MULTI-CHANNEL, FREQUENCY-AGNOSTIC, PORTABLE RECEIVER DESIGN FOR MAGNETIC RESONANCE IMAGING AND SPECTROSCOPY

A Dissertation

by

EDWIN PARKER EIGENBRODT

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
In partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Mary P. McDougall
Committee Members,	Christian Hilty
	Jim Ji
	Robert Nevels
Head of Department,	Miroslav M. Begovic

May 2016

Major Subject: Electrical Engineering

ABSTRACT

Despite great potential, non-^{1}H magnetic resonance imaging and spectroscopy (MRI/MRS) studies have not been adopted into standard clinical use. This is largely because the signals generated by non-^{1}H nuclei have limited signal-to-noise ratio (SNR) due to the nuclei's lower Larmor frequency and lower relative abundance. Exploiting the increased SNR provided by array coils is a natural direction to turn in addressing this; however, it is highly unusual for scanners to be equipped with multi-channel, multi-nuclear receivers due to cost and complexity. This leads to a "chicken or the egg" conundrum where scanners are not equipped with second-nuclei receivers because of the lack of any current widespread clinical adoption of second-nuclei studies, but studies are rare because there are not readily available receivers. The application of frequency domain multiplexing (FDM) to MRI has been investigated as a low cost alternative to expensive multi-channel receivers, and has been applied to non-^{1}H nuclei.

This dissertation describes the work done on a six channel, inexpensive, frequency domain multiplexed receiver, agnostic to the nuclei of interest or magnetic field strength, and implemented using off-the-shelf products. The receiver is designed to be portable and easily used in conjunction with any system with two programmable trigger lines. In addition, the architecture is straightforwardly scalable to 16 channels at an additional cost of approximately $1300 per channel. This work describes the receiver architecture and compares its performance to a commercial Varian Inova system. The flexibility and portability of the receiver are demonstrated by application to multiple channel imaging and spectroscopy of various nuclei at different field strengths, and on different scanners in different locations.

ACKNOWLEDGEMENTS

I would like to acknowledge my advisor, Dr. Mary McDougall for supporting me and believing in me through this long process. Your endless patience was underserved and greatly appreciated. Thank you for instilling in me a love and knowledge of MRI. I would also like to thank my other committee members, Dr. Christian Hilty, and Dr. Jim Ji, and Dr. Robert Nevels for their time and support.

I would like to acknowledge Dr. Wright for his help over the years in so many ways.

I would like to acknowledge my parents for helping instill in me a love of science at an early age. Thank you for never doubting my ability, always pushing me and always loving and supporting me, even when I was too stubborn appreciate it.

I would like to acknowledge all of my lab mates, for the countless hours of troubleshooting and conversation. Without you guys, I would have gone crazy and this system would not be working.

Finally, and most importantly, I would like to acknowledge my lovely wife, Julia. Your unwavering faith in my potential pushed me to do my best. Without your love, advice, and support, none of this would have been possible.

NOMENCLATURE

^{1}H	Hydrogen atom, often dubbed “proton”
^{7}Li	Lithium-7 isotope
^{13}C	Carbon-13 isotope
^{31}P	Phosphorous-31 isotope
B_0	Static magnetic flux density
B_1	RF magnetic flux density
FDM	Frequency domain multiplexing
MRI	Magnetic resonance imaging
NMR	Nuclear magnetic resonance
RF	Radio Frequency

TABLE OF CONTENTS

Page

ABSTRACT ii

ACKNOWLEDGEMENTS iii

NOMENCLATURE iv

TABLE OF CONTENTS v

LIST OF FIGURES vii

LIST OF TABLES ix

1 INTRODUCTION AND OBJECTIVE 1

1.1 Background: Multi-nuclear Spectroscopy and Limitations 2
1.2 Background: RF Array Coils in MRS 6
1.3 Background: Receiver Metrics 7
1.4 Design Parameters 9

2 RECEIVER HARDWARE 11

2.1 Limiter 11
2.2 Switch 12
2.3 Low Noise Amplifier 14
2.4 Variable Attenuator 15
2.5 Image Reject Filter 16
2.6 Mixer 17
2.7 Second Stage Amplifier 17
2.8 Channel Selection Bandpass Filter 18
2.9 Power Combiner 20
2.10 Digitizer 20
2.11 LO Generation/Amplification 20
2.12 Digitizer Computer 22
2.13 Triggering Circuit 24
2.14 Digitizer Cooling Solution 25
2.15 Digitizer Power Requirements 26
2.16 Fully Assembled Receiver 27
2.17 Cost 29

3 SOFTWARE 31

3.1 Digitizer Control Software 31
3.2 Post Processing Images 34
3.3 Post Processing Spectra 38

4 EXPERIMENTAL METHODS 40

4.1 Six-Channel Hydrogen 40
4.2 Two-Channel Carbon 43
4.3 Low-field Desktop Magnet 45

5 RESULTS 48

5.1 Six-Channel ^{1}H Array 48
5.2 Two-Channel ^{13}C Array 51
5.3 Lab Class ^{1}H Coil 52

6 CONCLUSIONS AND FUTURE WORK 53

6.1 SNR of Six-Channel Hydrogen Data 53
6.2 SNR Data from Two-Channel Carbon Data 54
6.3 Consistency of Channel to Channel SNR 55
6.4 SNR Comparison between FDM Receiver and Varian Inova 56

REFERENCES 58

APPENDIX A 61

Preparations for Use 61
The Desktop Control of the Digitizer 73
Compiling the Driver 78

APPENDIX B 81

Digitizer Control Software 81
Image Processing Code 91
Carbon Spectra Processing Code 93
MATLAB Function ‘findphase’ 95
MATLAB Function ‘correctphase’ 95

LIST OF FIGURES

Page

Figure 1: Full receiver diagram 11

Figure 2: a) An acquisition taken without using the switch, showing the saturation of the receiver during transmit. b) When the switch is used to attenuate the transmit signal, the digitizer is not saturated. 13

Figure 3: The LO generation and triggering box 21

Figure 4: a) the 500 MHz clock that drives the LO board. b) The LO generation board. c) The LO amplifiers. d) The triggering board for the digitizer. 21

Figure 5: The digitizer computer 23

Figure 6: a) The PCB layout for the triggering board. b) The circuit diagram of the triggering board. The red box denotes input and the orange box denotes output. 25

Figure 7: The triggering board. 25

Figure 8: a) The cooling solution without digitizer card installed. b) The cooling solution with digitizer card installed. c) The digitizer card being plugged into the PCI-e bus extender. 26

Figure 9: The fully assembled receiver on its cart. 27

Figure 10: a) The power combiners. b) The power supplies for the system. c) The LO generation and triggering board box. d) The boxes containing the amplifiers, mixers, and attenuation. e) The power connectors. f) The USB cables for controlling the LO board. 28

Figure 11: Reconstructed images with phase correction (right image) and without phase correction (left image) 38

Figure 12: The surface coil array inside of the birdcage volume coil on the loading system 41

Figure 13: a) The housing for the surface coil array. b) The phantom. c) The transmit birdcage coil. 41

Figure 14: The pulse sequence used when acquiring data from the six channel mouse array. Trigger 1 is used for triggering the digitizer and trigger 2 is used for turning off the switch at the input to the receiver. 42

Figure 15: a) The transmit Helmholtz coil. b) The geometrically decoupled surface coils. The olive oil phantom can be seen beneath the surface coils. 44

Figure 16: Carbon spectra acquisition pulse sequence 45

Figure 17: The modular receiver set up for imaging in the on campus lab 46

Figure 18: The pulse sequence used while receiving data from the lab class system 47

Figure 19: Images taken from each coil by FDM receiver and by Varian Inova receiver 48

Figure 20: On the left is the original channel 4 image. On the right is the channel 4 image with signal area denoted by yellow and noise area denoted by green 49

Figure 21: A graph showing what percentage of the SNR of the Inova receiver the FDM was able to achieve on each surface coil 50

Figure 22: The ^{13}C spectra received by both the Varian Inova receiver and the FDM receiver 51

Figure 23: Image of a test tube acquired using the lab class system 52

LIST OF TABLES

Page

Table 1: Filter properties 20

Table 2: Total cost of FDM receiver 29

Table 3: Cost of additional Channels 30

Table 4: SNR values from each coil 50

Table 5: SNR values for ^{13}C Spectra 51

Table 6: The percent of the SNR of the Inova receiver achieved by the FDM receiver in the hydrogen experiment 54

Table 7: The percent of the SNR of the Inova receiver achieved by the FDM receiver in the carbon experiment 55

Table 8: Noise factor vs device order 56

1 INTRODUCTION AND OBJECTIVE

The benefits of using array coils to increase the signal to noise ratio (SNR) in magnetic resonance imaging (MRI) are well established [1, 2], with the improvement in SNR typically exploited to increase the resolution or decrease the scan time of the imaging experiment. The benefits to in-vivo magnetic resonance spectroscopy (MRS) are analogous, with the increase in sensitivity perhaps even more meaningful for second-nuclei (nuclei other than hydrogen) [3-7]. Despite this, multi-channel multi-nuclear spectroscopy is far less explored than multichannel ^{1}H imaging. This is at least partially due to the fact that multiple channel receivers are not commercially available for second-nuclei frequencies. This leads to a "chicken or the egg" conundrum where scanners are not equipped with second-nuclei receivers because of the lack of any current widespread clinical adoption of second-nuclei studies, but studies are rare because there are not readily available receivers. The typical model for making new technology commercially available in the MR field begins with investigations using low-cost solutions generated from off-the-shelf components. This doctoral work investigates a low-cost multi-channel receiver built from off-the-shelf components for second-nuclei spectroscopy applications.

In 1946, Bloch and Purcell demonstrated the Nuclear Magnetic Resonance phenomenon in liquids and solids by showing that liquids and solids in a static magnetic field absorb and then re-emit RF energy at a specific frequency that is dependent on the static magnetic field strength [8-10]. That frequency is known as the Larmor frequency and is defined in Equation 1, where ω is the Lamor frequency in angular frequency, γ is the gyromagnetic ratio, and B_0 is the static magnetic field strength.

The phenomenon observed by Bloch and Purcell is caused when nuclei with a spin number of an odd multiple of ½ are put into a static magnetic field and the nuclei align either with or against the field. The spins that are aligned with the field are in a low energy state and the spins that are aligned against the field are in a high energy state. The low energy state has a slightly larger population than the high energy state.

$$\omega = \gamma * B_0 \qquad (1)$$

When energy at the Larmor frequency is absorbed by the system, spins are pushed from the lower to the higher energy state. When the spins fall back to the lower energy state, RF energy is re-released at the Larmor frequency. The frequency at which spins absorb energy is dependent on their local magnetic environment. For instance, if there are two different molecules that both contain hydrogen and they are placed in the same magnetic field, the hydrogens may resonate at different frequencies depending on the shielding caused by their respective molecular environments. This allows for NMR spectra to provide information about the molecular structure containing the nuclei of interest in vivo. NMR spectroscopy can give physicians and scientists the ability to take a non-invasive look at the molecular composition of tissue and take a picture of ongoing molecular processes. This has already had an impact in understanding of physiology and is being used to help provide biomarkers for disease.

1.1 Background: Multi-nuclear Spectroscopy and Limitations

Non-proton magnetic resonance spectroscopy is a useful tool in many clinical situations ranging from detection of metabolic processes and cancer diagnosis [11-13] to the study of mental illnesses [14-17]. Nuclei in the body with a spin number greater than 0, such as ^{1}H, ^{13}C, ^{31}P, and ^{7}Li, can be imaged with nuclear magnetic resonance. However, second-nuclei

exist in smaller concentrations and have lower Larmor frequencies and hence provide lower signal than ^{1}H nuclei.

Of specific relevance to this proposal, ^{13}C is a good example of a low concentration element with interesting clinical significance. Adipose tissue can be analyzed with magnetic resonance spectroscopy of ^{13}C and this has been used to show that fat composition *in vivo* can be used as a strong indicator of the diet of the individual [18]. MRS experiments determining the saturated vs. unsaturated fat content of fatty acids have also been used in other studies looking for correlations to type-2 diabetes [19, 20]. ^{13}C can also exist in sugars and has been used to measure the metabolism of tissue. The variation in metabolism of tissue can help determine the aggressiveness and location of cancer in mice [21, 22]. In the referenced study, the mice were injected with ^{13}C enriched pyruvate and the increased uptake and metabolism of that pyruvate was a good indicator of cancerous tissue and its activity.

Lithium is a main component in pharmaceuticals used to treat bipolar disorder as well as other affective disorders. In-vivo NMR spectroscopy of lithium provides the ability to track the location and amount of lithium in the brain [16, 17]. This can help elucidate how lithium actually works in alleviating mental health issues as well as gauging the effectiveness of the delivery of the drugs. In a study performed by Gyulai et al [16], ^{1}H, phosphorous 31 (^{31}P), and lithium 7 (^{7}Li) spectra were taken in the head and in the calf. They used two different sets of coils. One set could transmit and receive ^{1}H and ^{7}Li and the other could transmit and receive ^{1}H and ^{31}P. The ^{31}P spectra were acquired to determine how much signal was collected from the brain and how much signal was collected from surrounding muscle.

Phosphorous 31 is not only useful for measuring muscle content but can also be used to look at tumor metabolism [11], to measure brain lipid content [23], or to evaluate liver

metabolism[7, 11]. Physicians can glean useful information about a variety of cardiac diseases such as coronary artery disease [24] and dilated cardiomyopathy [25, 26] using ^{31}P spectroscopy.

The above examples are just a few of many uses of in-vivo NMR spectroscopy of non-^{1}H nuclei. In-vivo NMR spectroscopy provides huge potential for breakthroughs in understanding of disease, disease prevention via screening, improved diagnosis, and tracking of disease during treatment. Despite the potential clinical significance of ^{13}C and other non-hydrogen nuclei, it remains extremely challenging to interrogate, particularly *in vivo*. Equation 2, below, describes the SNR of an MRI experiment. In Equation 2, ω is the Larmor frequency of a nuclei in a given magnetic field, ΔV is the voxel size, M_{xy} is the magnetization in the transverse plane, B_{1t} is the field from the RF coil, T_{acq} is the acquisition time, k is the Boltzmann constant, T is the temperature, Δf is the bandwidth of the receiver, R is the coil resistance, B_0 is the main magnetic field strength, and γ is the gyromagnetic ratio.

$$SNR \propto \frac{\gamma B_0 \Delta V M_{xy} |B_{1t}| \sqrt{T_{acq}}}{\sqrt{4kT\Delta f R}} \qquad (2)$$

The sensitivity in a magnetic resonance experiment is proportional to both the abundance and to the cubed gyromagnetic ratio (γ) of the nuclei, taking into account a γ^2 term contained in the transverse magnetization [27]. The noise also has a linear dependence on γ and so the SNR increases as γ^2 [28]. The greatest limitation in the push towards second-nuclei spectroscopy lies in the lack of MR sensitivity to these nuclei. The isotopic abundance of ^{13}C is 1.1% compared to 99.8% for ^{1}H. Also, the Larmor frequency of ^{13}C is ¼ that of ^{1}H. These two effects lead to the absolute sensitivity of ^{13}C being .000175 that of ^{1}H in an MR experiment if there were equal numbers of each element. However, 63% of the atoms in the body are

hydrogen and only 12% are carbon. This further reduces the sensitivity of ^{13}C *in vivo* to 3.3e-5 compared to ^{1}H.

A few ways to increase ^{13}C sensitivity in a spectroscopy experiment are using ^{1}H decoupling; introducing molecules into the sample that have been artificially enriched with ^{13}C (such as glucose); imaging at a higher field strength; or receiving using a more sensitive radio frequency (RF) coil.

If it is desired to watch the metabolism of a certain sugar, the subject can be injected with a sugar artificially enriched with hyperpolarized ^{13}C, which does not change the chemical properties of the sugar. Limitations of this procedure are that it requires an injection into the patient and the hyperpolarization of the ^{13}C lasts only a short time. The useable increase in signal achieved by current hyperpolarization techniques persists on the order of several minutes. However, during the lifespan of the hyperpolarized signal, the signal sensitivity can be increased by a factor as much as 10^4.

In a higher field strength, the signal is greater since the signal sensitivity in an MR experiment is proportional to the main magnetic field strength (B_0), as given in Eq. 1. The higher field strength gives other advantages aside from increasing the signal such as spreading out the peaks in the NMR spectrum. The disadvantages are cost and availability of these high field strength magnets.

Finally, it can be seen in Equation 1 that SNR is proportional to B_{1t}, the field produced by the RF coil. Surface coils allow for a higher B_1 compared to volume coils, and hence higher SNR, at the cost of field of view (FOV). This limited FOV can be ameliorated by using an array of surface coils to provide a larger FOV without losing the SNR advantage [1, 3, 29].

The disadvantage of using an array of surface coils is the cost of the receiver to be able to record signals from all of them simultaneously. This dissertation addresses the potential to enable increasing the sensitivity of second-nuclei NMR detection via RF array coils by providing a blueprint for a low-cost, multi-channel second nuclei receiver.

1.2 Background: RF Array Coils in MRS

Since the idea of phased arrays was introduced to the MR world, it has been demonstrated that RF coil arrays give an SNR advantage in ^{1}H MRI. In 1990, Hayes and Roemer demonstrated the SNR benefit of using four surface coils rather than two surface coils or a single body coil as the receive coil. The four surface coils returned a signal with 1.38 times higher SNR than the two surface coils and with 1.8 times higher SNR than the body coil when measuring at the center of the phantom [2]. Wright et al. compared the SNR in an MR experiment using a volume coil to using an array of surface coils the same size as the volume coil, assuming sample loss dominance. Wright both calculated and demonstrated that the array never suffered lower SNR than the volume coil and that the array also generated higher SNR near the surface coils [3]. The benefits of adding array channels to ^{1}H MRI have led to the use of 32 channels in state-of-the-art clinical work and up to 128 channels in research [30, 31]. Similar advantages can be expected when increasing the channel count for second-nuclei arrays.

The addition of array channels for second-nuclei spectroscopy is already beginning to exhibit the same advantages seen with the addition of ^{1}H channels. Even relatively small arrays have shown marked SNR improvement over volume coils in a ^{31}P MRS application [5, 32]. In the ^{31}P spectroscopy experiment by Avdievich, a threefold increase in SNR was measured near the array coils compared to the same area acquired by the volume coil. Near the center

of the phantom, there was no loss of SNR when compared to the volume coil. Arrays as large as eight channels have now been created for ^{31}P spectroscopy [7]. It is anticipated that this SNR advantage is eventually going to be essential to acquiring *in vivo* spectra with reasonable SNR from isotopes such as ^{13}C.

While adding receiver channels for second-nuclei seems like a worthwhile endeavor, the expense and complexity is a strong counterargument when the benefits are largely unexplored. One possible solution to alleviate the expense of adding new channels is to use multiplexing in order to take greater advantage of the equipment already in-house. This allows for multiple new channels to be added without adding costly digitizers. Time domain multiplexing has already been successfully used to acquire MRS data [4]. However, time domain multiplexing can be difficult to synchronize correctly and can have bandwidth limitations in certain circumstances. Frequency domain multiplexing (FDM) has been demonstrated with promising results, both in research arena and in the commercial world [33-35].

The basic idea behind FDM is to combine the output of multiple coils onto a single coaxial cable and then sample the combined signals at the same time with a single digitizer. This technique as applied to NMR will be described further below.

1.3 Background: Receiver Metrics

After the coil picks up signal from the nuclei of interest, it must be somehow recorded before it can be used to produce a spectrum or image. The signal coming off of the coil is always very small (around the order of -50 dBm), but can vary dramatically depending on the situation. There are several metrics to describe receivers. The noise figure and the dynamic range are two metrics that describe a receiver. A measure of the quality of the signal is the

signal to noise ratio (SNR). The components are chosen in order to create a receiver with a good noise figure and a dynamic range that meets the requirements of the application.

The SNR is a metric of signal quality measuring the ratio of how much signal there is vs how much noise. For images, the SNR can be measured by taking the average of a signal area divided by the average of a noise area. Alternatively, the standard deviation of the noise can be used in place of the average of the noise. When calculating the SNR of a spectrum rather than an image, the area under a peak of interest is used for the signal.

The noise figure is a measure of how much a device degrades the SNR. The equation for noise factor is:

$$F = \frac{SNR_{in}}{SNR_{out}} \tag{3}$$

The noise figure is given by:

$$NF = 10\,log_{10}(F) \tag{4}$$

The noise factor will always be greater than 1 since the SNR going into a device is always greater than the SNR of the signal output by the device. The noise factor of a collection of devices, such as a receiver, can be calculated using Friis Forumla:

$$F = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 * G2} + \frac{F_4 - 1}{G_1 * G_2 * G_3} + \cdots + \frac{F_n - 1}{G_1 * G_2 * G_3 * \ldots * G_{n-1}} \tag{5}$$

In the above equation, F_n and G_n are the noise factor and the gain, respectively of the nth device in the chain.

Dynamic range is the measure of the smallest signal that the receiver can detect vs the largest. For instance, a receiver that can detect a 1 mV minimum and 1 V maximum signal has a dynamic range of 20*log(1/.001)=60 dB. Another example is a 12 bit digitizer with a

maximum input voltage of 1 V can detect signals down to 1 V/4096 (ideally). This means that the dynamic range is 20*log(4096)=72.2 dB.

1.4 Design Parameters

The receiver should be:

-Frequency agnostic across the nuclei and field strengths of interest: Many MRI receivers do not have the capability to receive more than one channel of non-^{1}H signal. The main goal and focus of this receiver is to make acquiring six channels of non-^{1}H signals straightforward.

-Able to accept signals with a wide range of amplitudes: Because of the design parameter to be able to acquire many different frequencies, it also becomes important to be able to acquire signals with a wide range of amplitudes. Using a hydrogen surface coil with a preamplifier on the coil will produce a vastly different signal amplitude from an unamplified carbon coil. Both signals should be within the range of the receiver's ability.

-Mobile: The receiver will be designed with the ability to work on many different magnets and interface with different systems. Mobility of the receiver is key to have the ability to move the receiver to the many systems with which the receiver is compatible. The receiver should be of a form factor that allows transport in a personal vehicle and safely loaded or removed by two people.

-Inexpensive: One of the biggest hurdles to overcome in acquiring multi-channel second-nuclei receivers is the cost.

-Modular: Modularity allows for ease of increasing the number of channels or upgrading the system in the future given future advances in technology.

This dissertation will be organized into six sections. Section 2 provides a description of the hardware that was built and tested. Section 3 outlines the software written to run the receiver and post process the acquired data. Section 4 describes the experiments that were conducted. Section 5 displays the results from the experiments. Section 6 describes future work and presents conclusions.

2 RECEIVER HARDWARE

A frequency domain multiplexed receiver was built, utilizing off the shelf components. The IF frequencies used for multiplexing are 500 KHz, 1 MHz, and 5 MHz. There are two channels at each IF frequency, totaling six channels. Figure 1 is a diagram of the full FDM receiver.

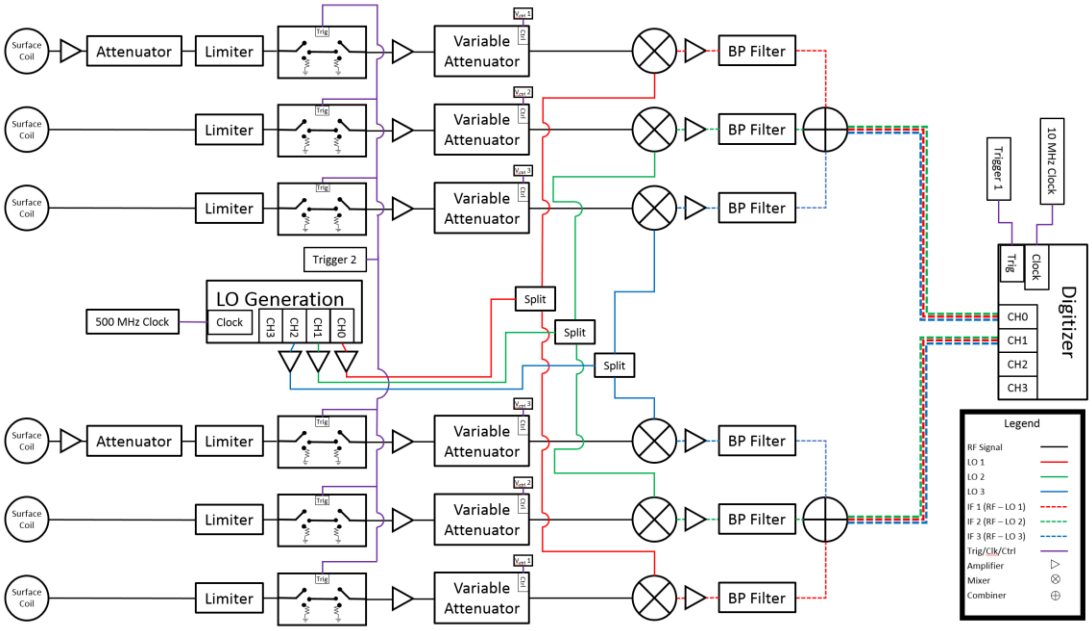


Figure 1: Full receiver diagram

2.1 Limiter

An RF limiter prevents any signals larger than a given power level from passing through it without attenuation. In this case, the limiter only lets 10 dBm signals or smaller pass without attenuation. When signals less than 10 dBm are passing through the limiter, the signal to noise ratio of the images recorded are not affected by its presence. While the noise factor should be somewhat affected by the presence of the limiter (~10% increase in noise factor),

the SNR of images acquired with and without the limiter did not change by a measureable amount.

The low noise amplifier has protection at its input and there is also a switch that provides some level of protection during the transmit pulse. This may make the RF limiter seem like an unnecessary level of protection, however given that it does not greatly affect the SNR and it is a passive device that will operate regardless of trigger timing, it is an added level of protection with minimal added cost and little to no degradation of image quality.

The signal level coming in during the acquisition period of the pulse sequence will never even come close to 10 dBm, usually staying much closer to -50 dBm to -60 dBm. A limiter with a lower power limit would have been a better fit and could have possibly eliminated the need for the switch as the next stage. The limiter that was chosen is the lowest level limiter that is sold by Mini-circuits, the VLM-52, and has one of the lowest thresholds that could be found from any commercial company. Because the limiter only caps the maximum signal level at 10 dBm, a switch is needed as the next stage to reduce the dynamic range of the input signal.

2.2 Switch

Because the transmit signal is recorded as well as the echo/FID, the dynamic range of the recorded signal is increased dramatically. The transmit signal that bleeds through into the receiver is much larger than the echo/FID signal and so the total recorded signal has a much larger maximum amplitude than if the echo/FID was the only recorded signal. The dynamic range of the signal must be reduced in some fashion and a large reduction in the transmit signal is a good way to accomplish this. A passive device that could attenuate any signals over a certain level seems like an ideal solution (see the above limiter), but the power levels of the

signal would be capped at too large of a level. The switch is a triggered device that has around -40 dB of coupling between the common port and the non-connected port. When the system is transmitting, the switch common port is connected to a 50 ohm load instead of the rest of the receiver. The transmit signal is large enough that the signal is still picked up by the receiver, albeit at a much lower level. This allows the acquired echo/FID to be phase corrected using the transmit signal while not increasing the dynamic range of the acquired signal. The effect of the switch on the acquired signal is shown in Figure 2.

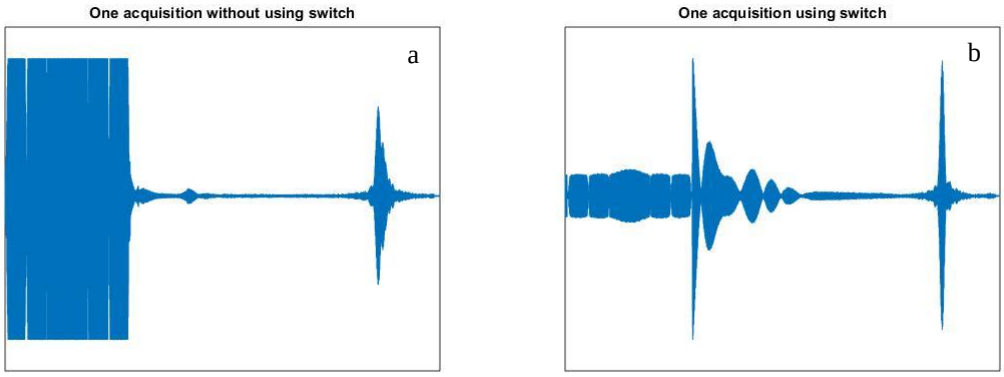


Figure 2: a) An acquisition taken without using the switch, showing the saturation of the receiver during transmit. b) When the switch is used to attenuate the transmit signal, the digitizer is not saturated.

Several different switches were tested. Three switches from Mini-circuits were tested and several switches were developed and built in the lab. The switches built in the lab were based around active PIN diode circuits. The PIN diode circuits added a significant amount of noise onto the RF line that was not present with the Mini-Circuits switches and so were not used in the final design.

If there is ever a case where the transmit signal is significantly reduced from what is expected, the switch would reduce that even further, making it difficult to use for phase

correction of the echo/FID. In that case, the switch can be set to always let all of the signal through and the transmit signal will then be large enough to use for phase correction.

While the switch reduces the dynamic range of the acquired signal, it also degrades the noise factor by quite a bit from 1.55 to 2. As was shown in Equation 5, the noise factor is greatly affected by the first stage of the receiver. If the low noise amplifier is the first stage, then its low noise figure and high gain dominate the rest of the stages. However, if the switch is put first then the noise figure is degraded before the gain of the amplifier can be the dominant factor.

2.3 Low Noise Amplifier

In a receiver, it is important to have a high quality low noise amplifier. The low noise amplifier can be subjected to a wide range of conditions and basically sets the noise figure for the rest of the receive chain. A Miteq AU-1647 was chosen to be the low noise preamplifier for the receiver for several reasons. It has a very wide usable frequency range from .1 to 400 MHz. This frequency range encompasses all of the nuclei from 1.5 T to 7 T. It is a very durable amplifier. It has passive input protection and reverse voltage protection. It also has a very high gain and extremely low noise figure. Those two aspects of the amplifier make sure that the noise figure for the rest of the receiver chain stays low. When testing a Mini-circuits amplifier with a slightly higher noise figure (3 dB noise figure compared to 1.4 dB) in the first stage, the SNR was reduced to 80% of the SNR when using the Miteq.

The Miteq amplifier also has fast recovery from large signals, which is good when acquiring spectroscopy data. In a ^{13}C spectroscopy acquisition, the received FID is very small and the transmit signal is quite large. Also, unlike a spin-echo, the desired signal (the FID) immediately follows the transmit signal. These two factors combined mean that if the amplifier

saturates under the transmit signal and doesn't have a very fast recovery time, the FID will be distorted or lost.

Finally, the Miteq amplifier has a high 1db compression point at 12dBm output power. If the compression point is much lower, then the compression point will be a limiting factor when recording larger signals and those larger signals will be distorted. Having a high available output power allows for a greater range of signals to be acquired without distortion from compression.

The Miteq amplifier is the single most expensive component used in every channel of the receiver at $400. However, all of the preceding factors make it the best choice in this situation, even when looking for a cost effective solution.

2.4 Variable Attenuator

The variable attenuator is extremely important because of the huge range of signals the receiver was designed to digitize. When receiving from a hydrogen surface coil with a preamplifier on the coil there is much more signal input to the system than from a non-amplified carbon coil. The attenuator that was chosen is the ZX73-2500+, a voltage controlled attenuator from Mini-Circuits. The control voltage for the ZX73-2500+ is adjusted via a bulkhead mounted potentiometer that can vary the control voltage from 0-12 V. When the control voltage is 0 V, the ZX73-2500+ is specified to attenuate by 40 dB and at 12 V the ZX73-2500+ is specified to attenuate by 3 dB. When testing on the bench, the bench, the ZX73-2500+ does in fact attenuate by 3 dB at the 12V control voltage. However, when the control voltage is changed to 0 V, the ZX73-2500+ attenuates by 60 dB instead of 40 dB. This range of attenuation is more than sufficient for the applications the receiver was built for.

There is one attenuator for each receiver channel and each must be given a control voltage. Individually having to adjust the attenuation for each channel would be a great inconvenience and so the control voltages were grouped into IF channels. There is one control voltage for the 5 MHz IF, one for the 1 MHz IF and one for the 500 kHz IF. While there may be vastly different signals being recorded on each of the different IF frequencies, it was assumed that a similar amplitude signal would be received by channels with the same IF frequency. For instance, while the 1 MHz channels might be receiving a very small carbon signal and the 500 kHz channels might be receiving a larger hydrogen signal, it is not possible to have both carbon and hydrogen on different 1 MHz channels. Since each IF frequency is reserved for only one nuclei, it is not a bad assumption that the signals being received on different channels at the same IF frequency will be of similar amplitude. Each set of channels at the same IF frequency can have their attenuation adjusted independently of the channels at different IF frequencies.

2.5 Image Reject Filter

No image reject filter was included in this design in order to make the receiver frequency agnostic. When mixing an RF signal against a local oscillator, there are always two RF frequencies that both mix down to the desired IF. The image reject filter allows through the desired RF frequency while blocking the "image" RF frequency. Using an image reject filter can help prevent spurious signals from mixing into the signal of interest as well as noise. However, using a fixed image reject filter fixes the input frequency of the receiver. There are adjustable band pass filters (such as cavity filters) that have useful properties such as low insertion loss and very high Q. However, these filters are very large, very expensive and require bench measurements to retune to a different frequency. It would be very inconvenient

if, for every change in nuclei, the bandpass filters had to be disconnected and tuned on the bench before being reconnected. In its current state, the receiver can be moved from receiving one nuclei to another simply by changing the LO frequency on a computer.

One possibility for a tunable filter is a MEMs (micro-electromechanical) filter. These filters are tunable and show great promise, but are not commercially available yet at these frequencies [36]. In the future, computer controlled MEMs filters may see enough development to be inexpensive and work in these frequencies of interest. If that becomes the case, the inclusion of an image reject filter would not be such an inconvenience in a similar receiver design.

2.6 Mixer

The mixer chosen is a Mini-Circuits ZX05-1L+. Mini-Circuits specify their mixers by LO drive level and frequency range. Since the RF level into the mixer is very low (~ -20 dBm) and it is easier to drive a lower power LO signal, the lowest level Mini-Circuits mixer was chosen. The mixer requires an LO drive of 3 dBm and Mini-Circuits specifies that the LO level should be about 10 dB above the RF level. Both of these requirements are met in this application.

2.7 Second Stage Amplifier

The second stage amplifier is a Mini-Circuits ZFL-500LN+. It is specified to have 24 dB of gain but was found to have around 30 dB of gain when tested on the bench. The 1 dB compression point is 5 dBm, which is high enough since the output of this amplifier is filtered and then sent to the digitizer. This Mini-Circuits amplifier is a more cost effective option than another Miteq amplifier would be. Since this amplification stage does not have the same effect

on the overall receiver chain as the first stage, putting in a less expensive amplifier allows the overall cost to be reduced while not significantly reducing the quality of the receiver chain.

2.8 Channel Selection Bandpass Filter

The bandpass filter after the mixer was chosen from KR electronics. Each different IF frequency has a bandpass filter with a different center frequency after the mixer. There are two 5 MHz bandpass filters, two 1 MHz bandpass filters, and two 500 KHz bandpass filters. These filters are used to greatly reduce the mixing products so that they don't show up in the bandwidths of interest for the other IF frequencies. The mixing products produced by a mixer can be calculated by the following equation:

$$F_{IF} = a * F_{LO} - b * F_{RF}$$

Using the above equation in the case of a 200 MHz RF signal and a 199.95 MHz LO, the spurs of interest can be calculated. In this case, the desired output frequency is 500 kHz, but there are also spurs that appear at 1 MHz, another one of our desired IF frequencies.

The receiver design needed three filters at low frequencies with non-overlapping bandwidths. The three most appealing choices, because of low center frequency and cost, all had center frequencies that could cause potential issues. The 1 MHz IF channel will produce mixing spurs at 5 MHz. However, the 1 MHz bandpass filter's bandwidth is tight enough that it should crush any of those mixing products. In the same way, the .5 MHz IF channel produces mixing products at 1 MHz. Again, the 0.5 MHz bandpass filter was thought to have a sufficiently high Q as to crush any mixing product that might appear at 1 MHz.

The mixers were evaluated for their -3 dB and -30 dB bandwidths and these values were recorded in Table 1. Each filter's response will have two frequencies where the attenuation reaches -3dB and two frequencies where the filter's response reaches -30dB. The "low" and "high" refer to the low frequency and high frequency where the filter's insertion loss reaches that value. Some values were not measureable due to the range of the network analyzer and these values are marked "NM."

Table 1: Filter properties

Bandpass Filters	Insertion Loss (dB)	3 dB low (MHz)	3 dB high (MHz)	30 dB low (MHz)	30 dB high (MHz)	ΔF 3 dB (MHz)	ΔF 30 dB (Mhz)
5MHz #1	3.54	4.82	5.2	4.66	5.41	0.38	0.75
5MHz #2	3.35	4.82	5.2	4.68	5.41	0.38	0.73
1MHz #1	2.77	0.907	1.09	0.792	1.25	0.183	0.458
1MHz #2	3.05	0.906	1.09	0.789	1.25	0.184	0.461
500 kHz #1	0.26	NM	0.695	NM	0.825	0.39	0.65
500 kHz #1	0.33	NM	0.711	NM	0.827	0.422	0.654

2.9 Power Combiner

The power combiner that was chosen is the Mini-Circuits ZSC-3-2B+. The ZSC-3-2B+ is used to power combine the three IF channels into a single channel of the digitizer. This power combiner works over .01 to 30 MHz, covering the useful IF frequencies. It can handle the necessary amount of power with no problem and has very little insertion loss.

2.10 Digitizer

The digitizer that was chosen is the GE ICS-1650a. It is a four channel digitizer that can sample up to 250 MS/s with 16 MB of onboard memory. The huge bandwidth of the GE digitizer makes it possible to stack several NMR signals into a single digitizer channel. The ICS-1650's sampling frequency can be adjusted from 80 MHz to 250 MHz. There is also the ability to decimate by a factor of 2, 4, 8, or 16. There is an input for a 10 MHz clock and a trigger line used to tell the card when to start its acquisition.

2.11 LO Generation/Amplification

Over the past 15 years, local oscillator (LO) technology has changed from large analog boxes for each LO channel to small digital boards that can support multiple channels. As technology continues to improve, the cost and size of these devices will only improve.

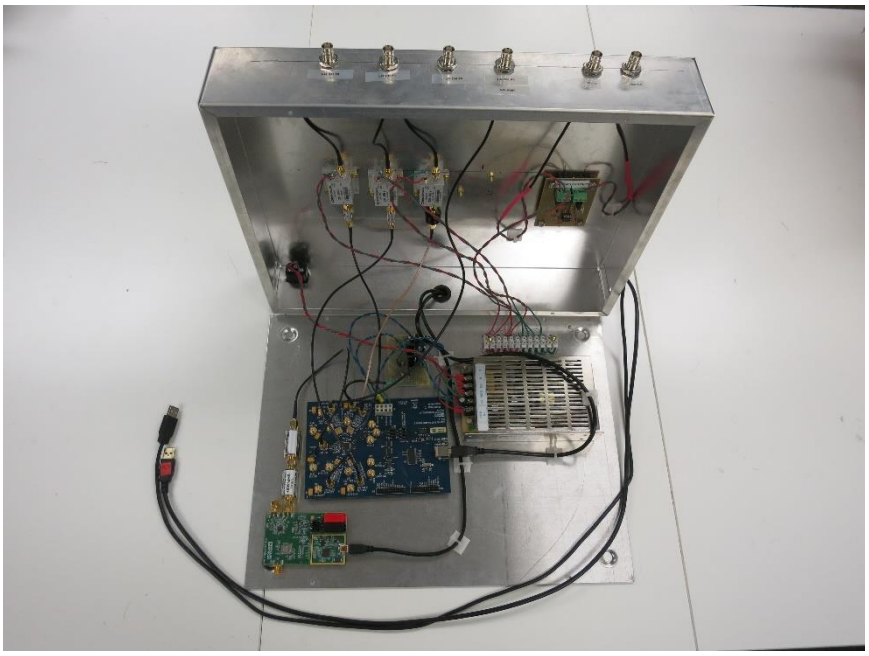


Figure 3: The LO generation and triggering box

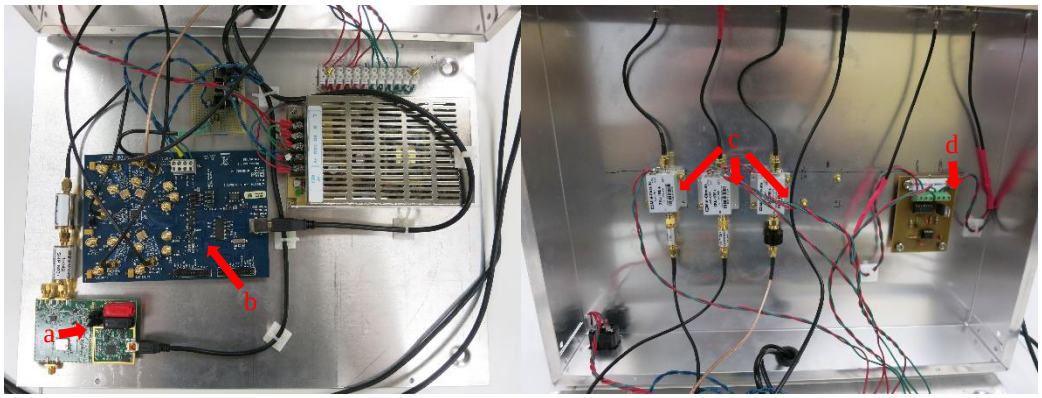


Figure 4: a) the 500 MHz clock that drives the LO board. b) The LO generation board. c) The LO amplifiers. d) The triggering board for the digitizer.

The LO generation card is an Analog Devices AD9959 Evaluation board. The heart of the board is a four channel direct digital synthesizer chip that has a maximum on board clock rate of 500 MHz. This allows the board to produce four independently controlled sinusoids with frequencies less than 200 MHz. Three of these channels are used for producing LO

signals to mix the RF against. The fourth is used to produce a clock for the digitizer card since it does not have the capability to produce its own clock without adding a crystal oscillator. The AD9959 board has the potential to have a reference clock on board when used with a crystal oscillator, but it does not have a sufficiently large multiplier to self-produce a 500 MHz clock on-board. Since the AD9959 board has the ability to take in an external clock, an AD4351 board is used to produce a 500 MHz clock for the AD9959. The AD4351 board has a high stability oscillator and is able to produce a very accurate 500 MHz clock to drive the AD9959. All of these components are shown in their enclosure in Figure 3 and Figure 4.

The AD9959 is inexpensive for a four channel oscillator, is small, and is straightforward to set up and use. In this modular receiver, the AD9959 board is controlled with the software included by Analog Devices. This allows the frequencies of any or all of the channels to be changed at any time. Since this device produces the LO signals for the receiver and those frequencies are so easy to manipulate, it also makes it straightforward and quick to change between different experiments acquiring different nuclei.

2.12 Digitizer Computer

The digitizer computer was designed and then built by a custom PC builder. This allowed for the computer to have the best attributes possible at the time for the price point. One choice was to have 24 GB of RAM so that when the digitizer is acquiring data or the data is being post processed, the RAM will never be a bottleneck. The processor is an Intel Core i7-920 and the graphics card is an Nvidia GTX 285, which combined make the computer capable of doing fast computations on large matrices. The PCI-e bus is one of the buses used by the computer to communicate with added cards and devices. The graphics card communicates with the computer over the PCI-e bus as well as the digitizer card. Many computer motherboards

will only have a very limited number of bus lanes so that only one card can fully use its bandwidth. The motherboard put into this computer has enough PCI-e bus lanes to support the full bandwidth of three cards. This means that the graphics card and the digitizer card will not compete for bandwidth with each other and both are fully supported.

To fully use 24 gigabytes (GB) of RAM, a 64 bit operating system must be used. This is because a 32 bit operating system cannot natively address more than 4 (GB) of memory. The 64 bit operating system can natively address up to 18 exabytes (EB) of memory. Using a 32 bit operating system would prevent the computer from natively being able to address more than $1/6^{th}$ of the installed RAM. The software development kits (SDK) written for the ICS-1650a digitizer card are written for 32 bit Windows and 64 bit Linux. Because 64 bit Linux was the only operating system that allowed for the full use of the RAM in the computer, that was the operating system chosen to be used to run the computer. One of the best supported distributions of Linux with the best documentation is Ubuntu Linux. Specifically, Ubuntu 10.04 64 bit edition is used. A copy of MATLAB has been purchased and installed on the computer for post processing the data. The desktop is shown in Figure 5.

Figure 5: The digitizer computer

2.13 Triggering Circuit

One of the attempts at increasing the stability of the digitizer card was cleaning and shortening the trigger sent to the digitizer. When viewed on an oscilloscope, the trigger signal going into the digitizer was ringing on both the rising and falling edge. It was believed that this ringing was retriggering the digitizer before it was ready and causing instability. This triggering circuit was built to try to reduce these issues to determine if they had an effect on the stability of the digitizer.

The first stage of the triggering circuit is a Schmitt trigger IC. This IC takes in a trigger signal that is either noisy or slowly rising and outputs a trigger signal that rises at a set rate and has very little oscillation. The receiver is designed to be used on a variety of systems and so the quality of the trigger entering the system is unknown. Having an IC that can take a dirty or unacceptable trigger and output a clean and standard signal is a great first step towards stability.

With some systems, the trigger to the digitizer might encompass the entire time of digitization. When testing stability of the digitizer, it was more stable when given a short, consistent trigger pulse. Because of this, the next stage in the triggering circuit takes a trigger signal of any length longer than 1ms and reduces it to a 500 μs trigger pulse. This is accomplished with a 555 timer circuit that can be tuned with two capacitors. The circuit diagram is shown in Figure 6. This circuit reduces the requirements on the trigger signal entering the receiver by cleaning it before sending it to the digitizer, increasing the stability and flexibility of the system. If the trigger signal sent by the MRI system is clean and less than 1 ms long, the trigger board can be easily bypassed and the digitizer can be directly interfaced with the MRI system.

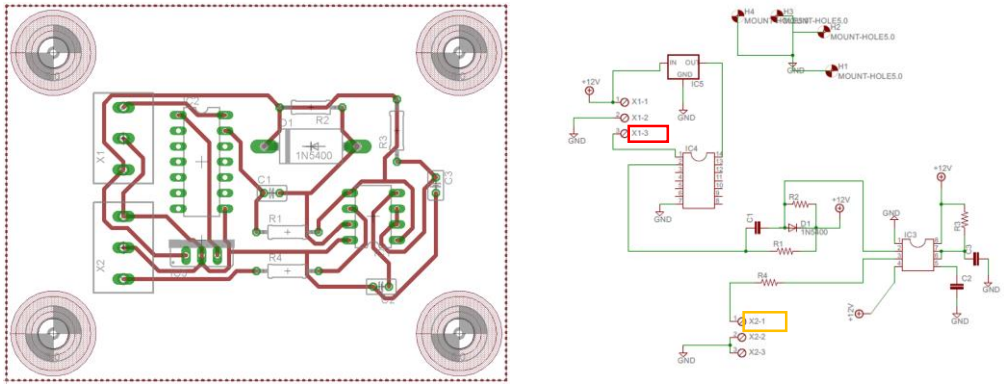


Figure 6: a) The PCB layout for the triggering board. b) The circuit diagram of the triggering board. The red box denotes input and the orange box denotes output.

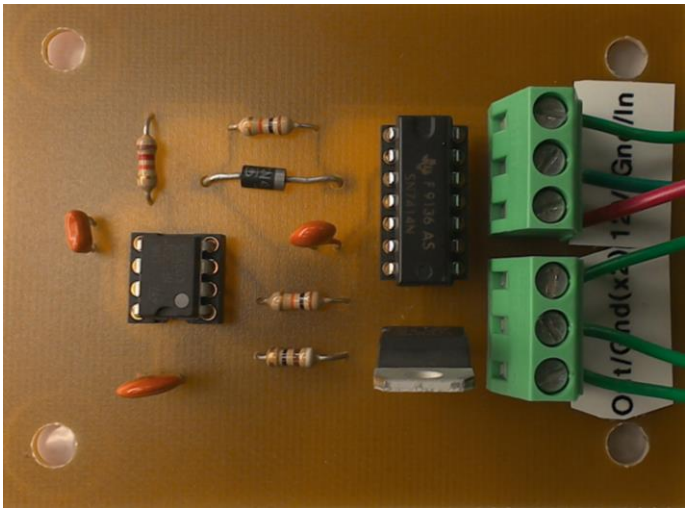


Figure 7: The triggering board.

2.14 Digitizer Cooling Solution

The trigger cleaner circuit increased the stability of the digitizer card somewhat, but the digitizer card continued to stop accepting data in the middle of a set of acquisitions rather frequently. Next, it was thought that the card might be overheating due to an insufficient fan to cool the card. Upon inspection with a FLIR handheld IR sensor, it was seen that the card was reaching temperatures that were beyond its manufactured specifications. A dual fan

cooling solution was built using Cooltron fans, one fan on each side of the card. Because the fans do not fit inside the computer chassis, a PCI-e extender cable is needed to bring the card outside of the computer case. This increased the card's stability dramatically, especially in cooler ambient environments.

Figure 8: a) The cooling solution without digitizer card installed. b) The cooling solution with digitizer card installed. c) The digitizer card being plugged into the PCI-e bus extender.

2.15 Digitizer Power Requirements

When looking at the designed power draw of the digitizer card, it was seen that the power requirements of the digitizer under load are very close to the maximum possible power output from the PCI-e bus. The maximum supported power draw over PCI-e is 25 W and the digitizer card is designed to draw 21.84 W, which should be supported under the PCI-e standard. However, there is also a quite powerful graphics card installed on the PCI-e bus for processing vector math. It was hypothesized that the two cards could be pulling more power than the PCI-e bus could support and so a device was installed to help increase the maximum amount of power that the PCI-e bus could support. The PCI-e bus power line provides 12 V and so another 12 V supply was tied into the PCI-e bus to supplement the power that could be provided. The additional power supply didn't change the voltage of the line in the PCI-e bus, but only increased the current capacity of the line.

Combining the trigger cleaner, the cooling solution and the increased power capacity of the PCI-e bus, the digitizer has been significantly more stable.

2.16 Fully Assembled Receiver

The fully assembled receiver fits onto a rolling cart that allows for easy mobility of the system. On the top shelf is the desktop computer that controls the digitizer and a small netbook for controlling the LO and clock sources. On the bottom shelf are all of the rest of the receiver components. One enclosure contains power supplies and provides power for the other boxes. One enclosure contains the LO source and the triggering circuit for the digitizer board. The other enclosures contain the amplifiers, mixers, switches, limiters and attenuators of the receiver. The enclosures are separated by IF frequency. One enclosure contains both channels at the 500 kHz IF frequency. The two channels are then summed into different power combiners and then digitized on different channels of the digitizer card. Figure 1 gives a detailed breakdown of the complete system.

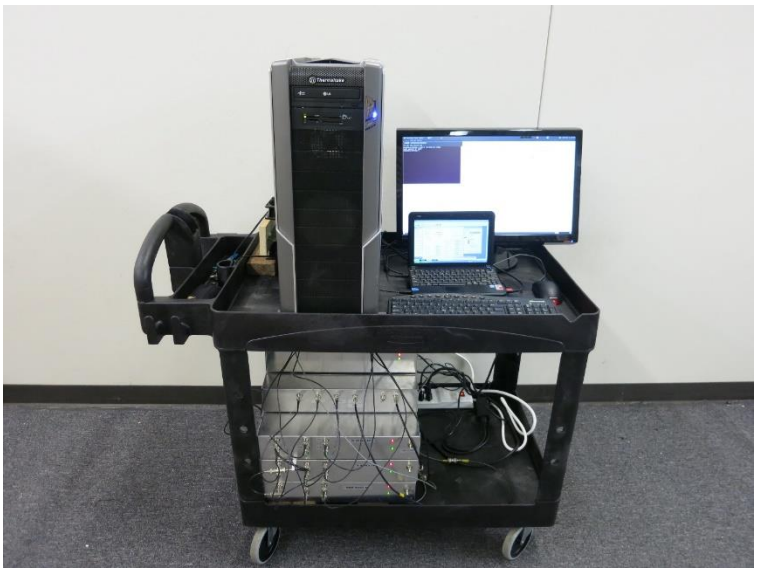


Figure 9: The fully assembled receiver on its cart.

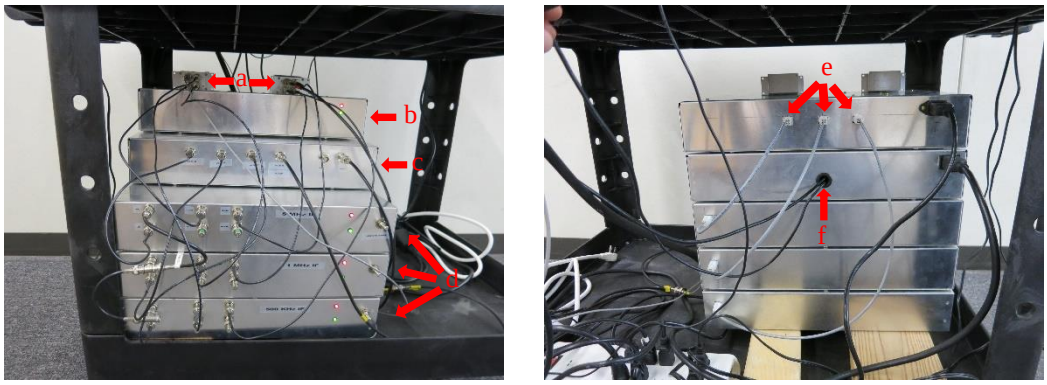


Figure 10: a) The power combiners. b) The power supplies for the system. c) The LO generation and triggering board box. d) The boxes containing the amplifiers, mixers, and attenuation. e) The power connectors. f) The USB cables for controlling the LO board.

For all of the boxes except for the LO generation box, the power is provided from a common set of power supplies. Each cable has +12 V, +5 V and -5 V provided and the connector is keyed such that it cannot be connected incorrectly.

2.17 Cost

The total cost of the system was tabulated and is shown in Table 2.

Table 2: Total cost of FDM receiver

Component	Part	Price	Number	Total
500 KHz Bandpass	KR 2851	228	2	456
1 MHz Bandpass	KR 3128	213	2	426
5 MHz Bandpass	KR 2938	129	2	258
Limiter	VLM-52	42	6	252
Mixer	ZX05-1L+	38	6	228
LO amplifier	ZKL-1R5+	150	3	450
Power Splitter	Z99SC-62-S+	65	3	195
Power Combiner	ZSC-3-2B+	62	2	124
Variable Attenuator	ZX73-2500+	50	6	300
Switch	ZYSWA-2-50DR	70	6	420
Various resistors/caps/enclosures		4	200	800
Second stage amplifiers	ZFL-500LN+	70	6	420
AU-1647 LNA	Low Noise Amplifier	435	6	2610
Computer		1500	1	1500
Lo generation board	AD9959 Eval + AD4351	500	1	500
Digitizer Card	ICS-1650a	5000	1	5000
				Total Cost
				$13939
				Cost per channel
				$2323.17

The cost per channel of the entire system is around $2300. However, the cost to add additional channels is significantly less and is only $1300 as can be seen in Table 3.

Table 3: Cost of additional Channels

Component	Part	Price	Number	Total
Bandpass	KR 2851	228	1	228
Limiter	VLM-52	42	1	42
Mixer	ZX05-1L+	38	1	38
LO amplifier	ZKL-1R5+	150	1	150
Power Splitter	Z99SC-62-S+	65	1	65
Power Combiner	ZSC-3-2B+	62	1	62
Variable Attenuator	ZX73-2500+	50	1	50
Switch	ZYSWA-2-50DR	70	1	70
Various resistors/caps/enclosures		1	90	90
Second stage amplifiers	ZFL-500LN+	70	1	70
AU-1647 LNA	Low Noise Amplifier	435	1	435
				Total Cost
				$1300

3 SOFTWARE

The receiver software is broken up into two main parts: the card control and the post processing. To be able to control the card, first the SDK must be compiled on the system. The card is initialized through a shell script and then control software can be run. Once the card collects the data, the data is processed using MATLAB scripts.

3.1 Digitizer Control Software

The digitizer software is an executable that can be run in the command line interface provided in Linux (the terminal). This allows inherent stability and flexibility for future implementations of this software. The digitizer software interacts with lower level code to be able to control the card and initializes the card with the correct parameters each time the card is initialized. If it was desired, a different executable could be created for each experiment. However, it is quite straightforward to make the desired changes to the digitizer software source code and recompile for a new experiment. Decoupling the digitizer software from the MATLAB image processing code allows the receiver to be used with whatever digitizer is desired. If advances in technology increase capabilities and decrease the price of new digitizers, a new one can be purchased and integrated much more easily than if all of the software was coupled together.

Initially, it was desired to control the digitizer from MATLAB since MATLAB would be used to do the post processing of the data. MATLAB can communicate with shared objects, written in a derivative of C, called mex files. These mex files are not C or MATLAB code, but can act as an intermediary between the MATLAB program and other C shared objects that you wish to use. It is theoretically possible that any C source code could be converted to a mex file, but this can get quite complicated in practice. A mex file was created that would interface

between a shared object, written in C, and a processing script, written in MATLAB. The mex file initialized the necessary memory space and then activated the C shared object. The C shared object communicated with digitizer card and then pushed the acquired data back to the mex file where it could be collected by MATLAB. This implementation of the software would have been slightly more convenient for pulling data into MATLAB, but the memory allocation in the mex file would become unstable at times. Also, having a mex file communicate between MATLAB and the code running the digitizer card meant that once an acquisition was started, it could not be aborted by the user. These negative factors led to the digitizer and post processing software being decoupled from each other.

The digitizer software goes through many steps of initializing the card for the experiment that is desired. The code is based on the demo software that was provided with the SDK, and builds off of it, making changes to the memory allocation, the digitization speed, the decimation factor, the triggering mechanism among others. The ICS-1650 SDK, written by GE, sets up structures that ferry the variables to lower level code which communicates with the card. There are structures for setting up the clock, the trigger, synchronization between cards, as well as several other data types created for the data coming off of the card. The 'clocking' structure takes in the clock frequency of the board (the sampling frequency), as well as several other variables that select whether the clock will be internal or from an external source, the speed of the reference clock, whether to output the internal clock to a connector for synchronization to another card. The 'triggerConfig' structure allows the user to choose whether there will be an external trigger or a simulated trigger using the keyboard as well as whether to trigger on the rising or falling edge of the trigger signal. Once these structures are created and populated with the desired parameters, they are sent to the card via other C

functions. The ‘ics1650ClockSet’ function takes the ‘clocking’ structure and converts it into card-readable binary words that are written to specific registers on the card. The ‘ics1650TriggerSet’ function does the same thing for the ‘triggerConfig’ structure. A few more parameters have to be set, such as the data type that will be acquired by the card, the voltage range for the card, and the decimation and gain. The memory where the acquired data will reside on the card is forced to all zeros. This prevents any data corruption from happening due to data already residing on the card’s memory. The data from the card needs a place to go after each acquisition before it can be written to a hard drive and so memory in the computer’s RAM is allocated. Now the card is ready to begin and the user has seen the ‘memory allocation success’ message in the terminal window. The software now enters a loop that will run for the number of acquisitions that was entered by the user. The loop pauses while waiting for a trigger and once the card sees a trigger it acquires the number of samples in one acquisition and saves that data to the card. The data is quickly pushed off of the card and into the computer’s RAM and the card waits for another trigger. While the card is waiting for a trigger, the RAM shuffles the data off onto the hard drive. The speed between the digitizer and the RAM is very fast because of the PCI-e bus that is utilized between the two. The digitizer can then go back into a state waiting for the next trigger while the RAM writes the data to the hard drive. The data is written to the hard drive as a binary file because that is the smallest representation of the data and quickest to write.

Finally, when the card has finished its last acquisition and the last data has been sent to the computer, the card resets all of its parameters to power on defaults. If an acquisition is terminated prematurely by the user instead of correctly resetting itself at the end of the

acquisition, the card will crash and requires a computer reboot to continue functioning correctly.

More detailed information can be found in Appendix A about use of the software and a copy of the source code can be found in Appendix B.

3.2 Post Processing Images

The MATLAB image processing script begins by loading the binary file that has been saved by the digitizer. Because of their large size, some of the binary files can take several seconds to load. Next the background math is set up for the rest of the script. Important parameters are entered such as the echo time, the sampling frequency, the center frequency of the IF, the number of points in the frequency encode direction, the bandwidth of the experiment and the length of the transmitted RF pulse. From these parameters, other parameters can be calculated such as the point at which the echo starts and ends in the file, the number of samples in each echo and the total number of echoes. Since the digitizer records the 180 degree RF pulse all the way through the echo, the echo does not start at the beginning of each acquisition. Instead, the 180 degree RF pulse appears first in each acquisition and then the echo. To find the start of the echo in each acquisition, the following equation is used:

$$echo\ start\ sample = \left(\frac{TE}{2} - \frac{T_{acq}}{2} + \frac{T_{tx}}{2}\right) * F_s + 1$$

Where TE is the echo time, T_{acq} is the acquisition time, T_{tx} is the length of the transmit pulses and F_s is the sampling frequency.

After Fourier transforming the processed echo at its IF frequency, there is a frequency spectrum with far too many points. The window of interest must be pulled out of the frequency domain. To do this, the point that represents the center of the bandwidth of interest is found

using the following equation where $echo_{samples}$ is the number of digitized samples per acquired echo, F_c is the center frequency of the acquired signal and F_s is the sampling frequency:

$$center\ profile\ point = \frac{echo_{samples}}{2} + \frac{echo_{samples}}{2} * \frac{F_c}{\left(\frac{F_s}{2}\right)}$$

Since all of the acquisitions are just a string of numbers, the data is reshaped from a single vector into a matrix with the dimensions of number of acquisitions by number of points per acquisition. Next, both the transmit signal and the echo are pulled out of each acquisition. Since there are three different IF signals, the transmit signal must be filtered before being used to determine the phase relative to the other acquisitions.

At this point in the program, the necessary computations have been done to know where the center of the frequency domain profile will be, a filtered transmit pulse has been created and the echo separated out. Next, the transmit signal's phase is found with the sub-function 'find phase'. To accomplish this, the transmit signal is IQ demodulated, providing a real and imaginary part at baseband. I-Q demodulation is also known as quadrature amplitude modulation. For an RF signal named A with center frequency F_c, A is multiplied by a cosine to produce the I component and multiplied by a sine to produce the Q component as seen in the following equations:

$$I = A * \cos(2 * pi * F_c * t)$$

$$Q = A * \sin(2 * pi * F_c * t)$$

The form of the demodulated signal allows for straightforward calculation of the phase of the signal using an arctangent.

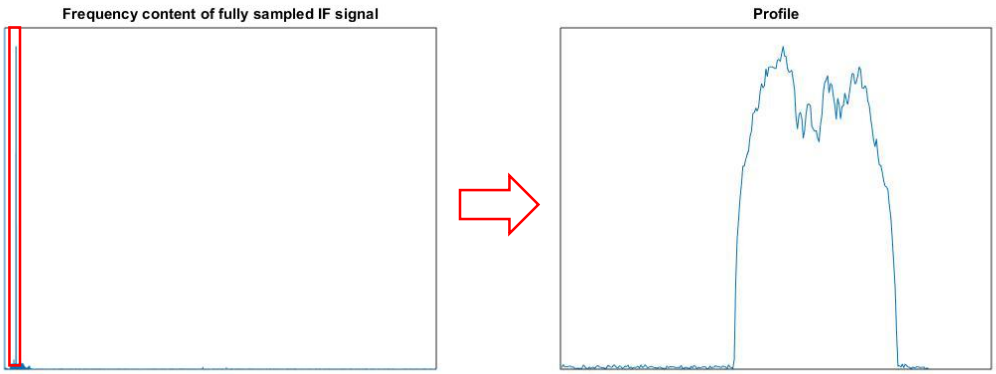
$$phase = \text{atan}\left(\frac{Q}{I}\right)$$

MATLAB's equivalent command is the 'angle' command, which takes in a complex number and outputs the phase angle. The phase is found for a set of points in the transmit pulse of interest and also in that same range in the original transmit pulse. The difference in phase is found between the current transmit pulse and the original. Because any single point is susceptible to noise, the average difference in phase is found between the current and original transmit signals. Now that the phase offset has been found for every acquisition, each can be corrected.

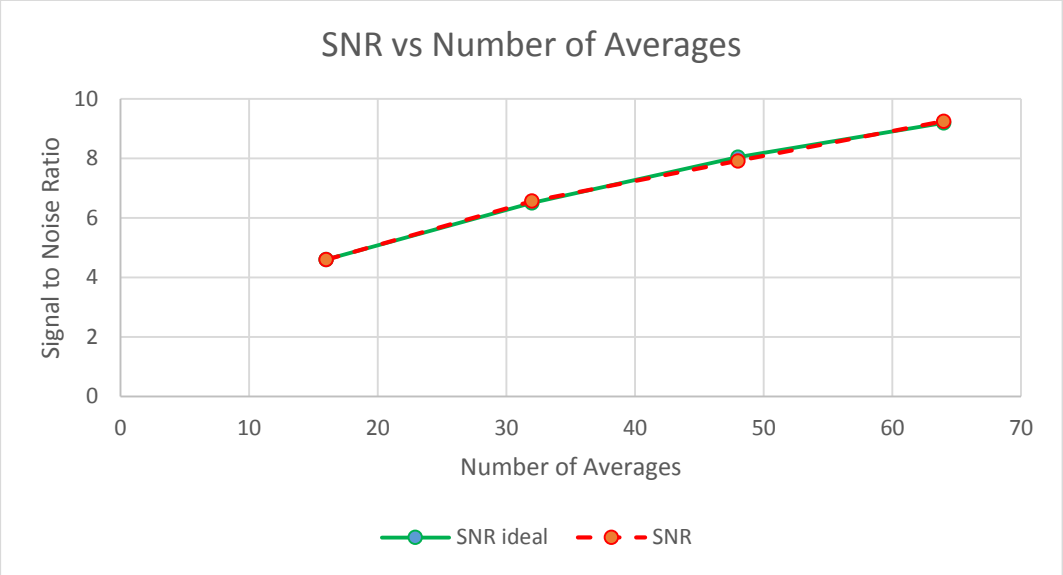
The phase is corrected in the sub-function 'correct_phase'. The phase correction necessary for each acquisition is known from the 'find_phase' function and now needs to be applied. In frequency space, a phase change can be applied by multiplying by a complex exponential.

$$phase\ corrected\ frequency\ content\ of\ signal = F(signal) * e^{-j*phase}$$

The 'correct_phase' sub-function takes in an echo, Fourier transforms it, and then applies a phase shift through a complex exponential. The phase shifted frequency domain representation of the fully sampled echo is then returned to the main MATLAB script. The returned echo still has the bandwidth dictated by the sampling rate, far wider than the bandwidth of interest. The center point of the bandwidth of interest was previously calculated and is now used to pull out the relevant frequency data.



A good test to determine whether the receiver is phase stable or not is to take many acquisitions of the same data and make sure that when averaged, the SNR improves by the square root of the number of averages. If the phase correction is not accurate, the signal has the potential to be reduced along with the noise. To test this, a profile was acquired 128 times and the SNR vs number of averages was plotted for 16, 32, 48, and 64 averages. The SNR at 16 averages is the baseline for the calculation of the ideal SNR. As can be seen in the following figure, the averaged profiles follow the ideal SNR to within 2% at 64 averages. The ideal SNR was calculated by improving the SNR of the 16 average profile by the square root of the number of averages.



Without phase correction, the change in phase between acquisitions becomes readily apparent in an acquired image. The image on the left was not phase corrected and was processed as it was recorded. The image on the right was phase corrected using the transmit signal as a phase indicator.

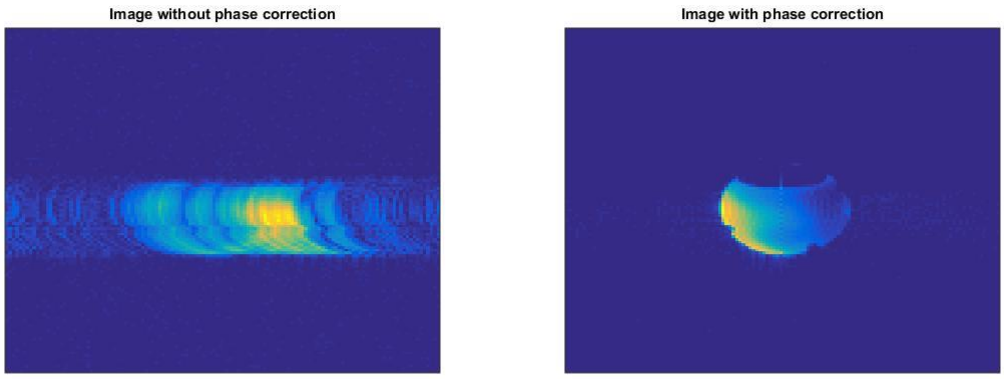


Figure 11: Reconstructed images with phase correction (right image) and without phase correction (left image)

3.3 Post Processing Spectra

The processing for a NMR spectrum is very similar to the processing of an image. Now instead of acquiring echoes, FIDs are acquired. The FID is Fourier transformed and phase

corrected based on the transmit signal. All of the acquisitions are averaged and the spectrum of the nuclei of interest is pulled out of the fully sampled, averaged, frequency space and then saved. Chemical spectra must be phase corrected and baseline corrected and ACD Labs software was used to accomplish this. Since the data that was saved earlier was the spectrum, the data is now inverse Fourier transformed to return it to a representation of the FID, the format of the data that ACD Labs can accept. ACD Labs does not directly import binary data, but only takes in files saved by commercial scanners. To allow ACD Labs to import the data from the modular receiver, it needs to be reshaped into a file like that from the commercial Varian Inova system. A sample acquisition is saved on the Inova scanner with the same pulse sequence and parameters as the acquisition acquired by the modular receiver. The FID file produced by the Inova scanner has many of the acquisition parameters saved as a header in the data and is necessary for processing by ACD labs. Then a MATLAB script strips the header out of the Inova binary file and it is added to the data recorded on the modular receiver. ACD Labs sees this new file as a file from an Inova scanner and can import it. Finally ACD Labs is used to phase and baseline correct the spectra and it can be exported in a variety of formats.

4 EXPERIMENTAL METHODS

Three experiments were conducted, highlighting the various usage cases of the receiver. Experiments were conducted on a 4.7 T research scanner as well as on a .08 T magnet used for a lab class. Data was acquired from a six channel array, a two channel array, and a single solenoidal coil. Finally, hydrogen and carbon data were both collected.

4.1 Six-Channel Hydrogen

The first images were collected from a six channel surface coil array that is paired with a shielded birdcage for the transmit coil. The array coils are overlapped with their neighbor to geometrically decouple them by removing the mutual inductance between the coils. Since the geometric decoupling is not sufficient for the coupling between non-adjacent coils, there are low impedance preamps on the coil that add additional decoupling between the coils. The decoupling between the surface coils and the birdcage is provided by an active PIN diode switch on every coil. The PIN diode switch on the surface coils are activated by a PIN diode driver that is controlled through a trigger line from the Inova system. When the PIN diode switch is turned on, the coil is shifted significantly off frequency and will not couple to another coil at its original frequency. The PIN diode switch on the surface coils needs to be activated during the transmit pulses, and so the output of the PIN diode driver is high during transmit. This helps alleviate any effects the surface coils have on the transmit coil. The birdcage PIN diode switch is also activated by a PIN diode driver that is controlled by the same trigger line. However, the PIN diode switch on the birdcage needs to be activated during receive and so it is high during the acquisition portion of the pulse sequence.

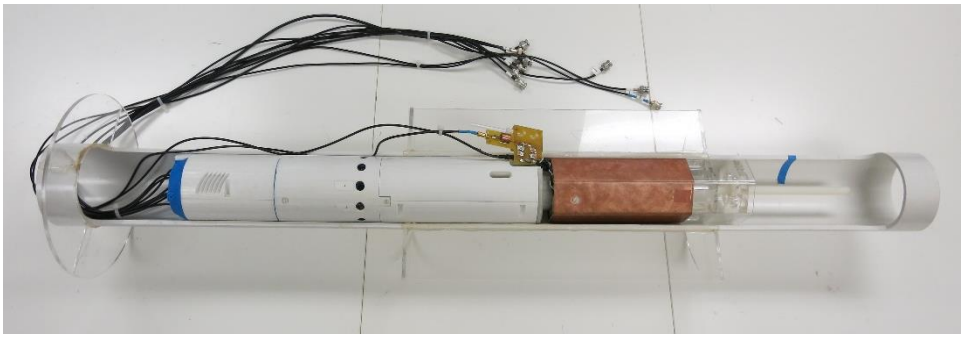


Figure 12: The surface coil array inside of the birdcage volume coil on the loading system

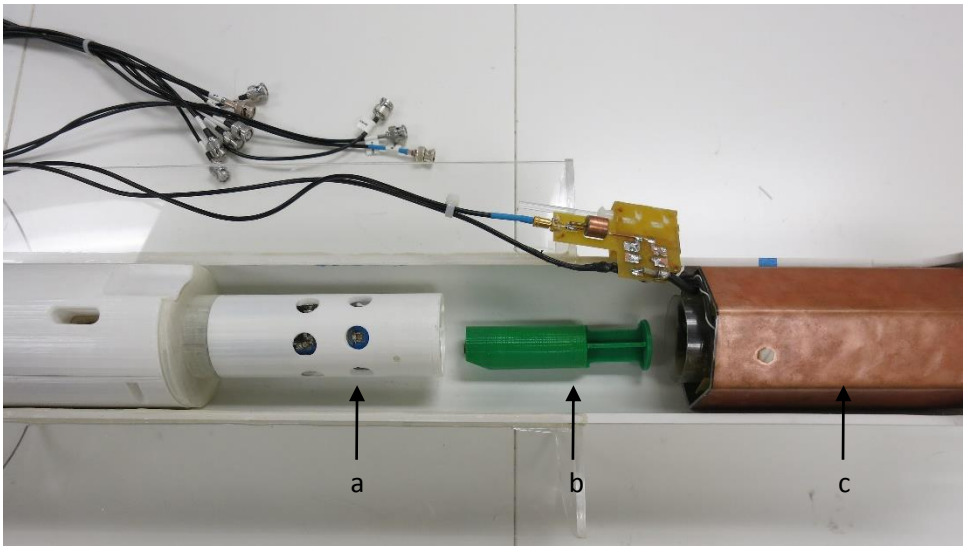


Figure 13: a) The housing for the surface coil array. b) The phantom. c) The transmit birdcage coil.

The experiment was performed in a 4.7 T research scanner at the University Services Building of Texas A&M. The magnet is paired with a Varian Inova scanner that is used for pulse sequence programming and control of all aspects of the NMR experiment. The modular receiver was interfaced with the Varian Inova through two trigger lines and a 10 MHz clock line. If the fourth channel of the AD9959 frequency generator is not being used as an LO source, it can be used to produce the 10 MHz clock. Some slight modification to the pulse sequence was necessary in order to program the two trigger lines. The pulse sequence used can be seen in Figure 14.

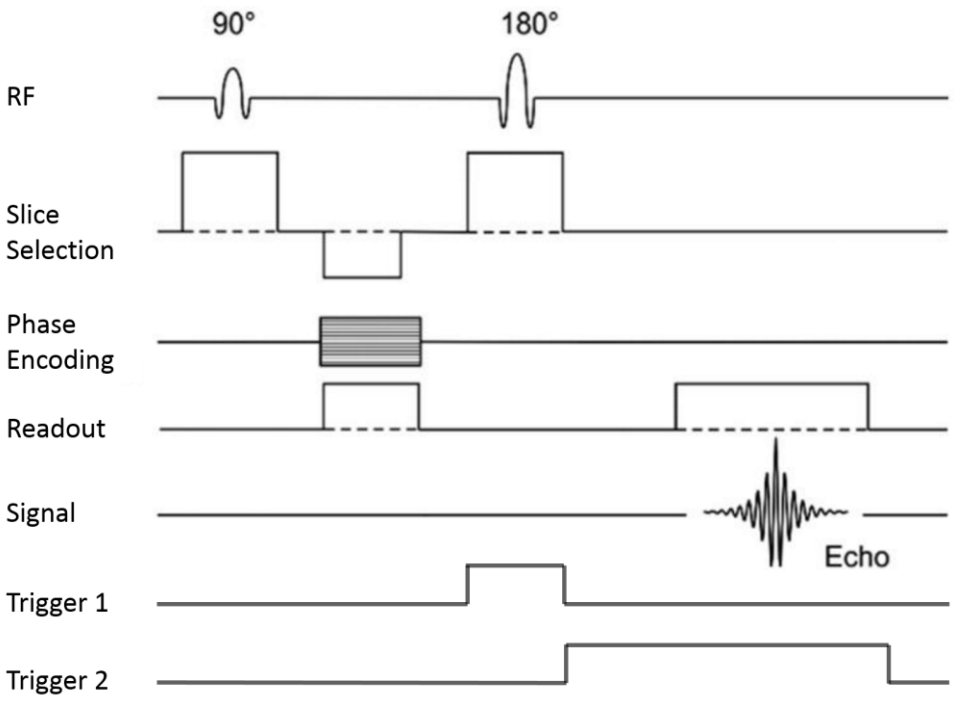


Figure 14: The pulse sequence used when acquiring data from the six channel mouse array. Trigger 1 is used for triggering the digitizer and trigger 2 is used for turning off the switch at the input to the receiver.

The first trigger is used to activate the digitizer at the beginning of the 180 degree RF pulse. Recording the transmit pulse gives a phase reference with which to correct the phase of the echoes. The second trigger line is used to activate the switch to allow signal through. Before the switch is activated, it is essentially a 30-40 dB attenuator that prevents the transmit signal from saturating all of the successive stages. It is triggered at the end of the 180 degree RF pulse to try to alleviate any delay in the switching that might occur before the data acquisition.

The phantom used is a refillable, 3D printed phantom that is calibrated to mimic the load of a mouse on the surface coil array. Each surface coil was used individually to acquire an image using only the Inova scanner's single channel receiver. Then all six channels were acquired simultaneously using the modular receiver.

The pulse sequence used the following parameters: repetition time (TR) of 250 ms, echo time (TE) of 20 ms, receiver bandwidth of 50 KHz, 128x128 points. Only one average was collected. The modular receiver was sampling at 50 MS/s and acquired 700000 samples every acquisition.

4.2 Two-Channel Carbon

The carbon spectra were acquired with two geometrically decoupled surface loops and a transmit Helmholtz coil. The surface coils and the transmit coils are geometrically decoupled from each other and there is no other active or passive decoupling present. Images of the Helmholtz transmit coil and the surface coils can be seen in Figure 15.

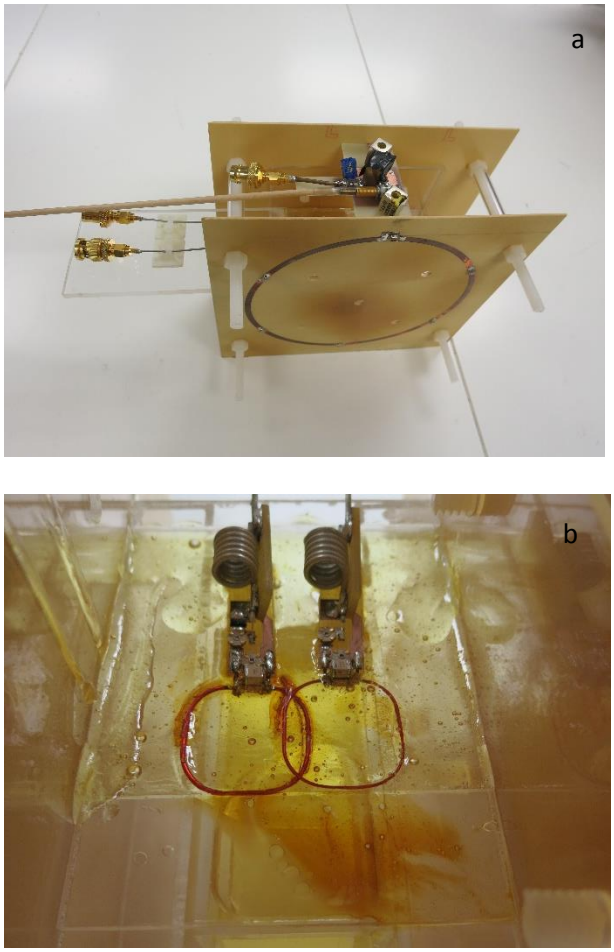


Figure 15: a) The transmit Helmholtz coil. b) The geometrically decoupled surface coils. The olive oil phantom can be seen beneath the surface coils.

The phantom used was an acrylic box filled with olive oil. The olive oil had a natural abundance of 13C and was not modified from its original form. The pulse sequence used can be seen in Figure 16.

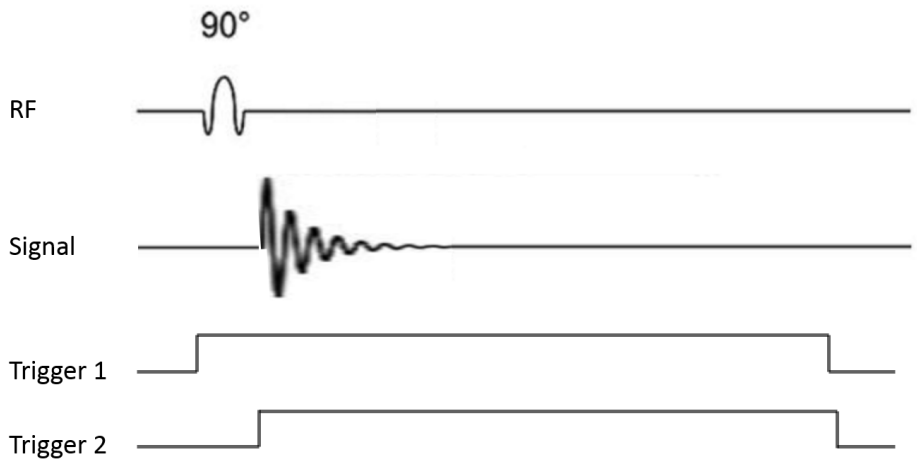


Figure 16: Carbon spectra acquisition pulse sequence

The transmit pulse length was 50 us, the acquisition time was 0.1024 s, and the spectral width was 10 kHz. The repetition time was 3 s and there were 64 averages collected. The modular receiver was sampling at 10 MHz and acquired 1100000 samples during each acquisition.

4.3 Low-field Desktop Magnet

The third set of data demonstrates not only the flexibility of the receiver but also its portability. The receiver was transported from an off campus lab to an on campus lab where it received an image from a very low field desktop magnet. The receiver was transported in a normal, consumer grade truck with no issues during transit.

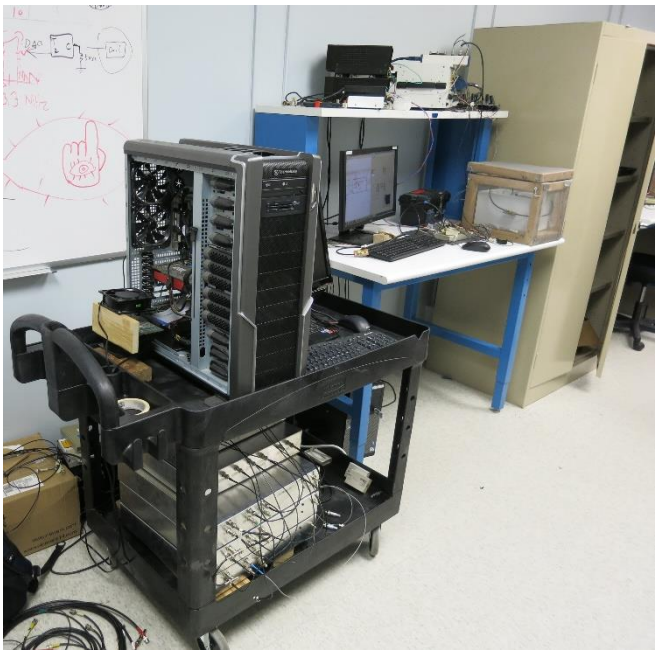


Figure 17: The modular receiver set up for imaging in the on campus lab

The receiver was interfaced with a desktop MRI system that is designed to be a semester long project for students in a lab class. The magnet is comprised of two square, neodymium magnets that are contained within a steel frame. There are two steel flux spreaders to help increase the homogenous region at the center of the magnet. The magnet is .08 T, and so water resonates at a much lower frequency than in the 4.7 T magnets previously used. Previously, the receiver was acquiring 200 MHz data and now it is tasked with acquiring 3.3 MHz data, demonstrating its wide range of applications. The desktop system is controlled via National Instruments (NI) hardware and software. The NI hardware includes slow and fast analog output channels as well as RF digitizers fast enough to digitize the 3.3 MHz RF signal without mixing. The NI hardware is all connected to a computer where it is controlled via NI LabVIEW software. LabVIEW is built off of virtual instruments (VI) instead of scripted functions. These VIs are built by drawing connections that send data between blocks. The

blocks act on the data, for instance an adding block will take in two numbers and output their sum. The VI for controlling the analog output lines was slightly adjusted to add two trigger lines to interface with the FDM receiver. The pulse sequence is outlined in Figure 18.

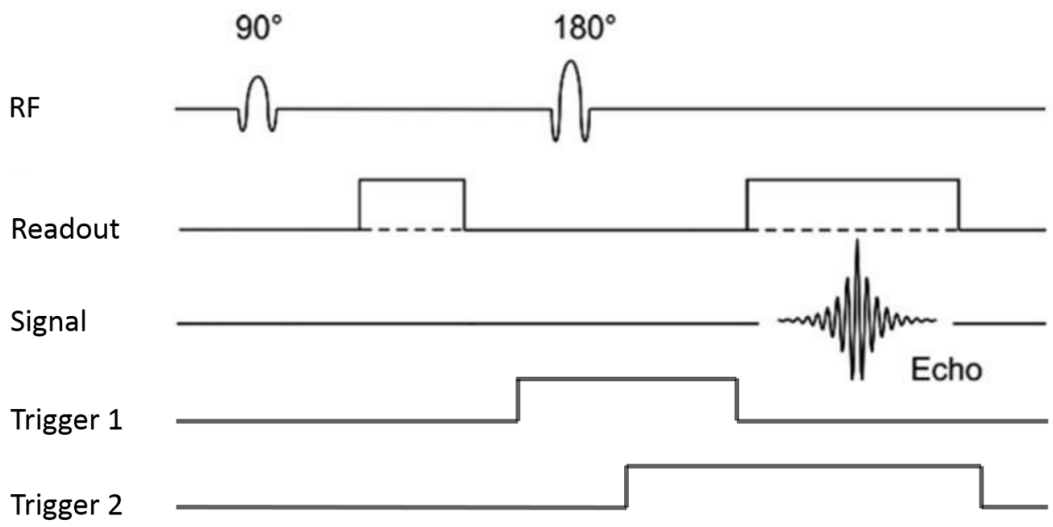


Figure 18: The pulse sequence used while receiving data from the lab class system

The phantom used was a 1 cm diameter test tube of water. The pulse sequence used a 25 ms TE, a 50 us 90 degree RF pulse and 100 us 180 degree RF pulse. The repetition time was 3 s and the acquisition time was 20 ms. Each profile was acquired with 128 averages. The low Larmor frequency of ^{1}H in the magnet and the inhomogeneity of the magnet made signal strength an issue, necessitating the high number of averages.

Unlike the phase encoded images taken on the Varian Inova system, this image was taken as a series of projections at regular angular intervals and then back-projected into an image. The reconstruction algorithm is the inverse radon transform.

5 RESULTS

The receiver was used to collect data on different systems in different locations. Six-channel ^{1}H images and two-channel ^{13}C spectra were collected on a 4.7 T research scanner. The FDM receiver was then transported to a different lab where data was acquired from a low-field desktop system.

5.1 Six-Channel ^{1}H Array

The images from the FDM receiver and the Varian Inova receiver are shown in Figure 19.

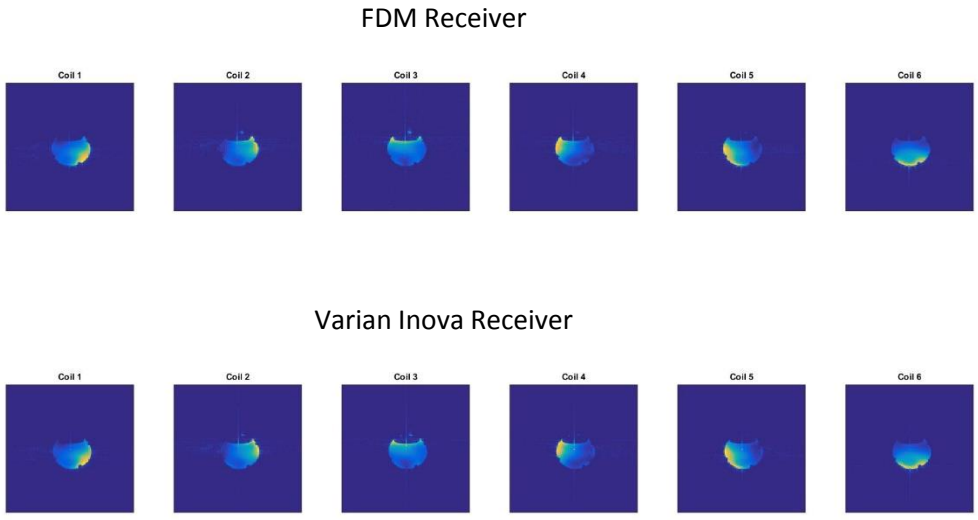


Figure 19: Images taken from each coil by FDM receiver and by Varian Inova receiver

The SNR from each image was calculated by using the mean signal divided by the standard deviation of the noise. The signal area was determined by separating the voxels with a higher value (pertaining to a signal area), from the voxels with a lower value (pertaining to a noise area). The threshold level used for the black/white cutoff was calculated by using Otsu's

method in the MATLAB function ‘graythresh’. The thresholding function used was MATLAB’s ‘im2bw’, which takes an image and the previously found threshold level and converts the image to black and white. The noise area was determined by using a much lower threshold to be sure that none of the signal area was included. The threshold used was 1/3 that given by Otsu’s method. Finally, the noise area was morphologically eroded using MATLAB’s ‘imerode’ function. This further pushed the noise boundary away from any possible signal. These signal and noise areas were calculated for each image before determining the SNR.

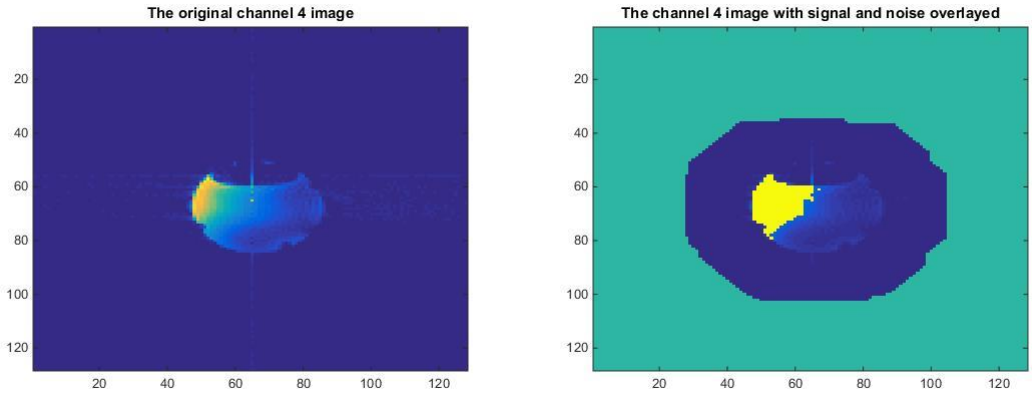


Figure 20: On the left is the original channel 4 image. On the right is the channel 4 image with signal area denoted by yellow and noise area denoted by green

The SNR was calculated for each coil from both the Inova receiver and the FDM receiver and the results are listed in Table 4.

Table 4: SNR values from each coil

Coil Number	Inova SNR	FDM Receiver SNR
1	122.28	103.60
2	83.52	70.31
3	57.03	37.75
4	116.43	88.98
5	121.84	103.47
6	115.43	93.37

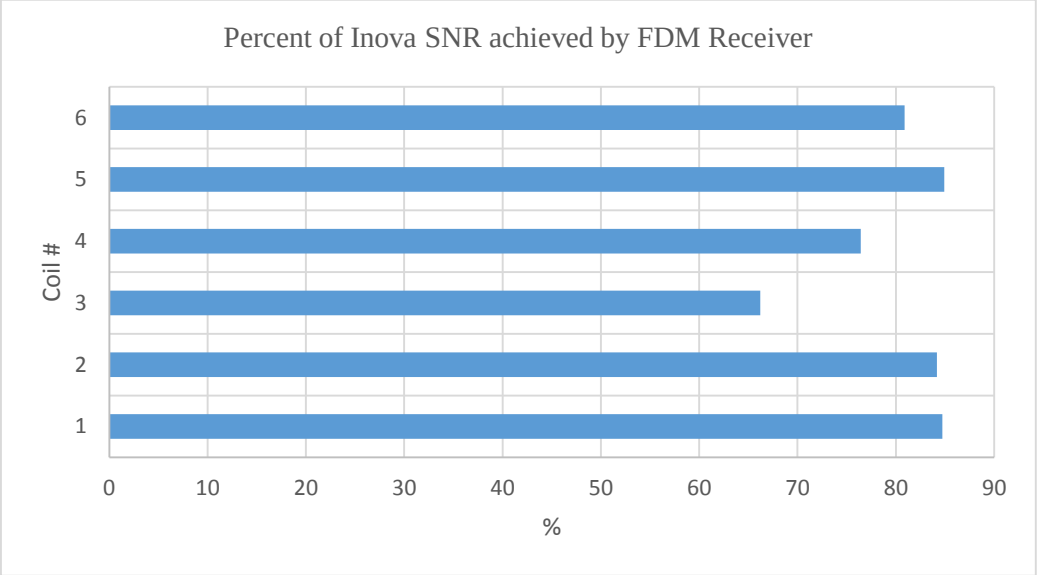


Figure 21: A graph showing what percentage of the SNR of the Inova receiver the FDM was able to achieve on each surface coil

As shown in Figure 21, the FDM receiver was able to reliably achieve 80-85% of the SNR of the Inova receiver with the exception of the signals coming from coils 3 and 4. The signals from coil 3 and coil 4 show reduced SNR compared to the other signals received by the FDM receiver. This discrepancy will be explained in the following section.

5.2 Two-Channel ^{13}C Array

The 13C spectra of olive oil, received from both the Varian Inova receiver and the FDM receiver are shown in Figure 22.

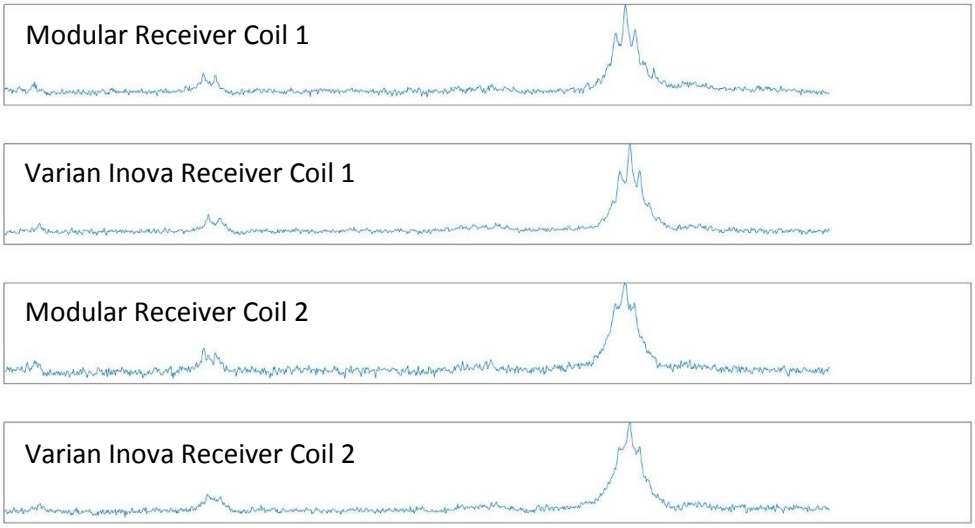


Figure 22: The ^{13}C spectra received by both the Varian Inova receiver and the FDM receiver

The SNR was calculated by taking the area under the large peak in the carbon spectrum and dividing that by the standard deviation of the noise. The SNR values for the carbon spectra can be found in Table 5.

Table 5: SNR values for ^{13}C Spectra

Coil Number	Inova SNR	FDM Receiver SNR
1	162	157
2	155	119

As was seen in the SNR of the ^{1}H images, the FDM receiver exhibits better SNR on some channels than others with respect to the Inova receiver.

5.3 Lab Class ^{1}H Coil

The image of a test tube filled with water, taken using the desktop magnet system, can be found in Figure 23.

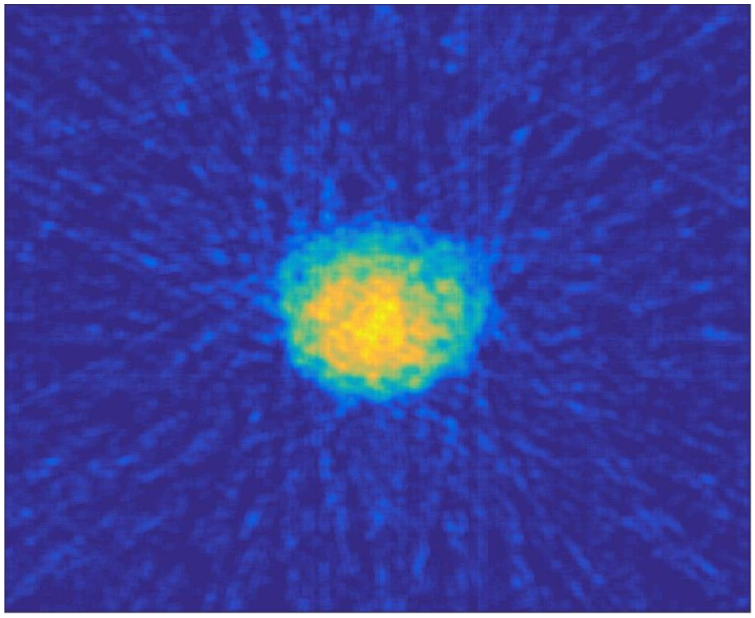


Figure 23: Image of a test tube acquired using the lab class system

No SNR data was taken for the test tube image because of the large quantity of post processing done to the image. This data was purely a demonstration of mobility and flexibility.

6 CONCLUSIONS AND FUTURE WORK

The digitizer successfully demonstrated a high degree of flexibility and portability, acquiring data from a six-channel hydrogen coil on a 4.7 T Varian Inova system, a two-channel carbon coil on a 4.7 T Varian Inova system, and a single hydrogen coil on a .08 T LabVIEW based lab class system. The system was straightforward to use and quite easy to switch from one experiment to another. However, with the ease of use, flexibility and portability, some tradeoffs were made. Looking at Table 4 and Table 5, it can be seen that the FDM receiver can stay within 80-85% of the SNR of the Varian Inova with a few exceptions. This section will address potential improvements in the design.

6.1 SNR of Six-Channel Hydrogen Data

There is a trend that can be seen in the SNR data given in Table 4 and Table 5. It is important to remember that in the six-channel hydrogen experiment, the signals from coils 1 and 2 were acquired on the 5 MHz IF channels, the signals from coils 3 and 4 were acquired on the 1 MHz IF channels, and the .5 MHz IF channels were used to acquire the signals from coils 1 and 2. Figure 21 describes how well the FDM receiver was able to compete with the Varian Inova and shows an interesting phenomenon. While the FDM receiver's SNR from coils 1, 2, 5, and 6 were mostly stable between 80-85% of the SNR of the Varian Inova, the FDM receiver's SNR from coils 3 and 4 dropped significantly.

Since each coil has a different sensitivity pattern and a different level of acquired signal, comparing the SNR from one channel to the next does not provide useful information. However, each IF channel can be represented by a comparison between that IF channel's SNR and the SNR of the Varian Inova, when both are acquiring the same signal.

Table 6: The percent of the SNR of the Inova receiver achieved by the FDM receiver in the hydrogen experiment

Coil Number	Percentage of SNR
1	84.723701
2	84.1779737
3	66.2026613
4	76.4224032
5	84.9257705
6	80.8905003

The data found in Table 6 is the SNR of the signals acquired by the FDM receiver divided by the SNR of the signals acquired by the Varian Inova using the six-channel hydrogen coil. The Varian Inova is assumed to produce consistent SNR results and so the data shown in Table 6 allow comparison between the SNR values of signals received on different IF channels of the FDM receiver. The coil 3 data, acquired on a 1 MHz IF channel, has 78% of the relative SNR compared to coil 1 data, acquired on a .5 MHz IF channel.

6.2 SNR Data from Two-Channel Carbon Data

The SNR of the acquired carbon spectra shows a similar trend. The signal from coil 1 was acquired on the .5 MHz IF channel and the signal from coil 2 was acquired on the 1 MHz IF channel. As can be seen in Table 5, there is a drop in SNR when moving from the .5 MHz IF channel to the 1 MHz IF channel. Again, the SNR values from channel to channel on the FDM receiver cannot be directly compared to each other.

Table 7: The percent of the SNR of the Inova receiver achieved by the FDM receiver in the carbon experiment

Coil Number	Percentage of SNR
1	96.91
2	76.77

Coil 2 data, acquired on a 1 MHz IF channel, has 79% of the relative SNR compared to coil 1 data, acquired on a .5 MHz IF channel. This is very consistent with the 78% found between the .5 MHz and 1 MHz channels in the six-channel hydrogen experiment. So while the channel to channel SNR seems inconsistent, the difference in SNR between channels stays consistent between experiments.

6.3 Consistency of Channel to Channel SNR

The most probable reason for the drop in SNR on the 1 MHz IF channels is the isolation between the .5 MHz IF channel and the 1 MHz IF channel. In Table 1 it can be seen that the 1 MHz and the .5 MHz bandpass filters cutoff long before reaching the 5 MHz IF band. There is no mixing product that will appear in the .5 MHz band, but the 1 MHz band is very close to the 30 dB cutoff for the .5 MHz bandpass filter. The mixing product from the .5 MHz channel will show up on top of the 1 MHz channel. This means that the .5 MHz bandpass filter is not sufficient to prevent interference with the 1 MHz channel as was originally hypothesized. One solution would be to add another .5 MHz bandpass filter in series with the .5 MHz bandpass filter currently installed. This will double the attenuation at 1 MHz from any signals generated by the .5 MHz channel that could interfere with the 1 MHz channel. Adding a second bandpass filter in series will also double the insertion loss of the effective filter. However, as can be seen in Table 1, the insertion loss for the .5 MHz bandpass filter is only ~.3 dB and doubling that

only brings the insertion loss up to ~.6 dB. This is negligible compared to the 3 dB of insertion loss in the other bandpass filters.

6.4 SNR Comparison between FDM Receiver and Varian Inova

On the 1 Mhz and 5 MHz channels, the SNR of the FDM receiver is 85% of the Varian Inova. The goal of equivalent SNR with the commercial Varian Inova scanner was not met. This is because of the noise factor of the system and how large an impact the switch has on the noise factor.

The switch at the input of the receiver is an important component in order to reduce the dynamic range of the acquired signal to the level of the echo or FID. However, it has a big effect on the noise factor of the receiver. The calculation of the noise factor for three components is as follows:

$$F = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 * G2}$$

Where F_n is the noise factor of the nth device and G_n is the gain of the nth device. Using this equation it can be calculated by how much the noise factor changes based on the order of the devices at the input to the receiver.

Table 8: Noise factor vs device order

Order of devices	Noise Factor
limiter -> switch -> amplifier	1.995
limiter -> amplifier -> switch	1.549
% difference	**28.79**

Only three devices were used in the noise factor calculation since anything after the amplifier has very little effect on the noise factor of the system. Even if the device order was

limiter → amplifier → switch → 30 dB attenuator, the noise factor only changes by .1% from the case with no attenuator. While anything after the amplifier does not affect the noise factor very much, Table 8 shows just how much the device in front of the amplifier affects the noise factor. Just by moving the switch in front of the amplifier, the noise factor decreases by almost 30%. This is a big loss and is why the FDM receiver lags the Varian Inova receiver in SNR. Several other topologies were tested to try to move the switch behind the amplifier, and hence dramatically increase the noise figure, but they were not feasible. Many of the other topologies that were tested did not work because there was no detectable transmit signal to be used for phase correction. The amplifier would saturate at around 12 dBm to 14 dBm output power and the -30 dB to -50 dB loss caused by the switch did not let enough of the transmit signal through to be useful for phase correction. There are, however, other possibilities to both increase the noise factor as well as recording the transmit signal. If a switch is found with slightly less isolation than the currently implemented switch, then the switch could be moved behind the preamp, greatly increasing the noise factor of the receiver. If the receiver is used with alongside a transmit system that has the same initial phase for each transmit signal, then it is not necessary to record the transmit signal. The switch can either be removed or moved behind the preamp for additional protection of the receiver chain.

This work demonstrates a receiver that is inexpensive, frequency-agnostic, portable, flexible, and straightforward to integrate with a variety of systems. This demonstration provides evidence of the feasibility of integrating a low-cost, flexible, multi-channel receiver into established systems, which should encourage other labs to pursue further studies in multi-channel, second-nuclei research.

REFERENCES

[1] P. B. Roemer, W. A. Edelstein, C. E. Hayes, S. P. Souza, and O. M. Mueller, "The NMR phased array," *Magnetic Resonance in Medicine,* vol. 16, pp. 192-225, 1990.

[2] C. E. Hayes, N. Hattes, and P. B. Roemer, "Volume imaging with MR phased arrays," *Magnetic Resonance in Medicine,* vol. 18, pp. 309-319, 1991.

[3] S. M. Wright and L. L. Wald, "Theory and application of array coils in MR spectroscopy," *NMR in Biomedicine,* vol. 10, pp. 394-410, 1997.

[4] J. A. Bankson and S. M. Wright, "Multi-channel magnetic resonance spectroscopy through time domain multiplexing," *Magnetic Resonance Imaging,* vol. 19, pp. 1001-1008, 9// 2001.

[5] C. J. Hardy, P. A. Bottomley, K. W. Rohling, and P. B. Roemer, "An NMR phased array for human cardiac 31P spectroscopy," *Magnetic Resonance in Medicine,* vol. 28, pp. 54-64, 1992.

[6] F. M. Meise, J. Rivoire, M. Terekhov, G. C. Wiggins, B. Keil, S. Karpuk, *et al.*, "Design and evaluation of a 32-channel phased-array coil for lung imaging with hyperpolarized 3-helium," *Magnetic Resonance in Medicine,* vol. 63, pp. 456-464, 2010.

[7] A. Panda, S. Jones, H. Stark, R. S. Raghavan, K. Sandrasegaran, N. Bansal, *et al.*, "Phosphorus liver MRSI at 3 T using a novel dual-tuned eight-channel 31P/1H coil," *Magnetic Resonance in Medicine,* pp. n/a-n/a, 2012.

[8] F. Bloch, *The principle of nuclear induction*: Kungl. boktryckeriet PA Norstedt & söner, 1953.

[9] F. Bloch, W. W. Hansen, and M. Packard, "The nuclear induction experiment," *Physical Review,* vol. 70, pp. 474-485, 10/01/ 1946.

[10] E. M. Purcell, H. C. Torrey, and R. V. Pound, "Resonance absorption by nuclear magnetic moments in a solid," *Physical Review,* vol. 69, pp. 37-38, 01/01/ 1946.

[11] R. Oberhaensli, P. Bore, R. Rampling, D. Hilton-Jones, L. Hands, and G. Radda, "Biochemical investigation of human tumours in vivo with phosphorous-31 magnetic resonance spectroscopy," *The Lancet,* vol. 328, pp. 8-11, 7/5/ 1986.

[12] L. L. Cheng, M. A. Burns, J. L. Taylor, W. He, E. F. Halpern, W. S. McDougal, *et al.*, "Metabolic characterization of human prostate cancer with tissue magnetic resonance spectroscopy," *Cancer Research,* vol. 65, pp. 3030-3034, April 15, 2005 2005.

[13] R. F. Lee, R. Giaquinto, C. Constantinides, S. Souza, R. G. Weiss, and P. A. Bottomley, "A broadband phased-array system for direct phosphorus and sodium metabolic MRI on a clinical scanner," *Magnetic Resonance in Medicine,* vol. 43, pp. 269-277, 2000.

[14] B.-Y. Choe, T.-S. Suh, K.-S. Shinn, C.-W. Lee, C. Lee, and I.-H. Paik, "Observation of metabolic changes in chronic schizophrenia after neuroleptic treatment by In vivo hydrogen magnetic resonance spectroscopy," *Investigative Radiology*, vol. 31, pp. 345-352, 1996.

[15] C. Stork and P. F. Renshaw, "Mitochondrial dysfunction in bipolar disorder: evidence from magnetic resonance spectroscopy research," *Mol Psychiatry*, vol. 10, pp. 900-919, 07/12/online 2005.

[16] L. Gyulai, S. W. Wicklund, R. Greenstein, M. S. Bauer, P. Ciccione, P. C. Whybrow, *et al.*, "Measurement of tissue lithium concentration by lithium magnetic resonance spectroscopy in patients with bipolar disorder," *Biological Psychiatry*, vol. 29, pp. 1161-1170, 6/15/ 1991.

[17] P. F. Renshaw and S. Wicklund, "In vivo measurement of lithium in humans by nuclear magnetic resonance spectroscopy," *Biological Psychiatry*, vol. 23, pp. 465-475, 3/1/ 1988.

[18] N. Beckmann, J. J. Brocard, U. Keller, and J. Seelig, "Relationship between the degree of unsaturation of dietary fatty acids and adipose tissue fatty acids assessed by natural- abundance 13c magnetic resonance spectroscopy in man," *Magnetic Resonance in Medicine*, vol. 27, pp. 97-106, 1992.

[19] G. Perseghin, P. Scifo, F. De Cobelli, E. Pagliato, A. Battezzati, C. Arcelloni, *et al.*, "Intramyocellular triglyceride content is a determinant of in vivo insulin resistance in humans: a 1H-13C nuclear magnetic resonance spectroscopy assessment in offspring of type 2 diabetic parents," *Diabetes*, vol. 48, pp. 1600-1606, August 1, 1999 1999.

[20] G. I. Shulman, D. L. Rothman, T. Jue, P. Stein, R. A. DeFronzo, and R. G. Shulman, "Quantitation of muscle glycogen synthesis in normal subjects and subjects with non-insulin-dependent diabetes by 13C nuclear magnetic resonance spectroscopy," *New England Journal of Medicine*, vol. 322, pp. 223-228, 1990.

[21] S. E. Day, M. I. Kettunen, F. A. Gallagher, D.-E. Hu, M. Lerche, J. Wolber, *et al.*, "Detecting tumor response to treatment using hyperpolarized 13C magnetic resonance imaging and spectroscopy," *Nat Med*, vol. 13, pp. 1382-1387, 2007.

[22] K. Golman, R. I. t. Zandt, M. Lerche, R. Pehrson, and J. H. Ardenkjaer-Larsen, "Metabolic imaging by hyperpolarized 13C magnetic resonance imaging for in vivo tumor diagnosis," 2006.

[23] V. Tugnoli, M. R. Tosi, A. Tinti, A. Trinchero, G. Bottura, and G. Fini, "Characterization of lipids from human brain tissues by multinuclear magnetic resonance spectroscopy," *Biopolymers*, vol. 62, pp. 297-306, 2001.

[24] R. G. Weiss, P. A. Bottomley, C. J. Hardy, and G. Gerstenblith, "Regional myocardial metabolism of high-energy phosphates during isometric exercise in patients with coronary artery disease," *New England Journal of Medicine*, vol. 323, pp. 1593-1600, 1990.

[25] S. Schaefer, J. R. Gober, G. G. Schwartz, D. B. Twieg, M. W. Weiner, and B. Massie, "In vivo phosphorus-31 spectroscopic imaging in patients with global myocardial disease," *The American journal of cardiology*, vol. 65, pp. 1154-1161, 1990.

[26] W. Auffermann, W. Chew, C. Wolfe, N. Tavares, W. Parmley, R. Semelka, *et al.*, "Normal and diffusely abnormal myocardium in humans: functional and metabolic characterization with P-31 MR spectroscopy and cine MR imaging," *Radiology*, vol. 179, pp. 253-259, 1991.

[27] R. Subramanian and A. G. Webb, "Design of solenoidal microcoils for high-resolution 13C NMR spectroscopy," *Analytical Chemistry*, vol. 70, pp. 2454-2458, 1998/07/01 1998.

[28] R. A. De Graaf, *In vivo NMR spectroscopy : principles and techniques*: Chichester, West Sussex, England ; Hoboken, NJ : John Wiley & Sons, 2nd ed., 2007.

[29] L. L. Wald, S. E. Moyher, M. R. Day, S. J. Nelson, and D. B. Vigneron, "Proton spectroscopic imaging of the human brain using phased array detectors," *Magnetic Resonance in Medicine*, vol. 34, pp. 440-445, 1995.

[30] M. Schmitt, A. Potthast, D. E. Sosnovik, J. R. Polimeni, G. C. Wiggins, C. Triantafyllou, *et al.*, "A 128-channel receive-only cardiac coil for highly accelerated cardiac MRI at 3 Tesla," *Magnetic Resonance in Medicine*, vol. 59, pp. 1431-1439, 2008.

[31] C. J. Hardy, R. O. Giaquinto, J. E. Piel, K. W. Rohling Aas, L. Marinelli, D. J. Blezek, *et al.*, "128-channel body MRI with a flexible high-density receiver-coil array," *Journal of Magnetic Resonance Imaging*, vol. 28, pp. 1219-1225, 2008.

[32] N. I. Avdievich and H. P. Hetherington, "4T Actively detuneable double-tuned 1H/31P head volume coil and four-channel 31P phased array for human brain spectroscopy," *Journal of Magnetic Resonance*, vol. 186, pp. 341-346, 2007.

[33] Q. Xu, H. Wang, Z. Xu, and G. Li, "Frequency domain multiplexing for parallel acquisition of MR images," *Electronics Letters*, vol. 42, pp. 326-327, 2006.

[34] W. He, X. Qin, R. Jiejing, and L. Gengying, "Four-channel magnetic resonance imaging receiver using frequency domain multiplexing," *Review of Scientific Instruments*, vol. 78, pp. 015102-015102-6, 2007.

[35] J. Iannotti, S. Go, K. Dufel, B. Platt, and E. Fiveland, "Frequency division multiplex for phased array receive coils " in *International Society of Magnetic Resonance in Medicine*, Honolulu, HI, 2009, p. 4732.

[36] H. Chandrahalim and S. A. Bhave, "Digitally-tunable mems filter using mechanically-coupled resonator array," in *Micro Electro Mechanical Systems, 2008. MEMS 2008. IEEE 21st International Conference on*, 2008, pp. 1020-1023.

APPENDIX A

INSTRUCTIONS FOR USE OF THE FDM RECEIVER

Preparations for Use

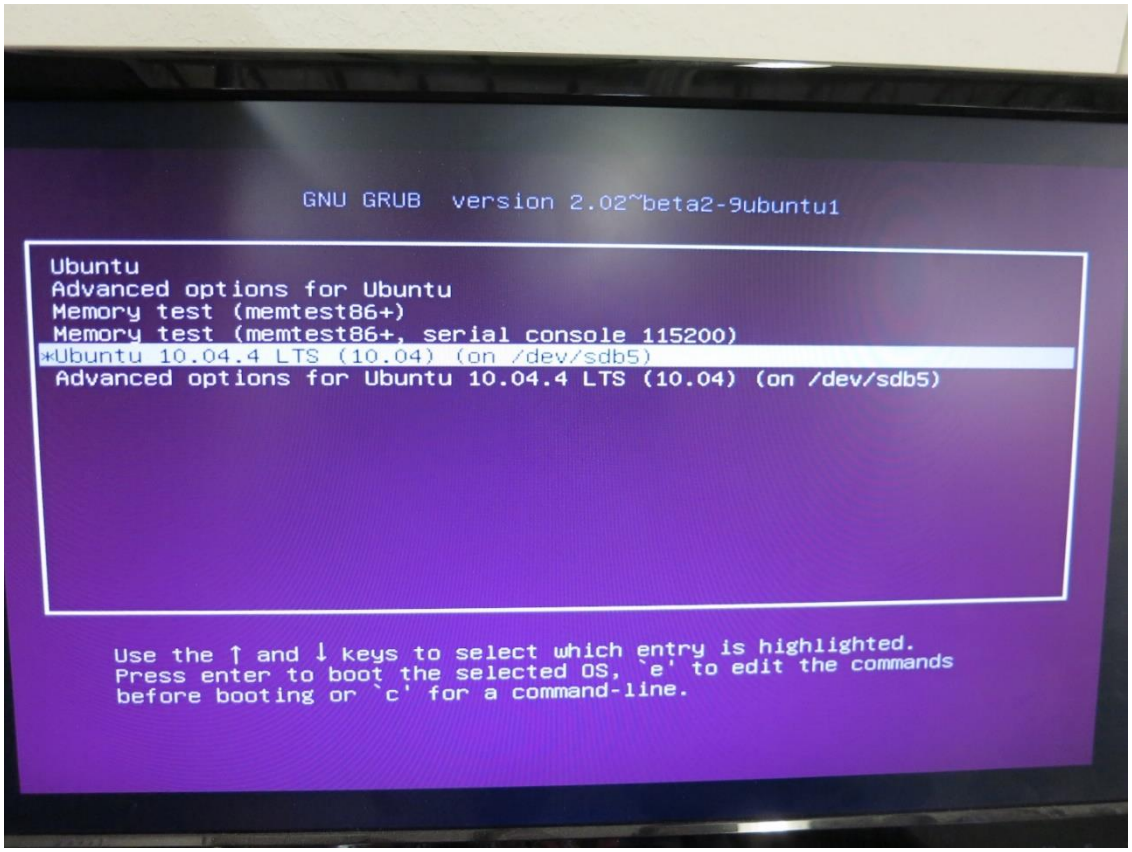
First, the digitizer card (ICS-1650) needs to be kept cool at all times to keep it working in a stable condition. This means that you need to make sure that the large fans that are on the top and bottom of the card are plugged in and turned on before or at the same time as the desktop is turned on.

Make sure that any ports on the receiver that are not being used are terminated in a 50 ohm load. The digitizer itself (ICS-1650 card) will be fine with open ports, but the rest of the receiver has many amplifiers that should not be left with an open input or output.

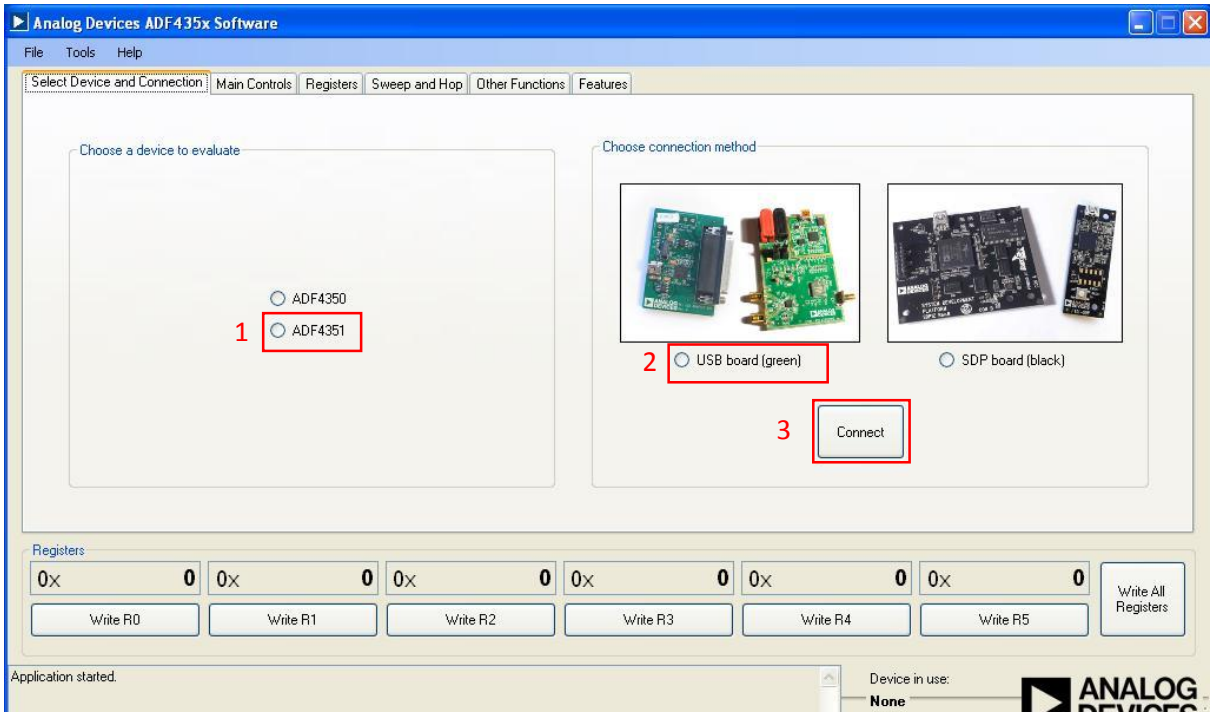
Turning the system on

You can now turn on the power supply for the desktop computer containing the digitizer by flipping the power supply switch in the back from ‘O’ to ‘I’. Next, power on the receiver by turning on the surge protector, followed by powering on the laptop and desktop. The desktop has two version of Ubuntu Linux on it.

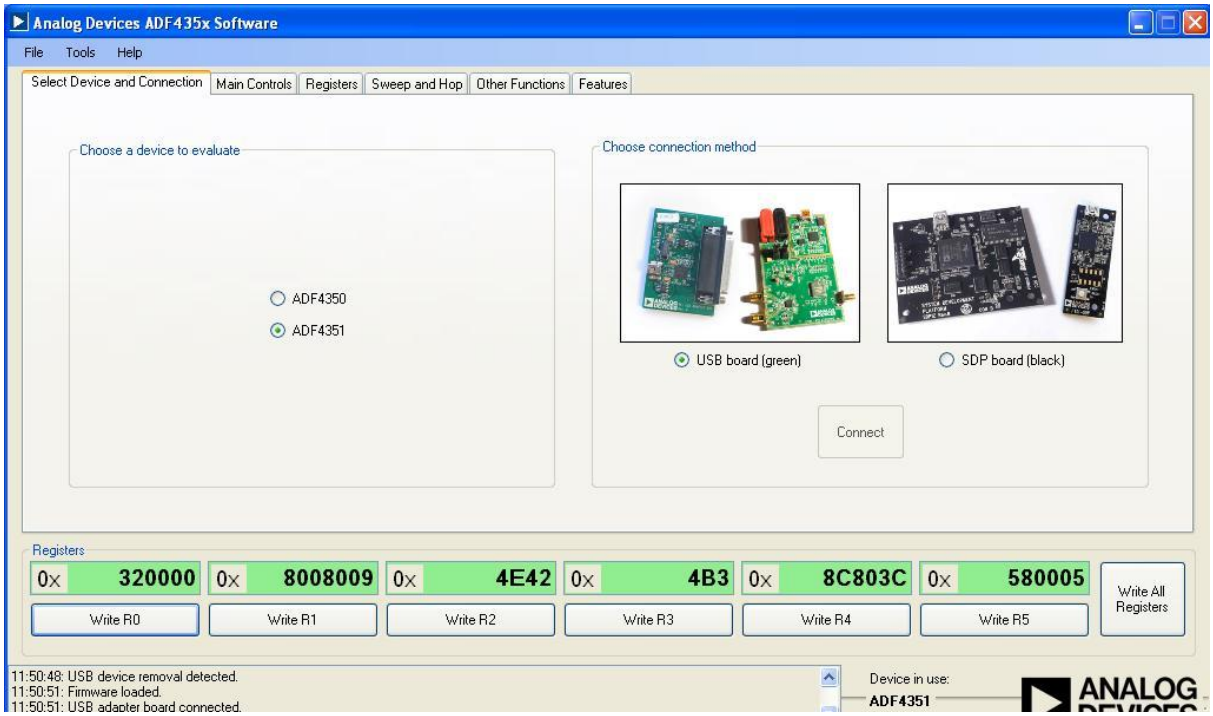
When the computer is booting, the screen will display a list of possible option on a maroon background. You want to scroll down with the “down arrow key” until you have “Ubuntu 10.04 LTS” highlighted and then press enter. If you don’t select the correct option quick enough, just reboot the computer using the onscreen menu and try again.



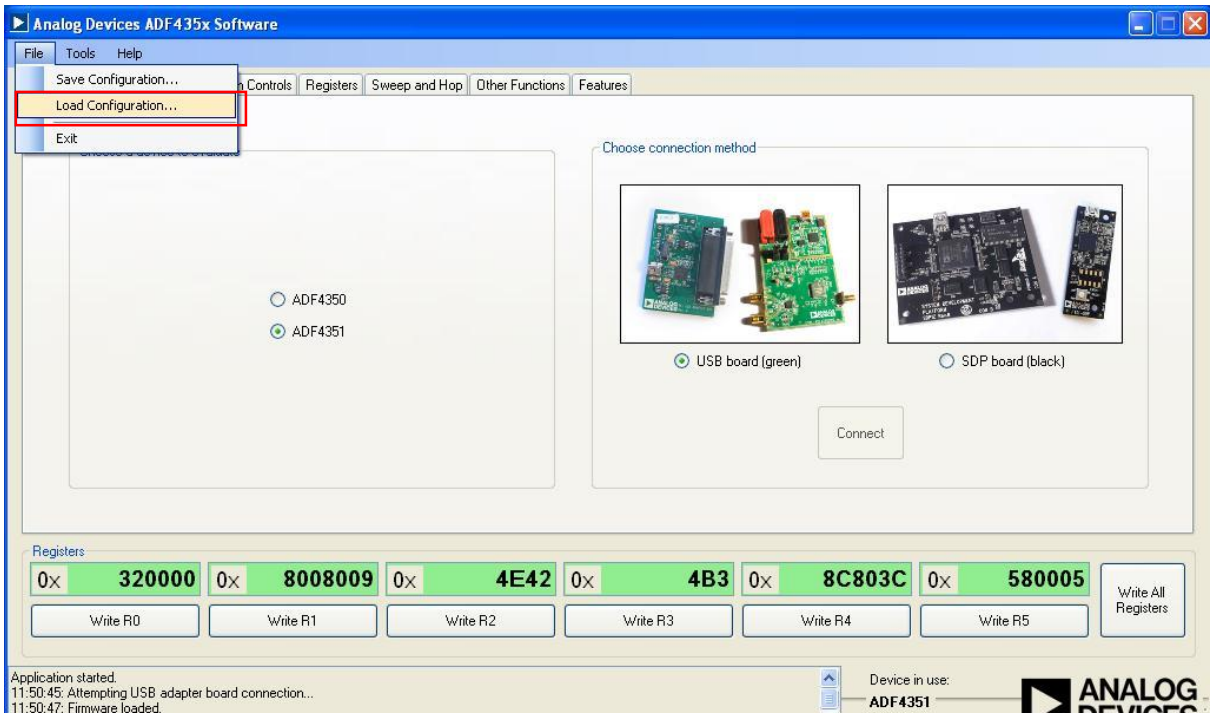
Make sure the two USB cables are plugged into the laptop. The USB cables are color coded with red and blue tape corresponding to the red and blue tape put over the USB ports on the laptop. After loading into Windows on the laptop and Ubuntu on the desktop, give each a few minutes to allow all the background processes to load. Once you have told both operating systems you don't want to upgrade, update or change anything, double click the AD4351 software icon on the desktop of the laptop.



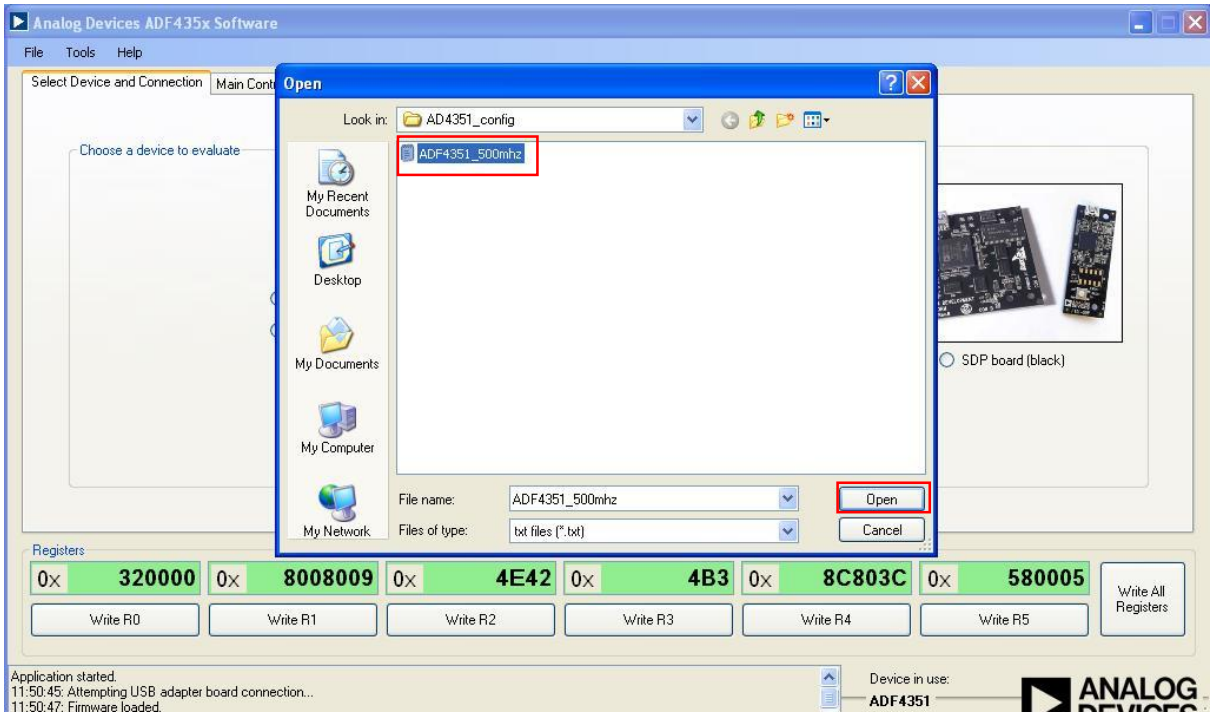
Now the AD4351 software is open as shown above. Select AD4351, green PCB, and click "Connect".



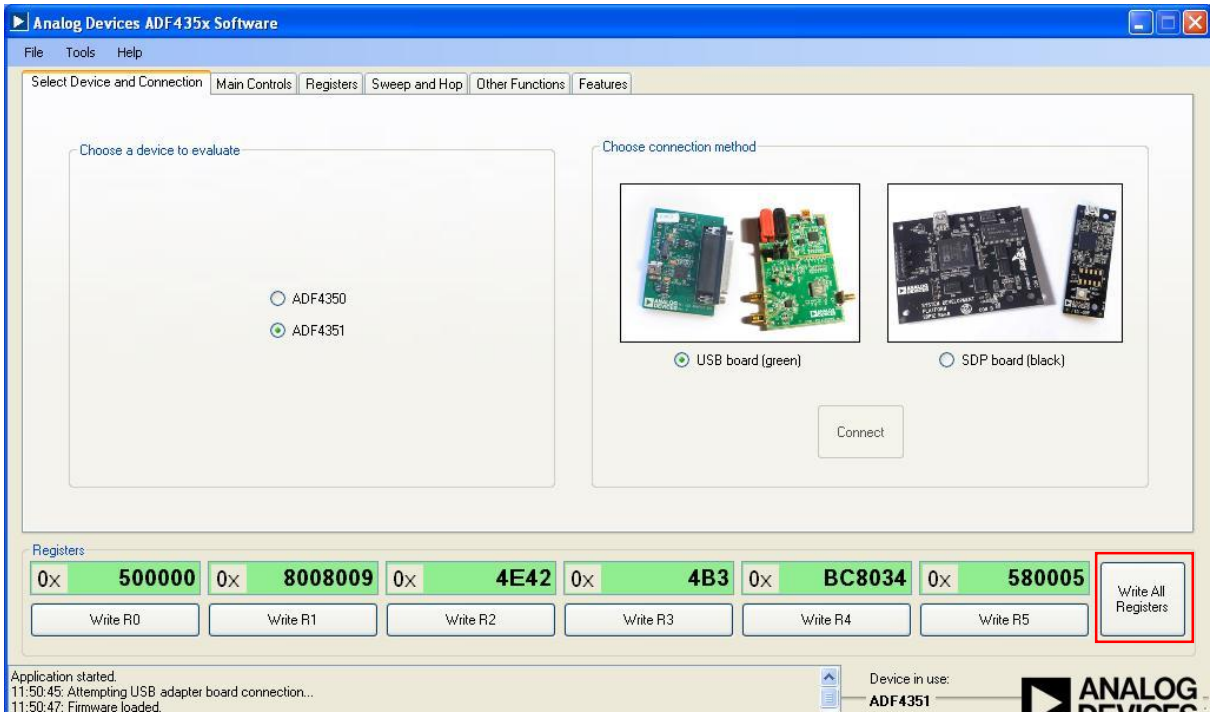
There will be a status bar that will let you know when it's done loading. At the bottom of the screen you can see "USB adapter board connected". All of the register boxes are green, meaning they have not been written yet. This is good because they are not the settings we want. Next we will load the configuration file to put in the correct settings.



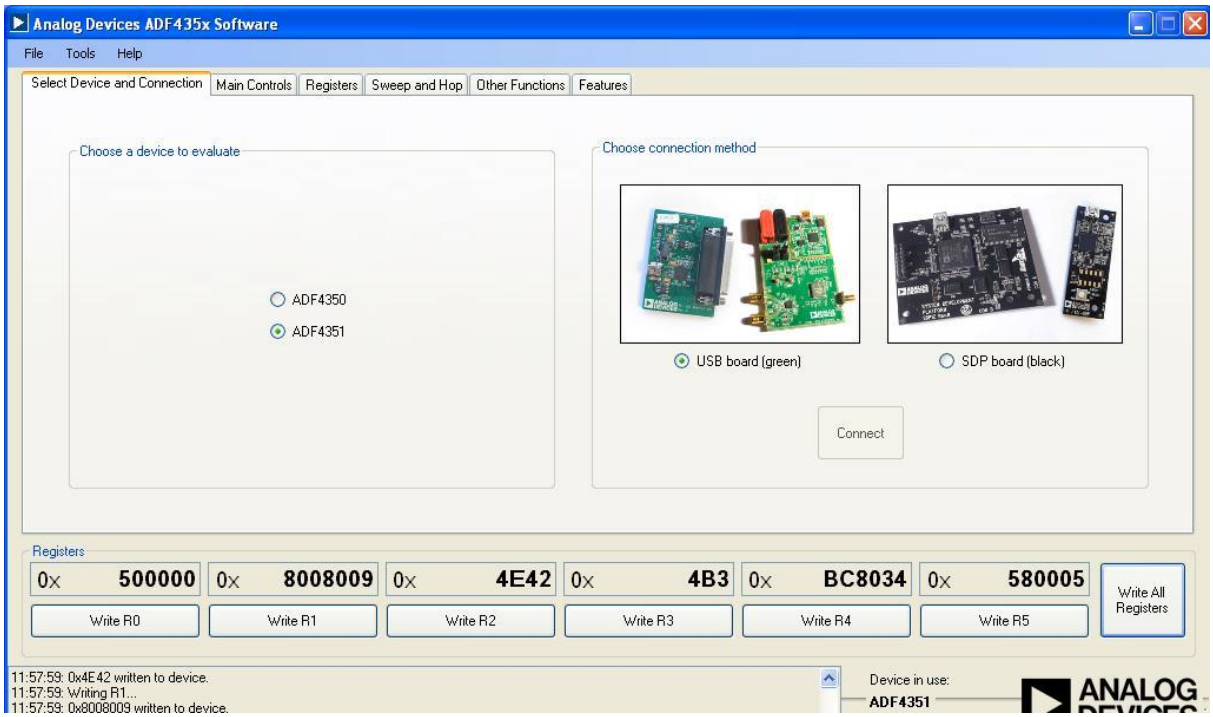
Click on "File" -> "Load Configuration" to find the configuration file.



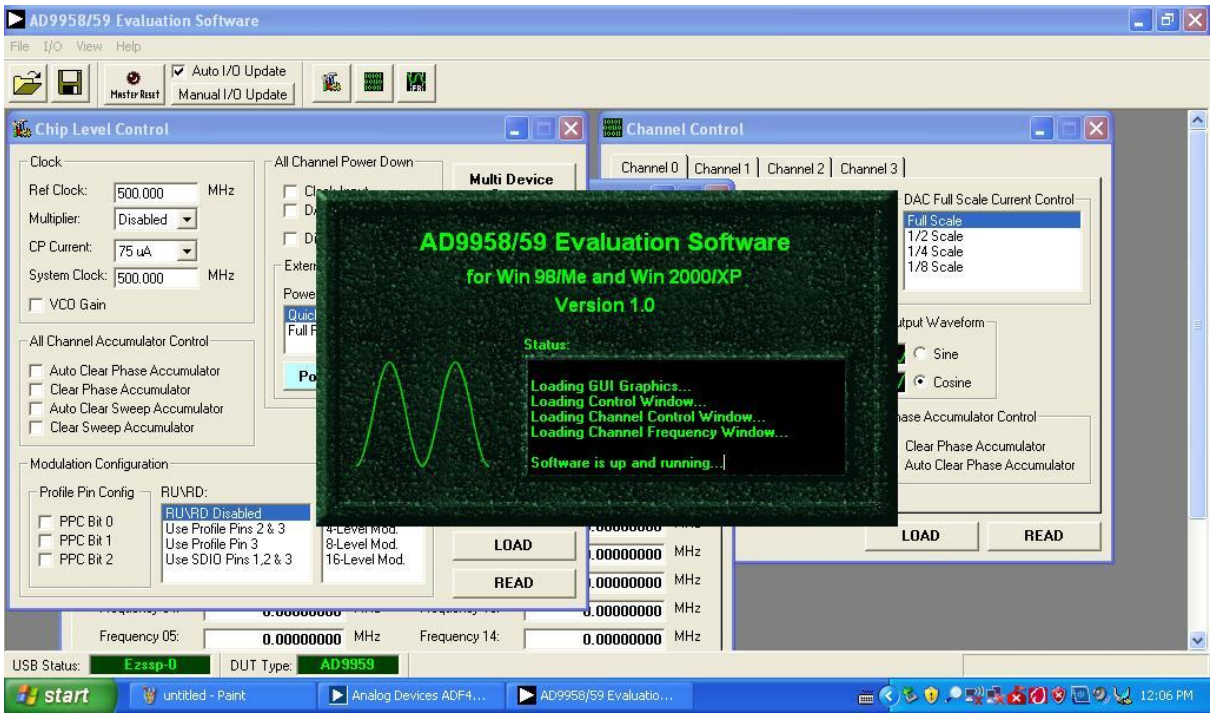
Click on the correct configuration file, ‘ADF4351_500mhz’, and then click open. You can create and save other configuration files if desired, but this should be the only one ever needed to run this receiver.



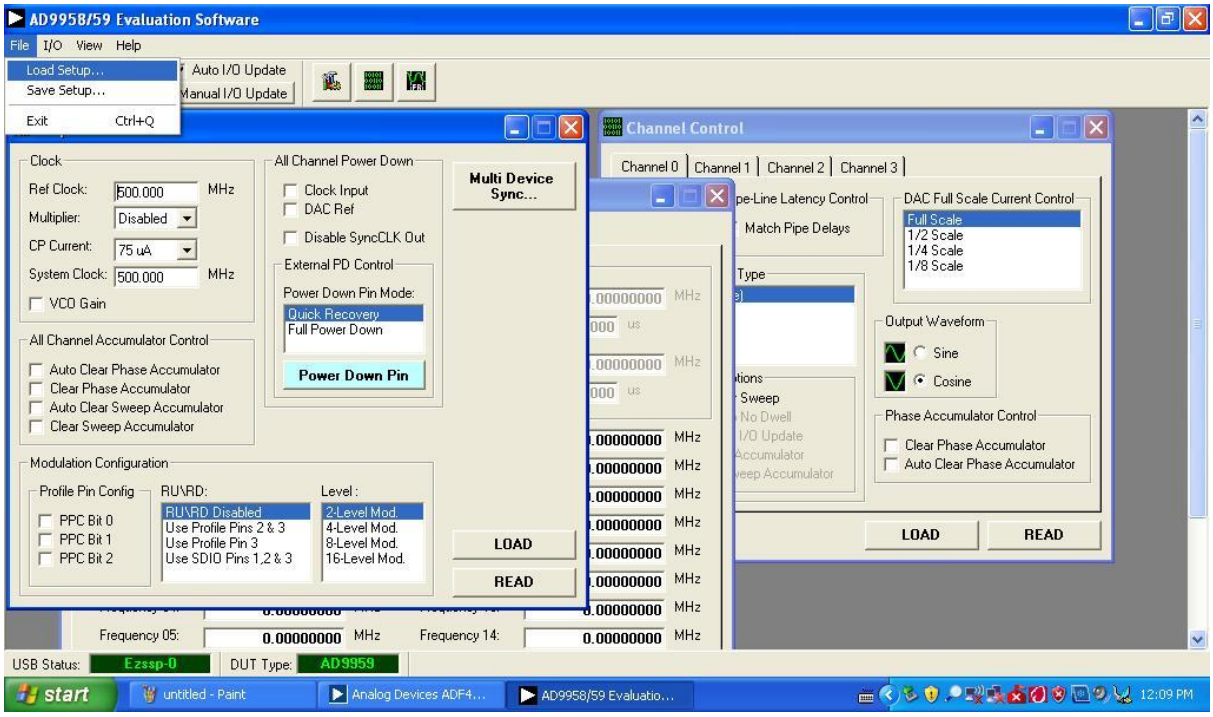
It can be seen that R0 and R4 have changed from the defaults. Now click “Write All Registers”.



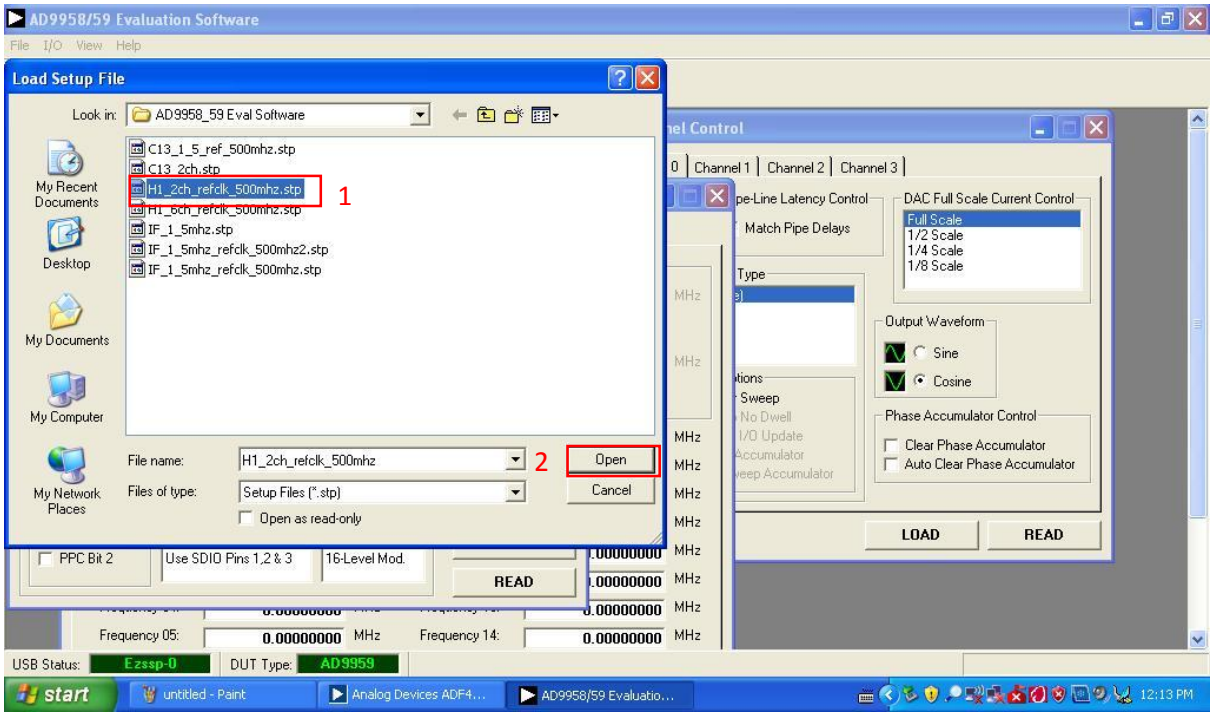
After clicking “Write All Registers,” none of the registers boxes are green anymore and the card should be outputting 500 MHz to the AD9959 LO board. Now the AD9959 LO board needs to be configured. We are done with the AD4351 software for now, so minimize the AD4351 software and double click the AD9959 icon on the desktop to begin.



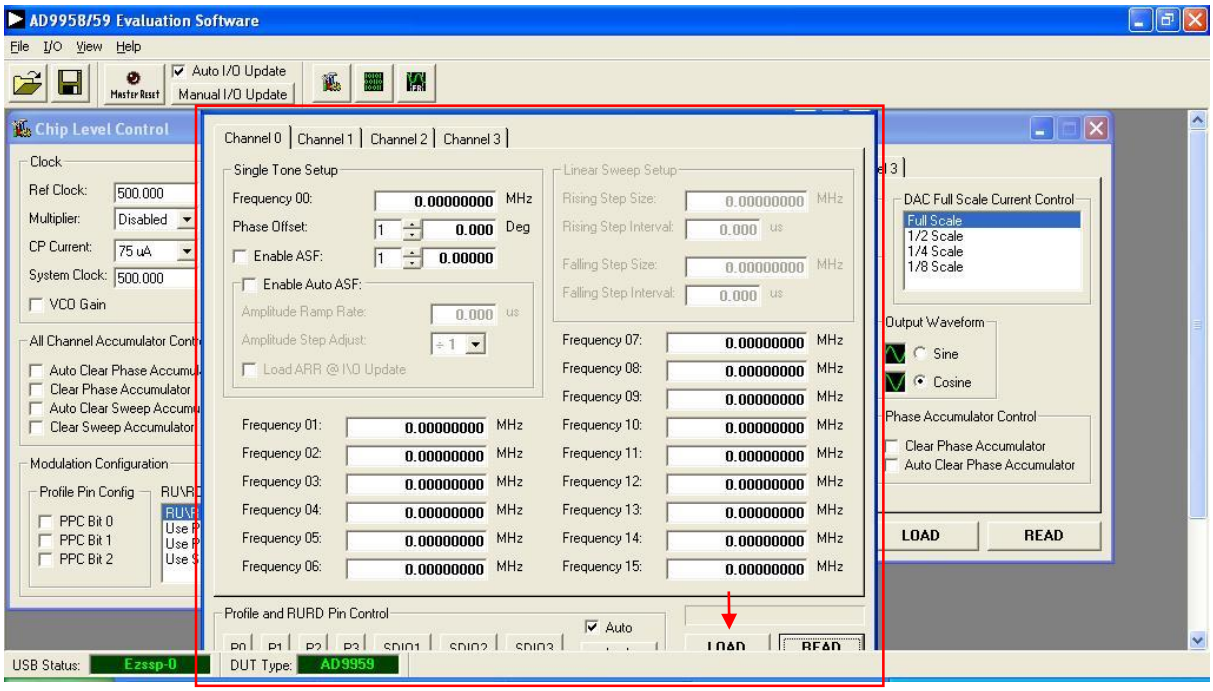
The AD9959 software should have opened and look like the above image. The splash screen will continue to show until you click outside of it. If you have forgotten to turn on the AD4351, or if the AD4351 is malfunctioning, there will be an error on the splash screen in red letting you know.



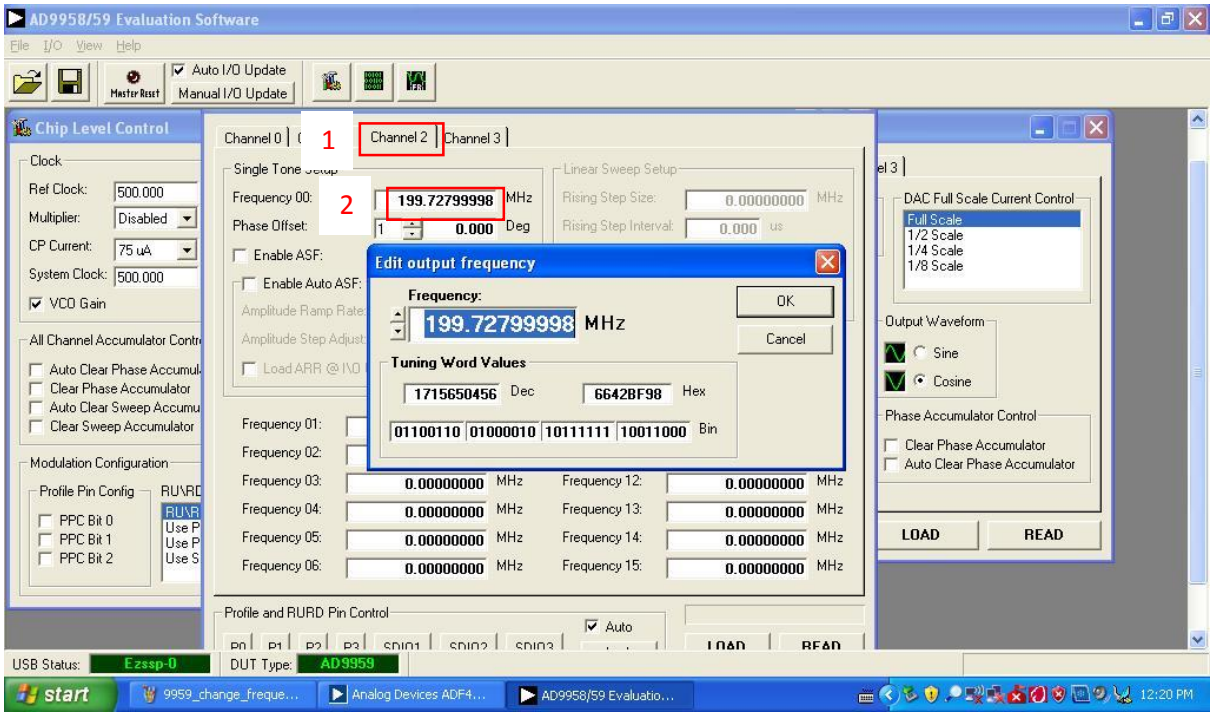
Next click ‘File’ → ‘Load Setup’. If you change parameters and wish to save your configuration file, click ‘File’ → ‘Save Setup’.



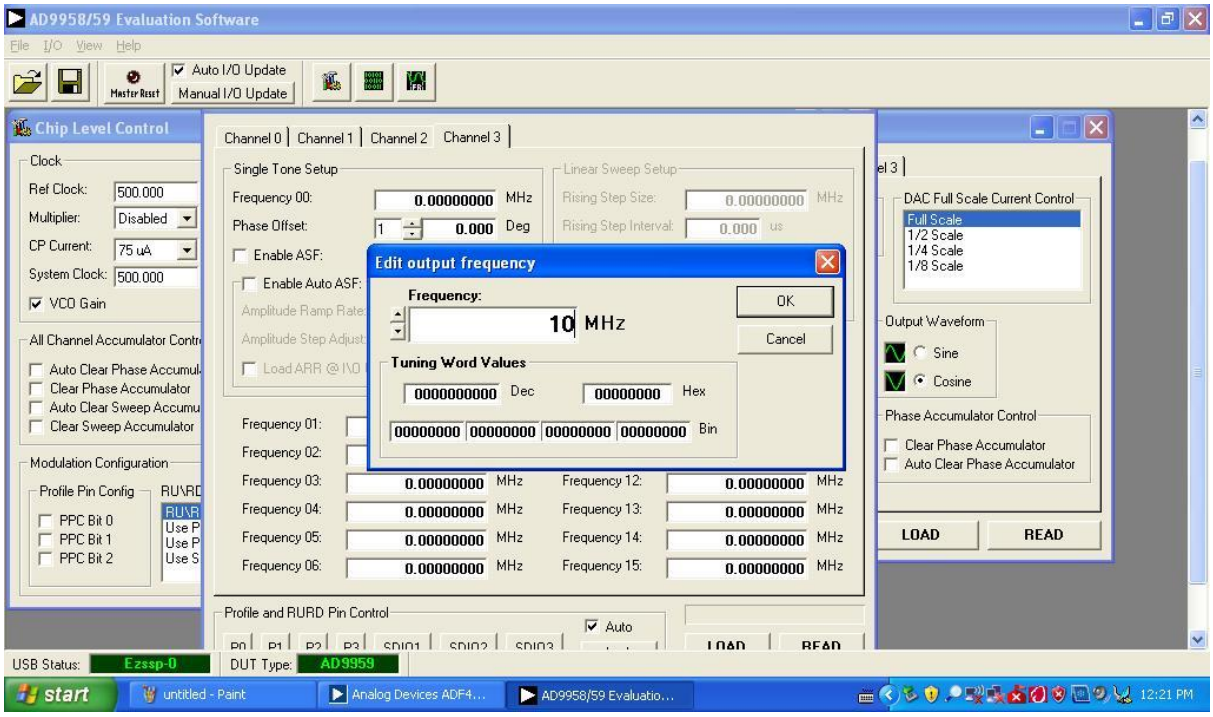
Select the configuration file you wish to load. The AD9959 (four channel local oscillator) can have a different file for every different application you have for it. It is also easy to configure on the fly. For a two channel hydrogen acquisition experiment, I choose ‘H1_2ch_refclk_500mhz’.



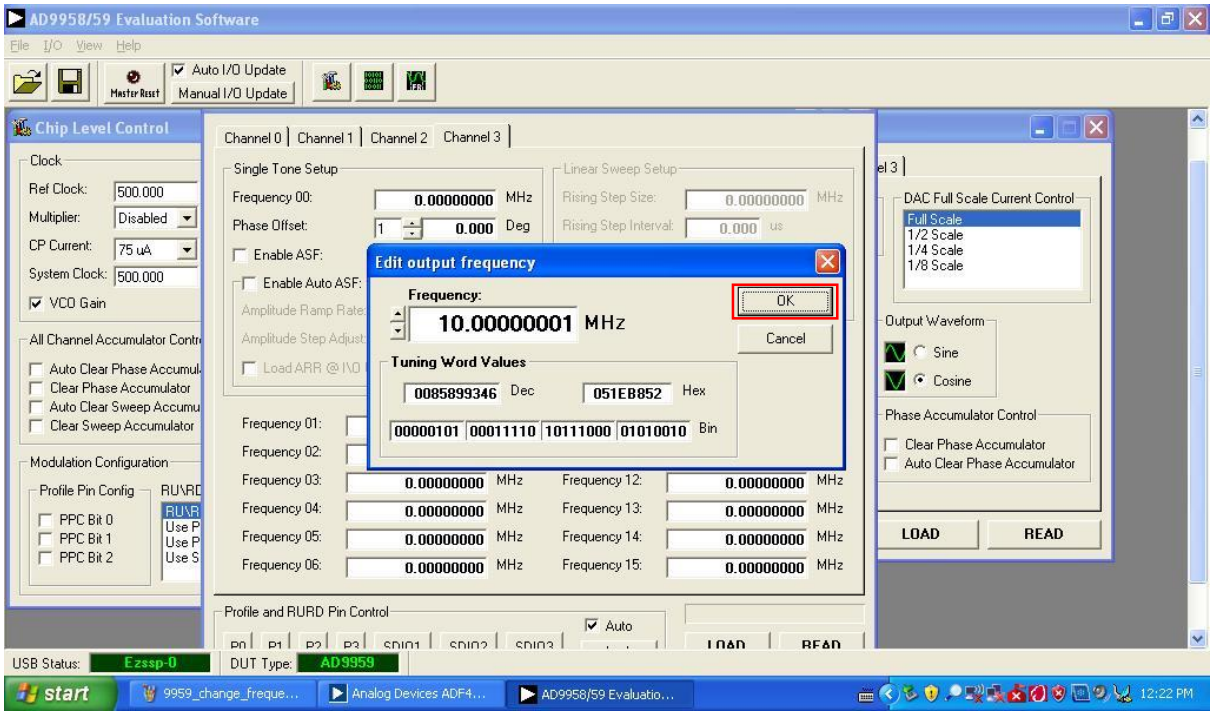
Now that the configuration file is loaded, we can make changes to the file. Click on the middle window and drag it up as far as the program will let you. This allows you to see the “LOAD” button at the bottom of the window.



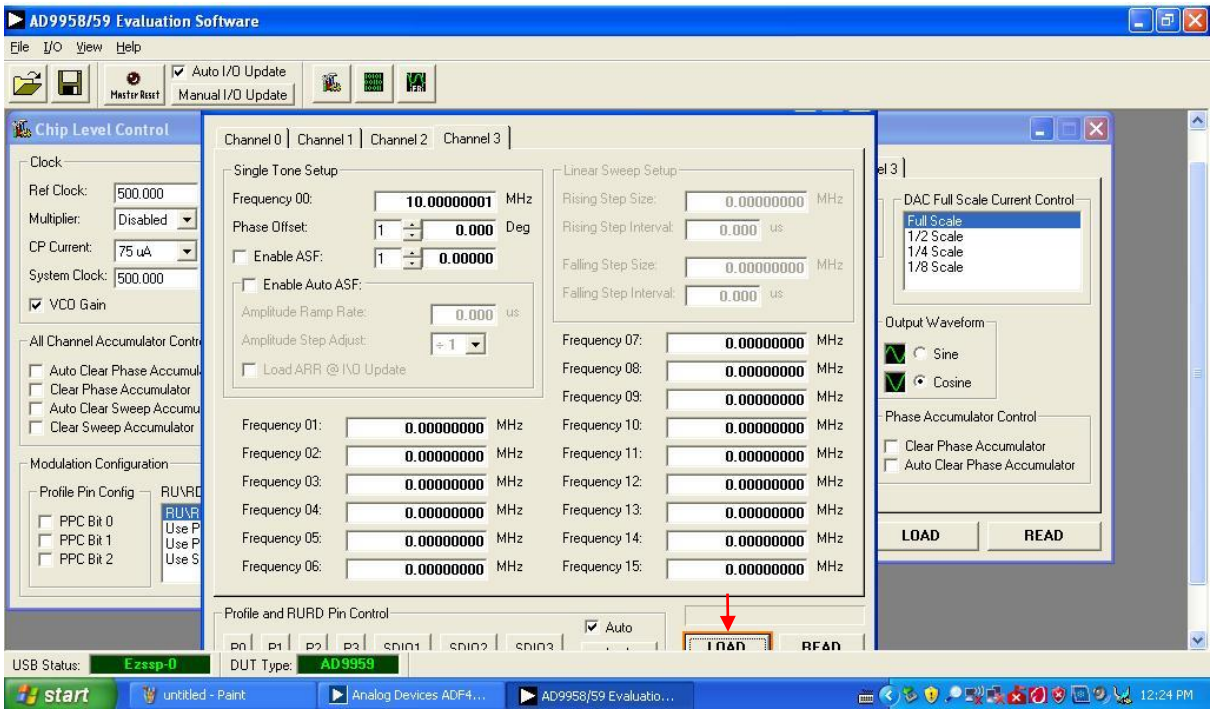
First click on the tab pertaining to the channel you want to change. In this case I'm changing the frequency of Channel 2. Next double click the "Frequency" box in order to be able to change it.



Enter the frequency you wish to use in MHz.

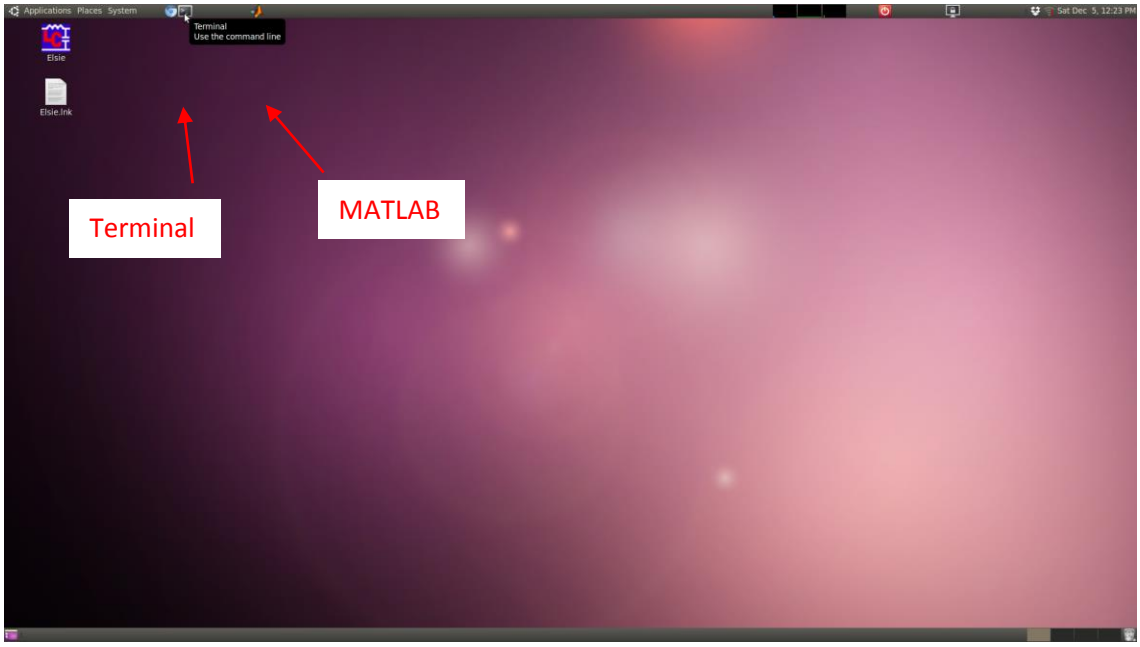


The frequency that can be used is limited by the number of bits that are used to represent the number. In this case, when you press ‘enter’, the ’10 MHz’ you wanted changes to the nearest value that can be represented by the card. Finally, click ‘OK’.

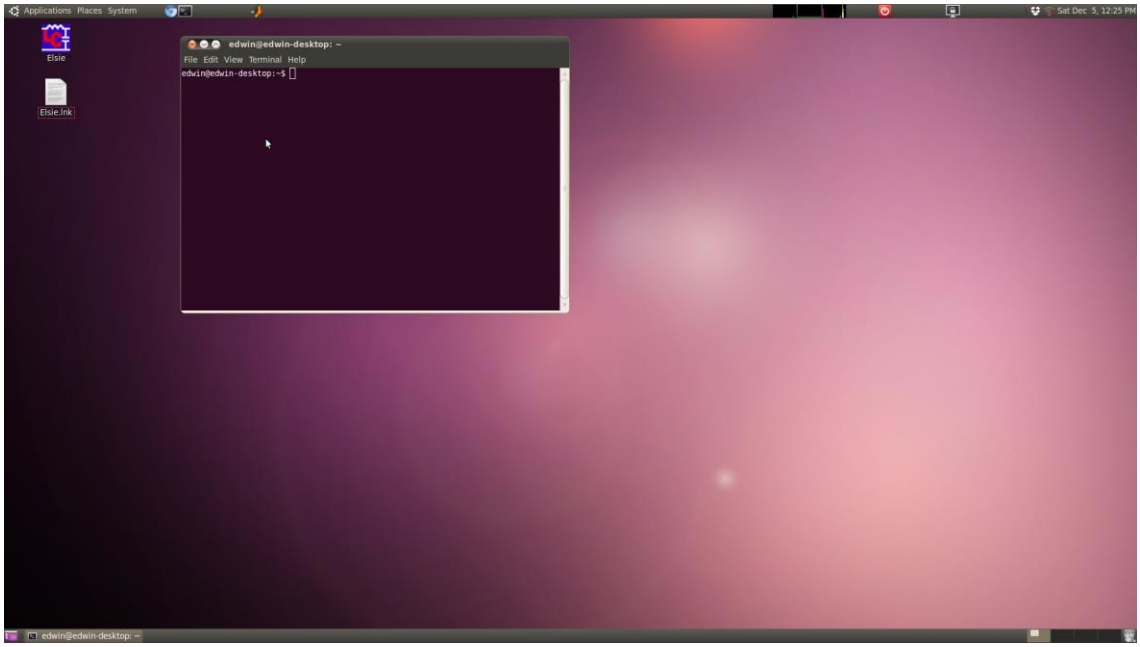


After you click ‘OK’ the ‘LOAD’ button has a flashing orange border, letting you know that you have changed parameters in the software that have not yet been written to the card. Click ‘Load’ and the card will update to the new settings. Now the LO sources are completely set up.

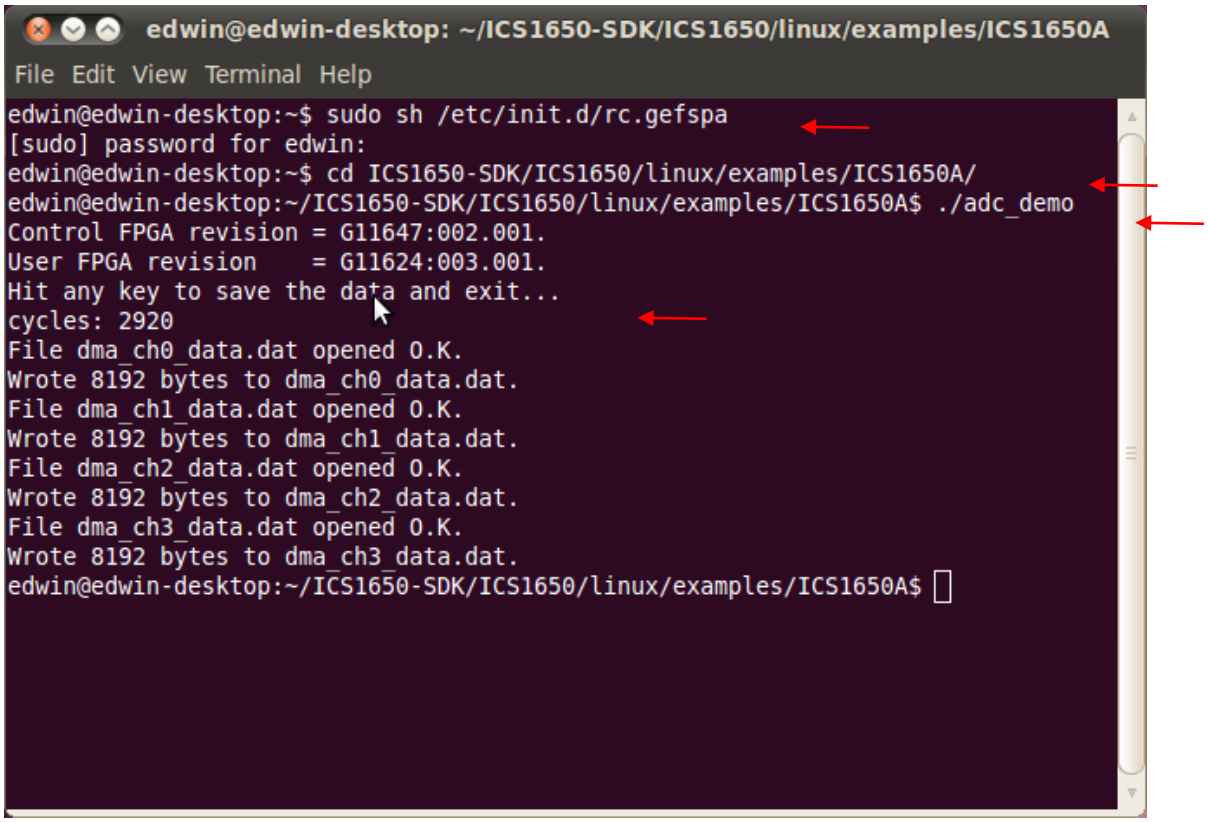
The Desktop Control of the Digitizer



There are two programs that we will use here: the terminal and MATLAB. First, use the terminal to set up the digitizer.



When the terminal opens, there is a blinking prompt allowing you to type commands.



The above image is after a few of the set up commands. Each interaction required by the user is marked by a red arrow. The first command is ‘sudo sh /etc/init.d/rc.gefspa’. After you type a command, press ‘Enter’. The administrator (super user) password for this account is ‘haha’. Let’s break down the commands. The ‘sudo’ portion is telling the computer that you are a ‘super user’ and to ‘do’ what you say. It will require a password because you are telling the computer to change settings and it wants to make sure that you really are a super user. The ‘sh’ is telling the computer that you will be running a shell script. A shell script is a text document with a list of commands for the computer. In this case, the shell script is setting up the computer to be able to work with the digitizer card. You have to do this after every shutdown or restart. Finally, the ‘/etc/init.d/rc.gefspa’ is pointing to where the shell script is. The shell script resides in the file ‘/etc/init.d’ and is named ‘rc.gefspa’.

The next command in the above prompt begins with ‘cd,’ which changes the working directory of the terminal. It’s just like in Windows when you’re looking in one directory and change to another. The files and executables that you can open are in the current working directory. Now that we’re in the ‘examples’ folder inside the digitizer software development kit, we can run the provided demo program to make sure the card is working correctly. The command ‘./adc_demo’ will start the demo program and pressing enter will end it. If at any point the demo program hangs, or there is some other oddity, pressing ‘CTRL + c’ will terminate the running program in the terminal (just like in MATLAB). This ‘adc_demo’ program is simply a test and is only used to make sure the digitizer is operating correctly. Typing ‘./adc_demo_test’ will start the digitizer in the mode we want for the receiver.

Many of the imaging parameters that you will want to change will be adjusted in the adc_demo_test.c file. When you make changes to adc_demo_test.c, you need to navigate to

the folder in the terminal and type “make”. This will compile the new adc_demo_test program and it will be ready to use. The terminal will display a message with errors if there was something wrong with the syntax of the c file. The main four variables that need to be changed in adc_demo_test.c are the sampling rate, the decimation rate, the number of samples taken, and the number of channels used to sample data.

The sampling rate controls how fast the clock on the ICS-1650 runs and how fast it takes samples. It cannot be set to a sampling rate less than 80 MHz. To get to a lower sampling frequency than 80 MHz, use decimation. If the sampling rate is set to 100 MHz and the decimation variable is set to 2, then the resulting equivalent sampling rate would be 50 MHz. A sampling rate of 50 MHz is recommended when using all six channels since the highest IF used is 5 MHz and 50 MHz oversamples 5 MHz without taking too much unnecessary data. When using less channels and just using the .5 and 1 MHz IF frequencies, lower sampling rates can be used to acquire longer samples without filling the card up completely. The number of samples you want to take can be determined by the sampling time multiplied by the effective sampling rate (sampling frequency divided by decimation). For instance if you are sampling at 10 MHz and want to sample for .5 seconds, you would need to take 5e6 samples.

The onboard memory of the card constrains the maximum number of samples that can be taken in one acquisition. The maximum number of samples that can be taken in one acquisition (from all channels combined) is 16e6 samples. Two of the digitizer channels on the ICS-1650 can acquire 8e6 samples each per acquisition or one digitizer channel on the ICS-1650 can acquire 16e6 samples. This becomes important when doing spectroscopy if you want to take much longer acquisitions. The memory cap on the card should not be a problem during normal spin echo or gradient echo sequences. The total size of the data file produced increases

with the number of samples and the number of acquisitions. Once the total file size of your acquired data increases beyond 2 GB, the file can no longer be easily moved off the computer onto a flash drive because of the formatting of the operating system. Again, this should not be a problem in most situations.

The number of digitizer channels used on the ICS-1650 can be changed from one to four. When only sampling with one channel, digitizer channel 1 (or CH0 in software) is used. When two digitizer channels are used, channel 1 and channel 2 are used. You cannot select which digitizer channels you wish to use, only the number that you wish to use. When using the full six receiver channels provided, two digitizer channels on the ICS-1650 are used (channels 1 and 2).

After each acquisition, the data from the card is written to a hard drive. At the beginning of the sequence, a binary file is created and is appended after every acquisition. If for some reason the card fails to trigger on an acquisition and the acquisition is not complete, there will still be many samples that are saved to the hard drive and you will need to run the "rm_data" MATLAB script. If the data acquisition was successful, you can view the raw data using the "read_data" script. To save the data use the script "rename_files". This script renames the binary file that is produced by the data acquisition executable.

When using the 'read_file' script, there are several parameters that need to match up with the acquisition. The number of samples that are acquired each acquisition needs to be set to the value used by the digitizer as well as the sampling frequency, 'Fs'. Finally, the center frequency, 'Fc', needs to be set to the IF frequency that you wish to view. The 'read_file' script will only display the raw, unaltered, time domain data as an indicator of the signal levels for each IF and the quality of the received time domain signal.

Compiling the Driver

GE provides a software developer kit (SDK) to the user to allow for the creation of code that can control the digitizer card. The SDK must first be compiled on the system, allowing use of the driver and the other functions that are shared by any executable that uses the card. The SDK has already been compiled on the current system and so the following instructions would only need to be followed in the event of a reinstall of the operating system or in the event of some kind of data loss.

All of the set up tasks such as installing the driver or compiling the necessary shared functions are taken care of through one shell script, 'install.sh'. To run this, open the terminal (as shown earlier), navigate to the folder containing the 'install.sh' shell script and run the command 'sudo sh install.sh'. The 'install.sh' file resides in the top level of the ICS1650-SDK folder. When you run this shell script, it installs the driver on the computer, sets up the 'rc.gefspa' script, compiles the executables necessary for running the digitizer as well as compiling a demo program for high level digitizer control. The digitizer cannot be installed on a computer running a Linux kernel later than version 2.6.31. This means that if Ubuntu Linux is being used only Ubuntu Linux version 10.04 or older can be used.

When the 'install.sh' shell script has been run in the terminal and the 'rc.gefspa' shell script has been run to initialize the digitizer, the digitizer can be used. If the digitizer is set up to accept an external trigger and is waiting for a trigger, it will not wait more than 5 seconds before triggering. This is because there is a hardcoded maximum wait time of 5 seconds built in. To get around this, a variable must be changed in the source code for the SDK. Another variable for timing should also be checked in the demo program itself. The first variable is

‘u32Timeout’ and the second is DEFAULT_TIMEOUT. Both of these variables have the potential to prevent the digitizer from operating in an expected manner.

The ‘u32Timeout’ variable can be found in the ‘ics1650ReadFile’ function. The function is defined in the ‘ics1650FileIoApi.h’ header file as seen in the following line of code:

```
ICS1650_INT32_T EXPORT ics1650ReadFile (ICS1650_HANDLE_T hBoard, ICS1650_UINT32_T dmaIdx, ICS1650_VOID_T* pMemory, ICS1650_LONG_T numBytes, ICS1650_ULONG_T* bytesRead, ICS1650_UINT32_T timeout);
```

The last variable defined in the function, ‘ICS1650_UINT32_T timeout’, is the variable of interest. The ‘ics1650ReadFile’ command is used in the demo program as well as in the receiver program to tell the card how much data to pull off of the card’s memory and where to put it in the RAM of the control computer. However, if the value that has been input for ‘timeout’ is hit before the digitizer finishes pulling the data off of the card, the acquisition will be aborted. The value of ‘u32timeout’ is defined in milliseconds and a value of 1000 should be sufficient. Because the PCI-e bus is so fast, the data transfer should be incredibly fast and 1000ms should have two orders of magnitude in buffer built in.

The next timeout variable that needs to be changed is the ‘DEFAULT_TIMEOUT’ that resides in the ‘drvbrh.h’ header file. The line of code is shown here:

```
#define DEFAULT_TIMEOUT 5000
```

The ‘DEFAULT_TIMEOUT’ variable is defined in milliseconds and is set to 5000 milliseconds by default. If the digitizer card is looking for a trigger and the trigger is not seen within the time window allotted by ‘DEFAULT_TIMEOUT’, then the card will trigger itself. Once the value of ‘DEFAULT_TIMEOUT’ has been changed to a more reasonable number, the ‘install.sh’ script must be run again to recompile the SDK. Until the SDK is recompiled, the card will continue to use the old ‘DEFAULT_TIMEOUT’.

APPENDIX B

SOURCE CODE

Digitizer Control Software

```
#include <stdio.h>
#include <time.h>
#include "ics1650FunctionalApi.h"

#if defined(WIN32) || defined(_WIN64)
#include <conio.h>
#endif

#define NUMBER_OF_ADC_CHANNELS 1
#define CONTINUOUS_BUFFER_SIZE_IN_SAMPLES 300000

void write_to_file (ICS1650_INT16_T * buffer, ICS1650_ULONG_T buffer_size, char * filename);

#if defined(LINUX)
#include <fcntl.h>
#include <sys/types.h>
#include <sys/poll.h>
#endif

ICS1650_HANDLE_T  hDevice = ICS1650_INVALID_HANDLE_VALUE;
ICS1650_INT16_T **ptrBuffer = NULL;
int main(int argc, char ** argv)
{

  int num_acq;
```

```
printf("Please enter the number of acquisitions: ");
int num=scanf("%d", &num_acq);
printf("You entered: %d acquisitions\n", num_acq);


char filename[80] = "dma_ch0_data.bin";
ICS1650_ULONG_T u32cycles=0, u32NumBytesToRead, u32NumBytesRead=0;
ICS1650_INT32_T u32Timeout=1000, i32ChannelIdx;
ICS1650_INT32_T i = 0;
ICS1650_INT32_T i32NumSamplesToAquire = CONTINUOUS_BUFFER_SIZE_IN_SAMPLES;

ICS1650_ADC           adc[NUMBER_OF_ADC_CHANNELS];
ICS1650_CLOCK         clocking;
ICS1650_TRIGGER       triggerConfig;
ICS1650_TRANSFER      acq[NUMBER_OF_ADC_CHANNELS];
ICS1650_CORES_REVISIONS coresRevision;
ICS1650_SYNCHRONIZATION synchronizationConfig;

memset(&clocking, 0, sizeof(ICS1650_CLOCK));
memset(&triggerConfig, 0, sizeof(ICS1650_TRIGGER));
memset(&synchronizationConfig, 0, sizeof(ICS1650_SYNCHRONIZATION));
for(i=0;i<NUMBER_OF_ADC_CHANNELS;++i){
  memset(&acq[i], 0, sizeof(ICS1650_TRANSFER));
  memset(&adc[i], 0, sizeof(ICS1650_ADC));
}

/* Setup configurable parameters */
clocking.dClockFrequency            = 80.0; /* MHz */
clocking.dReferenceFrequency        = 10.0;  /* MHz */
```

```
clocking.u32ClockDividerADC          = F_ICS1650_CLOCK_DIVIDE_BY_1;
clocking.u32ClockSelect              = F_ICS1650_INTERNAL;
clocking.u32ReferenceSelect          = F_ICS1650_EXTERNAL;
clocking.u32ClockConnectorMode       = F_ICS1650_INTERNAL_CLOCK_OUTPUT;
clocking.u32ClockDividerInternalOutput = F_ICS1650_CLOCK_DIVIDE_BY_2;

triggerConfig.u32Select              = F_ICS1650_EXTERNAL;
triggerConfig.u32ConnectorMode       = F_ICS1650_EXTERNAL_TRIGGER_INPUT;
triggerConfig.u32ExternalMode        = F_ICS1650_TRIGGER_MODE_RISING_EDGE;

synchronizationConfig.u32Select      = F_ICS1650_INTERNAL;
synchronizationConfig.u32ConnectorMode = F_ICS1650_INTERNAL_SYNCHRONIZATION_OUTPUT;
synchronizationConfig.u32ExternalMode  = F_ICS1650_SYNCHRONIZATION_MODE_RISING_EDGE;

/* Open Device Handle */
if (ICS1650_OK != ics1650CreateFile (&hDevice, 0)) {
  printf ("ics1650CreateFile failed!\n");
  goto ErrorExit;
}
else {
  printf ("ics1650CreateFile success!\n");
}

/* Get Cores Revision */
memset(&coresRevision, 0, sizeof(ICS1650_CORES_REVISIONS));
if (ICS1650_ERROR == ics1650CoresRevisionGet (hDevice, &coresRevision)) {
  printf("ics1650CoresRevisionGet failed. \n");
  goto ErrorExit;
}
else {
```

```
  printf ("ics1650CoresRevisionGet success!\n");
}
printf("Control FPGA revision = %s.\n",coresRevision.cptrControlFPGARevision);
printf("User FPGA revision    = %s.\n",coresRevision.cptrUserFPGARevision);

/* Board Reset */
if(ICS1650_OK != ics1650BoardInitialize(hDevice)) {
  printf("ics1650BoardInitialize failed.\n");
  goto ErrorExit;
}
else {
  printf ("ics1650BoardInitialize success!\n");
}

/* Set ADC Clock */
if(ICS1650_OK != ics1650ClockSet(hDevice, &clocking)) {
  printf("ics1650ClockSet failed.\n");
  goto ErrorExit;
}
else {
  printf ("ics1650ClockSet success!\n");
}

/* Set Trigger Control. */
if(ICS1650_OK != ics1650TriggerSet(hDevice, &triggerConfig)) {
  printf("ics1650TriggerSet failed.\n");
  goto ErrorExit;
}
else {
  printf ("ics1650TriggerSet success!\n");
```

```
}

/* Configure ADC Front End */
memset(adc, 0, NUMBER_OF_ADC_CHANNELS*sizeof(ICS1650_ADC) );
for(i32ChannelIdx=0; i32ChannelIdx<NUMBER_OF_ADC_CHANNELS; i32ChannelIdx++) {
 adc[i32ChannelIdx].u32SampleType    = F_ICS1650_ADC_SAMPLE_TYPE_ADC;
 adc[i32ChannelIdx].u32SampleFormat  = F_ICS1650_ADC_FORMAT_LSB;
 ICS1650_INPUT_VOLTAGE_RANGE(1, adc[i32ChannelIdx].u32VoltageRange); /* 1 V */
 adc[i32ChannelIdx].u32Decimation    = 8;   /* Decimation */
 adc[i32ChannelIdx].dGain            = 1.0;
 //printf("adc[i32ChannelIdx].u32VoltageRange = 0x%x\n", adc[i32ChannelIdx].u32VoltageRange);
}

for (i32ChannelIdx=0; i32ChannelIdx<NUMBER_OF_ADC_CHANNELS; i32ChannelIdx++) {
 if (ICS1650_OK != ics1650ADCSet(hDevice, i32ChannelIdx, &adc[i32ChannelIdx])) {
  printf("ics1650ADCSet failed.\n");
  goto ErrorExit;
 }
        else {
  printf ("ics1650BoardInitialize success!\n");
 }
}

/* Configure and Enable Transfer */
memset(acq, 0, NUMBER_OF_ADC_CHANNELS*sizeof(ICS1650_TRANSFER) );
for(i32ChannelIdx=0; i32ChannelIdx<NUMBER_OF_ADC_CHANNELS; i32ChannelIdx++) {
 acq[i32ChannelIdx].u32SampleCount = i32NumSamplesToAquire;
 acq[i32ChannelIdx].u32DataType    = F_ICS1650_DATA_ADC;
 acq[i32ChannelIdx].u32SelectADC   = i32ChannelIdx;
 acq[i32ChannelIdx].u32EnableQDR   = F_ICS1650_ENABLE;
```

```
  acq[i32ChannelIdx].u32NumChannels = NUMBER_OF_ADC_CHANNELS;
 }

 for (i32ChannelIdx=0; i32ChannelIdx<NUMBER_OF_ADC_CHANNELS; i32ChannelIdx++) {
  if (ICS1650_OK != ics1650TransferSet(hDevice, i32ChannelIdx, &acq[i32ChannelIdx])) {
   printf ("ics1650TransferSet failed.\n");
   goto ErrorExit;
  }
 }

 /* Enable the transfer */
 for (i32ChannelIdx=0; i32ChannelIdx<NUMBER_OF_ADC_CHANNELS; i32ChannelIdx++) {
  if (ICS1650_OK != ics1650TransferEnable(hDevice, i32ChannelIdx)) {
   printf ("ics1650TransferEnable failed.\n");
   goto ErrorExit;
  }
 }

 /* Set Sync */
 if(synchronizationConfig.u32Select == F_ICS1650_INTERNAL) {
  if(ICS1650_OK != ics1650SynchronizationTrigger(hDevice)) {
   printf("ics1650SynchronizationTrigger failed.\n");
   goto ErrorExit;
  }
 }

 /* Allocate and Initialize Data Buffers */
 if((ptrBuffer = (ICS1650_INT16_T **)malloc(NUMBER_OF_ADC_CHANNELS *
sizeof(ICS1650_INT16_T *))) == NULL) {
  printf("Could not allocate memory.\n");
```

```
  goto ErrorExit;
 }

 for(i32ChannelIdx=0; i32ChannelIdx < NUMBER_OF_ADC_CHANNELS; i32ChannelIdx++) {
  if(acq[i32ChannelIdx].u32SampleCount == 0)
   u32NumBytesToRead = CONTINUOUS_BUFFER_SIZE_IN_SAMPLES * sizeof(ICS1650_INT16_T);
  else
   u32NumBytesToRead = acq[i32ChannelIdx].u32SampleCount * sizeof(ICS1650_INT16_T);

  if((ptrBuffer[i32ChannelIdx] = (ICS1650_INT16_T *)malloc(u32NumBytesToRead)) == NULL) {
   printf("Could not allocate memory.\n");
   goto ErrorExit;
  }
        else {
   printf ("memory allocation success!\n");
  }
  memset(ptrBuffer[i32ChannelIdx], 0, u32NumBytesToRead);
 }
 while(u32cycles<num_acq){
  /* Reset Acquisition Pipeline */
  if(ICS1650_OK != ics1650TransferReset(hDevice)) {
   printf("ics1650TransferReset failed.\n");
   goto ErrorExit;
  }
  /* Perform blocking read on each channel */
  for(i32ChannelIdx=0; i32ChannelIdx<NUMBER_OF_ADC_CHANNELS; i32ChannelIdx++) {
   if (ICS1650_OK != ics1650ReadFile (hDevice, i32ChannelIdx, ptrBuffer[i32ChannelIdx],
u32NumBytesToRead, &u32NumBytesRead, u32Timeout)) {
     printf ("DMA failed. Channel %d.\n",i32ChannelIdx);
     goto ErrorExit;
```

```
    }
   }
   printf("cycles: %ld \n", u32cycles++);
            /* Write last data acquisition to file */
   for(i32ChannelIdx=0; i32ChannelIdx<NUMBER_OF_ADC_CHANNELS; i32ChannelIdx++) {
   filename[6] = (char) ('0' + i32ChannelIdx);
   write_to_file(ptrBuffer[i32ChannelIdx], u32NumBytesToRead/sizeof(ICS1650_INT16_T),
filename);
   }

  }
ErrorExit:

 /* Free Buffers */
 if(NULL != ptrBuffer) {
  for(i32ChannelIdx = 0; i32ChannelIdx < NUMBER_OF_ADC_CHANNELS; i32ChannelIdx++) {
    if(ptrBuffer[i32ChannelIdx] != NULL) {
      free(ptrBuffer[i32ChannelIdx]);
      ptrBuffer[i32ChannelIdx] = NULL;
    }
  }
  free(ptrBuffer);
  ptrBuffer = NULL;
 }

 /* Close Handle */
 if (ICS1650_INVALID_HANDLE_VALUE != hDevice) {
  /* Reset any active transfers */
  for (i32ChannelIdx=0; i32ChannelIdx<NUMBER_OF_ADC_CHANNELS; i32ChannelIdx++) {
    if(ICS1650_OK != ics1650TransferDisable(hDevice, i32ChannelIdx)) {
```

```
        printf("ics1650TransferDisable failed.\n");
     }
              else{
                printf("transfer disable success. \n");
              }
   }
   if(ICS1650_OK != ics1650TransferReset(hDevice)) {
      printf("ics1650TransferReset failed.\n");
   }
            else{
              printf("transfer reset success. \n");
            }
   /* Board Reset */
   if(ICS1650_OK != ics1650BoardInitialize(hDevice)) {
     printf("ics1650BoardInitialize failed.\n");
   }
   ics1650CloseFile(hDevice);
  }
  return 0;
}

/* Function to write data to file ************************/
void write_to_file (ICS1650_INT16_T * buffer, ICS1650_ULONG_T buffer_size, char * filename)
{
  FILE * fd;


  if( NULL == (fd = fopen (filename, "a")) ) {
   printf ("Cannot open file %s.\n", filename);
   return;
```

```
  } else {
    //printf ("File %s opened O.K.\n", filename);
  }

  fwrite(buffer, sizeof(signed short int), CONTINUOUS_BUFFER_SIZE_IN_SAMPLES, fd);

  fclose(fd);

  //printf("Wrote %ld bytes to %s.\n", buffer_size*sizeof(ICS1650_INT16_T), filename);
}
```

Image Processing Code

```
clear all; close all; clc;
addpath('Subfunctions');
tic

%% load the data into 'A'
base='nov_11_15_ch';
%
filename=strcat(base,num2str(1),'_surface_coils_20db_attn_shifted_IF.bin')
;
filename=strcat(base,num2str(0),'_surface_ch1and3and5_ismrm.bin');

fileopen=fopen(filename);
A(1,:)=fread(fileopen,'int16');

disp('load success')

%% Background math

TE=10e-3; %1/2 echo time
Fs = 50e6;  % Sampling Frequency
Fc=[.5e6]; %IF Center Frequency
points=128; %matrix size in one direction (assuming square)
SW=50e3; %spectral width of the MR experiment
t_tx=4e-3; %transmit length
t_rx=1/SW*points; %How long the receiver would sample for

lower_bound=(TE-1/2*t_rx+1/2*t_tx)*Fs+1; %the point at which the echo
starts
upper_bound=lower_bound-1+t_rx*Fs; %the point at which the echo ends
echo_samples=upper_bound-lower_bound; %total number of samples in the echo
samples=700000;
echoes=max(size(A))/samples;

period_points=Fs./Fc; % points per period
cprofile=echo_samples./2+echo_samples/2*Fc/(Fs/2);


%% reshape A

data=reshape(A,samples,echoes);
clear('A');


%% separate into echo and transmit and filter

echo=data(lower_bound:upper_bound,:);
tx=filter_data(data(2e5:3e5,:),Fc(1),Fs);



locationtx=3e4;
```

```
[~,locationrx]=max(echo(:,1));


%% phase correct the echoes


for ll=1:echoes
    ll

phasediff(ll)=findphase(tx(:,1),locationtx,tx(:,ll),locationtx,Fc(1),Fs);
%     fullspectrum(:,ll)=correctphase(echo(:,ll),phasediff(ll),Fc,Fs);
    fullspectrum(:,ll)=correctphase(echo(:,ll),phasediff(ll),Fc,Fs);

end


%% pull the profile out of the fully samples echo

ft_onedim=fullspectrum(cprofile-63:cprofile+64,:);

image=fftshift(fft(fftshift(ft_onedim')))';
image=fliplr(abs(image));


imagesc(image)
title('coil with preamp on modular receiver')

toc

% save('myrx_ch6','image')


sigpoints=[46,79;67,98;];
noisepoints=[11,8;120,45;];

%% SNR calculation

% [SNR, sigpoints,noisepoints]=imageSNR(image);
% disp(SNR)

figure;
sigpoints=[55 65; 60 75;];
noisepoints=[7 5; 51 38;];
SNR2=imageSNR(image,sigpoints,noisepoints);

disp(SNR2)
```

Carbon Spectra Processing Code

```
clear all; close all; clc;
addpath('Subfunctions');
tic

%% load the data into 'A'
base='nov_10_15_ch';
filename=strcat(base,num2str(0),'_13c_64avgs_1024s.bin');

fileopen=fopen(filename);
A(1,:)=fread(fileopen,'int16');

disp('load success')

%% Background math

Fs =10e6;  % Sampling Frequency
Fc=.5e6; %Center Frequency

lower_bound=1750; %the point at which the echo starts
upper_bound=lower_bound+.1024*Fs-1; %the point at which the echo ends
echo_samples=upper_bound-lower_bound+1; %total number of samples in the
echo
samples=1100000;
echoes=max(size(A))/samples;

period_points=Fs./Fc; % points per period
cprofile=echo_samples./2+echo_samples/2*Fc/(Fs/2);


%% reshape A

data=reshape(A,samples,echoes);
clear('A');

%% create the weighting function

t=1:echo_samples;
weight_exp=exp(-t/100000)';

weight=ones(echo_samples,echoes);
%
% for i=1:echoes
% weight(1:echo_samples,i)=weight_exp;
% end

%% separate into echo and transmit and filter

echo=data(lower_bound:upper_bound,:).*weight;

tx=filter_data(data(200:1000,:),Fc(1),Fs);
```

```
locationtx=400;
[~,locationrx]=max(echo(:,1));


%% phase correct the echoes

for ll=1:echoes
    ll

phasediff(ll)=findphase(tx(:,1),locationtx,tx(:,ll),locationtx,Fc(1),Fs);
%     fullspectrum(:,ll)=correctphase(echo(:,ll),phasediff(ll),Fc,Fs);
    fullspectrum(:,ll)=correctphase(echo(:,ll),phasediff(ll),Fc,Fs);

end


%% pull the profile out of the fully samples echo

ft_onedim=fullspectrum(cprofile-511:cprofile+512,:);

newft=sum(ft_onedim,2);

plot(1:max(size(ft_onedim)),newft)

save('nov_10_15_13c_left.mat','newft')
```

MATLAB Function 'findphase'

```
function [phasediff] =
findphase(vector1,location1,vector2,location2,Fc,Fs)


[Re Im]=demod(vector1,Fc,Fs,'qam');
y(1,:)=complex(Re,Im);

[Re2 Im2]=demod(vector2,Fc,Fs,'qam');
y(2,:)=complex(Re2,Im2);


phasediff=mean(unwrap(angle(y(1,location1-5:location1+5)))-
unwrap(angle(y(2,location2-5:location2+5))));
```

MATLAB Function 'correctphase'

```
function [corrected] = correctphase(vector1,phase_offset,Fc,Fs)


%% shift the second sinusoid to be in phase with the first sinusoid

% phase_shift=2*pi*Fc/Fs*corr_angle.*t;
phase_offset=phase_offset*Fs/(Fc*2);

phase_shift=linspace(-1*phase_offset,phase_offset,max(size(vector1)))';

corrected=fftshift(fft(vector1)).*exp(-1i.*phase_shift);
```