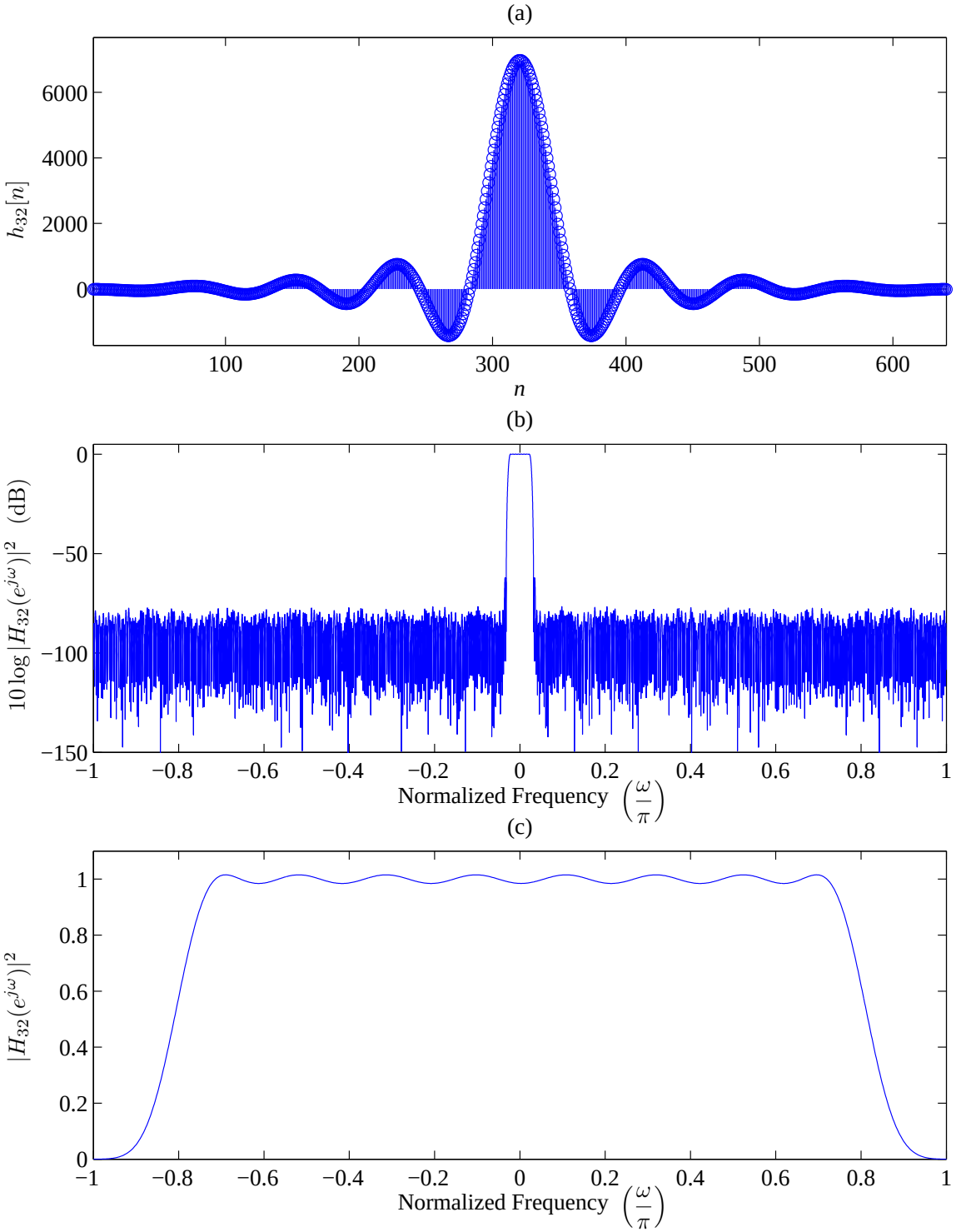


Figure B.17: (a) 640 tap decimate-by-32 anti-aliasing bandpass filter; (b) Normalized frequency response of $h_{32}[n]$ in dB, $10\log|H_{32}(e^{j\omega})|^2$; (c) Normalized frequency response of $h_{32}[n]$ after decimation by 32, $|H_{32}(e^{j\omega})|^2$. Without normalization, $h_{32}[n]$ has a passband gain of 108.3717 dB.

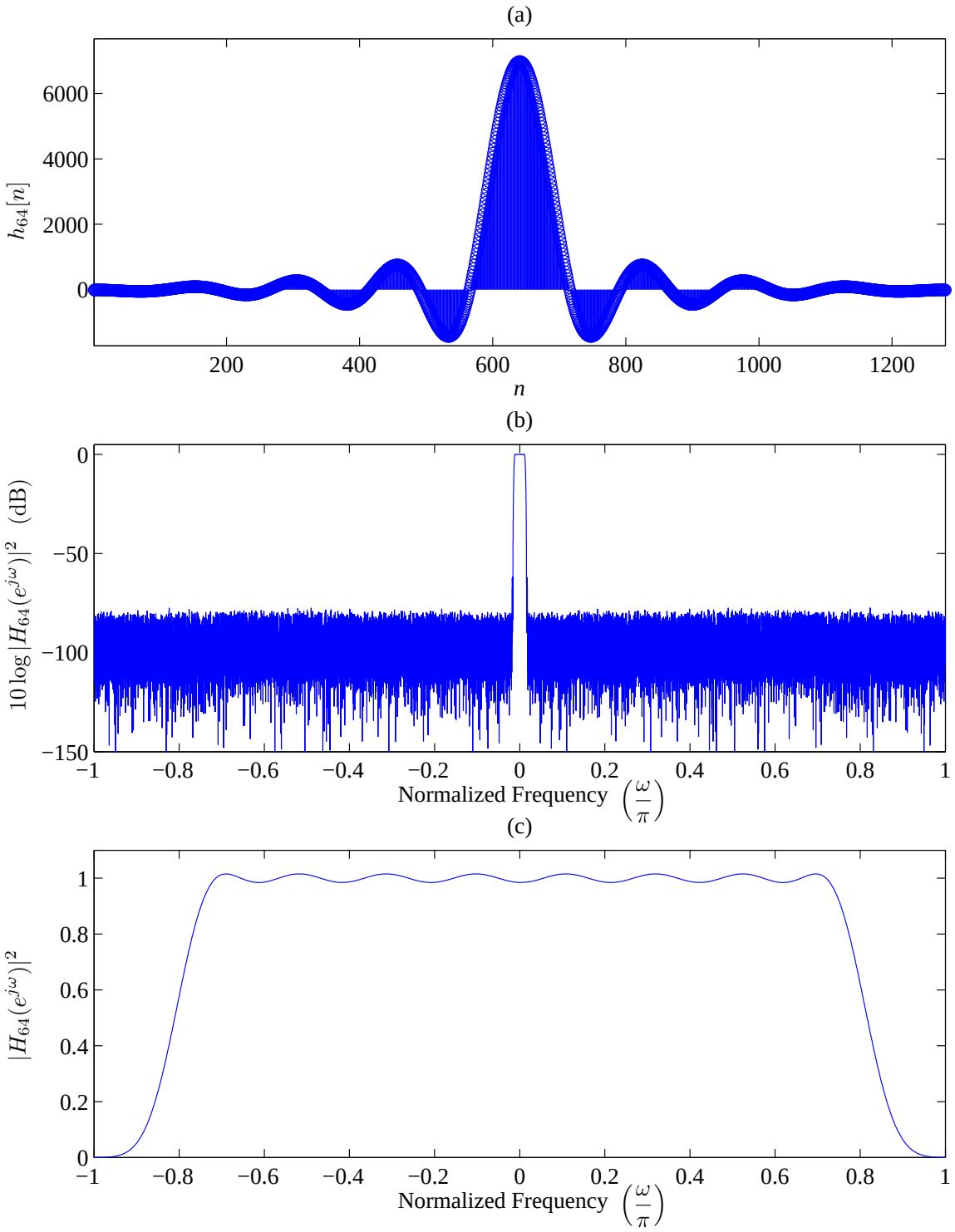


Figure B.18: (a) 1280 tap decimate-by-64 anti-aliasing bandpass filter; (b) Normalized frequency response of $h_{64}[n]$ in dB, $10\log|H_{64}(e^{j\omega})|^2$; (c) Normalized frequency response of $h_{64}[n]$ after decimation by 64, $|H_{64}(e^{j\omega})|^2$. Without normalization, $h_{64}[n]$ has a passband gain of 114.3924 dB.

Appendix C

Green Bank Test Details

C.1 GBT Scheduling

In order to schedule optimal test times with the GBT, I compiled calendars detailing when GLONASS 789 was above the horizon (see Section 4.7.5). Initially I had planned on visiting Green Bank in November or December 2002, but due to scheduling difficulties, the tests were postponed. For this reason, I created calendars from November 2002 to mid-February 2003. These calendars are found on the pages that follow. The reason for including the calendars is to demonstrate the relatively rare instances for which GLONASS 789 was simultaneously above the horizon with highly red-shifted sources. A similar calendar could also be used for scheduling observational times when especially problematic satellite(s) are below the horizon.

Ultimately, I was able to schedule tests for January 30–February 1, 2003. The final GBT schedules detailing my allocated observational times follow the GLONASS calendars.

Table C.1: December 2002 GLONASS 789 calendar

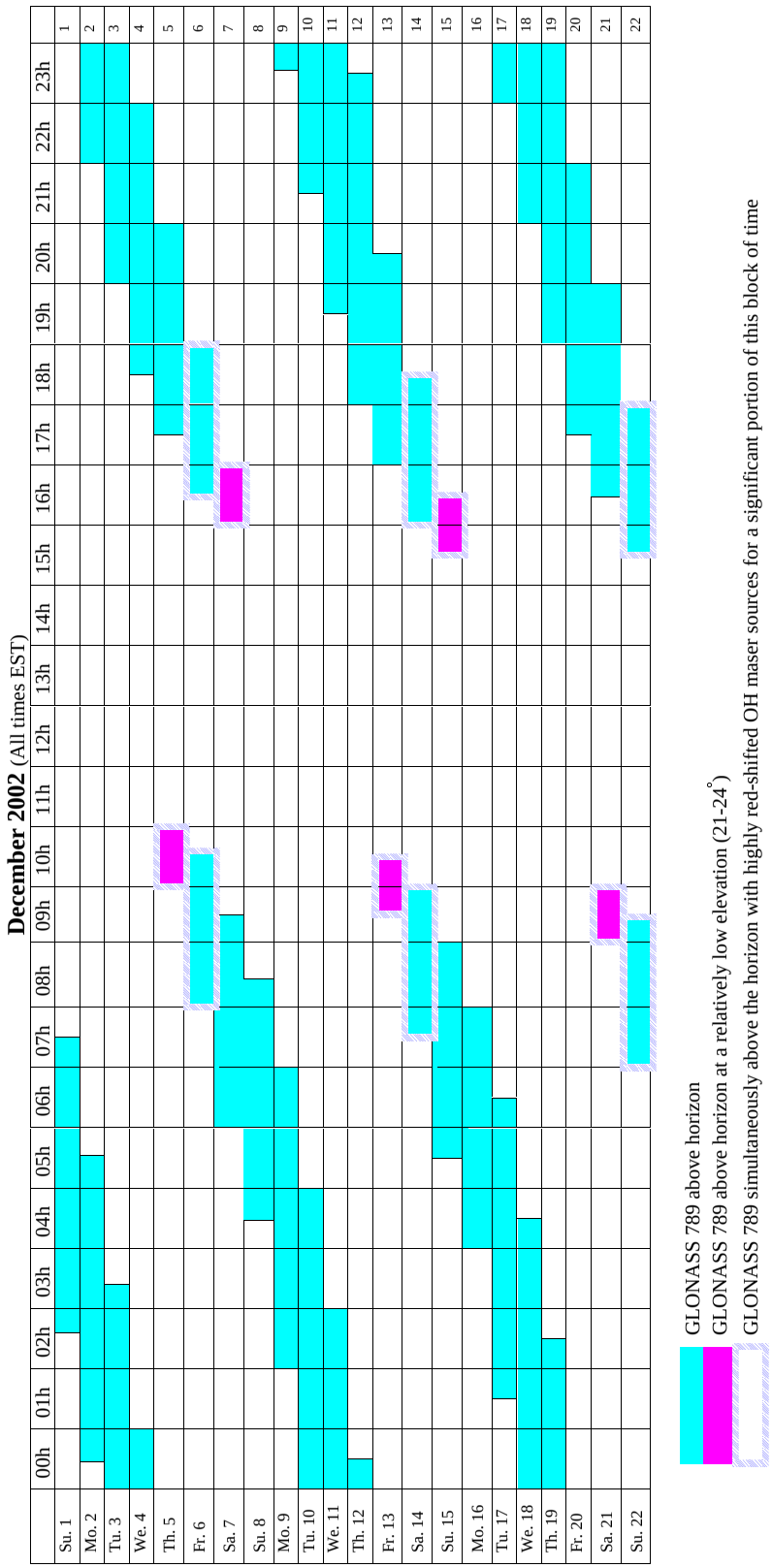


Table C.2: January 2003 GLONASS 789 calendar

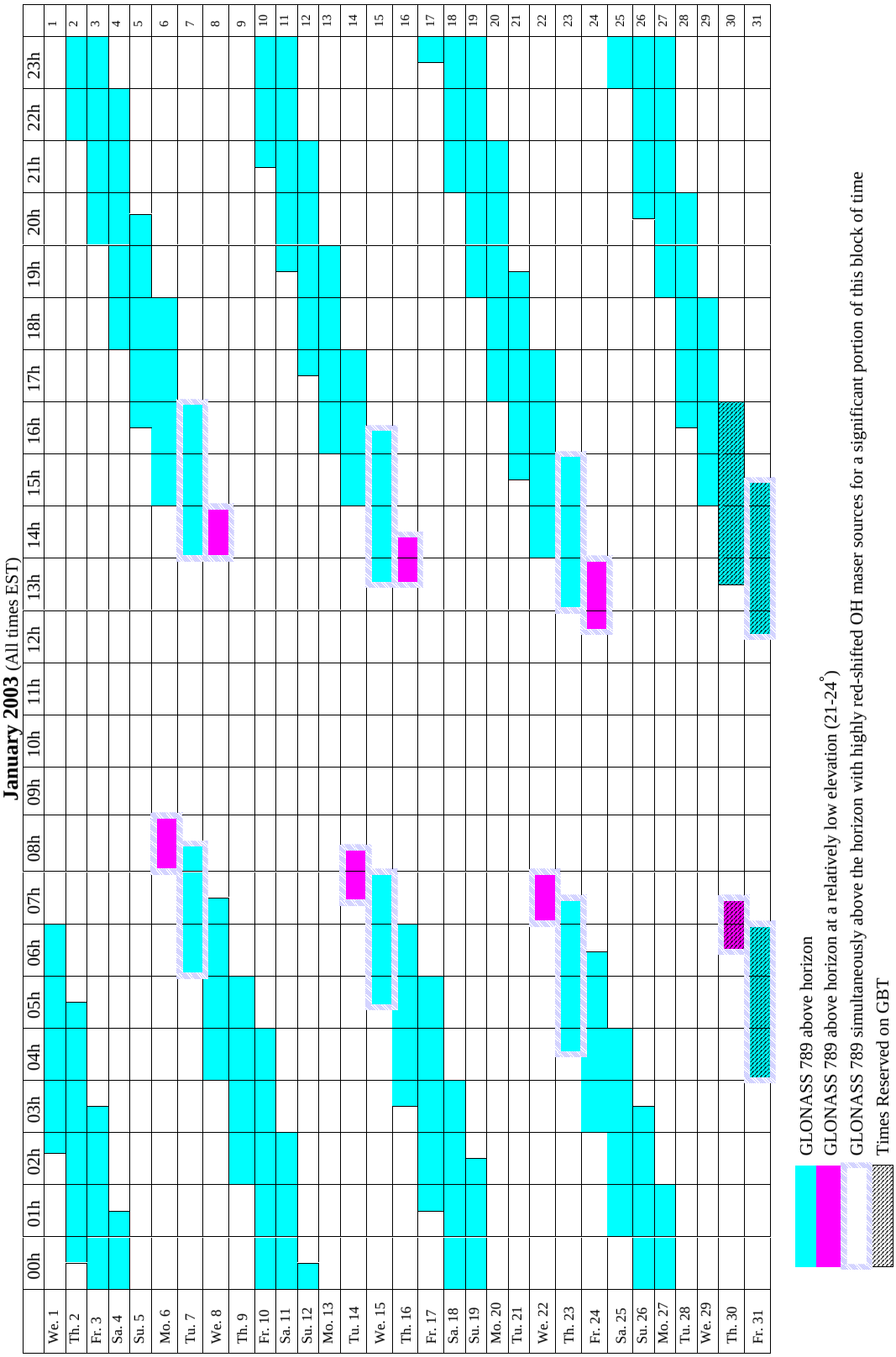
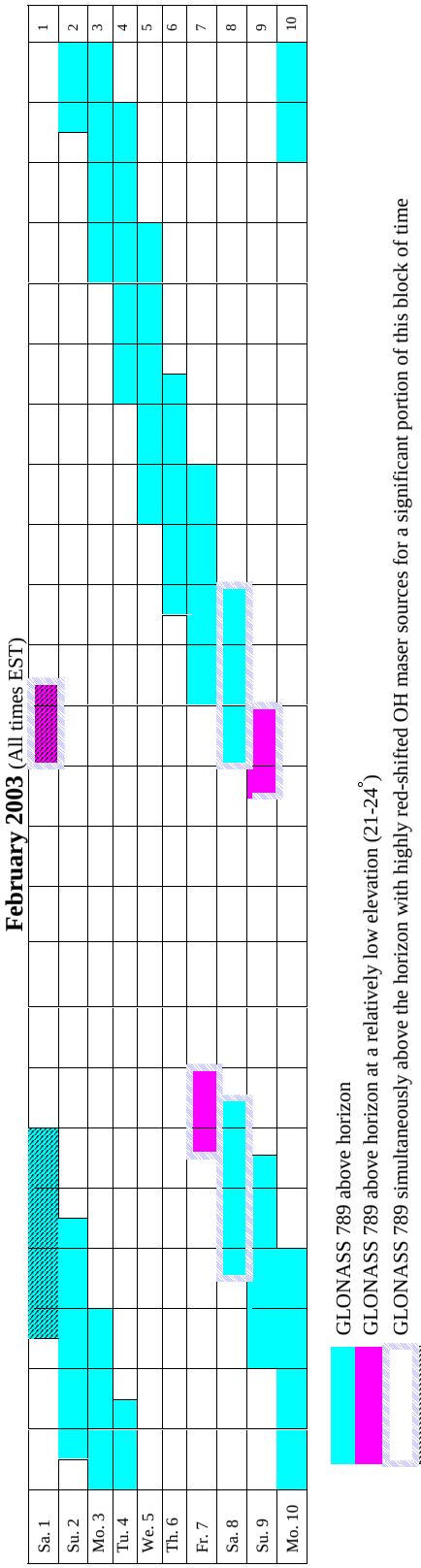


Table C.3: February 2003 GLONASS 789 calendar



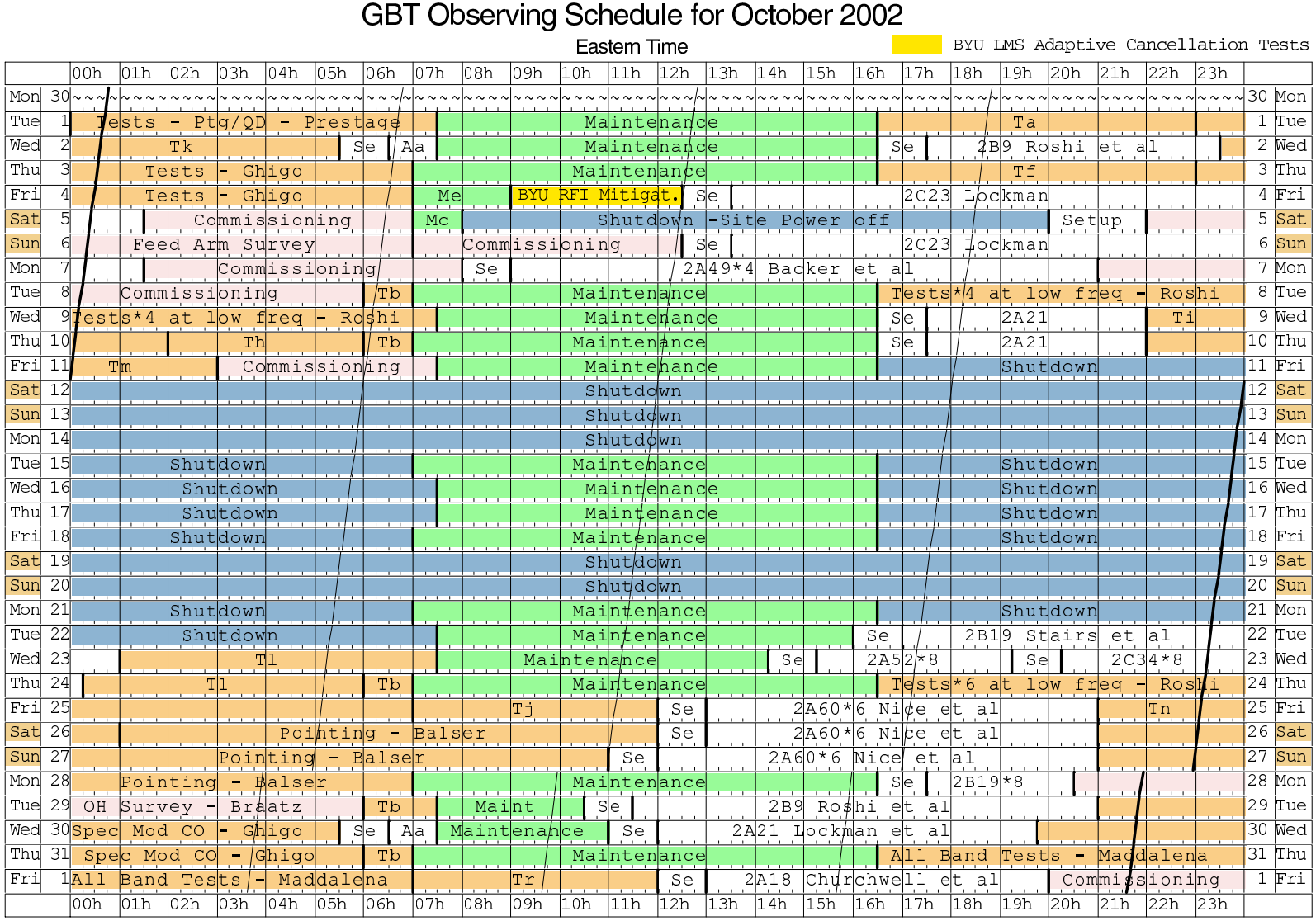
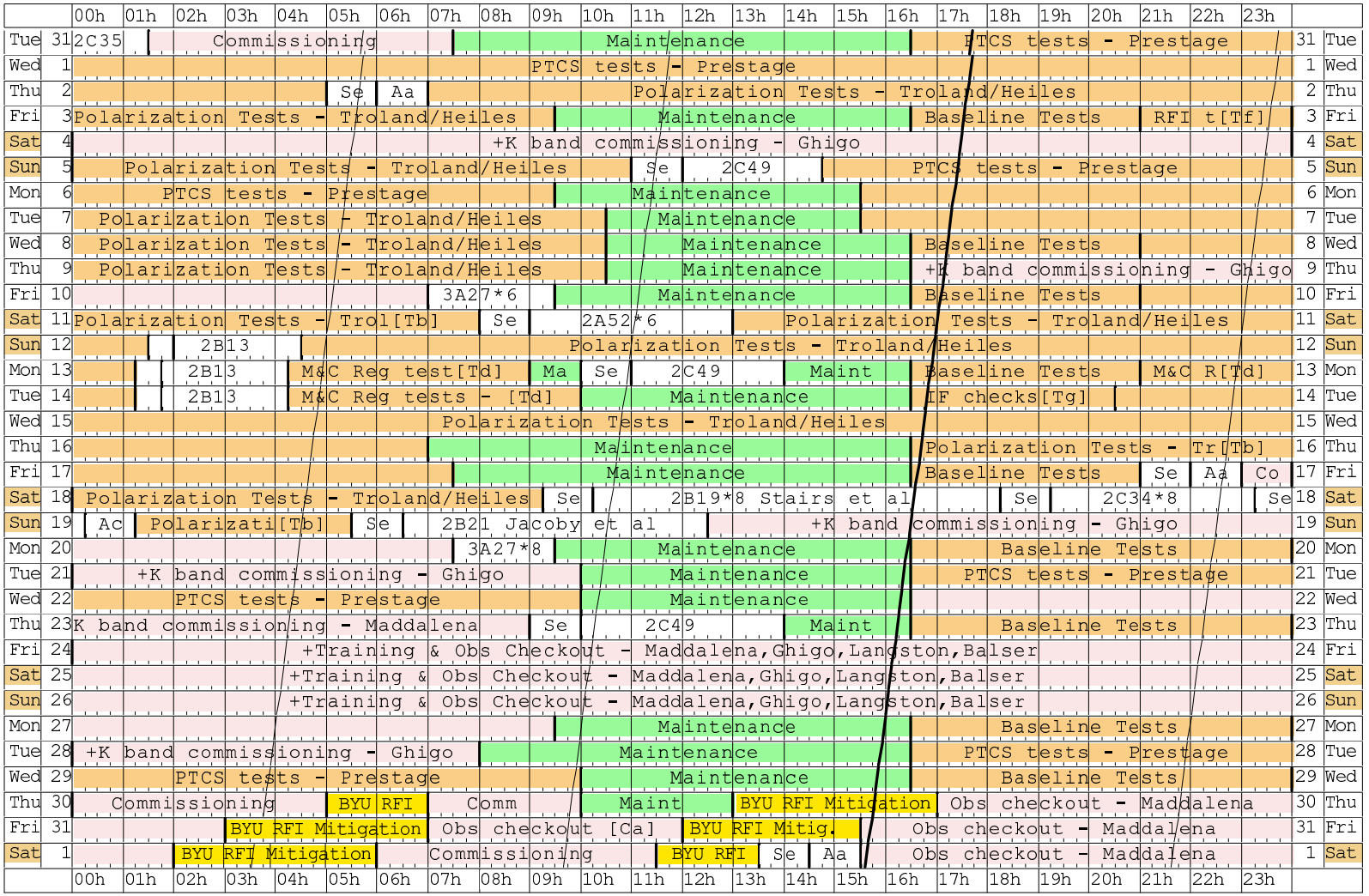


Table C.4: October 2002 GBT schedule

GBT Observing Schedule for January 2003

(before test cancellation due to track problems)

Eastern Time — BYU LMS Adaptive Cancellation Tests



Version 1.3 01/15/2003

Table C.5: January 2003 GBT schedule (before test cancellation)

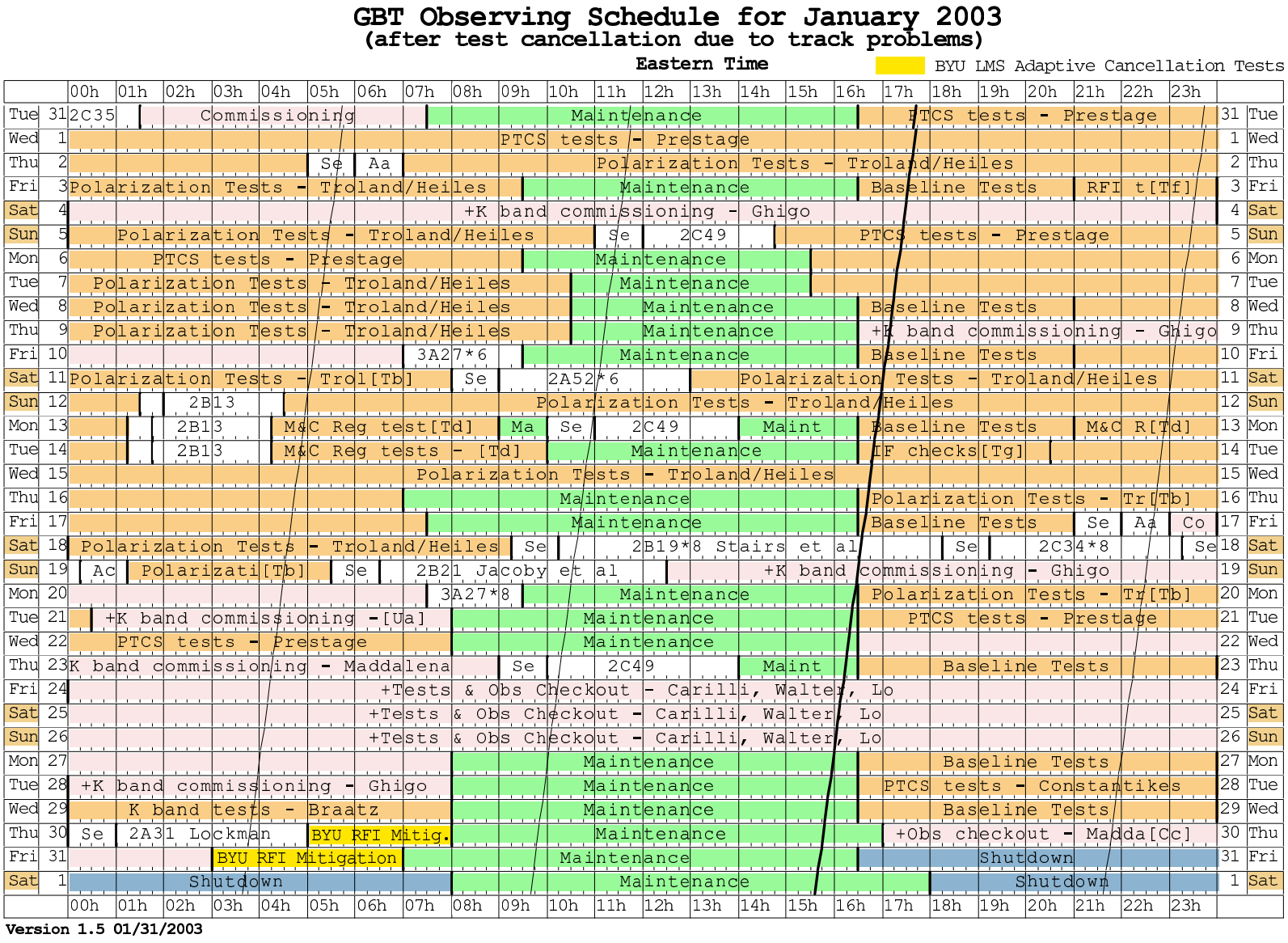


Table C.6: January 2003 GBT schedule (after test cancellation)

C.2 GBT Track Problems

This section contains copies of two e-mails given to me by Rick Fisher, detailing some of the specific problems encountered with the GBT azimuth track during my visit to Green Bank (see Section 4.7.6). The first e-mail is from Bob Anderson and Richard Prestage detailing the finding of a severely cracked track wear plate. The second is a message from John Ford explaining how the azimuth track closure would affect my tests (only relevant portions of this e-mail are included).

Date: Fri., 31 Jan 2003 14:04:10 (EST)
From: Richard Prestage
To: gbstaff@nrao.edu
Subject: telescope status

All:

Yesterday, we discovered another cracked track wear plate. The crack was over 75% through the thickness of the plate, and it's orientation is such that, if it cracks all the way through, it could leave a piece free. We are taking steps to bolt the piece in place. We are trying to get the tooling to do this, but at present we will not have the job completed until Monday. Since we do not want to risk driving over this piece without someone watching it until the bolt is in place, we are temporarily suspending azimuth movement of the telescope. We will use the time through this weekend for a crew to inspect every plate, grind smooth the bottom edges and perform shimming work. Our original plan was to get the inspection done by mid-February; the possibility of another plate being cracked as bad as this one is driving us to complete this inspection by early next week.

Accordingly, the telescope will be closed for general operations from today, Friday 31st, until the evening of Tuesday 4th February at the earliest. Depending upon progress (and weather) resumption of general operations may not happen until Wednesday 5th.

Arrangements have been made to perform other telescope work during this time, including receiver room RFI work and software and other testing activities. If you have any additional activities that could be done over the weekend or Monday or Tuesday, please let John Ford know.

Bob Anderson
Richard Prestage

Date: Fri, 31 Jan 2003 14:01:44
From: John Ford <jford@nrao.edu>

From our meeting, I got the following:

General: 1) Track work to be done Saturday, possibly Sunday, Monday, and Tuesday. Wednesday if needed...

3) We'll let Rick's scheduled time go on as planned, with the antenna in an arbitrary (snow dump, stow, or access) elevation. L band receiver in the focus...

The Details:

Today: Rick's time as scheduled, except that the telescope is parked in access. Roger to do some cursory checks of the K band system. (??? -> 4:30) Will not interfere with Rick...

Rick's time as scheduled tonight/tomorrow. Once the L band is in place, the subreflector will be positioned and then the servo system will be locked out until sometime Tuesday. I think Toney can't use the M&C regression test time scheduled for Saturday morning, so I propose to let Rick carry on until a civilized hour (8:00?), if he wants to...

C.3 Red-shifted OH Sources

Hydroxyl (OH) masers typically contain double spectral lines due to expanding shells from OH stars [1]. The lower line is referred to as the red-shifted peak, while the upper is the blue-shifted peak. Not only are the two peaks affected by the Doppler effect, but the OH stars are moving relative to the observer. I searched several different OH catalogs looking for highly red-shifted sources coincidentally above the horizon with GLONASS 789. The following tables are a result of these catalog searches. For each IRAS name, I tried to include either the 1950 or 2000 epoch for right ascension and declination. In some of the catalogs, the right ascension and declination were not provided (the source coordinates were given in galactic longitude and latitude). In these cases, the online version of the catalog converts the galactic coordinates to right ascension and declination in either the 1950 or 2000 epoch [49]. SL and SH are the peak power levels in Janskies (Jy) for the low and high spectral peaks, respectively. VL and VH are the velocities of the red- and blue-shifted peaks, respectively. In each of the following tables, the highly red-shifted sources are shaded in gray.

In the source column, the primary author and the year of publication are included. More detailed references are provided in the Bibliography: Hekkert, 1989 [50]; Hekkert, 1991 [51]; Sevenster, 1997 [52]; and Sevenster, 2001 [53]. Since much of the data was transcribed by hand from a printed copy to the tables, there may be some errors. Therefore, when using this list for making observations, it would be wise to use original copies of each catalog to verify the source coordinates before tracking.

Table C.7: Higher power red-shifted 1612 MHz OH spectral lines

IRAS Name	RA 2000 epoch	DE 2000 epoch	RA 1950 epoch	DE 1950 epoch	SL (Jy)	SH (Jy)	VL (km/s)	VH (km/s)	Source
05151+6312			05 15 06.0	+63 12 51	15.00	11.00	37.0	66.0	Hekkert, 1989
07113-2747	07 13 23	-27 52.9	07 11 23	-27 47.8	6.03	2.82	87.0	102.4	Hekkert, 1991
07445-2613	07 46 38	-26 20.5	07 44 34	-26 13.1	3.75	1.65	76.9	86.3	Hekkert, 1991
08089-3511	08 10 48	-35 20.8	08 08 54	-35 11.8	2.51	1.26	59.4	87.0	Hekkert, 1991
09429-2148	09 45 17	-22 01.9	09 42 58	-21 48.1	5.69	4.79	27.8	52.2	Hekkert, 1991
16260+3454	16 27 51	+34 48.1	16 26 00	+34 54.7	2.27	0.49	42.1	68.8	Hekkert, 1991
17164-3226	17 19 40.776	-32 29 51.69			2.332	1.615	73.5	101.2	Sevenster, 1997
17305-2751	17 33 43	-27 53.0	17 30 34	-27 51.0	1.29		122.8		Hekkert, 1991
17348-3051	17 38 01.627	-30 53 17.31			2.444	2.436	100.6	129.8	Sevenster, 1997
17358-2711	17 39 00.981	-27 13 23.65			1.214	0.714	211.7	201.4	Sevenster, 1997
17404-2713	17 43 37	-27 14.8	17 40 29	-27 13.5	6.38	5.20	32.0	58.3	Hekkert, 1991
	17 43 37	-27 14.8	17 40 29	-27 13.5	0.63	0.32	215.8	243.1	Hekkert, 1991
17430-2844	17 45 55.797	-28 45 18.85			1.755	2.035	145.9	173.6	Sevenster, 1997
17484-2350	17 51 29.404	-23 51 39.95			2.522	1.113	106.5	134.2	Sevenster, 1997
17486-2345	17 51 43	-23 46.5	17 48 40	-23 45.7	3.31	0.86	105.7	134.8	Hekkert, 1991
17543-3102	17 57 33.587	-31 03 02.08			3.353		121.4		Sevenster, 1997
17552-2339	17 58 17.649	-23 39 14.60			2.274	1.246	93.3	105.0	Sevenster, 1997
17560-2027	17 59 05.007	-20 27 23.95			4.251	2.576	191.1	217.4	Sevenster, 1997
	17 59 04.975	-20 27 24.77			3.017	1.568	190.7	217.9	Sevenster, 2001
18004-2131	18 03 55.720	-21 31 37.67			1.741	0.299	192.9	202.0	Sevenster, 2001
18006-3213	18 03 53	-32 13.0	18 00 37	-32 13.2	2.17	1.27	169.1	184.4	Hekkert, 1991
18167-1209			18 16 47.4	-12 09 27	6.0	6.0	166.0	186.0	Hekkert, 1989
18257-1000	18 28 28	-09 58.3	18 25 43	-10 00.3	16.00	28.17	97.2	134.7	Hekkert, 1991
18257-1052	18 28 30.818	-10 50 52.99			5.805	3.056	118.0	154.4	Sevenster, 2001
18262-0735	18 28 59	-07 33.4	18 26 17	-07 35.4	1.33	1.80	65.5	93.3	Hekkert, 1991
18277-1059	18 30 33	-10 57.5	18 27 46	-10 59.6	1.30	1.40	85.9	121.6	Hekkert, 1991
18282-0943	18 31 02	-09 41.2	18 28 17	-09 43.4	3.74	2.65	108.0	152.5	Hekkert, 1991
18257-1000	18 28 30.937	-09 58 14.33			18.928	17.786	97.6	133.9	Sevenster, 2001
18257-1052			18 25 44.3	-10 52 50	12.70	6.00	118.1	154.5	Hekkert, 1989
15308-0911			18 30 49.0	-09 09 00	1.40	2.70	76.7	108.7	Hekkert, 1989
18308-1000			18 30 49.2	-09 59 56	5.20	6.10	103.5	129.5	Hekkert, 1989
18310-0806			18 31 06.5	-08 06 22	1.30	8.10	96.5	117.5	Hekkert, 1989
18355-0712			18 35 33.4	-07 12 34	9.70	5.50	130.3	156.0	Hekkert, 1989
18396-0807	18 42 22	-08 05.0	18 39 39	-08 07.9	3.08	3.25	79.0	105.7	Hekkert, 1991
18432-0149	18 45 53	-01 46.7	18 43 17	-01 49.9	11.38	6.71	49.5	84.7	Hekkert, 1991
18440-0020	18 46 38	-00 17.3	18 44 04	-00 20.5	2.08	3.33	125.4	149.9	Hekkert, 1991
18043-2116	18 07 20.861	-21 16 10.86			8.394	0.260	71.4	99.2	Sevenster, 1997
18488-0107	18 51 26	-01 03.9	18 48 51	-01 07.5	9.67	5.40	55.7	96.7	Hekkert, 1991
18460-0151			18 46 07.2	-01 51 56	7.00	9.30	111.8	141.6	Hekkert, 1989
18460-0254	18 48 40	-02 50.6	18 46 03	-02 54.0	51.46	48.13	79.1	119.9	Hekkert, 1991
	18 48 41.947	-02 50 29.23			29.220	35.136	79.3	117.8	Sevenster, 2001
18540+0302	18 56 35	+03 06.7	18 54 04	+03 02.7	4.81	2.54	84.2	121.6	Hekkert, 1991
18549+0208	18 57 26	+02 12.2	18 54 55	+02 08.1	20.73	15.87	64.9	92.3	Hekkert, 1991
18551+0159			18 54 56.0	+02 08 14	18.00	14.50	64.0	92.0	Hekkert, 1989
18596+0315	19 02 06.259	+03 20 5.47			6.643	4.966	74.7	102.0	Sevenster, 2001
19017+0608	19 04 09	+06 13.2	19 01 42	+06 08.7	3.20	4.15	135.3	163.8	Hekkert, 1991
	19 04 09.722	+06 13 5.89			3.507	5.853	133.7	163.2	Sevenster, 2001
19026+0336	19 05 10	+03 40.9	19 02 40	+03 36.3	2.07	1.11	101.4	107.0	Hekkert, 1991
19244+1115	19 26 47	+11 21.2	19 24 26	+11 15.1	96.25	10.00	45.7	107.1	Hekkert, 1991
19067+0811			19 06 43.8	+08 11 41	5.00	24.00	43.1	75.8	Hekkert, 1989
19361-1658	19 39 01	-16 51.9	19 36 09	-16 58.8	1.46	2.51	50.0	65.1	Hekkert, 1991

Table C.8: Lower power red-shifted 1612 MHz OH spectral lines

IRAS Name	RA 2000 epoch	DE 2000 epoch	RA 1950 epoch	DE 1950 epoch	SL (Jy)	SH (Jy)	VL (km/s)	VH (km/s)	Source
09089-2149	09 11 10	-22 02.1	09 08 55	-21 49.8	1.58	0.76	77.3	107.4	Hekkert, 1991
10436-3459	10 45 59	-35 15.0	10 43 40	-34 59.2	0.10	0.15	126.5	149.1	Hekkert, 1991
17112-3445	17 14 09.820	-34 45 28.03			0.237	0.379	137.2	268.6	Sevenster, 1997
17179-2729	17 21 07	-27 32.8	17 17 59	-27 29.8	0.19		228.2		Hekkert, 1991
17238-3027	17 27 11.923	-30 30 18.88			0.399		170.8		Sevenster, 1997
17306-3328	17 33 42.686	-33 30 25.46			0.231		270.0		Sevenster, 1997
17305-2751	17 33 43.249	-27 53 00.72			0.926		121.7		Sevenster, 1997
17377-2859			17 37 43.0	-29 00 34	0.30	0.20	248.0	277.0	Hekkert, 1989
17424-2806			17 42 24.0	-28 04 27	0.20	0.40	235.0	246.0	Hekkert, 1989
17434-2858	17 46 35.313	-28 58 56.71			0.895	1.689	99.2	126.9	Sevenster, 1997
17433-2921	17 46 35.615	-29 21 52.19			0.312	0.476	169.2	202.8	Sevenster, 1997
17433-2918	17 46 38.067	-29 19 28.38			0.561	0.301	195.5	220.3	Sevenster, 1997
17433-2828	17 46 39.049	-28 28 05.79			0.437	0.570	153.2	183.8	Sevenster, 1997
17443-2839	17 47 25.081	-28 36 34.22			0.911	1.686	129.8	153.2	Sevenster, 1997
17350-2413	17 38 09	-24 14.8	17 35 05	-24 13.1	0.27		225.9		Hekkert, 1991
17351-2317	17 38 10	-23 18.9	17 35 07	-23 17.2	0.10	0.13	166.5	189.1	Hekkert, 1991
17361-2757	17 39 18	-27 58.9	17 36 09	-27 57.3	0.41	0.35	136.6	173.4	Hekkert, 1991
17376-2648	17 40 48	-26 50.4	17 37 40	-26 48.8	1.65	0.74	82.5	107.9	Hekkert, 1991
17379-2618	17 41 06	-26 20.4	17 37 59	-26 18.9	0.37	0.21	199.8	227.0	Hekkert, 1991
17382-2531	17 41 20	-25 32.9	17 38 14	-25 31.4	0.53		153.8		Hekkert, 1991
17387-2617	17 41 52.457	-26 19 10.82			0.336	0.443	201.4	227.7	Sevenster, 1997
17404-2713	17 43 37	-27 14.8	17 40 29	-27 13.5	6.38	5.20	32.0	58.3	Hekkert, 1991
	17 43 37	-27 14.8	17 40 29	-27 13.5	0.63	0.32	215.8	243.1	Hekkert, 1991
	17 43 37.355	-27 13 07.86			0.369	0.295	216.0	242.2	Sevenster, 1997
17411-2843	17 44 14.949	-28 45 08.08			0.648	0.726	132.7	124.0	Sevenster, 1997
17426-2804	17 45 35.664	-28 04 44.83			0.302	0.644	243.7	238.6	Sevenster, 1997
17432-1952	17 46 11	-19 53.4	17 43 13	-19 52.3	0.13	-0.09	107.3	124.1	Hekkert, 1991
17434-2858	17 46 35.313	-28 58 56.71			0.895	1.689	99.2	126.9	Sevenster, 1997
17433-2921	17 46 35.615	-29 21 52.19			0.312	0.476	169.2	202.8	Sevenster, 1997
17433-2918	17 46 38.067	-29 19 28.38			0.561	0.301	195.5	220.3	Sevenster, 1997
17433-2828	17 46 39.049	-28 28 05.79			0.437	0.570	153.2	183.8	Sevenster, 1997
17443-2839	17 47 25.081	-28 36 34.22			0.911	1.686	129.8	153.2	Sevenster, 1997
17466-2727	17 50 01.932	-27 25 02.16			0.355	1.878	105.0	132.7	Sevenster, 1997
17471-3235	17 50 26.232	-32 36 09.72			1.204	0.822	156.4	178.3	Sevenster, 1997
17479-2947	17 51 10.102	-29 48 48.25			0.267		333.1		Sevenster, 1997
17484-1511	17 51 20	-15 12.4	17 48 28	-15 11.7	1.56	0.51	75.0	110.0	Hekkert, 1991
17488-2716	17 52 04.416	-27 17 39.24			0.260	0.516	141.5	176.5	Sevenster, 1997
17489-2824	17 52 08.755	-28 24 56.25			0.269	0.337	296.3	319.6	Sevenster, 1997
17506-2955	17 53 50.715	-29 55 28.92			0.312	0.758	171.0	192.9	Sevenster, 1997
17518-2054	17 54 47	-20 54.5	17 51 48	-20 54.1	0.28		155.2		Hekkert, 1991
17522-2504	17 55 13.586	-25 07 00.91			0.920	0.585	194.1	229.1	Sevenster, 1997
17528-3052	17 56 04.419	-30.52 56.62			0.397	0.286	271.8	296.6	Sevenster, 1997
17531-1807	17 56 02	-18 07.8	17 53 06	-18 07.4	0.40	0.37	132.2	165.3	Hekkert, 1991
17532-1311	17 56 04	-13 11.7	17 53 14	-13 11.4	0.66		129.4		Hekkert, 1991
17541-2706	17 57 16	-27 06.6	17 54 08	-27 06.2	0.45	0.33	233.0	260.2	Hekkert, 1991
17545-1900	17 57 30.037	-19 00 08.92			0.388		195.6		Sevenster, 1997
17545-2817	17 57 41.359	-28 17 58.47			0.207		127.2		Sevenster, 1997
17543-2837	17 57 52.256	-28 39 19.01			0.315		365.2		Sevenster, 1997
17554-2816	17 58 31.636	-28 16 24.06			0.413	0.417	293.7	303.9	Sevenster, 1997
17557-2657	17 58 50.408	-26 53 08.10			0.237	0.226	168.1	203.1	Sevenster, 1997

Table C.9: Lower power red-shifted 1612 MHz OH spectral lines (continued)

IRAS Name	RA 2000 epoch	DE 2000 epoch	RA 1950 epoch	DE 1950 epoch	SL (Jy)	SH (Jy)	VL (km/s)	VH (km/s)	Source
17565-2035	17 59 34	-20 36.1	17 56 36	-20 35.9	0.86	0.37	190.4	217.9	Hekkert, 1991
17572-2934	18 00 30	-29 34.1	17 57 18	-29 34.1	0.13	0.17	194.8	229.8	Hekkert, 1991
17576-2653	18 00 49.500	-26 53 12.52			0.417		138.9		Sevenster, 1997
17577-2641	18 00 50	-26 41.1	17 57 43	-26 41.0	0.20	0.53	225.9	248.7	Hekkert, 1991
17582-2619	18 01 21.546	-26 19 36.75			1.022	0.904	163.7	185.6	Sevenster, 1997
18006-2528	18 03 43.929	-25 28 32.95			0.708		137.4		Sevenster, 1997
18009-2128	18 03 55.725	-21 31 37.76			1.390	0.437	192.6	201.4	Sevenster, 1997
18016-2646	18 04 25.917	-26 43 47.15			0.320		257.2		Sevenster, 1997
18020-1548	18 04 55	-15 48.5	18 02 03	-15 48.8	0.32	0.11	142.2	172.9	Hekkert, 1991
18039-1903	18 06 53	-19 03.1	18 03 56	-19 03.5	0.87	0.54	143.3	170.1	Hekkert, 1991
17594-1938	18 02 26.823	-19 38 14.62			0.057	0.264	181.6	206.5	Sevenster, 2001
18040-2726	18 07 08	-27 26.0	18 04 00	-27 26.4	1.25		157.5		Hekkert, 1991
18042-2131	18 07 22.392	-21 30 54.10			0.698	0.767	176.5	210.1	Sevenster, 1997
18042-2131	18 07 22.383	-21 30 54.61			0.372	0.910	177.0	208.8	Sevenster, 2001
18044-2324	18 07 28.723	-23 23 49.35			0.849	0.309	149.1	173.9	Sevenster, 1997
18051-2415	18 08 12.819	-24 14 36.73			0.685	0.264	105.3	134.5	Sevenster, 1997
18061-3140	18 09 24	-31 39.4	18 06 09	-31 40.0	0.31	0.17	214.9	243.9	Hekkert, 1991
18077-2614	18 10 54	-26 13.5	18 07 47	-26 14.2	1.01	0.34	85.5	113.6	Hekkert, 1991
18083-2128	18 11 22	-21 28.1	18 08 22	-21 28.8	0.64	1.08	104.3	130.9	Hekkert, 1991
	18 11 22.210	-21 28 09.35			0.473	0.768	103.9	130.1	Sevenster, 1997
18087-1512	18 11 37	-15 11.4	18 08 45	-15 12.2	0.21	0.13	178.5	212.6	Hekkert, 1991
18116-1443	18 14 27	-14 42.8	18 11 36	-14 43.8	0.98	0.57	124.3	147.7	Hekkert, 1991
18118-2420	18 14 58	-24 19.5	18 11 54	-24 20.5	0.18	0.25	102.5	131.5	Hekkert, 1991
18151-2839	18 18 16	-28 38.0	18 15 06	-28 39.2	0.26	0.22	95.3	124.0	Hekkert, 1991
18151-2629	18 18 13	-26 28.3	18 15 06	-26 29.5	0.09	0.11	83.6	115.5	Hekkert, 1991
18191-0707	18 21 48	-07 06.4	18 19 06	-07 07.9	0.34	0.28	80.9	103.6	Hekkert, 1991
18195-2804	18 22 40	-28 03.1	18 19 31	-28 04.7	0.14	0.39	97.7	130.3	Hekkert, 1991
18254-2418	18 28 30	-24 16.1	18 25 27	-24 18.1	0.24	0.16	115.0	134.0	Hekkert, 1991
18376-0848	18 40 20	-08 45.8	18 37 36	-08 48.6	0.59	0.49	114.9	146.7	Hekkert, 1991
18477+0243	18 50 13	+02 47.2	18 47 43	+02 43.6	0.21	0.20	108.3	136.7	Hekkert, 1991
18490+0302	18 51 36	+03 05.9	18 49 05	+03 02.3	0.57	0.21	113.8	149.2	Hekkert, 1991
19029-0208	19 05 34	-02 03.8	19 02 58	-02 08.4	0.15	0.14	85.4	120.6	Hekkert, 1991
19147+5004	19 16 03	+50 09.5	19 14 45	+50 04.1	0.07	0.08	245.3	270.4	Hekkert, 1991
19244+0946	19 26 50	+09 52.3	19 24 27	+09 46.2	0.15	0.07	162.6	167.6	Hekkert, 1991
19309+2022	19 33 07	+20 29.0	19 30 56	+20 22.5	0.13	0.09	98.9	108.1	Hekkert, 1991
19581+4723	19 59 39	+47 31.5	19 58 09	+47 23.3	0.09		135.4		Hekkert, 1991

C.4 GBT Delay Predictions

This section contains the precalculated data realignment parameters for tests performed with the GBT between January 30–February 1, 2003 (see Sections 4.3 and 4.7.3). All times are EST. Az and El are the azimuth and elevation of GLONASS satellite #789 in ten minute intervals. Parameters d_w and t_w are the relative wave propagation distance/delay between channels, while t_{net} is the net delay (further described in Section 4.3). As detailed in Figure 4.13, the relative cable delay, $t_c = 10.73$ μs, was included in these computations. Note that the filter orders, p, detailed in the tables correspond to their highest respective bandwidths shown in Table 4.2. The final seven columns contain the necessary delay (in samples) for each respective filter order. A positive number suggests the auxiliary channel should be delayed.

Table C.10: Data realignment parameters—Thursday, January 30, 2003

Time	Az	El	d_w(m)	$t_w(\mu s)$	$t_{net}(\mu s)$	$p = 42$	$p = 30$	$p = 12$	$p = 5$	$p = 4$	$p = 3$	$p = 2$
5:30	337.7	7.3	-443.80	-1.48	9.25	-17.18	-10.06	3.31	18.08	21.70	28.56	34.55
5:40	334.9	10.8	-504.12	-1.68	9.05	-17.27	-10.16	3.10	17.63	21.19	27.91	33.78
5:50	331.6	14.0	-571.31	-1.91	8.82	-17.36	-10.28	2.88	17.13	20.61	27.18	32.92
6:00	327.8	16.9	-644.04	-2.15	8.58	-17.46	-10.41	2.64	16.59	19.99	26.39	31.99
6:10	323.5	19.3	-721.61	-2.41	8.32	-17.57	-10.55	2.37	16.02	19.33	25.55	30.99
6:20	318.8	21.3	-800.69	-2.67	8.06	-17.68	-10.69	2.11	15.43	18.65	24.69	29.98
6:30	313.7	22.7	-881.43	-2.94	7.79	-17.79	-10.84	1.84	14.83	17.96	23.82	28.94
6:40	308.3	23.6	-960.96	-3.21	7.52	-17.90	-10.98	1.57	14.24	17.28	22.95	27.92
6:50	302.7	23.8	-1038.69	-3.46	7.27	-18.00	-11.12	1.31	13.67	16.62	22.11	26.93
7:00	297.1	23.4	-1111.18	-3.71	7.02	-18.10	-11.25	1.07	13.13	16.00	21.33	26.00
7:10	291.0	22.4	-1183.12	-3.95	6.78	-18.20	-11.38	0.83	12.59	15.38	20.55	25.07
7:20	285.6	20.8	-1241.79	-4.14	6.59	-18.28	-11.48	0.63	12.16	14.88	19.91	24.32
7:30	280.5	18.7	-1291.37	-4.31	6.42	-18.35	-11.57	0.46	11.79	14.46	19.37	23.69
7:40	275.6	16.2	-1331.57	-4.44	6.29	-18.41	-11.64	0.33	11.49	14.11	18.94	23.17
7:50	271.1	13.5	-1360.32	-4.54	6.19	-18.45	-11.69	0.23	11.28	13.87	18.63	22.80
8:00	266.8	10.4	-1379.51	-4.60	6.13	-18.47	-11.73	0.17	11.14	13.70	18.42	22.56
13:00	156.1	2.6	479.43	1.60	12.33	-15.91	-8.41	6.41	24.93	29.59	38.57	46.39
13:10	153.9	6.9	524.25	1.75	12.48	-15.85	-8.33	6.56	25.27	29.98	39.06	46.96
13:20	151.8	11.4	561.78	1.87	12.60	-15.80	-8.26	6.68	25.54	30.30	39.46	47.45
13:30	149.6	16.0	595.40	1.99	12.72	-15.75	-8.20	6.80	25.79	30.58	39.83	47.88
13:40	147.3	20.8	623.27	2.08	12.81	-15.72	-8.16	6.89	26.00	30.82	40.13	48.23
13:50	144.9	25.7	644.12	2.15	12.88	-15.69	-8.12	6.96	26.15	31.00	40.36	48.50
14:00	142.2	30.6	660.64	2.20	12.93	-15.66	-8.09	7.01	26.28	31.14	40.53	48.71
14:10	139.1	35.6	671.56	2.24	12.97	-15.65	-8.07	7.05	26.36	31.24	40.65	48.85
14:20	135.5	40.6	676.30	2.26	12.99	-15.64	-8.06	7.07	26.39	31.28	40.70	48.91
14:30	131.1	45.4	677.85	2.26	12.99	-15.64	-8.06	7.07	26.41	31.29	40.72	48.93
14:40	125.5	50.0	676.52	2.26	12.99	-15.64	-8.06	7.07	26.40	31.28	40.71	48.92
14:50	118.6	54.2	670.41	2.24	12.97	-15.65	-8.07	7.05	26.35	31.23	40.64	48.84
15:00	109.9	57.6	664.96	2.22	12.95	-15.66	-8.08	7.03	26.31	31.18	40.58	48.77
15:10	99.4	60.1	656.89	2.19	12.92	-15.67	-8.10	7.00	26.25	31.11	40.49	48.67
15:20	87.5	61.3	649.27	2.17	12.90	-15.68	-8.11	6.98	26.19	31.05	40.41	48.57
15:30	75.3	61.1	641.51	2.14	12.87	-15.69	-8.12	6.95	26.14	30.98	40.33	48.47
15:40	64.1	59.4	637.73	2.13	12.86	-15.70	-8.13	6.94	26.11	30.95	40.29	48.42
15:50	54.8	56.6	635.82	2.12	12.85	-15.70	-8.13	6.93	26.09	30.93	40.27	48.40
16:00	47.6	52.9	638.87	2.13	12.86	-15.69	-8.13	6.94	26.12	30.96	40.30	48.43
16:10	42.2	48.7	644.65	2.15	12.88	-15.69	-8.12	6.96	26.16	31.01	40.36	48.51
16:20	38.3	44.1	655.34	2.19	12.92	-15.67	-8.10	7.00	26.24	31.10	40.48	48.65
16:30	35.6	39.3	670.46	2.24	12.97	-15.65	-8.07	7.05	26.35	31.23	40.64	48.84
16:40	33.9	34.5	689.99	2.30	13.03	-15.62	-8.04	7.11	26.50	31.39	40.85	49.09
16:50	32.9	29.7	712.90	2.38	13.11	-15.59	-8.00	7.19	26.67	31.59	41.10	49.38
17:00	32.4	24.9	737.93	2.46	13.19	-15.56	-7.95	7.27	26.85	31.80	41.37	49.70
17:10	32.5	20.1	768.16	2.56	13.29	-15.52	-7.90	7.38	27.08	32.06	41.70	50.09
17:20	33.0	15.5	799.81	2.67	13.40	-15.47	-7.84	7.48	27.31	32.33	42.04	50.50
17:30	33.9	11.0	833.95	2.78	13.51	-15.43	-7.78	7.60	27.56	32.62	42.41	50.94

Table C.11: Data realignment parameters—Friday, January 31, 2003

Time	Az	El	d_w(m)	$t_w(\mu s)$	$t_{net}(\mu s)$	$p = 42$	$p = 30$	$p = 12$	$p = 5$	$p = 4$	$p = 3$	$p = 2$
3:40	333.6	5.9	-537.58	-1.79	8.94	-17.31	-10.22	2.99	17.38	20.90	27.54	33.35
3:50	332.8	10.3	-551.34	-1.84	8.89	-17.33	-10.25	2.95	17.28	20.78	27.40	33.17
4:00	331.6	14.6	-570.06	-1.90	8.83	-17.36	-10.28	2.88	17.14	20.62	27.19	32.93
4:10	329.8	18.9	-596.75	-1.99	8.74	-17.39	-10.33	2.79	16.95	20.39	26.90	32.59
4:20	327.5	23.0	-628.29	-2.10	8.63	-17.44	-10.39	2.69	16.71	20.13	26.56	32.19
4:30	324.5	27.0	-666.40	-2.22	8.51	-17.49	-10.45	2.56	16.43	19.80	26.15	31.70
4:40	320.8	30.7	-709.58	-2.37	8.36	-17.55	-10.53	2.42	16.11	19.43	25.68	31.15
4:50	316.3	34.0	-757.85	-2.53	8.20	-17.62	-10.62	2.25	15.75	19.02	25.16	30.53
5:00	310.9	36.9	-810.04	-2.70	8.03	-17.69	-10.71	2.08	15.36	18.57	24.59	29.86
5:10	304.8	39.0	-865.15	-2.89	7.84	-17.76	-10.81	1.89	14.95	18.10	23.99	29.15
5:20	298.0	40.5	-919.99	-3.07	7.66	-17.84	-10.91	1.71	14.55	17.63	23.40	28.45
5:30	290.7	41.1	-975.29	-3.25	7.48	-17.92	-11.00	1.52	14.14	17.16	22.80	27.74
5:40	283.4	40.8	-1027.45	-3.43	7.30	-17.99	-11.10	1.35	13.75	16.71	22.23	27.07
5:50	276.1	39.7	-1075.68	-3.59	7.14	-18.05	-11.18	1.19	13.39	16.30	21.71	26.45
6:00	269.3	37.8	-1118.39	-3.73	7.00	-18.11	-11.26	1.04	13.07	15.94	21.25	25.90
6:10	263.0	35.3	-1153.62	-3.85	6.88	-18.16	-11.32	0.92	12.81	15.63	20.87	25.45
6:20	257.3	32.2	-1182.00	-3.94	6.79	-18.20	-11.37	0.83	12.60	15.39	20.56	25.09
6:30	252.2	28.7	-1201.72	-4.01	6.72	-18.23	-11.41	0.76	12.46	15.22	20.34	24.84
6:40	247.6	25.0	-1211.24	-4.04	6.69	-18.24	-11.43	0.73	12.38	15.14	20.24	24.71
6:50	243.5	21.1	-1212.03	-4.04	6.69	-18.24	-11.43	0.73	12.38	15.14	20.23	24.70
7:00	239.7	17.1	-1202.91	-4.01	6.72	-18.23	-11.41	0.76	12.45	15.21	20.33	24.82
7:10	236.2	13.0	-1185.00	-3.95	6.78	-18.20	-11.38	0.82	12.58	15.37	20.53	25.05
12:00	130.9	4.2	989.68	3.30	14.03	-15.21	-7.50	8.12	28.72	33.95	44.10	52.93
12:10	127.8	8.1	1031.92	3.44	14.17	-15.15	-7.43	8.26	29.03	34.32	44.56	53.47
12:20	124.6	12.1	1066.44	3.56	14.29	-15.11	-7.37	8.38	29.29	34.61	44.93	53.92
12:30	121.2	16.1	1093.71	3.65	14.38	-15.07	-7.32	8.47	29.49	34.84	45.23	54.27
12:40	117.6	20.2	1111.58	3.71	14.44	-15.04	-7.28	8.53	29.62	35.00	45.42	54.50
12:50	113.6	24.1	1123.07	3.75	14.48	-15.03	-7.26	8.57	29.71	35.10	45.55	54.64
13:00	109.2	27.9	1125.84	3.76	14.49	-15.02	-7.26	8.58	29.73	35.12	45.58	54.68
13:10	104.3	31.4	1121.39	3.74	14.47	-15.03	-7.27	8.56	29.70	35.08	45.53	54.62
13:20	98.8	34.6	1108.98	3.70	14.43	-15.05	-7.29	8.52	29.60	34.97	45.39	54.46
13:30	92.7	37.3	1089.86	3.64	14.37	-15.07	-7.32	8.46	29.46	34.81	45.19	54.22
13:40	86.1	39.4	1064.43	3.55	14.28	-15.11	-7.37	8.37	29.27	34.59	44.91	53.89
13:50	79.0	40.8	1033.56	3.45	14.18	-15.15	-7.42	8.27	29.05	34.33	44.58	53.50
14:00	71.6	41.3	999.75	3.33	14.06	-15.20	-7.48	8.15	28.79	34.04	44.21	53.06
14:10	64.4	41.0	963.07	3.21	13.94	-15.25	-7.55	8.03	28.52	33.73	43.81	52.59
14:20	57.4	39.8	924.98	3.09	13.82	-15.30	-7.62	7.90	28.24	33.40	43.40	52.10
14:30	51.0	37.9	885.79	2.95	13.68	-15.36	-7.69	7.77	27.95	33.07	42.98	51.60
14:40	45.4	35.2	850.03	2.84	13.57	-15.40	-7.75	7.65	27.68	32.76	42.59	51.14
14:50	40.7	32.1	816.41	2.72	13.45	-15.45	-7.81	7.54	27.43	32.47	42.22	50.71
15:00	36.7	28.5	785.55	2.62	13.35	-15.49	-7.87	7.43	27.20	32.21	41.89	50.32
15:10	33.5	24.6	759.42	2.53	13.26	-15.53	-7.91	7.35	27.01	31.99	41.61	49.98
15:20	30.9	20.4	736.74	2.46	13.19	-15.56	-7.95	7.27	26.84	31.79	41.36	49.69
15:30	29.0	16.2	719.75	2.40	13.13	-15.58	-7.98	7.21	26.72	31.65	41.18	49.47
15:40	27.6	11.8	707.34	2.36	13.09	-15.60	-8.01	7.17	26.62	31.54	41.04	49.31
15:50	26.7	7.4	699.87	2.33	13.06	-15.61	-8.02	7.15	26.57	31.48	40.96	49.22
16:00	26.3	3.1	698.18	2.33	13.06	-15.61	-8.02	7.14	26.56	31.46	40.94	49.20

Table C.12: Data realignment parameters—Saturday, February 1, 2003

Time	Az	El	d_w(m)	$t_w(\mu s)$	$t_{net}(\mu s)$	$p = 42$	$p = 30$	$p = 12$	$p = 5$	$p = 4$	$p = 3$	$p = 2$
2:00	325.8	9.5	-702.43	-2.34	8.39	-17.54	-10.52	2.44	16.16	19.49	25.76	31.24
2:10	326.8	13.9	-672.76	-2.24	8.49	-17.50	-10.47	2.54	16.38	19.75	26.08	31.62
2:20	327.4	18.5	-647.18	-2.16	8.57	-17.46	-10.42	2.62	16.57	19.96	26.36	31.95
2:30	327.6	23.1	-625.91	-2.09	8.64	-17.44	-10.38	2.70	16.73	20.15	26.59	32.22
2:40	327.4	27.9	-607.43	-2.03	8.70	-17.41	-10.35	2.76	16.87	20.30	26.79	32.46
2:50	326.6	32.7	-595.09	-1.98	8.75	-17.39	-10.33	2.80	16.96	20.41	26.92	32.61
3:00	325.0	37.5	-590.22	-1.97	8.76	-17.39	-10.32	2.82	16.99	20.45	26.97	32.68
3:10	322.7	42.3	-587.95	-1.96	8.77	-17.38	-10.31	2.82	17.01	20.47	27.00	32.71
3:20	319.2	47.0	-592.44	-1.98	8.75	-17.39	-10.32	2.81	16.98	20.43	26.95	32.65
3:30	314.3	51.3	-603.57	-2.01	8.72	-17.40	-10.34	2.77	16.89	20.34	26.83	32.50
3:40	307.7	55.2	-618.35	-2.06	8.67	-17.42	-10.37	2.72	16.78	20.21	26.67	32.32
3:50	299.1	58.4	-636.54	-2.12	8.61	-17.45	-10.40	2.66	16.65	20.05	26.47	32.08
4:00	288.4	60.5	-659.27	-2.20	8.53	-17.48	-10.44	2.58	16.48	19.86	26.23	31.79
4:10	276.4	61.2	-684.83	-2.28	8.45	-17.52	-10.49	2.50	16.29	19.64	25.95	31.46
4:20	264.3	60.4	-711.89	-2.37	8.36	-17.55	-10.54	2.41	16.09	19.41	25.65	31.12
4:30	253.2	58.3	-736.99	-2.46	8.27	-17.59	-10.58	2.32	15.90	19.20	25.38	30.79
4:40	244.0	55.1	-760.95	-2.54	8.19	-17.62	-10.62	2.24	15.73	18.99	25.12	30.49
4:50	236.5	51.1	-781.83	-2.61	8.12	-17.65	-10.66	2.17	15.57	18.81	24.90	30.22
5:00	230.5	46.6	-797.96	-2.66	8.07	-17.67	-10.69	2.12	15.45	18.67	24.72	30.01
5:10	225.8	41.8	-810.02	-2.70	8.03	-17.69	-10.71	2.08	15.36	18.57	24.59	29.86
5:20	221.9	36.6	-817.81	-2.73	8.00	-17.70	-10.72	2.05	15.30	18.51	24.51	29.76
5:30	218.6	31.8	-813.95	-2.72	8.01	-17.69	-10.72	2.07	15.33	18.54	24.55	29.81
5:40	215.8	26.8	-806.24	-2.69	8.04	-17.68	-10.70	2.09	15.39	18.60	24.63	29.91
5:50	213.3	21.9	-791.29	-2.64	8.09	-17.66	-10.68	2.14	15.50	18.73	24.79	30.10
6:00	210.9	17.1	-767.52	-2.56	8.17	-17.63	-10.63	2.22	15.68	18.94	25.05	30.40
6:10	208.7	12.4	-738.47	-2.46	8.27	-17.59	-10.58	2.32	15.89	19.18	25.37	30.78
6:20	206.5	7.9	-701.53	-2.34	8.39	-17.54	-10.52	2.44	16.17	19.50	25.77	31.25
11:30	94.4	9.7	1359.83	4.54	15.27	-14.70	-6.84	9.36	31.47	37.12	48.11	57.68
11:40	90.2	12.8	1354.47	4.52	15.25	-14.71	-6.85	9.34	31.43	37.07	48.06	57.61
11:50	85.8	15.6	1339.77	4.47	15.20	-14.73	-6.88	9.29	31.32	36.95	47.90	57.42
12:00	81.1	18.2	1314.97	4.39	15.12	-14.76	-6.92	9.21	31.13	36.74	47.63	57.10
12:10	76.1	20.4	1281.22	4.27	15.00	-14.81	-6.98	9.10	30.88	36.45	47.26	56.67
12:20	70.8	22.1	1239.24	4.13	14.86	-14.87	-7.06	8.96	30.57	36.09	46.81	56.13
12:30	65.3	23.4	1188.49	3.96	14.69	-14.94	-7.15	8.79	30.19	35.65	46.26	55.48
12:40	59.7	24.0	1132.54	3.78	14.51	-15.02	-7.25	8.60	29.78	35.18	45.65	54.76
12:50	54.1	24.0	1071.43	3.57	14.30	-15.10	-7.36	8.39	29.33	34.65	44.99	53.98
13:00	48.7	23.4	1007.62	3.36	14.09	-15.19	-7.47	8.18	28.85	34.11	44.30	53.16
13:10	43.4	22.2	939.88	3.14	13.87	-15.28	-7.59	7.95	28.35	33.53	43.56	52.29
13:20	38.5	20.5	872.10	2.91	13.64	-15.37	-7.71	7.72	27.85	32.95	42.83	51.42
13:30	34.1	18.2	807.53	2.69	13.42	-15.46	-7.83	7.51	27.37	32.40	42.13	50.60
13:40	30.1	15.4	744.59	2.48	13.21	-15.55	-7.94	7.30	26.90	31.86	41.44	49.79
13:50	26.6	12.3	685.33	2.29	13.02	-15.63	-8.04	7.10	26.46	31.35	40.80	49.03
14:00	23.6	8.9	631.19	2.11	12.84	-15.71	-8.14	6.92	26.06	30.89	40.22	48.34

C.5 Hints for Choosing the LMS Adaptive Constant, μ

After literally hundreds of experiments, I have found that selecting μ while utilizing the original LMS vector update equation,

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \mu e[n]\mathbf{x}_n^*, \tag{C.1}$$

is relatively simple with a little experience and familiarity with the test scenario. Usually after only a couple trial adaptive constants, I was able to select a μ which would serve well for a long series of subsequent tests, especially if other test parameters such as bandwidth and filter order remained constant.

When selecting an adaptive constant, it is wise to remember a few general rules. The first is illustrated by Figure 4.25; as the filter order decreases, with all other parameters constant, eventually a larger adaptive constant will be required. Notice that the normalized LMS algorithm takes this into account,

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \frac{\mu}{\|\mathbf{x}_n\|^2} e[n]\mathbf{x}_n^*. \tag{C.2}$$

As the filter order decreases, so will $\|\mathbf{x}_n\|^2$, effectively creating a larger adaptive constant. Remember that $\mathbf{x}_n$ is a vector containing the p previous samples of $x[n]$ (see Eq. 4.5). Second, as the power in $x[n]$ increases, the adaptive constant should decrease in order to retain constant performance. Again, the normalized algorithm accounts for this.

There are tell-tale signs of an excessive adaptive constant; fortunately they are easy to detect. Full-scale instability results when the filter taps grow uncontrollably large. When this occurred in our platform, the filter taps exceeded the allowable range of even floating point precision, resulting in "NaN" or "not a number." This mostly occurred during the initial testing phase, before determining acceptable ranges for μ. More often, the presence of an unstable filter manifests itself through characteristics demonstrated in Figure C.1. A sure sign of an overly aggressive filter is one which adds to the overall input noise power at any frequency. In examples (a) and (b), noise suppression did occur at frequencies coincident with the dominant noise term, while noise amplification resulted at other frequencies. In one case where the dominant

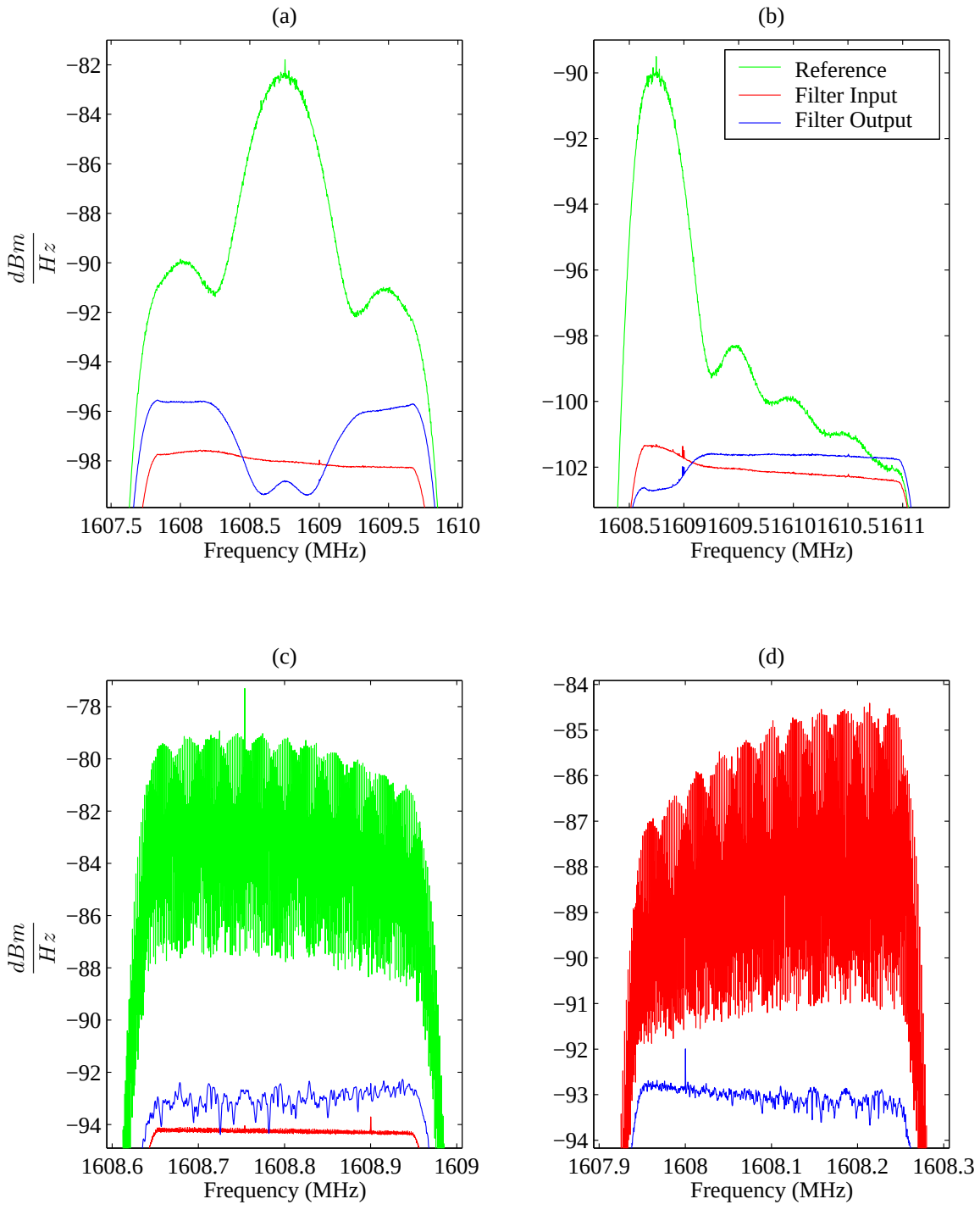


Figure C.1: Examples of the effect of an overly aggressive LMS adaptive constant, μ, resulting in unstable systems and undesired noise amplification. As a rule of thumb, if the signal power increases in any frequency bin after filtering, the system is adding noise to the signal, most likely due to an excessively large μ.

noise term spanned the entire bandwidth (see example (c)), noise was amplified over the entire bandwidth. This was not the case for all unstable filters, however. Example (d) demonstrates that significant interference cancellation can occur over the entire bandwidth, even revealing the hidden test tone at 1608 MHz, and still result in an unstable filter. Notice the misshaped baseline, particularly at the upper frequencies coincident with the strongest interference. Proper baselines are relatively smooth, without the bubbled appearance seen in examples (c) and (d), unless, of course, they are due to spectral characteristics of the desired signal (which was not the case in these examples). It is important to note that of the hundreds of tests I ran with the VSA and GBT, very few resulted in unstable systems. I found the LMS adaptive filter to be very robust and dynamic in a wide range of signal scenarios.

Bibliography

[1] B. F. Burke and F. Graham-Smith, *An Introduction to Radio Astronomy*, Cambridge University Press, second edition, 2002.

[2] N. L. Johnson and D. M. Rodvold, *Europe and Asia in Space 1993-1994*, Kaman Sciences Corporation, 1994.

[3] J. F. Bell, P. J. Hall, R. J. Sault, and L. Kewley, "Implementing Interference Suppression: Impacts on SKA System Design", in *Technological Pathways to the SKA*, August 2000.

[4] R. D. Ekers and J. F. Bell, "The Future of Radio Astonomy: Options for Dealing with Human Generated Interference", in *Preserving the Astronomical Sky*. International Astronomical Union (IAU), July 1999, number 196, pp. 33–42.

[5] A. Moffet, "JPL Work on Superconduction Filters", in *Proceedings of the Interference Identification and Excision Workshop*, 1982, pp. 91–95.

[6] J. R. Fisher, "An Impulse Noise Suppresser and other Thoughts on Interference Reduction", in *Proceedings of the Interference Identification and Excision Workshop*, 1982, pp. 100–111.

[7] R. Morrison, *Grounding and Shielding Techniques*, John Wiley & Sons, Inc., fourth edition, 1998.

[8] C. Barnbaum and R. F. Bradley, "A New Approach to Interference Excision in Radio Astronomy: Real-time Adaptive Cancellation", in *Astronomical Journal*, November 1998, vol. 116, pp. 2598–2614.

[9] A. Leshem and A.J. van der Veen, "Radio-Astronomical Imaging in the Presence of Strong Radio Interference", in *IEEE Transactions on Information Theory*, August 2000, vol. 46, pp. 1730–1747.

[10] A. B. Smolders, "Array Feed Cross-correlation for Beam Forming and RFI Suppression", in *Proceedings of the URSI 26th General Assembly*, August 1999, Toronto.

[11] A. B. Smolders and G. Hampson, "Deterministic RF Nulling in Phased Arrays for the Next Generation of Radio Telescopes", in *IEEE Antenna's and Propagation Magazine*, August 2002, vol. 44.

[12] M. Davis and T. Ghosh et. al., "RFI Assessment and Excision at Arecibo", in *Proceedings of the URSI 26th General Assembly*, August 1999, Toronto.

[13] R. Weber, B. Lamarque, and R. Canals, "Two Examples of Real-time RFI Detectors for Time-blanking", in *Proceedings of the URSI 26th General Assembly*, August 1999, Toronto.

[14] S. W. Ellingson, J. D. Bunton, and J. F. Bell, "Removal of the GLONASS C/A Signal from OH Spectral Line Observations using a Parametric Modelling Technique", in *The Astrophysical Journal Supplement Series*, July 2001, number 135, pp. 87–93.

[15] G. C. Bower, "A Radio Frequency Interference Mitigation Strategy for the Allen Telescope Array", in *URSI General Assembly XXVII*, August 2002.

[16] D. A. Mitchell and G. C. Bower, "Cancelling Satellite Interference at the Rapid Prototyping Array. A Comparison of Voltage and Power Domain Techniques", in *ATA Memo #36*, August 2001.

[17] F. Briggs and M. Kesteven, "RFI Subtraction with a Reference Horn: Application to Pulsars and VLBI", in *URSI General Assembly XXVII*. Nat. Academies of Sciences and Engineering, August 2002, in review.

[18] S. W. Elingson, "RFI Suppression for Radio Astronomy: Frequency-, Time-, Space-, and Multidomain Approaches", in *Proceedings of the URSI 26th General Assembly*, August 1999, Toronto.

[19] G. Swarup, "Characterization and Suppression of RFI to the Giant Metrewave Radio Telescope", in *Proceedings of the URSI 26th General Assembly*, August 1999, Toronto.

[20] J. F. Bell et. al., "Adaptive Interference Mitigation Strategies for SKA", in *Proceedings of the URSI 26th General Assembly*, August 1999, Toronto.

[21] L. Li, B. D. Jeffs, A. J. Poulsen, and K. Warnick, "Analysis of Adaptive Array Algorithm Performance for Satellite Interference Cancellation in Radio Astronomy", in *URSI General Assembly XXVII*. Nat. Academies of Sciences and Engineering, August 2002, Maastricht the Netherlands.

[22] B. Jeffs, K. Warnick, and L. Li, "Improved Interference Cancellation in Synthesis Array Radio Imaging Using Auxiliary Antennas", in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 2003, Hong Kong.

[23] B. Jeffs, L. Li, and K. Warnick, "Auxiliary Antenna Assisted Interference Cancellation for Radio Astronomy Imaging Arrays", in *IEEE Transactions on Signal Processing*, February 2003, in review.

[24] C. Hansen, K. Warnick, and B. Jeffs, "Adaptive Interference Cancellation Using an Array Feed Design for Radio Telescopes", in *Proceedings of the IEEE APS-URSI Symposium*, June 2003, Columbus, Ohio.

[25] A. J. Poulsen, B. Jeffs, K. Warnick, and R. Fisher, "Real-time Adaptive Cancellation of GLONASS Interference in OH Signal Observations at the Green Bank Telescope", in *Proceedings of the IEEE APS-URSI Symposium*, June 2003, Columbus, Ohio.

[26] B. Widrow and M. E. Hoff Jr., "Adaptive Switching Circuits", in *IRE WESCON Convention Record*, 1960, vol. 4, pp. 96–104.

[27] "Small Radio Telescope (SRT)", http://web.haystack.mit.edu/SRT/, January 2003.

[28] "Northern Lights Software Associates", http://www.nlsa.com/, May 2003.

[29] B. T. Walkenhorst, "Development of a Radio Telescope Receiver for Research in Radio Frequency Interference Mitigation", Master's thesis, Brigham Young University, August 2003.

[30] "Pentek", http://www.pentek.com, May 2003.

[31] M. S. Bartlett, "Smoothing Periodograms from Time Series with Continuous Spectra", in *Nature*, May 1948, vol. 161, pp. 686–687.

[32] P. D. Welch, "The use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging over Short Modified Periodograms", in *IEEE Trans. Audio and Electroacoust.*, June 1967, vol. AU-15, pp. 70–73.

[33] R. B. Blackman and J. W. Tukey, "The Measurement of Power Spectra", 1958, vol. AU-15, Dover, New York.

[34] J. Capon, "High-resolution Frequency-Wavenumber Spectrum Analysis", in *Proceedings IEEE*, August 1969, vol. 57, pp. 1408–1418.

[35] J. P. Burg, "Maximum Entropy Spectral Analysis", in *Proceedings IEEE*, August 1969, vol. 57, pp. 1408–1418.

[36] R. Schmidt, "Multiple Emitter Location and Signal Parameter Estimation", in *Proceedings of the RADC Spectrum Estimation Workshop*, 1979, pp. 243–258.

[37] M. H. Hayes, *Statistical Digital Signal Processing and Modelling*, John Wiley & Sons, Inc., 1996.

[38] "Graychip, Texas Instruments, Digital Radio Chips for Wireless Infrastructure", http://www.ti.com/graychip/index.shtml, July 2003.

[39] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*, Prentice Hall, 2000.

[40] H. Stark and J. W. Woods, *Probability and Random Processes with Applications to Signal Processing*, Prentice Hall, third edition, 2002.

[41] B. D. Van Veen and K. M. Buckley, "Beamforming: A Versatile Approach to Spatial Filtering", in *IEEE ASSP Magazine*, April 1988.

[42] H. L. Van Trees, *Optimum Array Processing*, vol. IV of *Detection, Estimation, and Modulation Theory*, John Wiley & Sons, Inc., 2002.

[43] "The GBT User Manual", http://www.gb.nrao.edu/gbt/GBTMANUAL/, March 2003.

[44] F. Ghigo, "VLBI on the GBT", http://www.gb.nrao.edu/ fghigo/gbtdoc/ vlbinfo.html, February 2003.

[45] L. Bogan, "Calculator for Distances between Geographical Locations", http://www.go.ednet.ns.ca/ larry/bsc/jslatlng.html, January 2003.

[46] "GBT Design", http://www.gb.nrao.edu/GBT/technicalterms.html, February 2003.

[47] "Creative Services Software", http://www.cssincorp.com/, July 2003.

[48] "Texas Instruments–Real World Signal Processing", http://www.ti.com/, July 2003.

[49] "The VizieR Catalogue Service", http://vizier.u-strasbg.fr/, March 2003.

[50] P. te Lintel Hekkert, H. A. Versteege-Hensel, H. J. Habing, and M. Wiertz, "A Catalogue of Stellar 1612 MHz Maser Sources", in *Astronomy & Astrophysics Supplement Series*, June 1989, 78, pp. 399–430.

[51] P. te Lintel Hekkert, J. L. Caswell, H. J. Habing, R. F. Haynes, and R. P. Norris, "1612 MHz OH Survey of IRAS Point Sources", in *Astronomy & Astrophysics Supplement Series*, October 1991, 90, pp. 327–353.

[52] M. N. Sevenster, J. M. Chapman, H. J. Habing, N. E. B. Killeen, and M. Lindqvist, "The ATCA/VLA OH 1612 MHz Survey. I. Observations of the Galactic Bulge Region.", in *Astronomy & Astrophysics Supplement Series*, April 1997, 122, pp. 79–93.

[53] M. N. Sevenster, H. J. van Langevelde, R. A. Moody, J. M. Chapman, H. J. Habing, and N. E. B. Killeen, "The ATCA/VLA OH 1612 MHz Survey. III. Observations of the Northern Galactic Plane.", in *Astronomy & Astrophysics Supplement Series*, February 2001, 366, pp. 481–489.