DESIGN AND IMPLEMENTATION OF LABVIEW BASED DATA ACQUISITION AND IMAGE RECONSTRUCTION ENVIRONMENT FOR METU-MRI SYSTEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MURAT EŞREF ÖZSÜT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2005

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. İsmet ERKMEN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. B. Murat EYÜBOĞLU
Supervisor

Examining Committee Members

Prof. Dr. Nevzat G. GENÇER (METU,EEE) _____

Prof. Dr. B. Murat EYÜBOĞLU (METU,EEE) _____

Prof. Dr. Adnan KÖKSAL (Hacettepe Un., EEE) _____

Assist. Prof. Dr. Yeşim SERİNAĞAOĞLU (METU,EEE) _____

Assist. Prof. Dr. Çağatay CANDAN (METU,EEE) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Murat Eşref ÖZSÜT

Signature :

ABSTRACT

DESIGN AND IMPLEMENTATION OF LABVIEW BASED DATA ACQUISITION AND IMAGE RECONSTRUCTION ENVIRONMENT FOR METU-MRI SYSTEM

Özsüt, Murat Eşref
M. Sc., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. B. Murat Eyüboğlu

September 2005, 155 pages

Data acquisition and image reconstruction tasks of METU Magnetic Resonance Imaging (MRI) System are used to be performed by a 15 year-old technology. This system is incapable of transmitting control signals simultaneously and has memory limitations. Control software is written mostly in assembly language, which is hard to modify, with very limited user interface functionality, and time consuming. In order to improve the system, a LabVIEW based data acquisition system consisting of a NI-6713 D/A card (to generate RF envelope, gradients, etc.) and a NI-6110E A/D card (to digitize echo signals) from National Instruments is programmed and integrated to the system, and a pulse sequence design, data acquisition and image reconstruction front-end is designed and implemented. Apart from that, a new method that can be used in Magnetic Resonance Current Density Imaging (MRCDI) experiments is proposed. In this method the readily built gradient coil of the MRI scanner is utilized to induce current in the imaging volume. Magnetic fields created by induced currents are

measured for various amplitude levels, and it is proved that inducing current with this method is possible.

Keywords: Magnetic Resonance Imaging, LabVIEW, Data Acquisition, Current Density Imaging

ÖZ

LABVIEW TABANLI VERİ TOPLAMA, GÖRÜNTÜ OLUŞTURMA ARAYÜZÜ DİZAYNI VE ODTÜ-MRG SİSTEMİNE UYGULANMASI

Özsüt, Murat Eşref
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği
Tez Yöneticisi: Prof. Dr. B. Murat Eyüboğlu

Eylül 2005, 155 sayfa

ODTÜ Manyetik Rezonans Görüntüleme (MRG) Sisteminin veri toplama ve görüntü oluşturma görevleri 15 yıl öncesinin teknolojisiyle yapılmaktaydı. Bu sistem kontrol sinyallerinin eşzamanlı olarak iletilmesi özelliğine sahip değildi ve hafıza kısıtlamaları vardı. Kontrol yazılımı genel olarak assembly dilinde yazılmıştı, eklemeler yapılması güçtü, çok kısıtlı kullanıcı etkileşimine sahipti ve zaman kayıplarına yol açıyordu. Sistemi tüm bu yönlerden geliştirmek amacıyla National Instruments firmasının ürünü, LabVIEW tabanlı, bir adet NI-6713 S/A kart (RF zarfını, gradyanları yaratmak vb.), ve bir adet NI-6110E A/S kart (yankı sinyallerini sayısallaştırmak için) programlanıp sisteme entegre edilmiş ve darbe dizisi tasarımı, veri toplama ve görüntü oluşturma uygulamaları tasarlanıp hazırlanmıştır. Bunların yanısıra Manyetik Rezonans Akım Yoğunluğu Görüntüeme (MR-AYG) deneylerinde kullanılabilecek yeni bir yöntem önerilmektedir. Bu yeni yöntemde MRG cihazının içinde hazır olarak bulunan gradyan sargısı görüntülenecek hacimde akım indüklemek için kullanılmıştır.

İndüklenen akım tarafından yaratılan manyetik alanlar değişik akım seviyeleri için ölçülmüş ve bu yöntemi kullanarak akım indüklenebileceği ispatlanmıştır.

Anahtar Kelimeler: Manyetik Rezonans Görüntüleme, LabVIEW, Veri Toplama, Akım Yoğunluğu Görüntüleme

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor Prof. Dr. Murat Eyüboğlu for his guidance and support throughout this study.

Very special thanks to my friends Emre Arpınar and Evren Değirmenci. These two guys are great for pouring (also drinking) antifreeze, killing cockroaches - giant spiders, touching high power circuits with conductor objects, fighting MR floods, eating high calorie food, and making geyik. Apart from these, their helps in every stage of this thesis are greatly appreciated.

I have been thankful to Evrim Çolak for manufacturing the PCB of the frequency synthesizer control card. He is also acknowledged for always finding the right stuff that I need from the junk he collected.

Thanks to my parents for directing me towards getting a MS degree although they did it rather forcefully (☺).

Thanks to Hüs, Ayhan, Serap, Yoldaş, and Dario for their support. My sincere thanks to Koray for the invaluable and motivating analyses that he presented loudly in the crowded cafeteria at the Biomed Conference.

Finally, I would like to thank Zekeriya Özgün and Veysel Eren at the mechanical workshop for their helps in the manufacturing of the boxes and phantoms that are used in this thesis.

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Magnetic resonance imaging

Magnetic resonance imaging is a tomographic imaging technique that utilizes the radiofrequency response of the spinning atomic nuclei to magnetic fields and radiowaves in order to form images that are extremely rich in information content.

Nuclear magnetic resonance phenomenon was first discovered independently by Bloch *et al* [1, 2] and Purcell *et al* [3] in 1946. This is followed by the discovery of the chemical shift which enables the identification of the nuclei in different chemical environments. Therefore nuclear magnetic resonance has become a powerful tool for chemistry to investigate the structure of the solids and liquids.

These spectroscopic techniques are followed by the discoveries related with imaging the proton density in tissues. The principle of using gradient fields to create a shift in the resonant frequencies of the spins as a function of their spatial position was proposed by Lauterbur [4] and by Mansfield and Grannell [5] in 1973. The first whole body image is published by Damadian *et al* [6] in 1977. Their pioneering research triggers the development of many other discoveries such as the fast imaging methods, T_1-T_2 contrast imaging etc. which create a multi-billion dollar industry.

1.2 METU magnetic resonance imaging system

Middle East Technical University - Magnetic Resonance Imaging System (METU MRI System) is a facility of the electrical and electronics engineering (EEE) department's Magnetic Resonance Imaging Research Laboratory (MRIRL).

This system is installed in the early 90's by the faculty and the research students of the EEE department. Similar to other MRI systems, this system is composed of a main magnet that creates a DC magnetic field, three gradient coils for creating gradient fields for signal localization and a RF coil for excitation of the spins and for receiving echo signals.

An MRI experiment that is performed by this system requires the specification of certain parameters that defines the amplitudes, phases, frequencies and timings of the waveforms to be applied to the above mentioned coils. The combination of all these parameters is called a pulse sequence.

A magnetic resonance image is depended on many properties of the imaging region such as the nuclear spin density ρ, the spin-lattice relaxation time T_1, the spin-spin relaxation time T_2, molecular motions, susceptibility effects and chemical shift differences.

The effects of these properties on the image can be suppressed or enhanced by setting some operator specified parameters of the MRI System such as the repetition time TR, the echo time TE, the flip angle α etc. appropriately. Therefore by appropriate manipulation of these parameters, different structures in the image can be reconstructed with high contrast. All of these parameters must also be defined in the pulse sequence.

1.3 Objectives of this study

Previously, data acquisition and image reconstruction tasks of the METU MRI system was performed by an A/D, a D/A and a TMS DSP card that are integrated to a 33 MHz 80486 PC. This hardware is programmed using assembler and C programming languages.

Although this hardware-software combination is a good solution in 1990's, current progress in technology makes it ineffective when compared to the new options that are available.

One of the drawbacks of the previous system lies in the fact that the simultaneous changing of gradients (actually any channels) is not possible. This situation unnecessarily lengthens the echo time, and also applications that require a simultaneous change in two or more channels become impossible to be performed.

The old system also has memory limitations. The burden caused by this fell onto the shoulders of the programmer. Even a simple operation like generating a sine wave must be cleverly planned in order to circumvent these limitations.

There is also a maximum instruction limitation of the TMS card. Therefore all of the modules that are written to perform specific tasks can not be present at the TMS card at the same time, instead they are loaded when necessary, causing a programming difficulty. Also the design of complex modules is prohibited by this limit.

Since TMS card is primarily a digital signal processing card instead of an acquisition card, the card has no timing and triggering functionalities. These functionalities are implemented by utilizing the cycle time of the TMS processor by using nop (no operation) commands to force TMS to wait until the right time

without performing any operation. These scheme is both hard to implement and inefficient in CPU time usage.

Another point to mention is the hard to program assembler language which is used in the programming of the TMS card. This language is difficult to understand and hard to modify causing an increase in the software development and modification times.

Also the old system has a complex integration scheme. The data acquisition tasks and waveform generation, control tasks are performed in different hardware. Also the TMS card is programmed in assembler language whereas there is another program that runs on PC that is written in the C language. These increase the complexity of the system, making new additions to the system hard to implement.

Although it can be said that the previous software is good for the required tasks considering the era it is written, it is evident that the software has completed its lifespan. Since the software lacks the graphical user interface functionality that is common to every software in today's world, it is hard to design and modify pulse sequences using this software. Also the image reconstruction components are primitive without providing additional image processing capabilities.

In order to improve the pulse sequence design, data acquisition and image reconstruction components of the system in all of the above mentioned aspects, new hardware-software combinations are designed and implemented in this study.

LabVIEW is chosen as the single programming environment for all of the tasks performed, since it provides powerful programming tools for acquiring, analyzing and presenting data. Multi-function acquisition hardware is purchased, programmed and integrated into the system to perform the data acquisition tasks. An additional card is designed and fabricated to control the frequency synthesizer unit remotely.

A software suit consisting of two powerful software is designed and implemented. One of them is the “Pulse Sequence Design and DAQ” software which is responsible for all of the pulse sequence design and data acquisition tasks as its name implies. The “Image Reconstruction” software on the other hand, provides image reconstruction, post processing and exporting functionalities.

Apart from providing all of the functionality of the old system, the new system also has additional functionalities such as the automatic tip angle adjustment, automatic center frequency adjustment, on the fly modification of the pulse sequence parameters, stand alone image processing functionalities, batch exporting capabilities etcetera. These software are designed with possible future expansions in mind, so they are written in a modular, easily understandable and scalable way.

In addition to the above mentioned studies, a novel method for inducing currents in a conductor object that can be used for current density imaging applications is proposed in this thesis. In this method, the already present gradient hardware of the MRI system is used for current induction purposes instead of additional external coils. A phantom is fabricated, a special pulse sequence is designed, experiments with varying amplitudes of current flowing through gradient coil are performed, results are analyzed and it is shown that the current induction by the proposed method is possible.

1.4 Outline of the study

In this thesis firstly, the hardware components of the METU MRI system is explained along with the discussions about how they are utilized in the software in Chapter 2. In Chapter 3 general programming concepts that are used during the creation of the software suit is explained to the reader. This is followed by the detailed explanation of the “Pulse Sequence Design and DAQ” software in

Chapter 4. After that the “Image Reconstruction” software is explained in detail in Chapter 5. The discussion about the current induction utilizing gradient coils is given in Chapter 6. Finally, Chapter 7 is the conclusion chapter.

There are four appendices of this thesis. In Appendices A and B the graphical user interface of the software suit is explained along with screenshots taken. In Appendix C, a categorized listing of all the subVIs (functions) that are written or used from LabVIEW libraries is given. Finally, Appendix D lists the conference presentation performed during this thesis.

CHAPTER 2

HARDWARE OF THE METU MRI SYSTEM

2.1 Introduction

Data acquisition and image reconstruction tasks of the METU Magnetic Resonance Imaging System are performed by a D/A (digital to analog) card and an A/D (analog to digital) card which are integrated in a PC based computer system. These cards are purchased from National Instruments company, and are programmed with LabVIEW which is a graphical programming environment specialized in acquiring, analyzing, and presenting data. LabVIEW has excellent integration with the above mentioned data acquisition hardware since it is also a product of the National Instruments Company.

The complete block diagram of the METU MRI System is given in Figure 2.1. The hardware components that are shown in this figure are explained throughout this chapter. In this figure the sections that are represented with green color are the components of the data acquisition subsystem which are designed and implemented during this study.

Also the other hardware components are explained in detail in [7], and they are summarized in [8]. Details about the current source unit can be found in [9].

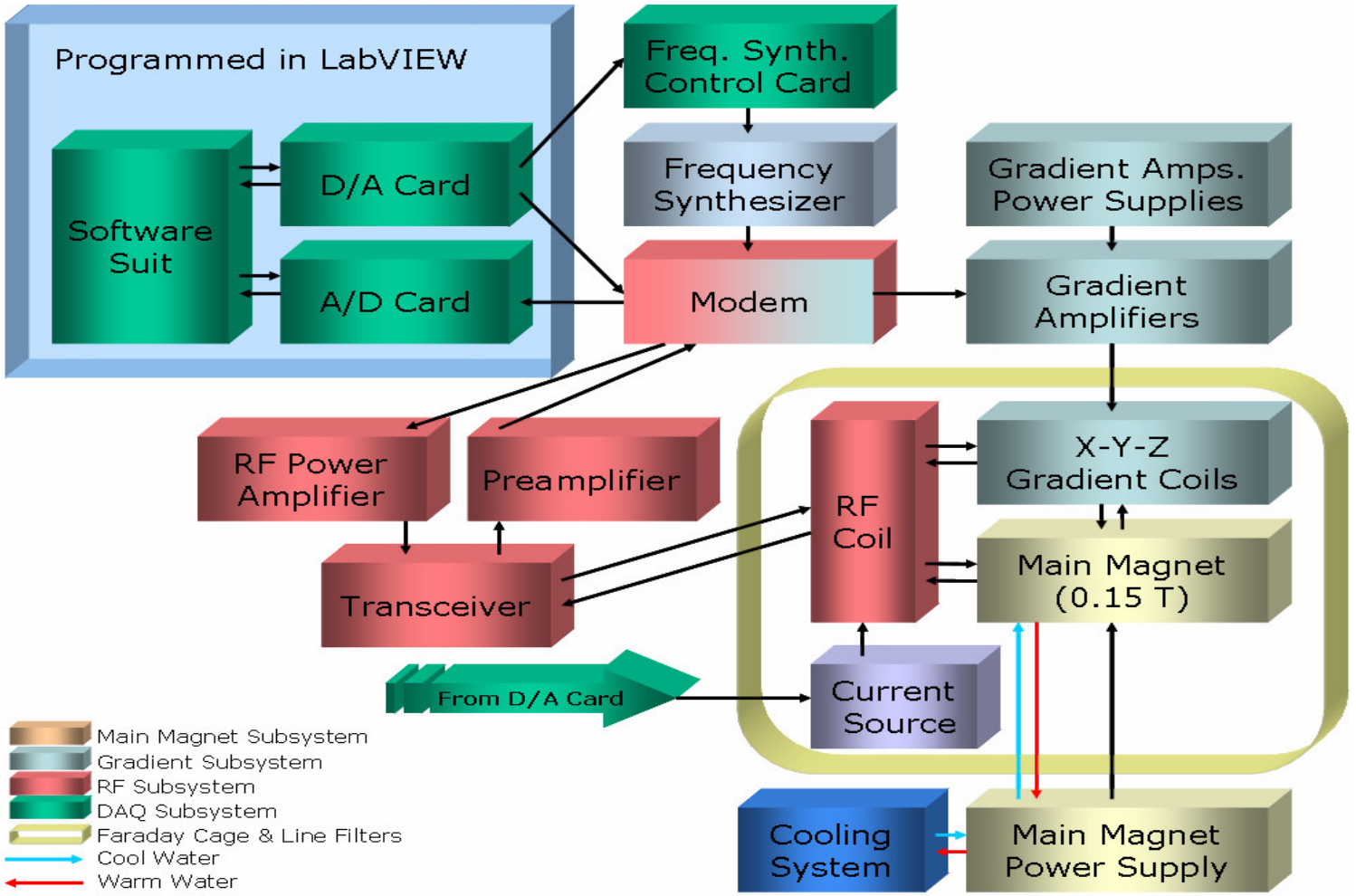


Figure 2.1: Block diagram of the METU MRI System.

In this chapter, firstly the functionalities of the data acquisition cards which are utilized during programming phase are explained (the programming details are given in Chapter 4), followed by the description of the computer system to which these cards are connected, then the designed and implemented frequency synthesizer control card is explained, after that the remote control of the current source is discussed, and finally the chapter ends with the explanation of the other components of the METU MRI System.

2.2 D/A Card

The main purpose of the D/A card is to generate the necessary waveforms that must be transmitted to the various parts of the MRI system. The card generates two RF envelope waveforms (real and imaginary parts), three gradient waveforms for slice selection, frequency encoding, and phase encoding, two signals that control the current source that is used for MR-CDI experiments, and a RF gating signal that turns on the RF power amplifier during the application of a RF pulse. To summarize the card generates six waveforms for MRI experiments and eight waveforms for MR-CDI experiments.

D/A card chosen for the system is NI PCI-6713. This card is a completely Plug and Play, analog output, digital and timing I/O device which is connected to the PC via the PCI bus. The card has eight channels of voltage outputs with 12 bits of resolution.

Since the card features a separate digital-to-analog converter (DAC) per channel, simultaneous driving of all of the eight channels is possible. This feature eliminates the image artifacts resulting from not being able to turn ON the gradient channels at the same time which is the case for the previous METU MRI hardware. The update rates are up to 1 MS/s per channel, and are listed in Table 2.1 as taken from product manual, measured with a 90 MHz Pentium machine [10].

Table 2.1: Maximum update rate (kS/s) using host PC memory.

Number of Channels	Maximum Update Rate
1 through 6	1000
7	869
8	769

2.3 A/D Card

This card is used to digitize the received echo signal. Echo signal is actually received as two signals, a separate signal for the real part and the imaginary part.

A/D card chosen for the system is NI PCI-6110E. This card is a completely Plug and Play, multifunction analog, digital and timing I/O device that is connected to a PC via the PCI bus.

The card has four analog input channels with 12 bits of resolution. Maximum sampling rate is at 5 MS/s regardless if one or four channels are acquiring data. Since the card features a separate analog-to-digital converter (ADC) per channel simultaneous sampling of echo signals are possible. The card also has two analog outputs which uses separate 16 bit D/A converters (DACs). Output rates are 4 MS/s when one channel is used and 2.5 MS/s when both of the channels are used.

2.4 Functionalities common to both of the cards

The block diagram of the PCI-6713 card is given in Figure 2.2 [10] and the block diagram of the PCI-6110E card is given in Figure 2.3 [23].

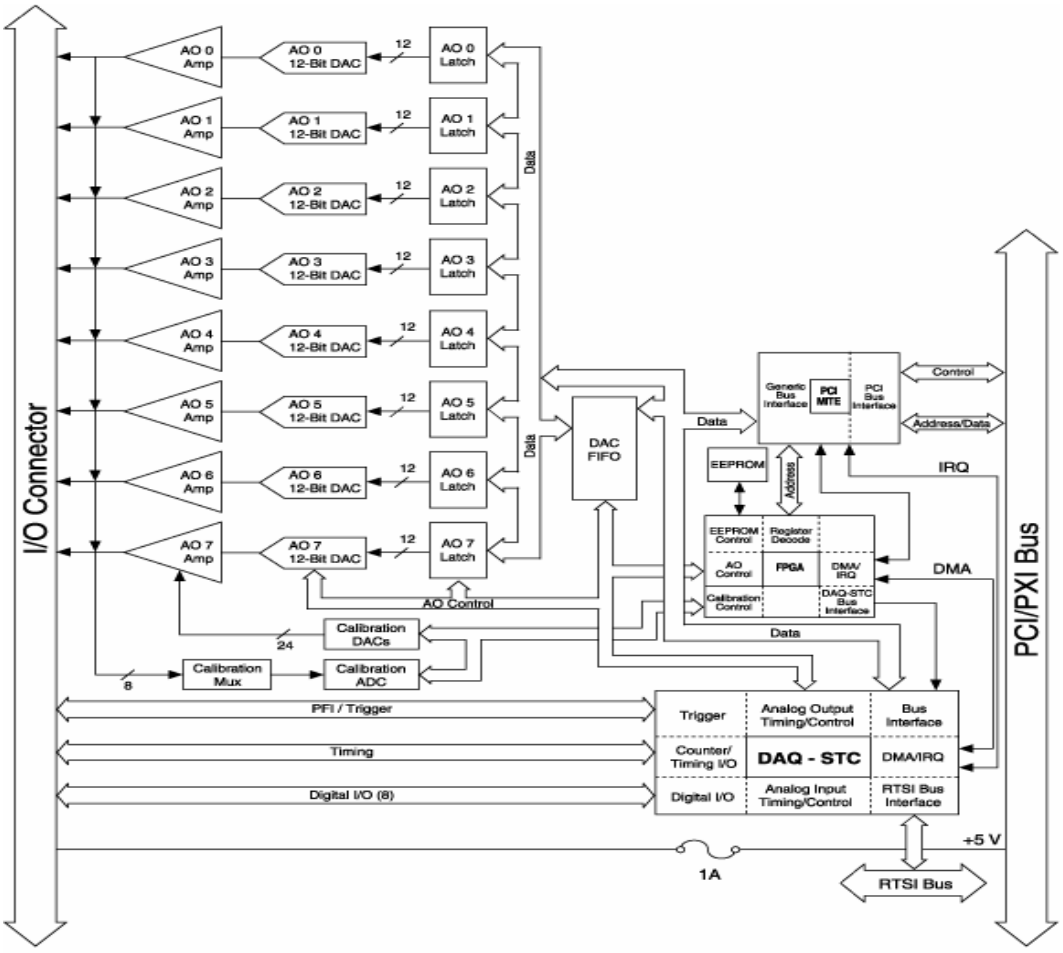


Figure 2.2: Block diagram of the PCI-6713 D/A card [10].

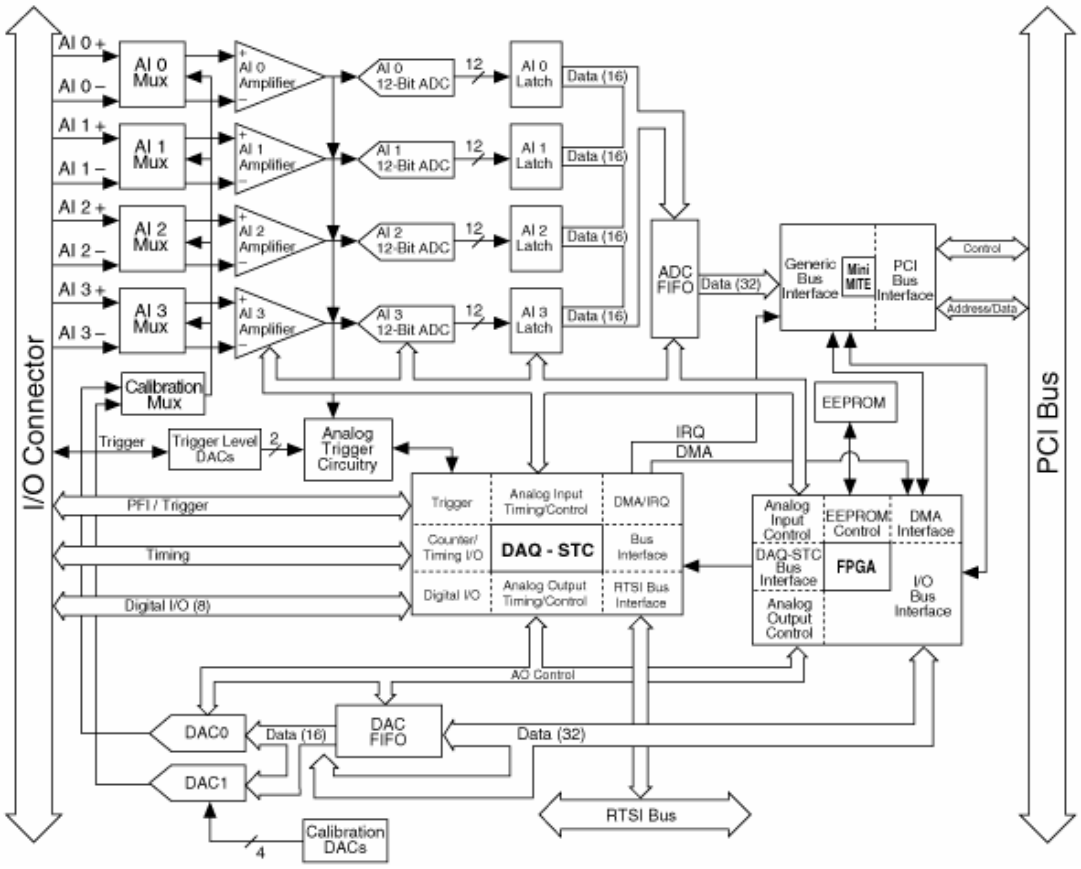


Figure 2.3: Block diagram of the PCI-6110E A/D card [23].

The cards have no DIP switches, jumpers, or potentiometers, and can be software-configured and calibrated. The MITE bus interface chip connects devices to the PCI I/O bus so that the interrupts and base memory addresses are all software configured.

The devices use DAQ-STC system timing controller for time related functions. The DAQ-STC consists of three timing groups that control analog input, analog output, and general purpose counter/timing functions. These groups include a total of seven 24 bit and three 16 bit counters, that can be configured with timing resolutions of 50ns or 10 µs. The analog input section of the DAQ-STC is unused

by PCI-6713 card. The DAQ-STC provides a very flexible interface for connecting timing signals to other devices or external circuitry. Precise timing of the pulse sequences is achieved utilizing the functionalities of the DAQ-STC.

The cards use the Real Time System Integration (RTSI) bus to interconnect timing signals between DAQ devices, and the Programmable Function Input (PFI) pins on the I/O connector to connect the cards to external circuitry.

Using PFI connections the cards can both control and be controlled by other devices and circuits. There are a total of thirteen timing signals internal to the DAQ-STC that can be controlled by external source. These timing signals can also be controlled by signals generated internally to the DAQ-STC, and these selections are fully software configurable. Many of these signals are also available as outputs on the RTSI pins and on the PFI pins. These PFI connections are used to trigger the DAQ cards for performing tasks such as data measurement and pulse train generation (are explained in detail in Chapter 4).

In order to synchronize measurement functions to a common trigger or timing event, the cards have the Real Time System Integration (RTSI) bus. The RTSI bus consists of RTSI bus interface and a ribbon cable to route timing and trigger signals between several functions on as many as five DAQ devices in a computer (two DAQ devices for METU MRI system). The seven RTSI trigger lines on the RTSI bus provide a flexible interconnection scheme for the devices sharing the RTSI bus. These bidirectional lines can drive any of the timing signals onto the RTSI bus and can receive any of these timing signals. The above mentioned trigger signals are routed between two DAQ cards via RTSI bus. Also RTSI bus is utilized for timebase synchronization operation which is explained later in this section and in more detail in Chapter 4.

The cards use the Programmable Function Input (PFI) pins to connect the timing signals to external circuitry. There are ten PFIs that are connected to the signal routing multiplexer for each timing signal, and software can select one of the

PFIs as the external source for a given timing signal. This flexible routing scheme reduces the need to change the physical wiring to the device I/O connector for applications requiring alternative wiring. Also PFI pins can be enabled to output a specific internal timing signal. By utilizing this functionality, timing signals can be used to trigger an oscilloscope while examining the waveforms generated by the cards during the programming phase.

For generating the necessary timing signals for A/D conversions, DAC updates, or general purpose signals at the I/O connector the cards use their internal 20 MHz timebase or a timebase received over the RTSI bus. Also if a device is configured to use its internal timebase, it can be programmed to drive its internal timebase over the RTSI bus to another device that is programmed to receive this timebase signal. This clock source, whether local or from the RTSI bus, is used directly by the device as the primary frequency source. This scheme is used to force the cards to be in perfect synchronization during operation as will be explained in Chapter 4 in more detail.

The cards have eight lines of TTL/CMOS compatible digital I/O for general purpose use. The lines are software configurable to be used as input or output. These lines are used in the remote control of the frequency synthesizer unit in the system.

BNC-2110 shielded connector blocks are used as the assemblies to connect the cards to the external devices. These connector blocks feature BNC connectors for the analog signals. They also provide access to the digital I/O and all of the counter/timing signals including the PFIs.

2.5 METU MRI front-end computer system

A powerful computer system is chosen as the host PC. This machine has a Pentium 4 processor at 2800MHz, and has 2 GB of RAM. Since monitors with CRT (cathode ray tube) technology do not perform well under magnetic fields

created by the MRI System, two LCD screens are chosen as the display devices. By this way simultaneous display of both the “Pulse Sequence Design and DAQ” software user interface and the “Image Reconstruction” software user interface is made possible, providing an optimal working environment. A picture of the METU MRI front-end computer system is given in Figure 2.4.

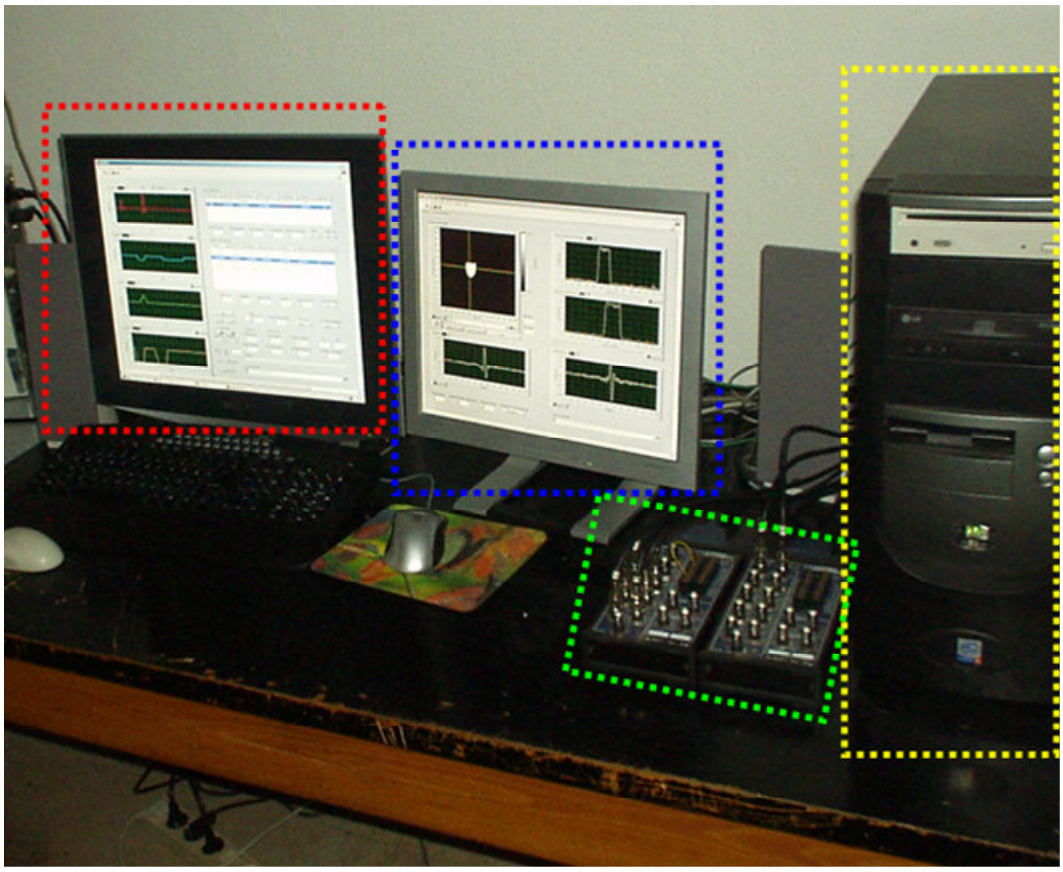


Figure 2.4: The METU MRI front-end computer system.

In this picture, LCD screens for simultaneous display of both of the software is shown in red and blue rectangles, the computer system that hosts the DAQ hardware is shown in the yellow rectangle whereas the connector blocks that are used for connecting the DAQ hardware to external circuitry are shown in the green rectangle.

2.6 Frequency synthesizer unit

The frequency synthesizer unit used in the METU MRI system is from Programmed Test Sources Incorporation, and its model number is PTS 160.

This unit generates a sinusoidal signal with a frequency specified by the operator. This signal is used by the modem unit as a carrier signal when modulation is performed to create the RF pulses and its frequency determines the central frequency of the spins in the imaging region. Also this signal is used in the receive phase for demodulation purposes. This unit can supply signals with frequencies from 0.1 MHz to 160 MHz with 0.1 Hz of resolution.

2.7 Frequency synthesizer control card

This card is designed and implemented to control the carrier frequency setting of the frequency synthesizer unit in METU MRI system.

Previously, the carrier frequency setting using the frequency synthesizer unit can only be made manually. By using the current hardware however, the operator can adjust the central frequency from the software and view the results simultaneously on another display.

The remote control of the frequency synthesizer is made possible by the parallel entry board circuitry of the unit. This board enables the unit to be controlled by

external devices using a 50 pin centronics connector that is located on the back panel of the frequency synthesizer unit.

The remote control format is parallel entry. The values of digits are given using binary coded decimals (BCD) for the digits in the 0.1 Hz to 1 MHz range, and hexadecimal for the 10 MHz digit. This means there are four lines that set the value of a digit; 1,2,4,8 respectively. For BCD digits the values in the 10-15 range are invalid. TTL logic levels are used and all commands are negative true. There are also five latch enable lines (since some digits share common latch enable lines) for storing the commands. Finally, there is a remote enable line that chooses between the local and remote frequency control modes. To sum up, there are 42 lines that must be driven parallel in order to control the frequency synthesizer remotely.

The DAQ cards in METU MRI system provide a total of 16 digital I/O for general purpose operations. Since this number is not enough to drive the needed 42 lines simultaneously, and it is not an optimized solution to use 42 digital I/O lines just for changing the frequency setting, a frequency synthesizer control card is designed and implemented. This card is a serial to parallel converter that consists of five registers.

The registers used in the card are CD4094BE registers. The functional diagram of the registers is given in Figure 2.5 [11]. These registers are eight-stage serial shift registers having a storage latch associated with each stage for strobing data from the serial input to parallel buffered three-state outputs. The parallel outputs are connected directly to the parallel entry board of the frequency synthesizer unit via a 50 line ribbon cable. Data is shifted on positive clock transitions. The data in each shift register stage is transferred to the storage register when the Strobe input is high. Data in the storage register appears at the outputs whenever the Output-Enable signal is high. Two serial outputs are available for cascading these devices. Data is available at the QS1 serial output terminal on positive clock edges to allow for high-speed operation in cascaded system in which the clock

rise time is fast. The same serial information, available at the QS2 terminal on the next negative clock edge, provides a means for cascading these devices when the clock rise time is slow.

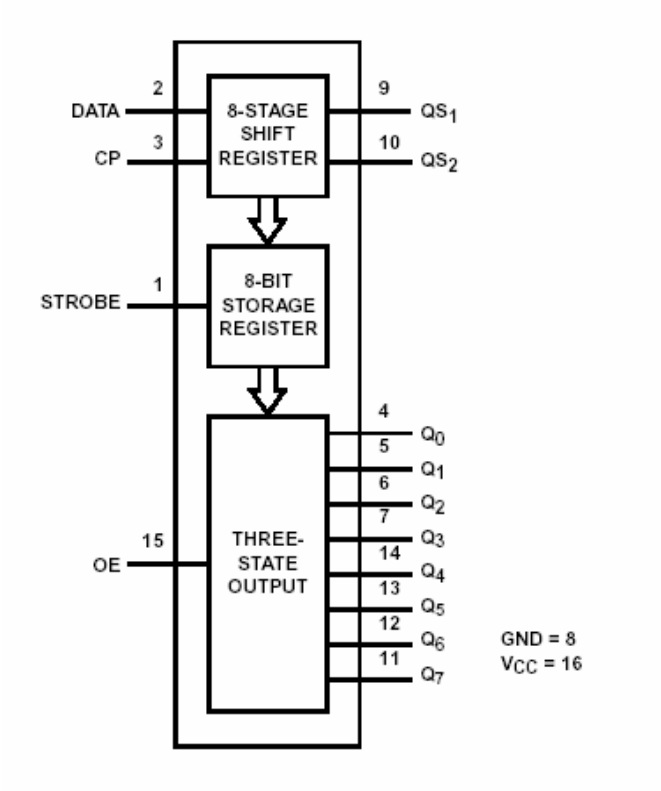


Figure 2.5: Functional diagram of 8-stage CD4094BE register [11].

The block diagram of the frequency synthesizer control card is given in Figure 2.6. In this card, five 8-stage CD4094BE registers are cascaded by connecting the QS1 serial output of a register to the data input of another one. QS2 serial outputs of the registers are unused. Strobe and output enable inputs are at all times set as high logic level. By this way the registers are used as 8-stage shift registers.

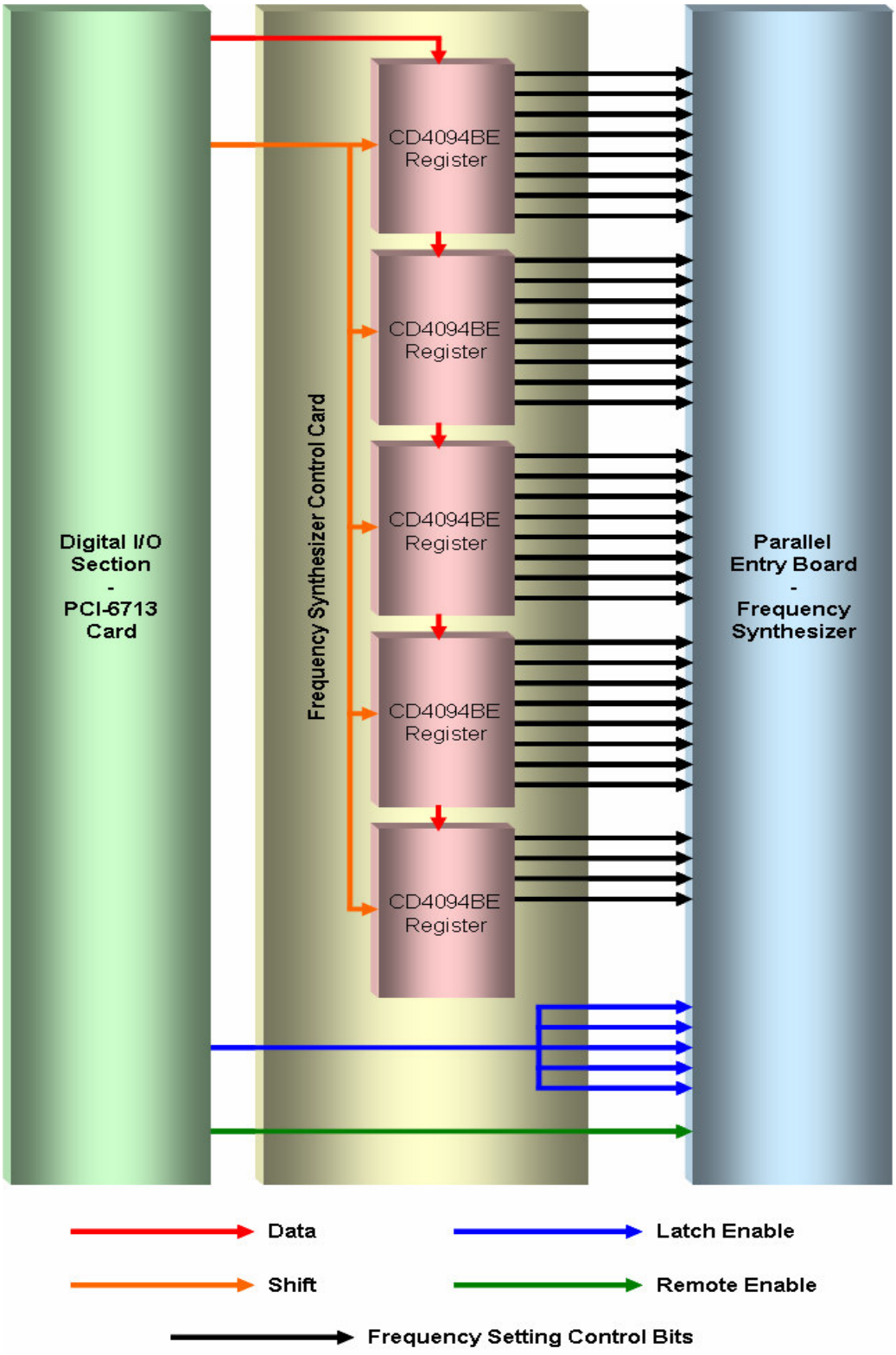


Figure 2.6: Block diagram of the frequency synthesizer control card.

There are four digital control signals that are transmitted from the digital I/O section of the DAQ card in the system. Either one of the cards can be used to transmit these signals, so the PCI-6713 card is selected to perform this operation. These control signals are; data, shift, latch enable, and remote enable.

Data signal is the serial data that will be converted to the parallel form by the card, and is the signal that sets the necessary pins in the parallel entry board to change the frequency setting. Since the frequency synthesizer unit has nine digits, and each digit needs a 4 bit control signal, this signal is 36 bits long.

This signal is driven serially to the five shift registers (the remaining 4 bits of the last register in the chain is unused), on every rising edge of the shift signal.

After that the latch enable signal which is connected directly to all of the five latch enable pins of the parallel entry board is applied so that the frequency synthesizer can latch the data that is located in the registers, and change its frequency.

The remote enable signal is connected to the remote enable pin of the parallel entry board, and is used to switch between local and remote mode.

2.8 Current source

Current source in METU MRI system is used in Magnetic Resonance–Current Density Imaging (MR-CDI) experiments [9] and Magnetic Resonance–Electrical Impedance Tomography (MR-EIT) experiments [12]. In both of these techniques bipolar current pulses are applied to the imaging region in synchronization with a spin echo pulse sequence, and the current source is responsible for injecting current into the object. The block diagram of the current source is given in Figure 2.7.

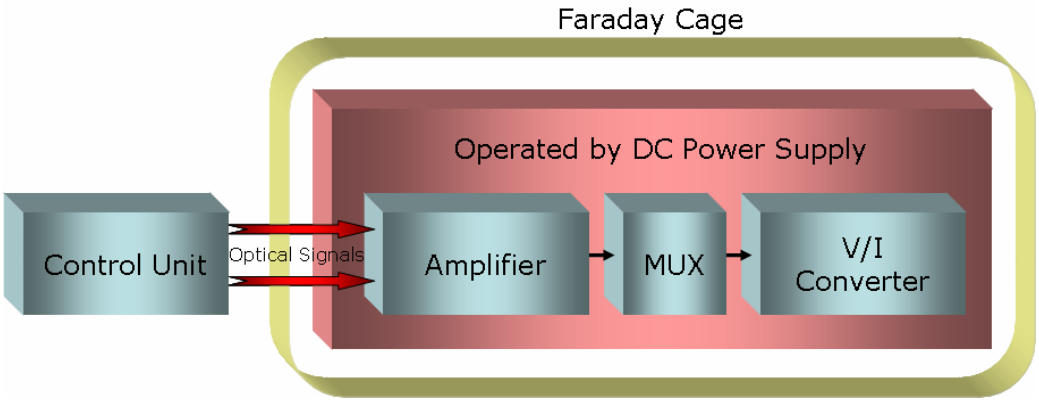


Figure 2.7: Block diagram of the current source.

The control unit converts the control signals that are transmitted by the DAQ card to optical signals, in order to avoid RF interference. As can be seen from Figure 2.7 the rest of the current source hardware resides in the Faraday cage, and these components are battery operated also for avoiding RF interference. The amplifier converts the optical control signals to digital TTL level signals, and these are used by the analog multiplexer to generate a 0V or 5V or -5V signal at its analog output. This signal is converted to current using the voltage to current (V/I) converter. The magnitude of the current is adjustable using the variable resistor located in the V/I converter. Current source in METU MRI hardware is made by Özbek, and details about this unit is available in his thesis [9].

2.9 Remote control of the current source

The current source is controlled by two signals, namely S0 and S1. Although these signals are digital TTL level signals, they are generated by the analog output section of the PCI-6713 D/A card because the DAQ cards does not provide a digital output start trigger to synchronize the digital I/O operations with the analog output operations. So, the current source control signals are generated

using the analog output section, and the triggering capabilities of this section is used to synchronize these signals with the other pulse sequence signals such as RF envelopes, gradient channels etcetera. The truth table for the current source control signals is given in Table 2.2.

Table 2.2: Truth table for the current source control signals.

S1	S0	Current Output State
0	0	Not Applied
0	1	Positive Polarity
1	0	Negative Polarity
Others are not used		X

2.10 Modem unit – modulator section

Modem unit has been built at METU in early 1990's. In this unit, the carrier signal supplied by the frequency synthesizer unit is modulated by the RF envelope signals generated by the D/A card in order to select off center slices (is explained in detail in Chapter 4). The type of modulation is quadrature amplitude modulation. Also a gain control knob which adjusts the amplitude of the created RF pulse is present for tip angle adjustment.

2.11 Gradient amplifiers and their power supplies

There are three gradient amplifier-power supply pairs in the system for creating the x, y and z gradients. These are purchased from Analogic Corporation. Their

model number is AN8210. They provide a maximum output of 168 A and 125 V with minimum rise time of 1ms.

2.12 RF power amplifier

This amplifier is Analogic brand, and is used to amplify the RF pulses created at the modulator output to the energy level that is needed to excite the spins. Its model number is AN8031. This device can supply maximum 1 KW peak envelope power, and has an operation range between 5 MHz - 90 MHz. Its output is gated and is controlled by the D/A card to prevent noise interference during signal reception.

2.13 Gradient coils

There are three sets of gradient coils present in the system for creating the x, y and z gradient fields that can be used for slice selection, frequency encoding and phase encoding purposes. The x and y gradient coils are built at METU, and are wounded on a fiberglass frame. They are designed as modified golay type coils to increase the linear region. They provide maximum field strength of 0.8 G/cm. The z gradient coil is the original coil provided by the Oxford Magnet Technologies, and is mounted on the magnet structure. This coil can provide maximum field strength of 0.2 G/cm, and is a Helmholtz type coil.

2.14 Main magnet power supply

This unit is purchased from Oxford Magnet Technologies. It has 300A and 200V maximum output ratings. It is working at 90 percent of its maximum rated power to generate the 0.15 T main magnetic field. This unit is cooled by water, and has a security board for detecting failures (water failures, overheating conditions etc.).

2.15 Main magnet

The main magnet is purchased from Oxford Magnet Technologies, and has four coils in the form of short solenoids. Utilizing these coils, this magnet generates the 0.15 T main magnetic field. This is a resistive magnet that needs to be powered up in order to create the magnetic field. It has 80 cm bore, and is water cooled.

2.16 Cooling system

Cooling system of the METU MRI System is a Carrier 30RA 040 air cooled liquid chiller. This unit has a net cooling capacity of 39.3 KW. The cooling system is a closed loop system. The unit cools a 1000 liter tank that is located under the unit. After that, the warm water coming from the system is mixed with the cold water in the tank, and then the cool water is given back to the system.

2.17 Faraday cage and line filters

A faraday cage is constructed around the magnet to suppress the interfering electromagnetic signals. The cage is installed by a local company, and approximately 80 dB suppression is achieved.

Line filters are mounted on the cage walls. Their function is to suppress the noise induced on the cables that are carrying signals to and from the components inside the faraday cage.

2.18 RF coil

In METU MRI System a single RF coil is used for both transmit and receive operations. This coil is built by METU EEE students, and it is tuned to operate at the resonant frequency of the spins.

2.19 Transceiver

This unit is built at METU, and is a directional coupler. During the transmit operation, this unit is used to direct the high power RF pulse to the coil while isolating the receiver amplifier. During the receive operation, this unit will direct the weak echo signal to the receiving amplifier while isolating the RF power amplifier port.

2.20 Preamplifier

In order to obtain a reasonable SNR in the image, the weak signal received by the RF coil must be amplified. A HP Avantek INA01170 amplifier is used for this purpose. This device has a voltage gain of 32.5 dB, with an approximately 3 dB measured noise figure [7].

2.21 Modem unit – demodulator section

This unit performs quadrature amplitude demodulation to remove the carrier signal at the resonant frequency of the spins from the received signal. Also this section has a variable attenuator to prevent the saturation of the circuits in further stages.

CHAPTER 3

LABVIEW PROGRAMMING CONCEPTS

3.1 Introduction

In order to implement the data acquisition and image reconstruction tasks of the METU MRI System, two separate software are developed in LabVIEW. One is the "Pulse Sequence Design and DAQ" software and the other is the "Image Reconstruction" software. From now on these two software will be referred as the software suit. The software suit is explained in Chapter 4 and Chapter 5. This chapter explains the necessary background for Chapters 4 and 5.

In this chapter, firstly an introduction to the LabVIEW which is the programming environment used for the creation of this software suit is given. After that, LabVIEW programming environment is discussed, followed by the explanation of the program structure of the software suit and the description of the event driven programming concept. Finally, the chapter ends with the explanation of the types of the events used in the software suit.

3.2 Introduction to LabVIEW

LabVIEW stands for **Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench, and is one of the leading tools in the market for acquiring, analyzing, and presenting data. It is an open environment which is designed to interface with any kind of measurement hardware, METU MRI System, in our case. It has more than 450 built-in functions (and more in its specialized modules) that are

designed specifically to extract useful information from any set of acquired data, to analyze measurements, and to process signals. Finally, LabVIEW provides tools for data visualization, user interface design, web publishing, report generation, data management, and software connectivity. Most of the above mentioned abilities of LabVIEW are utilized, and are used in the implementation of the software suit written for the METU MRI System.

The concept of "Virtual Instrumentation" is the keystone of the LabVIEW environment. A virtual instrument consists of an industry-standard computer equipped with powerful application software written in LabVIEW, cost-effective hardware such as plug-in boards purchased for METU MRI System with appropriate driver software (NI-DAQmx in our case) and the system, namely the METU MRI System, which the above mentioned hardware and software will interact.

All of these will work together to perform the functions of the traditional instruments causing a fundamental shift from the hardware-centered instrumentation systems to the software-centered systems that exploit computing power, productivity, display, and connectivity capabilities of desktop computers.

Flexibility of the LabVIEW environment combined with modular hardware solutions resulted in an extremely customized, user-defined, and scalable data acquisition and image reconstruction environment for the METU MRI System.

3.3 LabVIEW programming environment

LabVIEW is an easy-to-use graphical programming environment that has the performance and flexibility of the traditional programming languages as well as a high level rapid development environment for measurement and automation applications. In LabVIEW icons are used to represent the functions, and they are wired together by the programmer to determine the flow of data throughout the program. Therefore code syntax (commas, brackets etc.) are not important in

LabVIEW; increasing the productivity, and decreasing the time required to develop applications.

Each program in LabVIEW is called a Virtual Instrument, or a VI. Each VI is composed of two main components, namely the front panel and the block diagram.

The front panel is the graphical user interface of the VI. It contains the controls and indicators such as buttons, slides, dials, tables, graphs etc. which user manipulates to interact with the VI.

The block diagram is the graphical code. Every front panel object has a terminal in the block diagram, and the programmer wires these terminals to functions (other VIs actually) that are represented as icons to design the measurement application's functionality. A complete block diagram resembles a flow-diagram.

LabVIEW is a dataflow programming language meaning that data flows from data sources such as controllers in the front panel or outputs of data acquisition functions (i.e. analog input functions) to sinks such as indicators in the front panel or inputs of data acquisition functions (i.e. analog output functions), and propagates throughout the program. Utilizing the dataflow architecture LabVIEW can execute multiple operations in parallel unlike traditional text-based languages that use sequential-execution architecture.

3.4 Program structure of the software suit

General program structure of both of the software is similar, therefore will be explained together in this section. It is possible to examine the execution of both software by dividing them into three main parts; namely the initialization, the operation and the housekeeping. Also throughout this thesis it is assumed that the reader is familiar with the magnetic resonance imaging concept, therefore only necessary details will be given.

3.4.1.1 Initialization

As explained before, dataflow determines the sequence of execution in LabVIEW so each software's dataflow is started by the initialization part. This is a stacked sequence structure, a special programming structure of LabVIEW, which is composed of frames that execute sequentially giving the programmer an option to order the execution of initialization operations. Also the order of execution can be easily changed by the programmer by changing the order of the frames, increasing the flexibility. Another reason of using a stacked sequence structure in the initialization part is that these structures can easily grow by adding additional frames when necessary, and growth does not increase the space occupied by these structures in the block diagram, therefore improving the readability of the code. In this initialization part; the initial values of the controllers and indicators on the front panel are set, data structures that stores critical information are initialized, and user event (will be explained later) registrations are performed.

3.4.1.2 Operation

After initialization part, dataflow reaches a "while" loop which executes until the user chooses to exit from the file menu of the software. An event structure resides in this while loop, and therefore will be executed continuously until exit. The event structure is a special programming structure for LabVIEW, and is the key for creating graphical user interfaces in LabVIEW.

An event is an asynchronous notification that something has occurred. Events can originate either from the user interface or programmatically generated by the programmer.

User interface events include mouse clicks, key presses, and so on. There are two types of user interface events; notify and filter. Notify events are an indication that a user action has already occurred, such as when the user changed the value of a control. Nearly all of the events written for the software suit are of this type. Filter events on the other hand, inform the programmer that the user has

performed an action before even LabVIEW processes it. This gives the programmer the ability to participate in the handling of the event by changing the event data or by completely discarding the event as it had never happened.

The other type of events is programmatically generated event, also called user event, and allows different parts of the application to communicate asynchronously.

All of these events are handled by the event structure. The event structure is a structure resembling a case structure that has a case associated with each event. By using the while loop mentioned above, the event structure wait for an event to occur, executes code to respond to the event, and reiterates to wait for the next event. Every action of the software suit such as performing data acquisition, changing pulse sequence waveforms, or reconstructing images are organized as events. This concept is called event-driven programming, and is used in both software.

It is also possible to poll the states of all front panel objects in a loop, and to check if any change has occurred, but this approach consumes significant amount of CPU time, and also can fail to detect changes if they occur too quickly therefore is not chosen.

Also subVIs (functions) for performing various tasks are written and are used in the event cases mentioned above.

The detailed explanation of the operations performed by the events and subVIs of the software suit will be given in subsequent chapters.

3.4.1.3 Housekeeping

After the event structure, dataflow reaches the housekeeping section. Reaching of the dataflow to this section means that the user has exited from the application. In

this part previously registered user events are unregistered, images that reside in the memory of the system are disposed etc., thus giving the name housekeeping to this part.

3.5 Types of events used in the software suit

The types of the notify events that are used in the software suit are; mouse enter, mouse leave, mouse down, mouse up, value change, menu selection (user), and key down. Apart from these notify events, filter form of mouse down event is also used. Also various user events are created, namely load, extract, process, and display events of the "Image Reconstruction" software and save, load, and plot events of the "Pulse Sequence Design and DAQ" software.

Mouse enter event is generated when the user moves the mouse pointer over the object that the event is defined. Mouse leave event is generated when the mouse pointer leaves the object.

Mouse down event is generated when the user clicks on the object. The mouse button used and whether or not the click is a double click can be quarried from the data fields of this event. In the filter form of this event, operations like discarding the event as it has never happened or changing of the event data fields (i.e. button used, double click performed) are possible.

Value change event is generated when the value of the object changes. Both the new value and the old value of that object can be read from the data fields of this event.

Menu selection (user) event is triggered when the user chooses a menu item from the user menu. User menu of the software is created by the programmer and saved as a LabVIEW menu file. LabVIEW automatically replaces the default menu it uses with this custom menu created. Item tag and item path data fields of this event gives information about the selected menu item.

Key down event for an object is generated when the user presses a key while the application focus is on the object. The last object that the user has interacted (clicked, changed value etc.) is the object that has the application focus. The pressed key and whether or not the pressing is performed together with a modifier key (shift, alt, ctrl etc.) can be quarried from the event data fields.

User events mentioned above are created by the programmer to perform specific tasks, and will be explained in the following chapters.

CHAPTER 4

PULSE SEQUENCE DESIGN & DAQ SOFTWARE

4.1 Introduction

In this chapter “Pulse Sequence Design and DAQ” software, which is developed in this thesis, will be explained in detail. Throughout this chapter the “Pulse Sequence Design and DAQ” software will be referred as software shortly.

This chapter starts by explaining the purpose of the software, followed by the list of waveforms generated and measured by the software. After that the design processes of RF envelope waveforms and the gradient waveforms are explained, followed by the explanation of how pulse sequences are generated. Then the programming of the DAQ cards is discussed along with some new functionality added to the system. This is followed by the explanation of the file formats that are used by the software. The software then ends with the explanation of the events of the software. Also the graphical user interface of the software is explained in Appendix A.

4.2 Purpose of the software

There are two primary functions of this software; a tool to design pulse sequences interactively and a program that controls all of the data acquisition tasks related with a MRI experiment.

4.3 Waveforms generated and measured by the software

This software is responsible for all of the generation and measurement tasks of the METU MRI system. The list of waveforms generated and measured by the software is given in Table 4.1.

Table 4.1: The list of waveforms handled by the software.

Name	Type
RF Real Envelope	Analog Output
RF Imaginary Envelope	Analog Output
X Gradient	Analog Output
Y Gradient	Analog Output
Z Gradient	Analog Output
RF Gating	Analog Output
Current Source Control - S0	Analog Output
Current Source Control - S1	Analog Output
Echo Real Channel	Analog Input
Echo Imaginary Channel	Analog Input
Data	Digital Output
Shift	Digital Output
Latch Enable	Digital Output
Remote Enable	Digital Output

4.4 Generation of RF envelope waveforms

In METU MRI System two types of RF pulse envelope functions are used; sinc and rectangular. The sinc envelope is used to design slice selective soft pulses

where the pulse excites nuclei resonating over a narrow bandwidth of frequencies centered around the excitation frequency of the pulse. This type of pulse is used with a slice selection gradient in order to only excite the spins residing in a finite thickness slice. Hard pulses created by rectangular envelope function on the other hand, excite a wide bandwidth of frequencies of the spin system, therefore are not slice selective. The software parameters related with RF pulse envelope creation are designed to enable a more suitable input of soft sinc envelope pulses and hard rectangular envelope pulses but a short duration sinc pulse envelope can be used to design a hard pulse, or a long duration rectangular pulse envelope can be used to design a soft pulse upon request.

4.4.1 Design of slice selective sinc envelope pulses

These pulses are used to excite the spins in a finite thickness slice of a spin system. As stated previously they must be applied together with a slice selection gradient in order to modify the resonance frequencies of the spins in the imaging volume as a function of position along slice selection direction.

Two important parameters in the design process of a slice selective sinc pulse are; the thickness and the position of the slice to be imaged. These parameters are inputted by the user to the software and are the most important parameters during the creation of the sinc envelope function.

4.4.1.1 Setting the slice thickness

To understand how a finite thickness slice is selected, one must consult to the frequency spectrum of the sinc envelope function which is a rect function. The well known Fourier transform relationship between the sinc and rect functions is given in Equation 4.1.

$$A\sin c(2\omega t) \Rightarrow \left(\frac{A}{2\omega}\right) rect\left(\frac{f}{2\omega}\right) \tag{4.1}$$

According to the above relationship, the reciprocal of the first zero crossing of the sinc function determines the bandwidth of the rect function as illustrated in Figure 4.1.

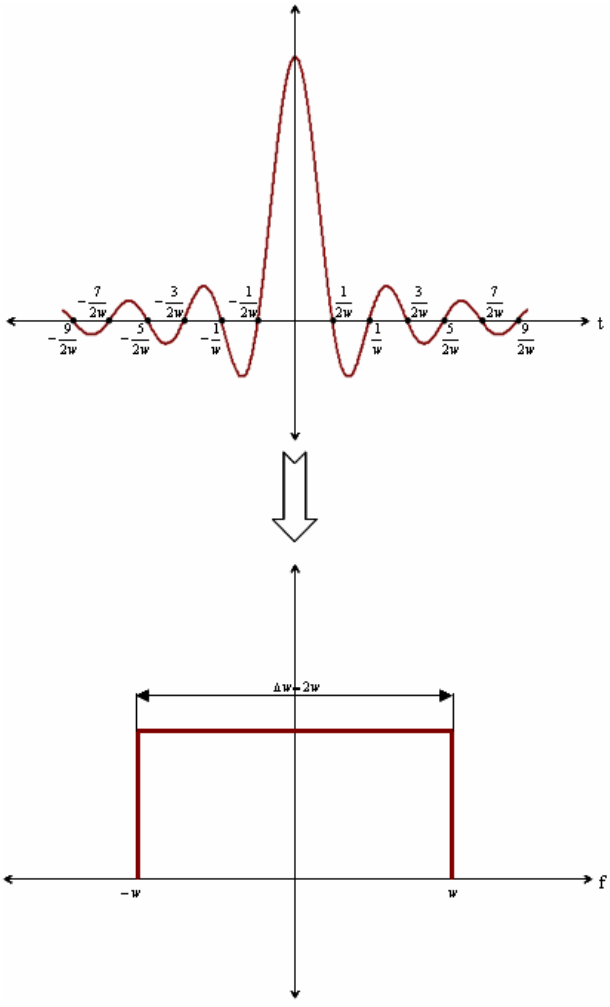


Figure 4.1: The Fourier pair of sinc and rect functions where Δω is the bandwidth of the rect function and also equals to the reciprocal of the first zero crossing of the sinc function.

Therefore a sinc envelope pulse will excite only the spins resonating in the Δω bandwidth discussed above. In the presence of a slice selective gradient the equation for the Larmor (resonance) frequency at position z is given below [13].

$$\omega(z) = \omega_0 + \gamma G_{sel} z \tag{4.2}$$

In this equation z is position along the slice selection direction, γ is the gyromagnetic ratio, ω_0 is the frequency due to the main magnetic field and G_{sel} is the slice selection gradient amplitude. According to this the slice thickness Δz can be given as:

$$\Delta z = \frac{\Delta\omega}{\gamma G_{sel}} \tag{4.3}$$

Since every point in the frequency domain corresponds to a different spatial position along the slice selection direction as a result of the slice selection gradient, a sinc envelope pulse will excite only the spins residing in the Δz volume.

The software takes the slice selection gradient amplitude and slice thickness parameters and calculates the necessary bandwidth. Then it generates the sinc pulse envelope by setting the first zero crossing of the sinc function according to the bandwidth value calculated and setting the amplitude of the pulse envelope according to the RF pulse amplitude parameter.

4.4.1.2 Setting the number of sidelobes and windowing

In order to achieve a perfect rectangular excitation profile in the imaging volume, an infinite length sinc envelope must be used. Since this is not possible in practice, sinc envelope must be truncated at some point. This truncation is defined by the number of sidelobes parameter. According to this parameter the

software only generates the given number of sidelobes along each side of the main lobe.

The truncation operation corresponds to the multiplication of the sinc envelope with a rectangular window in the time domain. This means that in frequency domain the Fourier transform of the sinc envelope is convolved with the Fourier transform of the rectangular window which is a sinc function. The result of this convolution process are the pulse truncation artifacts, namely the nonuniform excitation profile across the slice and the cross-talk artifact which is the unwanted excitation of the spins in the neighboring slices to a varying degree [13].

In order to lessen these artifacts, number of sidelobes can be increased up to the pulse time of 10 ms. which is a limit imposed by the RF amplifier unit in the system. Another method that can be used is the application of a window to the RF pulse envelope. The window can be applied by turning on the window parameter, and is the Blackman-Harris window that is used from the LabVIEW signal processing functions. Its equation is given below.

$$Y_i = X_i\left[0.422323 - 0.49755\cos(\omega) + 0.07922\cos(2\omega)\right] \quad \text{for } i = 0,1,2,\ldots,n-1 \quad \omega = \frac{2\pi i}{n} \tag{4.4}$$

In this equation X is the input sequence, Y is the output sequence and n is the number of elements in X.

4.4.1.3 Slice positioning

In METU MRI System the positions of slices parameter defines the position(s) of the slice or slices in the experiment. Using this parameter the software creates the necessary envelope functions for the RF channels.

In a MRI experiment where none of the gradient fields are turned on, the spins in the imaging volume precess at a frequency ω_0 that is directly proportional to the main magnetic field B_0 [13], as shown in Equation 4.5.

$$\omega_0 = \gamma B_0 \tag{4.5}$$

In order to meet the resonance condition and excite the spins, a RF pulse with a center frequency which is equal to the frequency of the spins must be applied. Therefore the frequency spectrum of the sinc envelope pulse which is given in Figure 4.1 is shifted as illustrated in Figure 4.2.

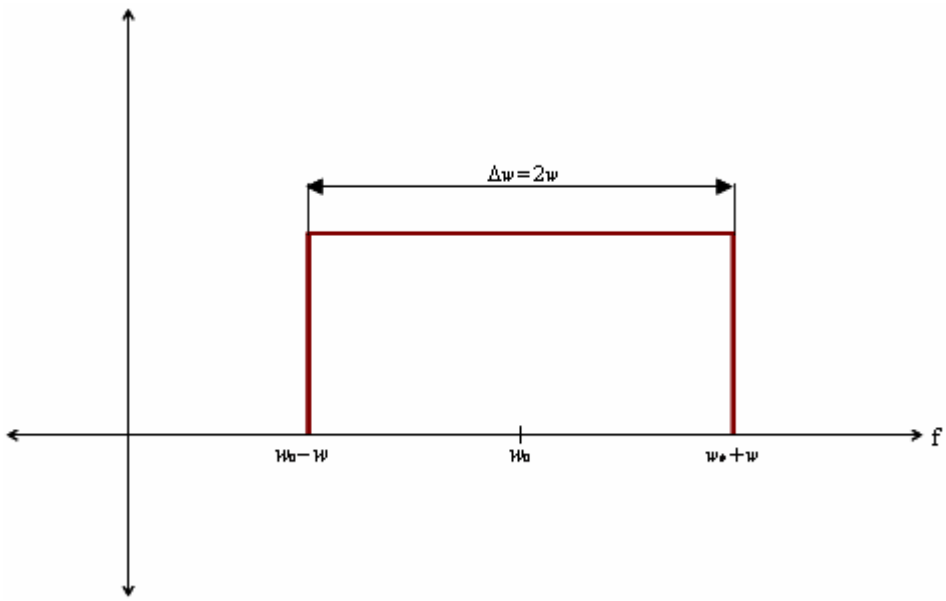


Figure 4.2: Frequency spectrum of the w_0 frequency sinc envelope pulse.

At the presence of a slice selection gradient, the precession frequencies of the spins in the imaging volume are modified along the slice selection direction according to the gradient. As a result spins at every different spatial coordinate

along the slice selection direction have a different precession frequency. In this situation to excite a slice other than the central slice, center frequency of the pulse must be shifted by additional ω_{sh} accordingly to the Equation 4.2. This case is illustrated for the sinc envelope pulse in Figure 4.3.

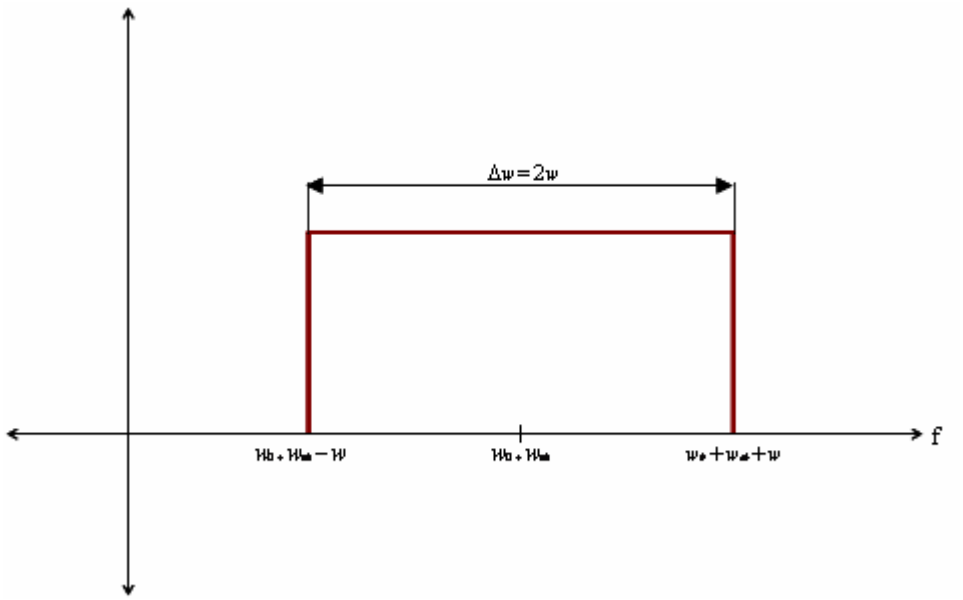


Figure 4.3: Frequency spectrum of a slice other than the central slice for a sinc envelope pulse.

By denoting the pulse envelope function as H(t), the equation for the applied pulse B_1 can be given as,

$$B_1(t) = H(t) \cdot \cos((\omega_0 + \omega_{sh})t) \tag{4.6}$$

By expanding the above equation we get,

$$B_1(t) = H(t) \cdot [\cos(\omega_0 t)\cos(\omega_{sh} t) - \sin(\omega_0 t)\sin(\omega_{sh} t)] \tag{4.7}$$

As can be seen from the above equation the RF pulses of the METU MRI System are quadrature amplitude modulated. This modulation is performed by the modulator part of the modem unit in the system.

The quadrature-phase envelope signals $H(t).\cos(w_{sh}t)$ and $H(t).\sin(w_{sh}t)$ are called the RF real channel envelope signal and RF imaginary channel envelope signal respectively. These signals are created by the software according to the parameters supplied by the user as explained previously, and are outputted from the D/A card of the system.

The high frequency $\cos(\omega_0 t)$ signal is generated by the frequency synthesizer unit of the METU MRI System. This signal and the phase shifted version of it are called the quadrature-phase carrier signals. These carrier signals are quadrature amplitude modulated by the RF pulse envelope signals in the modem unit, after that the created RF pulse signal is amplified by the RF power amplifier and finally, transmitted from the RF coil.

4.4.2 Design of non slice selective rectangular envelope pulses

These pulses are used to excite the spins in the whole imaging volume. Since rect envelope function is used for the pulse envelope, excitation profile will be in the form of a sinc function. A rect envelope pulse with duration τ_p will excite the spins over a frequency range defined by Equation 4.8 centered on the excitation frequency ω_{rf} [13]. As can be seen from this equation, by setting the pulse duration τ_p as small as possible, a wide range of spins can be excited. The software uses RF pulse duration and RF pulse amplitude parameters to generate this type of envelopes.

$$\left|\omega - \omega_{rf}\right| < \frac{2\pi}{\tau_p} \tag{4.8}$$

4.4.3 DC offset elimination

A DC offset is added to the echo signal after the reception due to the hardware components along the signal path. If this offset is not eliminated it will appear as a DC artifact in the image. In order to eliminate this offset a method common to MRI systems is utilized. In this method every step of a pulse sequence is applied once again with the RF envelope channels negated. These two repetitions are called the positive and negative DC offset elimination steps. As a result the magnetization vector is aligned with the RF coil such that the sensitivity of the RF coil will be in the opposite direction in each of these elimination steps whereas the offset will be the same in each step since it is added after the reception of the echo signal. When the data acquired from the negative DC offset elimination step is subtracted from the positive DC offset elimination step and the result is divided by two, the offset is eliminated as shown in Equation 4.9.

$$\frac{\left[\overbrace{(Echo + DCOffset)}^{PositiveStep} - \overbrace{(-Echo + DCOffset)}^{NegativeStep}\right]}{2} = Echo \tag{4.9}$$

DC offset elimination process is performed automatically by the software. Also the user has the option to view the RF envelope channel for each of the DC offset elimination steps when designing the pulse sequence.

4.4.4 Hahn and CPMG echoes

The echo type parameter of the software chooses between the Hahn and CPMG (Carr-Purcell-Meiboom-Gill) echo types [13]. These two techniques differ from each other only in the application of the 180 degrees pulse. In Hahn echo there is no phase difference between the 90 degrees and 180 degrees pulses whereas in CPMG echo, the 180 degrees pulse is 90 degrees out of phase from the 90 degrees pulse. The software changes the cos or sin functions, explained in the discussion about Equation 4.7, accordingly to generate this phase difference.

4.5 Generation of gradient waveforms

There are three gradient coils in the METU MRI System as explained before. These gradient fields generated by these coils are called X, Y and Z gradient fields. Also in MR-CDI experiments current is applied to the object to be imaged and since this also acts a gradient field, the current source control waveforms S0 and S1 are also evaluated as gradient channels by the software.

4.5.1 Gradient parameters

For the creation of the gradient waveforms, certain parameters must be inputted by the user. According to these parameters blocks of gradient waveforms can be designed, and by the concatenation of these blocks (performed by the software according to the timing information) gradient waveforms are created. These parameters are; channel, start time, amplitude, ramp up time, duration, ramp down time, and phase encoding.

Channel parameter specifies the gradient channel that the waveform block will be applied, and can be x, y, z, current-S0, and current-S1 (S0 and S1 are explained previously in Chapter 2). Start time parameter specifies the starting time of the gradient block in seconds. Amplitude parameter defines the amplitude in the units of volts. The gradient amplifiers of METU MRI System have a minimum rise and fall times of 1 ms., which is the reason for the usage of the ramp up time and the ramp down time parameters. Duration parameter specifies the time that the gradient will be on. Phase encoding parameter can be either ON or OFF, and determines whether the gradient block's amplitude will be varied with each phase encoding step or not. A sample block of a constant amplitude gradient waveform (i.e. slice selection, frequency encoding) is shown in Figure 4.4. The generation of a gradient with varying amplitude (i.e. phase encoding) will be explained in the next section.

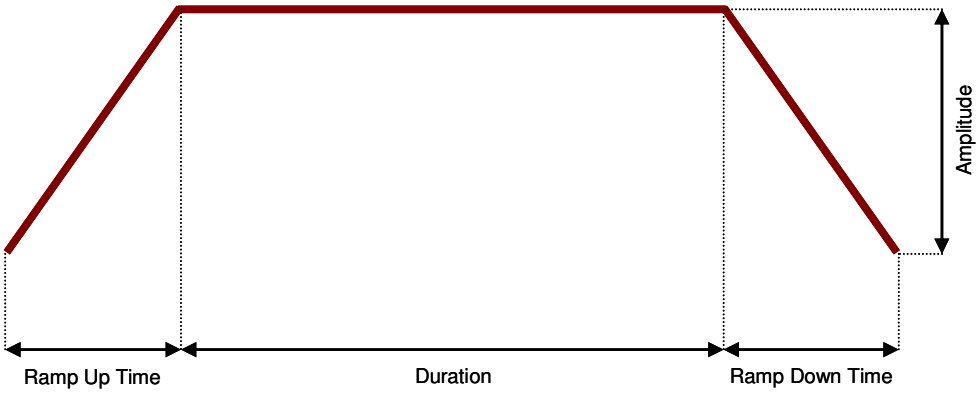


Figure 4.4: A sample block of a constant amplitude gradient waveform.

4.5.2 Generation of variable amplitude gradient waveforms

This type of gradient waveforms is used for generating the phase encoding gradient. In this type of gradient waveform, at every new phase encoding step, new gradient amplitude is generated. The amplitude parameter in this type of gradient waveform is interpreted as the maximum or minimum amplitude (according to its sign) of the gradient. Then a step size is calculated according to the Equation 4.10 where ΔA is the step amplitude, A is the amplitude parameter and N is the number of phase encoding steps which is defined by the number of phase encoding steps (image size - phase direction) parameter.

$$\Delta A = \frac{|A|}{N/2} \tag{4.10}$$

If a positive value is entered as amplitude parameter, the software starts at that value for the first phase encoding step, and decreases that value by ΔA in each phase encoding step. If a negative value is entered as amplitude parameter the opposite of the above operation is performed. By this way phase encoding steps are generated in a staircase manner which is illustrated in Figure 4.5.

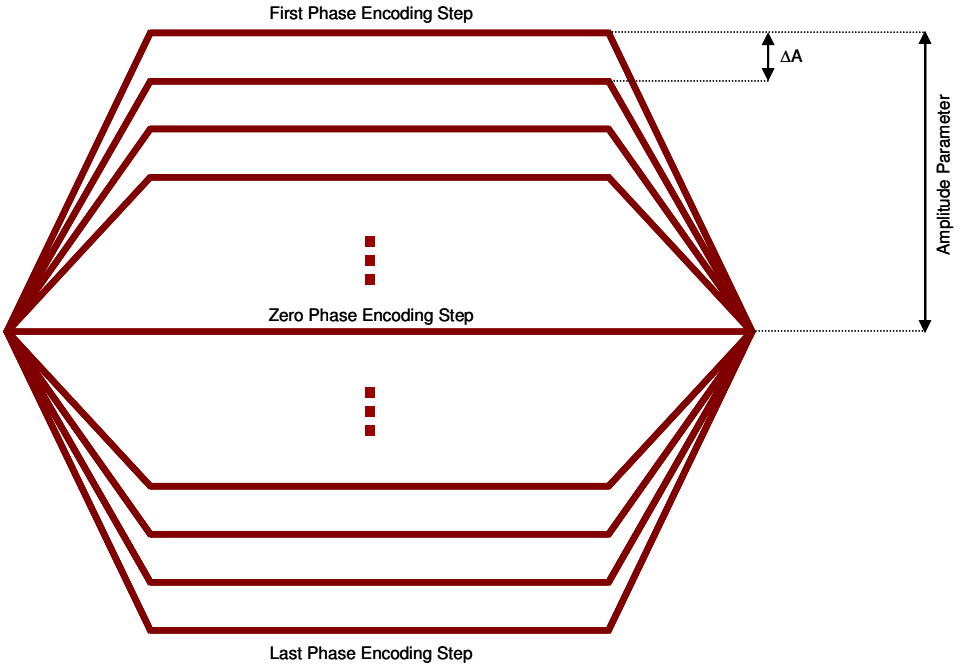


Figure 4.5: Variable amplitude gradient waveform generation for a positive amplitude parameter. For the negative case just the outputting order of the phase encoding steps are reversed.

4.6 Generation order of the cycles of a pulse sequence

A pulse sequence can be thought as the combination of all kinds of waveforms that are needed to get meaningful data from the imaging volume, namely the RF channel envelopes, gradients, current control and RF gating. These waveforms are generated and if necessary modified during generation process according to some software parameters which are explained below.

The number of phase encoding steps or image size - phase direction parameter defines the number of phase encoding steps that will be performed during the application of the pulse sequence. According to this parameter phase encoding gradients are created as explained previously.

The number of averages parameter defines the number of experiments (shortly denoted as NEX) that will be performed. The advantage of the averaging process is the increase in the SNR (signal to noise ratio) by the square root of the number of experiments done. The drawback is the increase in the experimentation time.

Number of slices in the experiment is defined by the number of entries in the positions of slices parameter, which is explained in the previous sections.

Before revealing further details about the application order of a pulse sequence, the definition of the term cycle must be given. Throughout this thesis one cycle of a pulse sequence corresponds to all the waveforms created for the duration that is equal to the repetition time T_R for a certain phase encoding step, a certain averaging, a certain slice and a certain DC offset elimination step. As an example a pulse sequence with 100 phase encoding steps, 4 averagings and 5 slices has 4000 cycles.

When generating the waveforms of a pulse sequence, the software starts with the first phase encoding step and the first averaging, and it generates the positive DC offset elimination steps for all of the slices in the sequence consecutively, then the negative DC offset elimination steps for all of the slices in the sequence are generated. After that the above steps are repeated again for the first phase encoding step but the second averaging. When all of the averagings are performed the software begins generating the waveforms for the second phase encoding step. The above procedure continues until all phase encoding steps are completed. This cycle generation order is illustrated in Figure 4.6.

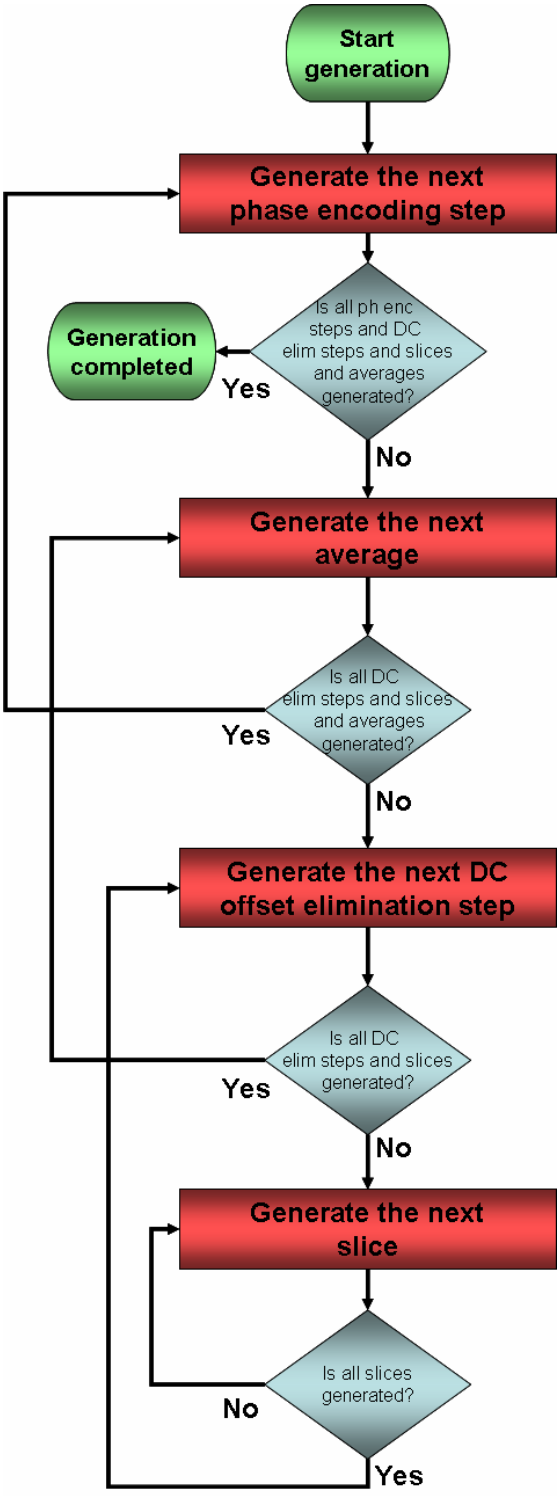


Figure 4.6: Cycle generation order of a pulse sequence.

4.7 Programming of the DAQ cards

The generation and receiving of MRI signals requires precise timing and synchronization across all system components. The hardware components of the A/D and D/A cards integrated to the system is previously explained in Chapter 2. In this section concepts related with the programming of these cards will be explained.

The cards are programmed under LabVIEW using DAQmx which is the latest generation driver for National Instruments family data acquisition cards. This driver has a complete set of VIs to access all the functionality of the cards. The PCI-6110E and PCI-6713 cards in the system are supported by the DAQmx versions 7.3 and later. The latest version of this driver at the time this thesis is written is 7.4.

Before explaining the programming details of the cards, the definition of the term task must be given. In LabVIEW programming a DAQmx task, or shortly task, is a collection of one or more channels with timing, triggering, and other properties. A task represents a generation or a measurement that the programmer wants to perform. In this software analog output, analog input, counter output, and digital output tasks are written.

These tasks are performed by the software in its two operation modes. These modes are called the single shot mode and the progressive mode. The single shot mode is the actual acquisition mode that is used to acquire data from the system. In this mode the software generates the pulse sequence and stores the results in a file. The progressive mode is a special mode where nearly all parameters of the pulse sequence and also some hardware settings of the system can be changed progressively on the fly while viewing the results simultaneously. These operation modes will be explained in detail in subsequent sections.

The analog output, analog input and counter output tasks are used for the generation of the pulse sequences and measurement of the echo signals, and they differ only slightly for the two modes of operation. Also these tasks are strongly depended on each other and performed together in a synchronized way, therefore they are explained together in the Section 4.7.1.

On the other hand digital output task is for controlling the frequency synthesizer unit and is not depended on the above trio so it is explained separately in Section 4.7.3. Another point to mention is that this task is performed only in the progressive operation mode of the software.

It is possible to divide the two operation modes of the software in to three main phases; the initialization, the main loop and the finalization. Therefore the above mentioned tasks will be explained as divided accordingly to these phases in the subsequent sections. General organization scheme for the data acquisition operations is illustrated in Figure 4.7.

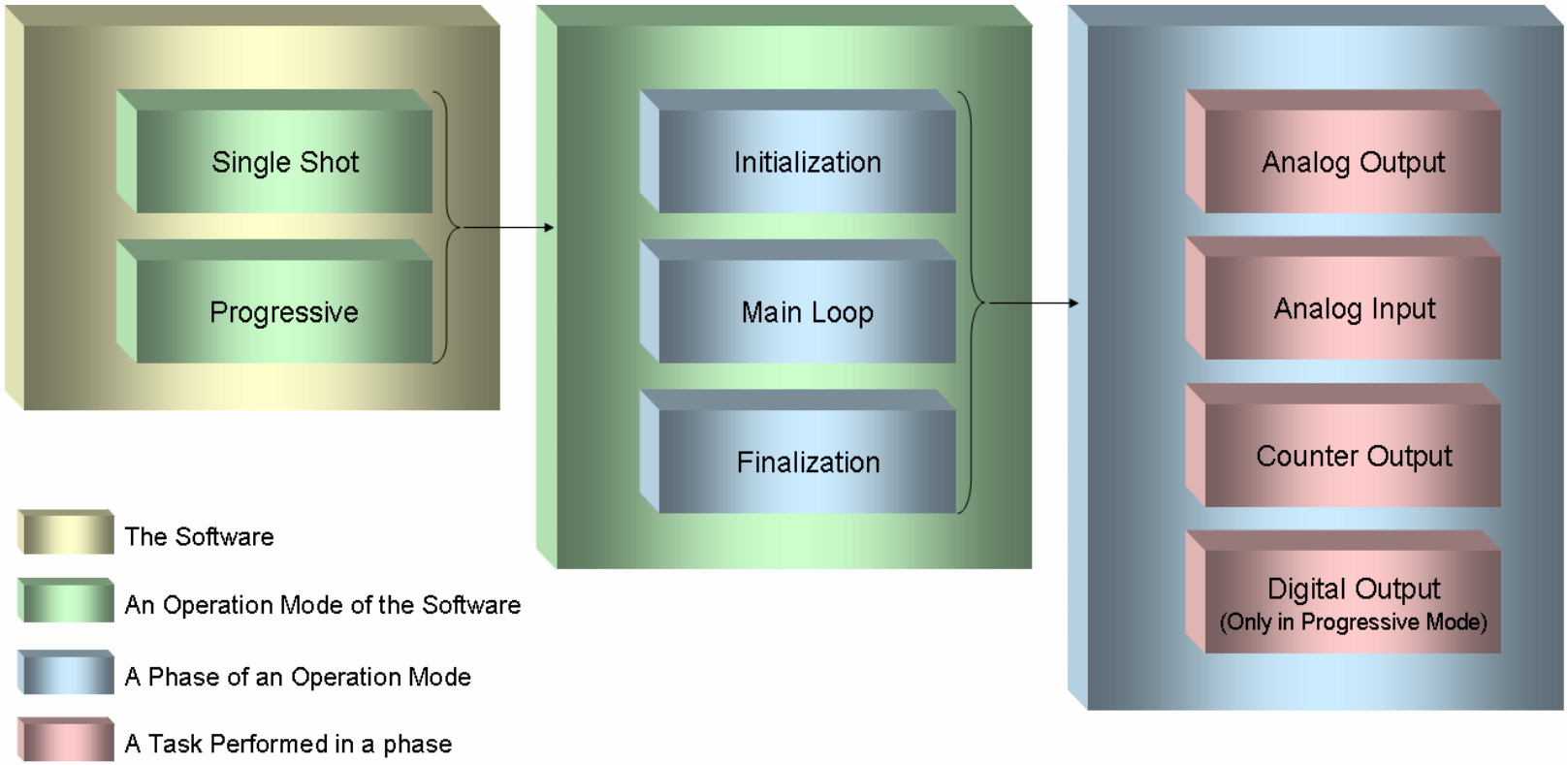


Figure 4.7: General organization scheme for the data acquisition operations.

4.7.1 Programming of the analog functionality

As shown in Table 4.1, the cards in the system output eight analog waveforms and input two analog waveforms during a pulse sequence. Analog output task is performed by the PCI-6713 D/A card whereas the analog input task is performed by the PCI-6110E A/D card. Counter output task is also performed, and will be explained in this section since it is used for the timing of the analog tasks. This counter output task is performed by the previously mentioned DAQ-STC section of the PCI-6110E card.

4.7.1.1 The initialization phase of the analog output task

For the analog output task, firstly channels are created according to the disable button on the software GUI for the current channel. This button controls whether or not the software will output the two current source control channels. If these are disabled then six analog output channels are created for analog output from PCI-6713 card, namely the RF real envelope, RF imaginary envelope, RF gating, x gradient, y gradient and z gradient. If current channel is not disabled then the software additionally creates the current source control S0 and current source control S1 channels.

As one may notice from the above channels, the RF gating channel and current source control channels are actually channels that contain digital waveforms. The reason for these channels to be outputted using the analog output section of the card is that they required to be in exact timing with the other channels, and the digital I/O section of the card does not provide the triggering functionality for that purpose.

After the creation of the channels the output rate is set according to the output rate parameter entered by the user and the card is programmed for continuous sampling mode using buffer. In this mode, an intermediate buffer is created in the host PC memory and the card generates the samples according to the data written in this buffer. This buffer is illustrated in Figure 4.8.

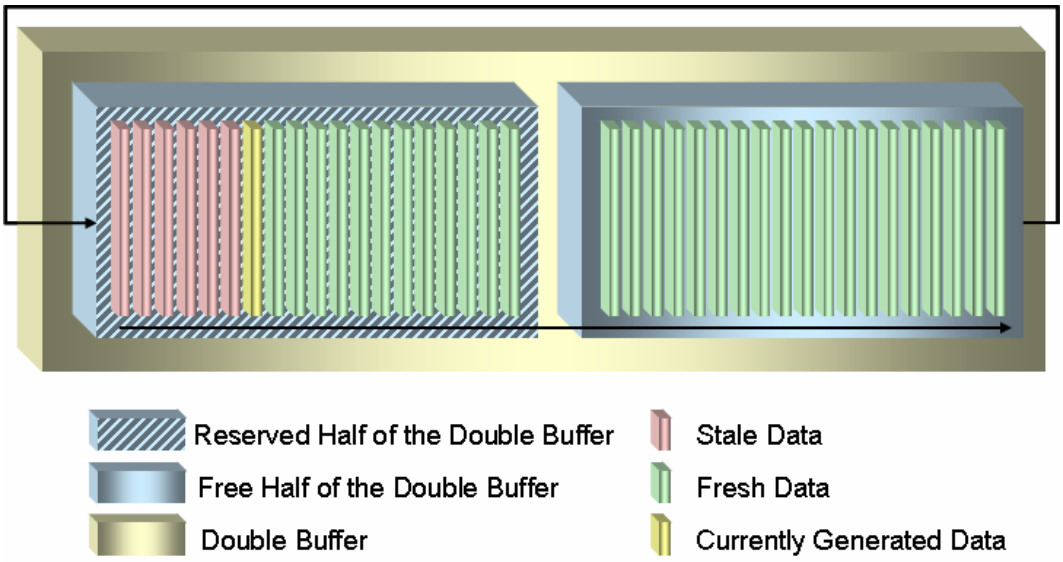


Figure 4.8: The double buffer used for the analog output task.

First in - first out logic is used for the generation of the data in the buffer and the generation process is continuous meaning that after generating all the data in the buffer, the card will try to generate the first sample in the buffer again if it is not replaced with fresh data. The continuous analog output process will continue until it is stopped by the programmer explicitly.

Allowing or not allowing of the regeneration of previously generated data in the buffer depends on the mode selected. Before explaining further issues, definitions of the terms fresh data and stale data must be given.

A data written in the buffer and waiting for its turn to be generated is called fresh data. After a fresh data is generated it is called stale data. In single shot mode the software is programmed in such a way that the regeneration of stale data is not allowed. However in progressive mode, regeneration is allowed, and stale data can be continuously regenerated until fresh data is written to the buffer. The reasons behind these regeneration settings will be explained in detail later in this chapter.

Other important issues concerning the buffer are the type, length and working principle of the buffer. The buffer created is a double buffer. The word double comes from the separation of the buffer into two halves. The size of the buffer is set in such a way that it is two times larger than the necessary space to hold all waveforms per pulse sequence cycle. The number of samples in a waveform (single channel) can be calculated by multiplying the T_R parameter with the output rate parameter. Therefore by multiplying this number with the number of analog output channels, the total number of samples in one cycle of a pulse sequence can be found. Finally the equation for the required buffer size in samples is given below where f_0 is the output rate and N_{ch} is the number of analog output channels which is either six or eight.

$$S_{buffer} = 2 \cdot T_R \cdot f_o \cdot \mathrm{N}_{\mathrm{ch}} \tag{4.11}$$

During the analog output operation the card reserves the first half of the buffer and start generating the samples located in that half. Since first half is reserved, the data located in it can not be changed until all the data in it is generated. But while first half is generated it is possible to modify the data in the second half. After the generation of first half is completed, the second half is immediately reserved and generation process begins. Clearly, now it is possible to modify the data in the first half. Because of the circular fashion of this process, the double buffer also called circular buffer.

The main advantage of using double buffer is that it creates CPU time for critical calculations. The calculations of the second half can be made during the time in which the first half is generated and vice versa. Ample CPU time is created for calculations since the generation time for one half of the buffer is equal to the repetition time (cycle time) T_R which is in 200-500 ms. interval in most cases This time is used for creation of RF envelopes and gradient channels for the next cycle, for processes that are related with the progressive mode (will be explained later), and for parallel running programs other than this software such as the “Image Reconstruction” software.

After the creation of the buffer, the state of the analog output task is changed to the commit state. In this state the hardware is not started for performing the task but programmed as much as possible according to the parameters that are mentioned above. By doing this, the fastest possible start of the analog output task in the main loop is guaranteed. This step completes the initialization phase of the analog output task.

4.7.1.2 The initialization phase of the analog input task

As similar to the analog output task, the first operation done in the initialization process of the analog input task is the creation of analog input channels. The names of these channels are; echo real channel and the echo imaginary channel. These channels are measured using the analog input section of the PCI-6110E card in the system.

After the creation of the channels, the sampling rate that will be used to acquire data is programmed according to the sampling rate parameter which is inputted by the user. Then the sampling mode is set as finite samples mode. In this mode the card will acquire finite samples of data in each pulse sequence cycle. The number of samples that will be acquired in each cycle (the data acquisition window) is determined by the image size - read direction (number of samples) parameter. Also the card is programmed in such a way that the task uses an intermediate memory buffer similar to the one in analog output task. But this time the buffer is a standard buffer with a size equal to the number of samples that will be acquired in each pulse sequence cycle.

The exact timing of the analog input task is very important since any shift in time corresponds to a shift in the acquisition window. The exact timing is achieved by programming a trigger for the analog input task for each pulse sequence cycle. The source of the trigger is programmed as the GPCTR0_OUT (general purpose counter 0 output) signal of the PCI-6110E card. GPCTR0_OUT signal is the output of the first counter of the previously explained DAQ-STC section of the

card. This trigger signal will be generated by the counter output task that is performed also on the PCI-6110E card. The type of the trigger is set as digital start trigger which means that the trigger is a TTL level trigger, and the analog input task will be started upon receiving it. The triggering operation for this type of trigger is edge sensitive, and rising edge is programmed as the triggering edge. This trigger signal is automatically routed by the DAQmx using the internal hardware of the PCI-6110E card since both the generation of the trigger and the task that is triggered are performed by the same hardware. Triggering scheme used is given in Figure 4.9. Triggering of counter output task which is also displayed in this figure will be explained in the next section.

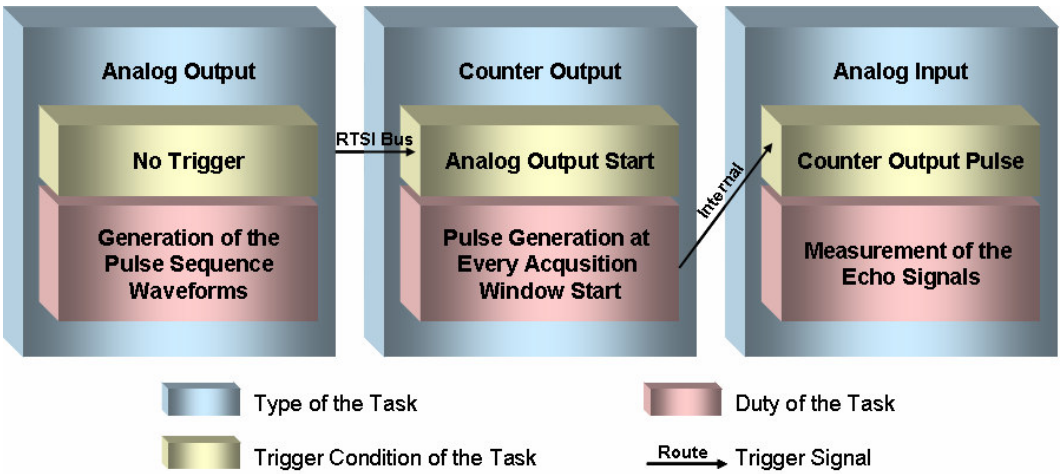


Figure 4.9: Triggering scheme for the analog tasks.

After setting the trigger, the card is programmed for not overwriting the unread data in the buffer. Therefore it is ensured that all data acquired is read and is processed by the software. An error will be returned by the software if the hardware tries to overwrite an unread sample in the buffer, but this will never

happen for the current version of the software since adequate time is provided to the software to empty the buffer.

The initialization phase for the analog input task is ended by committing the hardware. Since the analog input task is a finite sampling task, it must be started and stopped in each pulse sequence cycle in the main loop phase. Therefore if committing is not performed in the initialization part (before the first start command given to the hardware), the card will spend time for changing its state to committed for every start command given. By committing the hardware in the initialization phase, these unnecessary delays are prevented.

4.7.1.3 The initialization phase of the counter output task

This task is started with the creation of the counter output channel. Then the counter is programmed to create a digital pulse at its output with period T_R. Also an initial delay time that is equal to the start time of the data acquisition window is programmed according to the T_E parameter, image size - read direction or number of samples (N_s) parameter and sampling frequency (f_s) parameter as given below.

$$T_{Delay} = T_E - \frac{N_s}{2 f_s} \tag{4.12}$$

By this way at every pulse sequence cycle a pulse is created at the start time of the data acquisition window. Individual pulses of this pulse train are used as start triggers by the analog input task for acquiring data.

After setting the parameters for the pulse train, a rising edge sensitive digital start trigger for the counter output task is programmed. The source of this trigger is set as the WFTRIG (waveform trigger) signal of the PCI-6713 card. WFTRIG signal is an internal trigger signal of the card that starts the analog output generation. Since the sampling mode of the analog output task is set as continuous (task will

run until it is stopped by the programmer explicitly), this trigger signal will be generated just one time in a pulse sequence when the analog output task is started. Triggering scheme for analog tasks is previously illustrated in Figure 4.9.

By setting this trigger, the start of the pulse train generation is synchronized with the start of the analog output task. Therefore the pulse sequence generation and the analog input start trigger generation tasks are synchronized with each other. This trigger will be automatically routed by the DAQmx from the PCI-6110E card to the PCI-6713 card via previously explained RTSI bus that connects these cards.

Finally, the task is started. This is an actual start, not a commit operation as the previous tasks. Therefore the task will now waits for its trigger signal, and upon receiving it, will generate the pulse train. Generation will continue until it is stopped by the programmer explicitly.

4.7.1.4 The main loop phase of all analog tasks

As explained previously, a pulse sequence is composed of cycles that have duration of T_R. The main loop is actually composed of multiple stacked loops for generating the pulse sequence for all phase encoding steps, for all averagings, for all DC offset elimination steps, and for all slices. Since the term cycle is introduced for simplicity these stacked loops will be shortly called the main loop from now on. Therefore for each iteration of the main loop, a new cycle of a pulse sequence is generated.

Tasks performed in the main loops for the single shot mode and the progressive mode differs for the digital output task which is explained in a different section. For the analog tasks and the counter output task they are mostly the same, therefore they are explained together in this section.

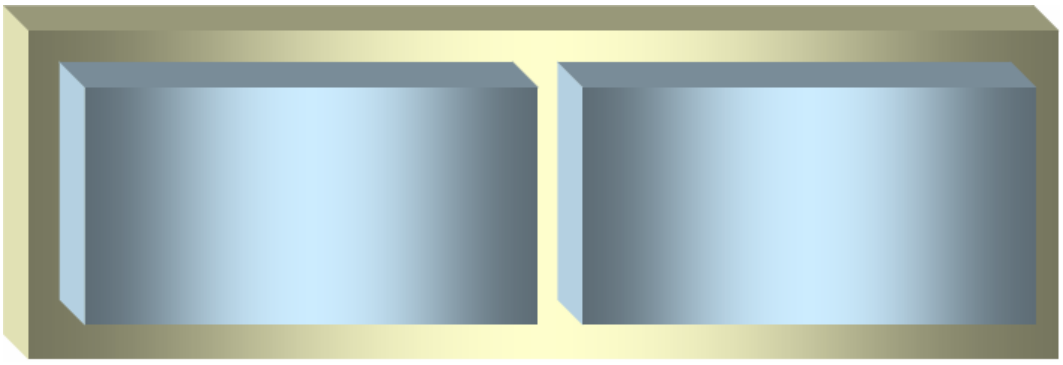
In the main loop of the software, waveforms for all channels are generated according to the previously explained parameters supplied by the user. Operations performed in the main loop can be examined in three cases according to the iteration number of the loop. These cases are for the first iteration, the second iteration and the third and greater iterations. The state of the double buffer during the operation of the main loop is illustrated in Figure 4.10.

At the first iteration of the loop which corresponds to the first cycle of the pulse sequence, the only operation performed is the writing of all channel data of the first cycle into the first half of the double buffer. It is important to notice that this operation is only a buffer write operation. Neither analog output nor analog input tasks are started. So generation and measurement operations are not started, also since previously started counter output task is waiting for the start trigger of the analog output task, the pulse train generation is also not initiated at this iteration.

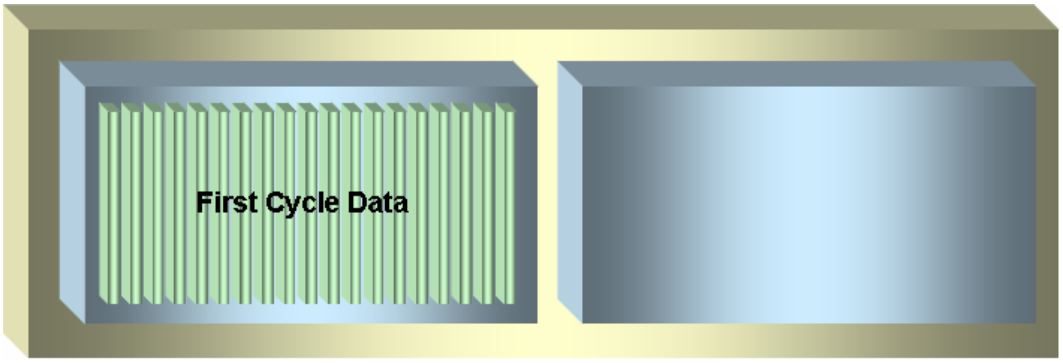
At the second iteration, writing of the all channel data of the second pulse sequence cycle into the second half of the buffer occurs firstly. After that the analog input task is started, so this task will now begin waiting for its trigger which will be generated by the counter output task. The analog output task is started next. So the card starts to generate all of the waveforms of the pulse sequence. An important point to mention is the samples generated by the cards. The generated samples are the data written into the double buffer during the first iteration of the main loop. Therefore at second iteration first cycle of the pulse sequence waveforms are generated whereas the waveforms for the second cycle are not generated but readied.

The start of the analog output task also triggers the counter output task, and the generation of the pulse train that determines the start of the data acquisition window has started. A buffer read operation for the analog input task is performed next. This operation waits for the card to perform the measurement of the echo signals initiated by the trigger that is generated by the counter output task.

(a)

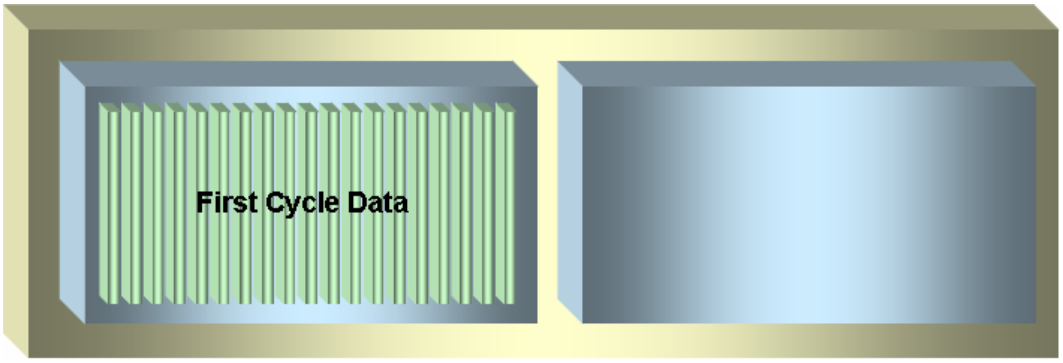


(b)

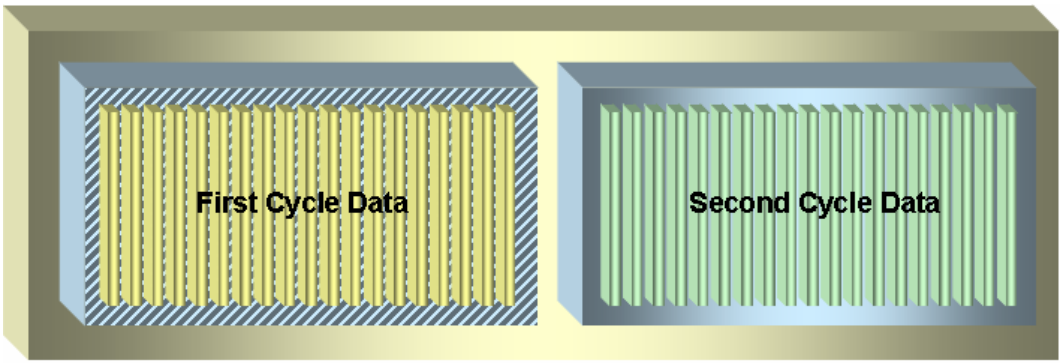


(c)

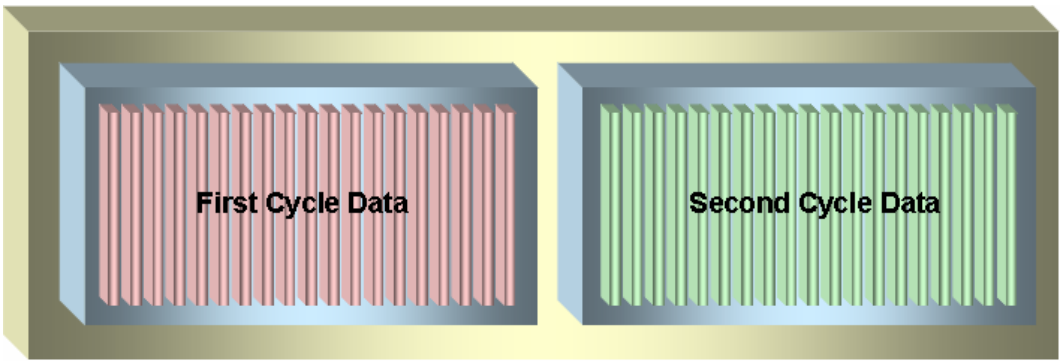
Figure 4.10: State of the double buffer during the operation of the main loop.
(a) Legend, (b) At the start of the first iteration, (c) During first iteration, (d) At the end of the first iteration - at the start of the second iteration, (e) During second iteration, (f) At the end of the second iteration - at the start of the third iteration, (g) During third iteration, (h) At the end of the third iteration - at the start of the fourth iteration, (i) During fourth iteration, (j) At the end of the fourth iteration - at the start of the fifth iteration.



(d)

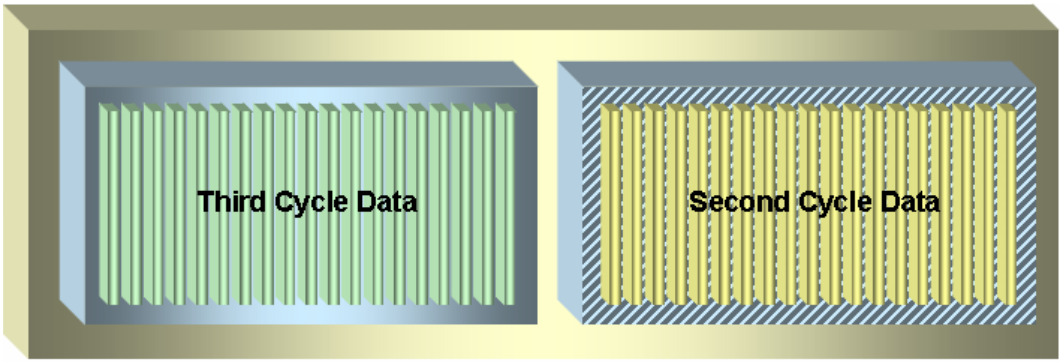


(e)

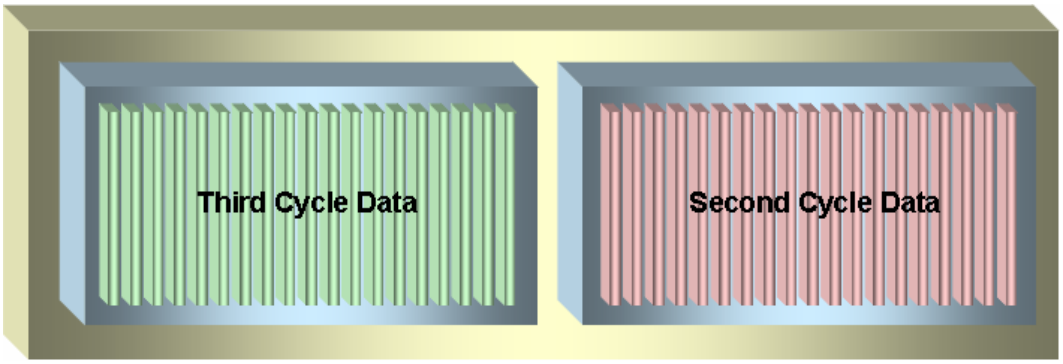


(f)

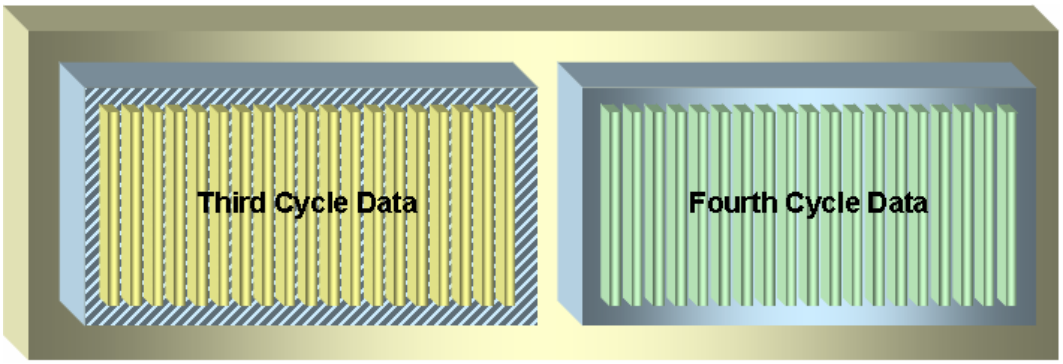
Figure 4.10 (continued)



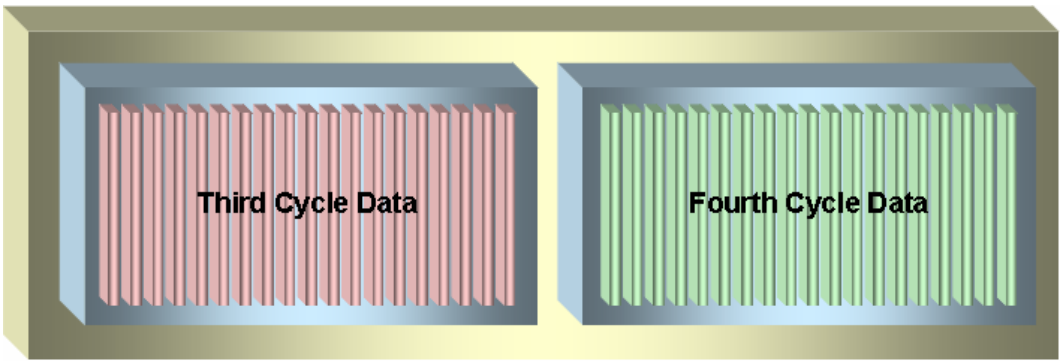
(g)



(h)



(i)



(j)

Figure 4.10 (continued)

The measured data is automatically written by the card into the previously configured input buffer. When this happens, buffer read operation reads this newly acquired data, and makes it available to the other components in the software. In both operation modes of the software, the acquired data is passed to the "display current data" VI which will be explained later. For the single shot mode the data is also stored to a file. The format of the file and the additional details of these file operations are given in the subsequent sections.

After storing the echo signals, the end of the analog input task is waited to make sure that the task is finished before stopping it. When the finishing of the task is confirmed, the task is stopped, and this is the end of the second iteration.

The operations performed for the third and greater iterations resemble the ones performed at the second iteration. The only difference is the analog output task is not started this time since it is a continuous sampling task that needs to be started just one time which happens at the second iteration.

Therefore at third iteration the first half of the output buffer previously containing the first cycle data will now be available, so the software writes the data for the third cycle there. Also the second cycle data that is written previously to the second half of the buffer will now be generated. This process will follow the same procedure for the subsequent cycles by switching the halves of the output buffer to be used. To be more specific; generating previous cycle's data from one half whereas writing the current cycle's data to the other half to be generated by the next iteration.

4.7.1.5 The finalization phase of all analog tasks

As one may notice from the above discussion, at the last iteration of the main loop all cycles of the pulse sequence except the last one are generated and their corresponding echo signals are acquired. Also the last cycle's data is written into the output buffer, and ready for generation. So first thing that is needed to be

done in the finalization phase is the generation of the waveforms and measurement of the echo signals for the last cycle.

Since the analog output task and the counter output task will continue until they are explicitly stopped by the programmer, the card will generate the last cycle and the counter will generate the pulse that will trigger analog input. So nothing is needed to be done for these tasks. But since the analog input task is a finite sampling task that must be started and be stopped for each cycle, the finalization phase begins with starting the analog input task for the last time in order to acquire the last cycle's echo signals. Then the acquired data is read from the buffer and passed to the "display current data" VI (is explained later in this chapter) in both operation modes and also stored in a file in single shot mode. After that, the end of the analog input task is waited, and when the end is confirmed the task is stopped. These steps are same as the ones performed in the main loop iterations except the first iteration.

After these, the analog input task is cleared. By clearing the task, the resources that were reserved previously by the task are released therefore enabling the usage of these resources (hardware) by the other parts of the software or by other programs. The analog output task is then stopped and cleared. This is followed by the stopping and clearing of the counter output task and this step concludes the finalization phase.

4.7.2 Timebase synchronization

This operation is performed in the initialization phase, and is very critical for the correct operation of the system. In this operation, synchronization of the timebases among two cards is performed. As explained before both of the cards use 20 MHz timebases for all of their internal operations. If this timebases are not perfectly synchronized, the tasks that are performed on separate cards will become out of synch. For example, the acquisition window will slightly shift in time which will result in the early or late acquiring of the echo signals. These

shifts will add up and after some time the echo signals will be completely out of the bounds of the acquisition window.

In order to prevent effects like the one explained above, the cards are programmed so that they share the same timebase. The timebase of the PCI-6110E card is used as the master timebase and the PCI-6713 card is set to ignore its own timebase but to use the other card's timebase. The timebase is shared via RTSI bus, and the routing of the timebase signal is performed automatically by the DAQmx.

4.7.3 Programming of the digital functionality

As given in Table 4.1, there are four digital output channels. These channels are generated by the digital output task, and are used for the control of the frequency synthesizer unit in the system. The names of these control signals are data, shift, latch enable and remote enable. Detailed explanations of these signals are given in Chapter 2.

As explained previously, the digital output task is only performed by the progressive operation mode of the software. Actually, there are four parallel running digital output tasks performed for each channel but for simplicity these tasks will be referred as a single task during the subsequent sections. TTL voltage levels are used for digital operations, so high logic and low logic corresponds to 5V and 0V signals respectively. Also all commands are negative true (i.e. to enable remote mode; the remote enable channel must be brought to 0V level) as stated previously.

4.7.3.1 The programming concepts related with the digital output task

As previously mentioned, both cards provide simple digital functionality since they are primarily designed for analog output and/or analog input operations. As a result, the cards do not provide triggering functionality for their digital I/O

lines. Therefore the digital output task can not be performed as a completely hardware timed task as the analog tasks. However since data channel must be driven according to the shift channel that provides pulses for the serial shift operation, a timing scheme between these two must be created. Also similar timing considerations apply to the latch enable channel.

Since hardware timing is not possible, the necessary timing is implemented in the software. This is accomplished by utilizing the dataflow based program execution architecture of LabVIEW. In this method, the error cluster inputs and outputs that are present in every DAQmx VI are utilized. These clusters are used primarily for carrying error information through out the program for error handling. But they can also be used in an alternative way by connecting the error out terminal of one DAQmx VI that performs an operation on one channel to the error in terminal of another DAQmx VI that performs an operation on a different channel. Therefore a data dependency between multiple channels is created in the software by ordering the value change times of the channels appropriately.

Since this timing solution is implemented purely on the software side, there will be small generation delays that will be depended on how fast the software is executing when a specific digital output operation is requested. As a result the high time or low time of a digital signal generated will be depended on the execution speed, but since the data dependency is created between the channels this will not cause any problems. Although timings will vary, data in each channel will be consistent with each other at all times. This situation is illustrated in Figure 4.11.

In this figure a 10 bits long data signal (actually data signal is 36 bits long) is driven serially into the card. The data signal used in the illustration is 1001010011. The frequency setting is correctly set despite varying duration pulses.

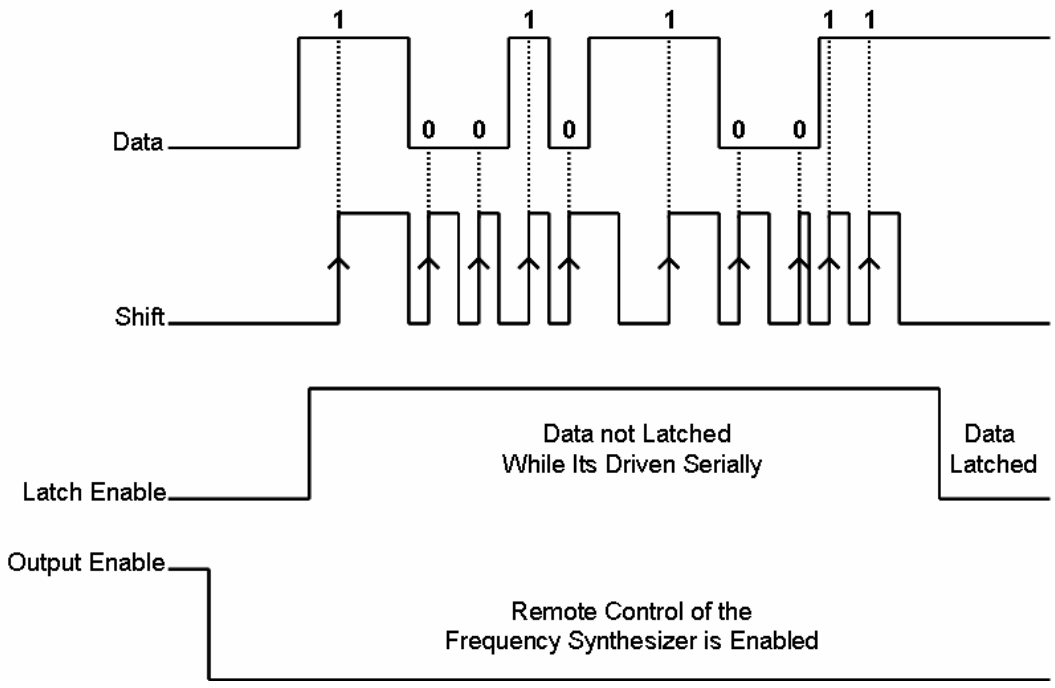


Figure 4.11: Frequency synthesizer control signals.

4.7.3.2 The initialization phase of the digital output task

The initialization phase of the digital output task is simple. Firstly, the above mentioned four digital output channels are created, and then the digital output task is started for all channels. The start of the task means that the channels are now ready for the digital output operations.

Since no timing scheme is specified for the generation of the samples in the initialization phase, LabVIEW uses the default which is the on demand timing. In on demand timing the samples are generated as soon as they are written to the buffer, and also it is ensured that all the samples in the buffer are generated before continuing program execution.

4.7.3.3 The main loop phase of the digital output task

The waveform generation scheme for the main loop phase of the digital output task is illustrated previously in Figure 4.11. The flowchart of the operations performed in this phase is given in Figure 4.12.

In the main loop phase, firstly the state of the local/remote mode selector parameter which switches the frequency synthesizer unit between the local and remote operation modes is checked. In the local mode, frequency that is inputted using the knobs on the device is outputted by the frequency synthesizer unit whereas in the remote mode, frequency is controlled by the user from the software. If the status of the local/remote mode selector parameter had been recently changed by the user, an output operation is performed according to this parameter; outputting high for local mode and low for remote mode.

After that the data signal that specifies the frequency setting inputted by the user is driven serially to the frequency synthesizer control card. Just like the above process, this is performed only if the user had recently changed the frequency setting from the software. To perform this operation, firstly the latch enable channel is brought to high level for preventing the frequency synthesizer unit to change its frequency simultaneously with the serial driving of the data into the card.

This is followed by a loop for driving the data into the card. This loop iterates for 36 times which is the total number of samples in the data signal that determines the frequency setting as explained in Chapter 2. At each iteration, first the shift channel is brought to low state, then the data channel is brought to the current sample's state, and finally the shift channel is brought to high state. The last operation causes a low to high transition (a positive edge) in the shift channel, which causes the shifting of the current data into the shift registers in the card. Therefore at the end of the last iteration, the frequency data had been completely shifted into the shift registers.

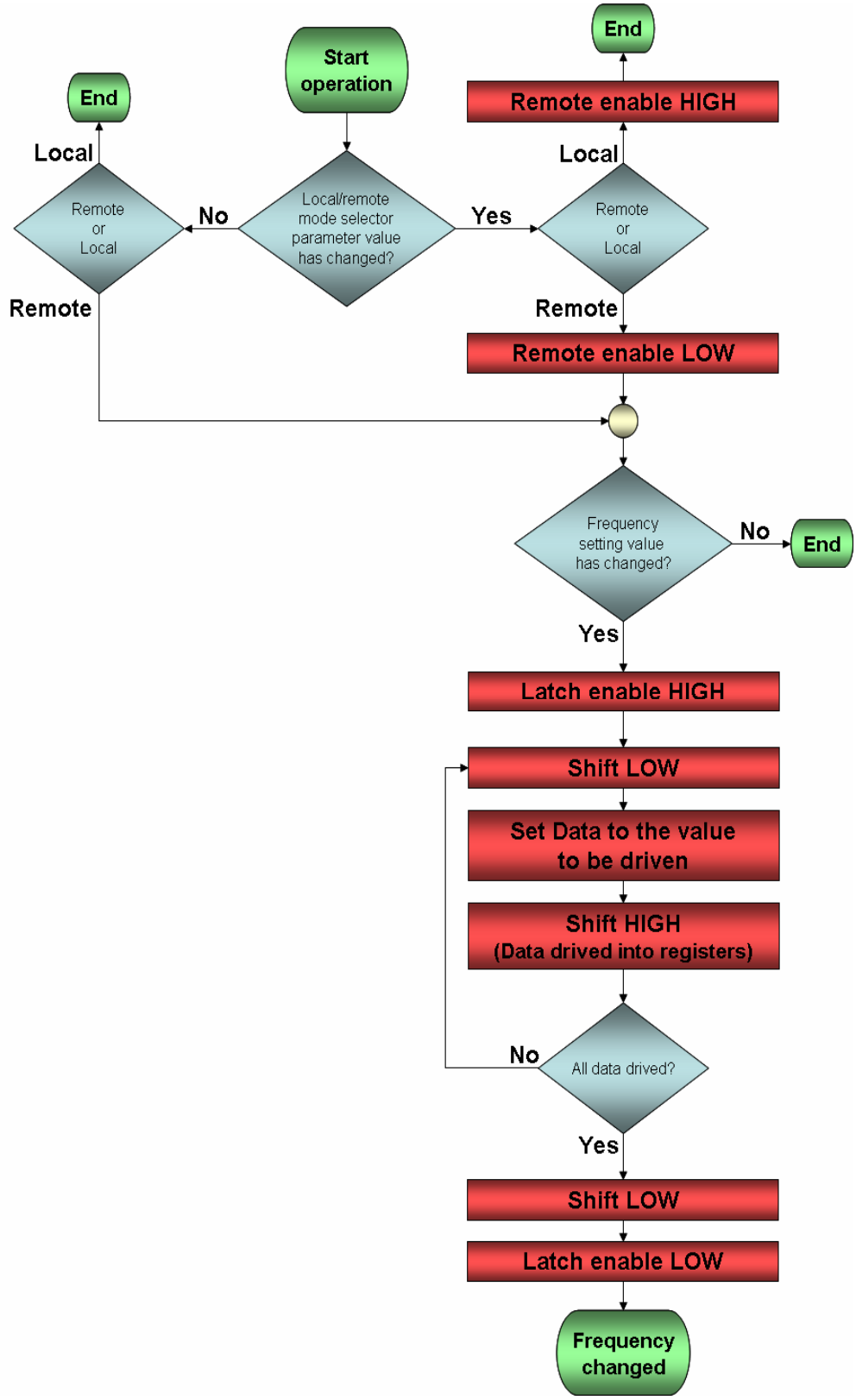


Figure 4.12: Flowchart of the operations performed in the main loop phase of the digital output task.

After that, the states of the shift channel and the latch enable channel are brought to low. The latter one causes the enabling of the five latch enable lines of the frequency synthesizer unit since all of these are driven by the latch enable channel. Therefore the frequency synthesizer unit reads the frequency data and changes its frequency according to it.

4.7.3.4 The finalization phase of the digital output task

In the finalization phase, digital output task (actually the four separate digital output tasks as previously explained) is stopped and then cleared, therefore the resources that are reserved by this task are released.

4.7.4 Functionalities of the progressive mode

There are some additional functionalities of the progressive mode apart from the ones explained before. These functionalities are discussed below.

In this mode disabling of any requested channel(s) is possible. When a channel is disabled, it is not generated from the card. Disabling is used primarily when adjusting the pulse sequence (i.e. disabling all channels except the RF envelopes and slice selection gradient to adjust the tip angle by changing the RF pulse amplitudes progressively). By this functionality powering the gradient amplifiers on and off all the time is no longer necessary.

As explained before, this mode allows the user to change the frequency setting of the frequency synthesizer unit. Utilizing this functionality, frequency sweeps can be performed. By this way, previously manually done center frequency setting task can be done automatically. Start, increment and end frequency values for the sweep is inputted by the user, then the software performs the sweep by detecting the peak amplitude of the received echo signals. After that the frequency value for the detected peak can be used during experimentation.

Also there is an option to change the amplitudes of all RF pulse envelopes simultaneously. If this option is selected, then when the user changes the first pulse's amplitude, the amplitudes of the other pulses are set as twice that amplitude. Also if the user changes the amplitude of a pulse other than the first one, the first pulse's amplitude is set as half of that amplitude. This method provides a very convenient way to adjust RF pulse amplitudes.

Voltage sweeps for sweeping the amplitudes of the RF pulse envelopes can also be performed in this mode. This feature provides automatic tip angle adjustment process which is previously done manually. Start, increment and end voltages are inputted by the user, and then the software performs the sweep with detecting the amplitudes of the received echo signal. After that the voltage value that corresponds to the peak amplitude can be used for experimentation.

4.8 Formats of the files used by the software

In this section format of the files that are used by the software are explained. The software uses configuration settings files with the extension "ini", pulse sequence files with the extension "psq" and also data files with the extension "dat". The purpose and format of these file types are explained in the following sections.

4.8.1 Configuration settings file

This file type is used by the software to store the critical hardware limits and conversion coefficients. These limits and coefficients are explained later in this chapter. Any change in this file must be made with great caution since the software uses the limits specified in this file while performing error checking on the pulse sequences created by the user. Entering these limits incorrectly may damage the MRI System hardware, especially the preamplifier, the RF amplifier, and the gradient amplifiers.

This file is a plain text file like any other ini file which is the standard format of a Windows family operating system to store configuration settings. The file is composed of names of main sections in brackets, and within each section keys that define the name of the parameters (limits, coefficients etc.) followed by an equal sign and the value associated with these parameters. The keys in this file can either be changed from software directly or can be edited by using a text editor.

4.8.2 Pulse sequence file

This file type is used by the software to store all the information related with a pulse sequence. The file format is shown in Figure 4.13.

The detailed explanation of the parameters stored in the file has been given throughout this chapter. This file is a binary file with the default file extension of "psq", but this file extension is just used for easy recognition of these files in the operating system, and can be omitted or can be changed. The details of the save and load operations will be explained in the subsequent sections.

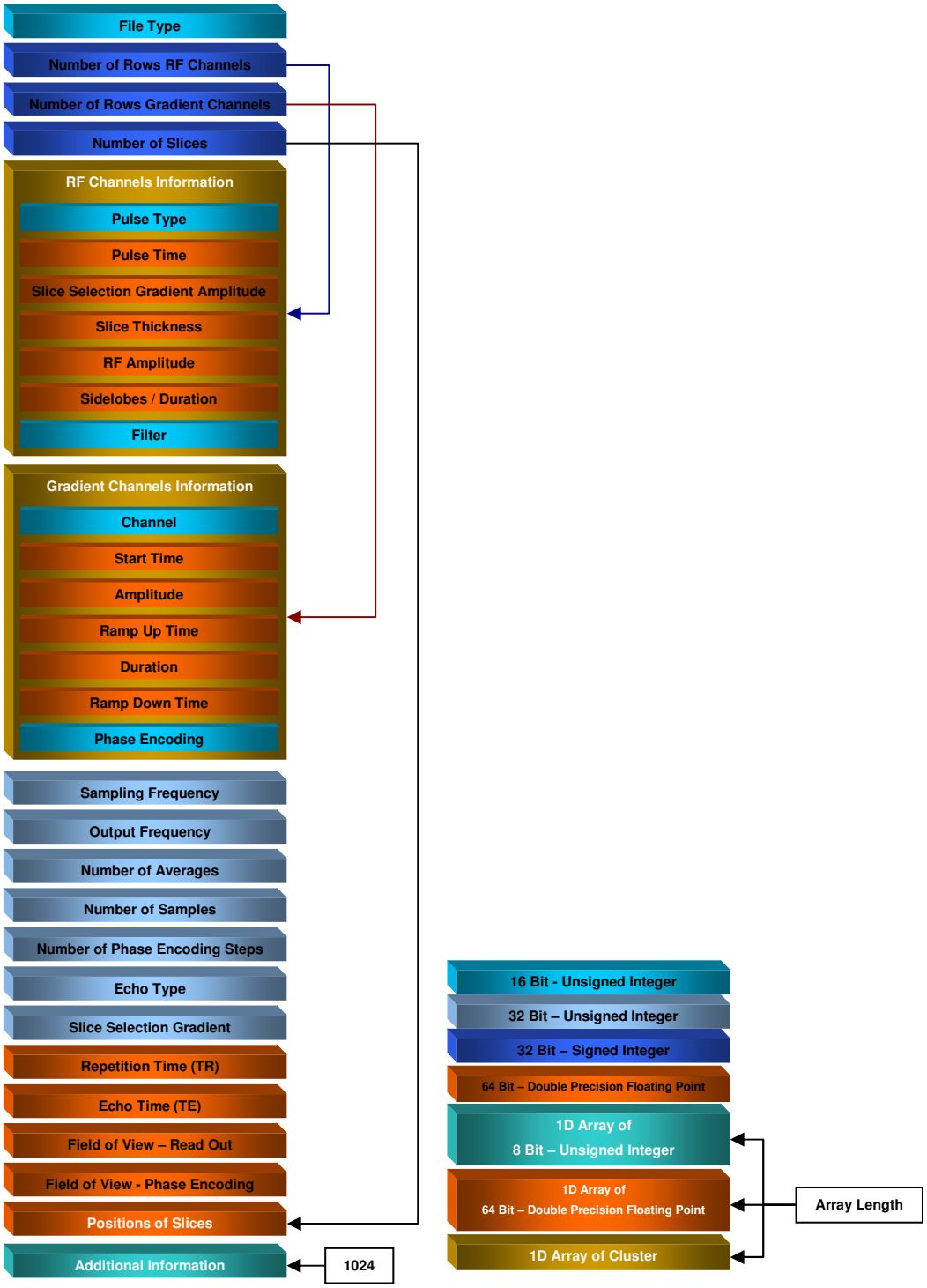


Figure 4.13: File format of a pulse sequence file.

4.8.3 Data file

This file is used primarily by the “Image Reconstruction” software but “Pulse Sequence Design and DAQ” software can also use it. This file is used for storing echo signals but also pulse sequence data is recorded at the beginning of this file, and this software can load that pulse sequences. This file type will be explained in detail in Chapter 5.

4.9 Events of the software

As mentioned previously, all functionality of the software is configured as events. In this section explanation of these events is given.

4.9.1 Single shot button - value change event

In this event, single shot mode data acquisition operations previously explained in this chapter are performed. As mentioned previously, in single shot mode received data is stored in a data file. So this event first creates this file and stores the pulse sequence information at the beginning of the file. By this way the user does not have to remember which data is acquired with which pulse sequence.

In single shot mode the regeneration of data in the output buffer is not allowed as mentioned previously. The reason for this choice is to make sure that the pulse sequence generation is done without any delays. Since the sampling mode is continuous for pulse sequence generation (analog output task), if host pc’s CPU is unable to calculate the waveforms in the time provided, then the card will try to regenerate a stale sample in the buffer. This situation may happen when the CPU is excessively used by another program that runs at the same time the data acquisition process is in progress. This causes an error and operation is stopped. By this way an automatic error checking mechanism is created.

Another thing to mention is that this event continuously displays generated waveforms cycle by cycle while generating the pulse sequence. It also displays the current progress of the pulse sequence in percents.

4.9.2 Progressive button - value change event

In this event, progressive mode data acquisition operations previously explained in this chapter are performed. In this mode every parameter of a pulse sequence can be modified simultaneously viewing the resulting echo signals in another subVI which is “display current data” VI. The front panel of this subVI is displayed on the “Image Reconstruction” software using the subpanel component of the LabVIEW. By using this component the front panel of a VI can be loaded and displayed on the front panel of another VI. The “Image Reconstruction” software is used for displaying the received echo signals since the software suit is displayed on two screens at the same time.

Since the user can modify every aspect of a pulse sequence in this event, he/she can also give erroneous inputs. When such thing occurs, the software detects this situation by error checking and displays an error message about the place and the type of the error. Naturally, time passes as the user reads this message. Since the sampling mode is continuous for pulse sequence generation (analog output task), the software will try to regenerate a stale sample in the buffer as a result of the time that is required by the user to respond to the error message. Therefore in this mode regeneration is allowed. During error conditions the software will continue to generate the last good state of the pulse sequence therefore no harm is done to the hardware.

Similar to the event for the single shot mode, this event continuously displays generated waveforms cycle by cycle while generating the pulse sequence and also displays the current progress of the pulse sequence in percents.

4.9.3 Stop button - value change event

This event is used to terminate the pulse sequence generation process at any time. In this event card is programmed to output zero for all channels.

4.9.4 Menu selection (user) event

This event is performed when the user clicked to a menu item of the software. Software menu is designed in LabVIEW and is saved into a rtm (real time menu) file which replaces the original LabVIEW menu.

Nearly all of the software functionality can be reached from the front panel of the software, so the menu of the software is simple. The software has a file menu to create, open, save pulse sequences and exit from the software. It has a zoom menu for controlling the zoom options to be used when displaying the pulse sequences. Also it has an about menu for displaying the version information about the software.

In this event the selected menu item is detected according to its item tag, and appropriate calls to generate specific user events (are explained later in this chapter) are performed. Load user event, save user event, plot user event is generated when the user gives open sequence, save sequence and any of the zoom commands respectively.

4.9.5 Save - user event

This user event is called to save the designed pulse sequence to a pulse sequence file. The file format is given in Section 4.8.2, and the parameters that are stored in the file are explained throughout this chapter. Only file type parameter is needed to be explained. This parameter defines the type of the file as a pulse sequence file or a data file, since both files start with pulse sequence information.

4.9.6 Load - user event

This user event is called to load a pulse sequence that is stored in a file (pulse sequence file or a data file). After pulse sequence is loaded, it is drawn to the screen.

4.9.7 Accept settings button - value change event

This event is written to change the minimum and maximum limits of the critical system parameters. A list of these limits along with their purposes and the faults that will be observed if these limits are not chosen with care is given in Table 4.2.

Also in this event, x, y and z gradient coefficients can be changed. These coefficients are used to convert the amplitude units that are used to display the gradient waveforms to each other. These units can be chosen as volts or G/cm. All pulse sequence amplitude information except the slice selection gradient amplitude (G_{sel}) is stored in the units of volts since volt is the actual unit that will be used by the cards and not depended on other factors.

These coefficients are used according to the following equation, where $G_{G/cm}$ and G_{Volts} are the amplitude of the gradient in G/cm and V units respectively, and C_G is the coefficient for that gradient channel.

$$G_{G/cm} = \frac{G_{Volts}}{C_G} \tag{4.12}$$

The currently displayed pulse sequence is redrawn according to the changes in these coefficients. Also an error check is performed to inform the user whether the previously designed pulse sequence complies with the current limits or not.

Table 4.2: Critical limits of the software.

Name of the limit	**Purpose of the limit**	**Faults observed if not chosen with care**
RF Amplifier turn on time before pulse	In order to allow a settling time for the RF power amplifier when it is gated	A noisy RF pulse is created causing an improper excitation
RF Amplifier turn off time after pulse		
RF Pulse minimum voltage	In order to limit the amplitude of the RF pulses created	Severe damage to the preamplifier unit may occur
RF Pulse maximum voltage		
X Gradient minimum voltage	In order limit the amplitude of the gradient waveforms created	Gradient amplifiers will give overload and over temperature faults and will not amplify until these faults are removed
X Gradient maximum voltage		
Y Gradient minimum voltage		
Y Gradient maximum voltage		
Z Gradient minimum voltage		
Z Gradient maximum voltage		

4.9.8 Plot - user event

This event is a user event and it is called from other events when necessary. The plot user event handles all necessary operations concerning the creation of a pulse sequence. In this event, pulse sequence blocks are created as previously explained in this chapter. Also the created pulse sequence waveforms are displayed.

4.9.9 Add button - value change event

This event is used to add new blocks to the pulse sequence waveforms. In this event, pulse sequence parameters that are inputted by the user to the input fields are read, and blocks are created according to them. The information of the blocks are displayed in the tables on the software's GUI. Also pulse sequence is redrawn to show the newly added blocks. Naturally, these operations are only performed if the given input can pass the error check.

4.9.10 Change button - value change event

This event resembles the add button - value change event. This time the currently displayed block of the pulse sequence is modified according to the input received from the user. All operations performed for the add button - value change event is also performed in this event.

4.9.11 Delete button - value change event

This event is used for deleting a block of a pulse sequence. The block to be deleted can be chosen from the pulse sequence information tables on the GUI. This event deletes all stored information related with the deleted block and redraws the final state of the pulse sequence.

4.9.12 Swap gradient channels button - value change event

As its name implies, this event is used to swap a gradient channel with another. This pair is specified by the user. An important point to mention is that the amplitude values which are stored in the units of volts are not directly swapped since the field strength of a gradient may be different than the other one. The operation is performed by converting these values to the G/cm units by using the gradient coefficients. By this way field strengths of the gradients are correctly swapped. Error checking is also performed to check that the requested swap operation is possible. Also necessary GUI components are updated according to the results of the swap operation.

4.9.13 Change selected line's channel button - value change event

This event resembles the swap gradient channels button - value change event. But this time a selected block of a gradient channel is redesigned for another channel. Other operations performed are similar to the ones in the swap gradient channels button - value change event.

4.9.14 Calculate read-out button - value change event

This event calculates the necessary read out gradient amplitude that must be applied to generate a specified field of view in the read out (frequency encoding) direction. Field of view in the read out direction is calculated according to the equation given below [13]. In this equation G_{rd} is the amplitude of the read out gradient, f_s is the sampling frequency, γ is the gyromagnetic ratio and FOV_{rd} is the field of view in read out direction.

$$G_{rd} = \frac{f_s}{FOV_{rd} \cdot \gamma} \tag{4.13}$$

4.9.15 Calculate phase encoding button - value change event

This event calculates the necessary phase encoding gradient amplitude that must be applied to generate a specified field of view in the phase encoding direction. Field of view in the phase encoding direction is calculated according to the equation given below [13]. In this equation G_{ph} is the amplitude of the phase encoding gradient, N is the number of phase encoding steps, FOV_{ph} is field of view in phase encoding direction, γ is the gyromagnetic ratio and A_{ph} is the area under the phase encoding gradient.

$$G_{ph} = \frac{N}{FOV_{ph} \cdot \gamma \cdot A_{ph}} \tag{4.14}$$

4.9.16 Calculate single area button - value change event

This event is for calculating the area under a single gradient block. The result is displayed in a table, and is given in both Vs and Gs/cm units.

4.9.17 Calculate all areas button - value change event

This event calculates the areas under all gradient waveforms and displays them in a table in both Vs and Gs/cm units.

4.9.18 Calculate amplitude / duration button - value change event

The amplitude or duration must be given to create a gradient block that has a certain area specified by the user is calculated in this event. The calculation is performed by keeping one of the amplitude or duration parameters constant to find the other one that creates the requested area.

4.9.19 Start sequence browsing button - value change event

Sequence browsing is an easy way of browsing multiple sequences in a directory. In this event the directory specified is analyzed for files containing pulse sequence information. Also filtering can be applied to search for files that contain a specific pattern in their names. Number of files found that meets the filtering criteria is displayed to the user. Also the first file that is found is loaded and is displayed to the user.

4.9.20 Sequence browse directory - value change event

This is the event performed if the user changes the directory to be analyzed for sequence browsing. This event is a duplicate of the start sequence browsing button – value change event.

4.9.21 Next and previous buttons - value change events

These events are also for sequence browsing. In these events the next or the previous file that contains pulse sequence information in the directory specified is loaded and displayed to the user.

4.9.22 RF channels info and gradient channels info - mouse down events

In these events the clicked row of the table that displays information about the RF envelope or gradient channel waveforms is detected. After that the information contained in this detected row is written into the user input fields of the GUI for user interaction.

4.9.23 RF channels info and gradient channels info - key down events

These events are used to delete the selected block of a pulse sequence when the delete key is pressed in keyboard. New status of the pulse sequence is redrawn, and information in the table is updated after delete operation.

4.9.24 Slice selection gradient - value change event

This event is performed to define the slice selection gradient. In this event, the amplitude parameter (G_{sel}) of the slice selection gradient which is stored in the pulse sequence in the units of G/cm is converted to volts according to the gradient coefficients if the user chooses to view them in the units of volts.

4.9.25 Info amplitude type - value change event

Amplitude unit for the gradient channels can be changed to either V or G/cm in this event. All necessary information in the tables is modified, and the pulse sequence is redrawn.

4.9.26 Additional information - value change event

There is an additional information text field that is present in every pulse sequence file to store any additional data such as notes taken or observations made about the MRI experiment performed. This text field is 1024 characters (therefore bytes) long. In this event, length of the information contained in this field is checked and if it exceeds 1024 characters limit, the user is informed and the characters after the limit are deleted.

4.9.27 Events that require the pulse sequence to be redrawn

There are ten events that fall into this category. These events are resulted from a parameter change that causes the pulse sequence to be redrawn. All of these

events generate a plot – user event to redraw the pulse sequence according to the new value of the parameters. These events are listed in Table 4.3.

Table 4.3: Events that cause the pulse sequence to be redrawn.

Name of the Event	**Type of the Event**
Positions of slices	Mouse down
Positions of slices	Value change
Slice no	Value change
Echo type	Value change
DC offset elimination	Value change
RF signal shown	Value change
Displayed phase encoding step	Value change
Output frequency	Value change
TR	Value change
Image size – phase direction	Value change

4.9.28 Tab selector - value change event

This event is used to load the limit and coefficient information from the configuration settings file and display them on their corresponding user input fields. This action is only performed when the user chooses the settings tab of the tab selector that contains components for most of the functionality of the software.

4.9.29 Events written for implementing radio button behavior

There are four events that fall into this category. These are the value change events for the voltage sweep radiobutton, frequency sweep radiobutton, RF channels radiobutton and gradient channels radiobutton. The first two and the last two of the above events are separately grouped. If one of the radiobuttons in a group is turned on these events turn off the other radiobutton in the group.

4.9.30 Events written for the channel selection

These events are used for the channel selection operations by the user. Subsequent commands to add, modify, analyze waveform components will be performed on the selected channel. Mouse enter, mouse leave, mouse down events are written for all of the RF envelope, x, y, z gradients and applied current channel.

A channel is highlighted when the user moves the mouse pointer over it (mouse enter event), it is selected when the user clicked on the channel (mouse down event) and it is not highlighted if the mouse pointer leaves the channel without clicking it (mouse leave event). Selected channel will be highlighted all the time and it can be deselected by clicking on it again.

CHAPTER 5

IMAGE RECONSTRUCTION SOFTWARE

5.1 Introduction

Image reconstruction and image processing functionalities of the software is written using the IMAQ and IMAQ Vision components of the LabVIEW. IMAQ is a freely distributed image processing function library for the LabVIEW. On the other hand, the IMAQ Vision is the primary component of the National Instruments Vision Development Module which is an add-on toolbox for the LabVIEW that extends the functionalities of the IMAQ further.

Throughout this chapter the "Image Reconstruction" software will be referred as software shortly. In this chapter; firstly the purpose of the software is explained, followed by the derivation of the signal equation. After that the format of the data file that is used for storing k-space data is explained, and finally the events of the software are explained. The graphical user interface of the software is explained in Appendix B.

5.2 Purpose of the software

The primary function of this software is to reconstruct images from the data collected by the "Pulse Sequence Design and DAQ" software. This software also provides means for analyzing images, performing operations on images and easy browsing of the images. Strong interaction with Matlab software is also provided by the export functionalities that are built into the software.

5.3 Signal equation

In MRI Systems a RF coil is used to detect the magnetic field created by the magnetization of the spins in the imaging volume. This RF coil is designed to have a uniform sensitivity over the imaging volume. As similar to other MRI systems, in METU MRI System, a RF coil that is designed for measuring the transverse component of the magnetic flux is used. The reason for measuring the transverse component lies in the precession characteristics of the net magnetization vector which has a slowly varying z component (z is the direction of the main magnetic field) compared to the x and y components [13].

Therefore the received signal is composed from the transverse magnetizations of all the spins in the imaging volume. These magnetizations are time varying. They are also functions of space since the objects that are imaged are inhomogeneous. Therefore by denoting the transverse magnetization vector as M(x,y,z,t), the expression for the time domain signal s(t) can be written as,

$$s(t) = \iiint_{xyz} M(x, y, z, t)\,dx\,dy\,dz \tag{5.1}$$

The expression for M(x,y,z,t) can be derived from the Bloch equations which describe the motion of the magnetization vector under a magnetic field [13], and is given below.

$$M(\vec{r}, t) = M^{0}(\vec{r})\, e^{-t/T_2(\vec{r})}\, e^{-i\omega_0 t}\, e^{-i\gamma\int_0^t G(\tau)\vec{r}\,d\tau}$$

$$\text{where,} \quad \vec{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{5.2}$$

In this expression; $\vec{r}$ is the position vector, $M^0(\vec{r})$ is the equilibrium value of the transverse magnetization vector, ω_0 is the center frequency and G(t) is a time

varying gradient field. Therefore the expression in Equation 5.1 can be rewritten as,

$$s(t) = \iiint_{xyz} M^0(\vec{r}) e^{-t/T_2(\vec{r})} e^{-i\omega_0 t} e^{-i\gamma\int_0^t G(\tau)\vec{r}d\tau} dxdydz \tag{5.3}$$

In METU MRI System, spin echo imaging method is utilized. Therefore slice selection is performed, and only the spins residing in a finite thickness slice are excited. As a result, for an axial slice with thickness Δz, at position z = z_0, and also for constant gradients G_x and G_y, the Equation 5.3 simplifies to,

$$s(t) = \iint_{xy} m(x,y) e^{-t/T_2(x,y)} e^{-i\omega_0 t} e^{-i(\gamma\cdot G_x\cdot t\cdot x + \gamma\cdot G_y\cdot t\cdot y)} dxdy$$

$$\text{where, } m(x,y) = \int_{z_0-\Delta z/2}^{z_0+\Delta z/2} M^0(x,y,z)dz \tag{5.4}$$

After signal reception, quadrature amplitude demodulation is performed in the modem unit to remove the $e^{-i\omega_0 t}$ term. Also the $e^{-t/T_2(x,y)}$ term can be neglected by assuming T_2(x,y) is large enough. After these simplifications, the signal can be expressed as,

$$s(t) = \iint_{xy} m(x,y) e^{-i(u\cdot x + v\cdot y)} dxdy$$

$$\text{where, } \begin{aligned} u &= \gamma\cdot G_x\cdot t \\ v &= \gamma\cdot G_y\cdot t \end{aligned} \tag{5.5}$$

In the above expression m(x,y) is the quantity that is to be imaged. Also since u and v specify spatial frequency, the above expression is a two dimensional Fourier transformation between the space domain and the spatial frequency domain. Therefore for changing values of u and v (performed by frequency encoding and phase encoding), data from the complete spatial frequency domain

(also named as the k-space) can be collected. Finally, by performing a two dimensional inverse Fourier transformation, m(x,y) can be reconstructed from the s(t).

5.4 Format of the file used by the software

The software uses data files with the default extension dat. This extension is not forced by the software, it is only used for easy recognition of the files in the operating system. The beginning of a data file contains complete pulse sequence information just the same way as it is stored in a pulse sequence file. This information is followed by the echo samples that are collected for each pulse sequence cycle. Therefore, it can be said that a data file is a data appended version of a pulse sequence file. The file format is shown in Figure 5.1.

This file type is used primarily by the “Image Reconstruction” software but can also be used by the “Pulse Sequence Design and DAQ” software. These files are binary files with the default file extension of “dat”. Similar to pulse sequence files this extension can be changed or can be omitted completely.

Data files contain both the raw data (echo signals) acquired in a MRI experiment and the pulse sequence used in the experiment. By storing the pulse sequence information in the data file; the need to record which data is taken with which sequence is eliminated, files can be kept in a more organized way, and user errors which cause the acquired data to become useless such as losing the relationship between the data file and the pulse sequence are eliminated.

The “Image Reconstruction” software uses these files to reconstruct images from raw data and the “Pulse Sequence Design and DAQ” software uses these files to show the user the pulse sequence used when acquiring the raw data. The “Image Reconstruction” software also loads primary information about the pulse sequence from the data file such as TR, TE, number of averages, position of slices etc., and displays them upon the request of the user.

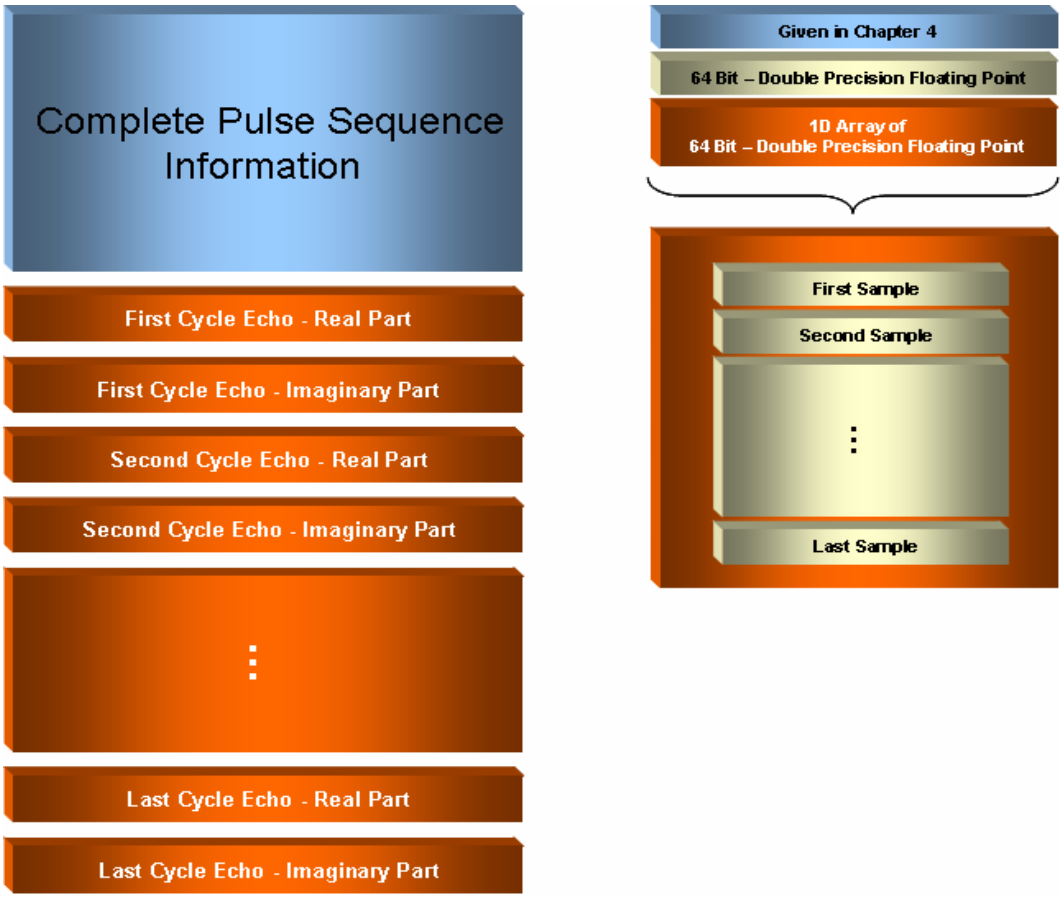


Figure 5.1: File format of a data file.

5.5 Events of the software

All functionality of the software is performed via events. Throughout this section, operations that are performed in these events are explained.

5.5.1 Load - user event

This event is for displaying specific echo signals to the user. The real and the imaginary parts of an echo signal that is acquired for a specific cycle of a pulse sequence is displayed. This event is called from other events when necessary.

In order to specify the cycle to be displayed, the user inputs which phase encoding step, which averaging, which DC offset elimination step and which slice that he/she wants it to be displayed by using the input fields. After that if the file is a data file, the offset from the beginning of the file to the point where the data that the user wants to see is calculated. Then the file pointer is rewound to the specified position by using this offset, and data for the specified echo signal is read. The data that is read is then displayed in the waveform graph components on the software GUI. Also the peak values of the echoes along with their indexes are calculated and displayed to the user.

5.5.2 Extract - user event

This event is also called from other events when necessary. In this event, firstly the pulse sequence information is read from a data file. According to this information (number of phase encoding steps, number of samples taken etc.), the k-space data that is stored in the data file is extracted and is moved into a two dimensional array structure. After these, a process - user event is generated with reconstruct command (is explained later in this section).

5.5.3 Process - user event

This event performs all image processing tasks that are supported by the software. It is possible to work with 17 images at the same time. These images can be displayed on the image area on the software GUI or on separate windows. All images are stored in an array for providing access to all of the events of the software.

Every image processing task is defined in terms of the source image, target image, operation name and on some occasions another image. When an image processing task is requested; the software reads the source image, performs the necessary operation as defined by the operation name and stores the result on the target image. Source image and target image parameters can be given as same.

Also some image processing tasks have only a target image and some only have a source image. There are also some image processing tasks that requires an additional image to be specified such as a mask image during mask image operation..

The IMAQ Vision supports the display of 16 image windows simultaneously and there is an image area on the software GUI, therefore it is possible to work with 17 images at the same time in the software. Any of these images can be specified as the source or target images that are mentioned in the previous discussion.

When a specific operation is requested, firstly the image that the requested operation will use as source is read. This image is the source image for most cases, but in some occasional cases target image can also be used. After that, the image is passed to a case structure that has a case defined for each operation type. The requested operation is performed in this case structure, and then the results are stored in the image array to the position of the target image specified. Naturally, if the operation does not requires a target image the last step is not performed. Also if the operation requires another image to be specified, this image is accessed in the case structure. Finally, if the result of the operation is an image, this image is displayed to the user appropriately by calling display - user event which will be explained later.

This scheme is very flexible and it enables the easy addition of the future expansions to the software. The program structure for passing all necessary data and checking the validity of user's inputs are present, therefore by utilizing this scheme a new functionality can be added by just adding a new case to the above mentioned case structure.

The image processing operations performed (cases of the case structure) are explained in the subsequent parts of this section.

5.5.3.1 Reconstruct operation

The signal equation and the principals of the reconstruction process from Fourier transform samples are given previously in Section 5.3. In this operation, different types of magnetic resonance images are reconstructed or extracted from k-space data acquired. These images are; magnitude, phase, real part, imaginary part of the complex image and magnitude, phase, real part, imaginary part of the k-space data.

One other point to mention about this event is the writing of the valid key into the image. Since the images will be stored in an array the software determines if an image in the array contains information or not by checking this key. At the initial run of the software and after an image is deleted the value of this key becomes false.

5.5.3.2 Mask operation

This operation is different from the mask image operation that will be explained later. This operation is for creating a binary image mask. Firstly, the user defines a reference pixel in the image by clicking on the image and the coordinates of the clicked pixel are detected. After that, specified pixel is used as an origin and an image mask is created by extracting a region surrounded by the reference pixel using a user specified tolerance value that defines the range of the intensity variations to be included in the mask based on the reference pixel. Using the origin's intensity, a search operation is performed on the neighboring pixels for an intensity that equals or falls within the tolerance value specified. The resulting image mask is a binary image. Also the contour of the image mask is drawn on both the source and target images. This operation is referred as the magic wand on the software GUI due to the nature of the process.

5.5.3.3 3D view operation

A three dimensional isometric view of the source image is calculated and stored in the target image. Each pixel of the source image is represented as a column of pixels in the 3D view such that it corresponds to the altitude. Also viewing orientations and angles can be specified.

5.5.3.4 Rotate operation

Source image is rotated by the specified amount degrees, and rotated image is stored in the target image. There is also an option for maintaining the size of the source image in the target image, and the replace parameter specifies the value to be filled in the image for the empty areas created by the rotation.

5.5.3.5 Threshold operation

A threshold is applied to the source image in this operation. A range defined by the upper and lower intensity values is inputted by the user, and the threshold is applied according to this range by either replacing the values in the range by a replace value specified or by keeping the values in the range untouched. Naturally, values that fall outside of the range are masked out. The result is stored in the target image.

5.5.3.6 Unwrap operation

Phase images reconstructed from the k-space data are uniquely defined only in the principal range [-π, π]. Therefore phase wraps in the phase image occur resulting from the values that fall outside of this range. The wrapped phase $\overline{\varphi}$ is related to the original phase φ as given in Equation 5.6 where l(x,y) is some integer [14].

$$\varphi(x,y) = \overline{\varphi}(x,y) \pm l(x,y)2\pi \tag{5.6}$$

In order to eliminate these phase wraps, the actual phase must be calculated either from the wrapped phase or from the original image. A model based phase unwrapping method [14] is used for unwrapping. This method is implemented in Matlab environment previously by Özbek (2002), and a detailed explanation of the method implemented can be found in his thesis [9].

The above mentioned Matlab algorithm is compiled into a binary executable file, and is called from software when requested. The data passing between the executable of the unwrapping algorithm and the software is performed via intermediate files that are deleted after the completion of the unwrapping operation.

A third-party VI that is distributed freely [15] is used for converting the data in LabVIEW to the Matlab mat file format. The phase unwrapping executable loads the data from this file, and unwraps the phase. After that, the unwrapped phase data is written to a binary file by the executable, and this file is read by the software. LabVIEW uses big-endian scheme for representing data whereas Matlab uses little-endian, therefore a bytewise endian conversion is performed to make the data usable to LabVIEW. Finally, the results are stored in the target image, and the temporary files are deleted.

5.5.3.7 Histogram operation

The histogram of the source image is calculated and displayed on a graph on the software GUI. The intensity range of the histogram and the number of classes that will be used are specified by the user. The mean value and the standard deviation of the pixels are also calculated and displayed.

5.5.3.8 Profile window operation

The profile of the pixels along the boundary of a ROI (region of interest) descriptor of an image window is displayed to the user. This ROI can be drawn on the image using the tools palette provided. Many types of ROIs (single point,

line, rectangular, circular etc.) can be drawn on the image to specify the region for which the consecutive operations will be performed on. Also the minimum-maximum pixel intensities, the mean value and the standard deviation of the pixel intensities and the number of pixels along the ROI boundary are calculated and displayed.

5.5.3.9 Linear averages operation

The average pixel intensity (mean line profile) of the image is calculated and displayed. Linear averaging operation is performed along rows, columns or diagonals of the image as specified by the user.

5.5.3.10 Shift operation

The source image is shifted according to the offset and replace value parameters, and the result is stored in the target image.

5.5.3.11 Mask image operation

In this operation a binary image mask that is specified by the user is applied to the source image, after that the masked image is stored in the target image.

5.5.3.12 Copy operation

The source image is copied to the target image by this operation.

5.5.3.13 Convert ROI to mask operation

In this operation a ROI that is drawn on the source image is converted to a binary mask, and stored in the target image. The value of the pixels that will be turned on (that are defined by the ROI) in the mask can also be specified by the user.

5.5.3.14 Copy ROI operation

This operation copies the ROI of the source image to the target image.

5.5.3.15 Convert mask to ROI

A mask in the source image is converted into a ROI in the target image. The external edges of the mask are used to create the ROI.

5.5.3.16 Profile along x index and profile along y index operations

These operations draw a line (ROI) on a specified row or column of the source image, and display the profile of that row or column on a graph on the software GUI.

5.5.3.17 Extract operation

This operation is similar to the mask image operation but this time the target image contains only the region specified by the mask, therefore the size of the target image is defined by the pixels that are turned on in the mask image.

5.5.3.18 Light meter window operation

This operation takes a rectangular ROI that is drawn on an image window. After that it calculates the histogram of the pixels in the region specified by the ROI and displays the histogram on a graph on the GUI of the software. This operation also calculates and displays the minimum, maximum, mean pixel intensities in the region along with the number and the standard deviation of the pixels residing in the region.

5.5.3.19 Change colormap operation

As a result of this operation, the colormap of the source image will be changed by the display - user event that will be triggered after process - user event. Nothing is

done in this operation. As mentioned previously, process - user event performs a validity check on the source image for checking whether it contains data or not, and then it passes the image to the case structure for performing the operation requested. Therefore this dummy operation is just added to make use of this initial check performed.

Another reason for adding this operation to the process - user event instead of directly performing the check and call the display - user event somewhere else is maintaining the software standardization. Every image processing operation of the software is initiated with an appropriate call to the process - user event, and after the processing is done, displayed by the display - user event which is called in the process - user event. In order not to create an exception to this routine this dummy operation is added.

5.5.3.20 Subtract operation

In this operation, an image that is specified is subtracted from the source image, and the result is stored in the target image.

5.5.3.21 Display in browser operation

In this operation, all 16 images that are displayed on windows are displayed to the user using a single image. This image is called the browser image, and is divided to four rows and four columns. In this operation, all window images are checked for validity, and then the valid ones are displayed at the appropriate places of the browser image. Browser image is a RGB image therefore all subimages are displayed correctly with their individual colormaps.

5.5.3.22 Save image operation

In this operation, the source image is saved into a file. Bitmap, jpeg, png, tiff and aipd (internal format of National Instruments) can be selected as the file type.

5.5.4 Display - user event

This event contains all image display functionality. This event is called by the process - user event to display the results. Apart from displaying images, this event also performs image naming and export variable naming tasks.

This event starts with the naming tasks. The image names are the names that are displayed on the window titles. The export variable names are the names to be used when exporting data to Matlab. These names are given automatically by the software, and can be changed by the user upon request.

The software has an autoname feature to give reconstructed or processed images names automatically. For example, “data2 (Phase Image) Masked” will be given automatically to a phase image which is reconstructed from a data2.dat file, and later masked. These predefined names are stored in an array, and is given to images according to the processing operations performed during the process - user event. The names of the export variables are also checked for length, Matlab naming conventions, invalid characters and duplicate variable names. Erroneous names are automatically corrected as to comply with Matlab.

If autoname functionality is not used then the software gives constant names to images and variables according to their position in the storage array.

After naming tasks are finished, the event displays the image using the selected colormap, in the specified window or in the image area on the GUI. Naming information, colormap used and validity information are written to keys in the image.

Finally, the image and export variable names are written to tables on the software GUI, and states of certain controls that are related with naming process are modified.

5.5.5 Tools - value change event

This event shows or hides the tools palette that is used for drawing ROIs on images that are displayed in windows (the image area has its own tools palette on the software GUI).

5.5.6 Software - mouse move event

IMAQ does not inform the software automatically for the closing of a window or the tools palette. Therefore a way of detecting close operations must be implemented since the statuses of some components are depended on these operations (i.e. leds on the software GUI that lights while an image is present).

In every execution of this mouse move event, windows and tools palette are checked for close operations, and then the modifications on the previously mentioned components are performed if necessary.

Another operation performed in this event is the detection of the last image that is operated on. The colormap of this image is automatically displayed in the active palette component that is present on the software GUI.

5.5.7 Close all windows - value change event

This event closes all opened image windows without deleting the actual image data. Also the states of the display leds that inform the user about the currently displayed images are set as all false.

5.5.8 Profile image area and light meter image area - value change events

Profile and light meter operations for an image residing in the image area are actually performed in the image area - mouse up event, and the results are shown

on graphs on the software GUI that are located at the same position. In these events a different graph is bring to front according to the operation specified.

5.5.9 Magic wand - value change event

This event is just written for disabling the profile and the light meter functionalities of the image area since they can not be performed together with magic wand masking operation.

5.5.10 Image area - mouse down event

In this event, the process - user event is called with operation specified as the mask operation if the magic wand setting is turned on in the GUI.

5.5.11 Image area - mouse up event

In this event, the profile image area and the light meter image area operations are performed if they are enabled on the GUI. Operations are the same as the ones that are performed during the window versions of these operations in the process -user event.

5.5.12 All or none (display) - value change event

This event is for closing or showing all the available windows instantly. Available windows are detected according to their valid key.

5.5.13 All or none (export) - value change event

This event is for exporting all available images to Matlab instantly to a specified mat file or for deleting all the previously stored information in a mat file instantly.

5.5.14 Defaults (window titles) and defaults (variable names) - value change event

These events return window titles and variable names back to their default values.

5.5.15 Export button - value change event

This event reads the states of the export leds on the software GUI that defines which images are to be exported, and according to that information exports the requested images to a Matlab mat file specified by the user.

5.5.16 Display - value change event

This event is triggered when the value of the display leds that define which images are to be displayed has changed. The requested windows are displayed to the user according to the statuses of the display leds.

5.5.17 Present - value change event

In this event, the states of the present leds that show which images are present in the memory are read. Then according to this, the images that are not chosen are cleared permanently from the memory.

5.5.18 Active palette - value change event

Current version of IMAQ do not support the display of a color ramp that shows the colormap that is used in an image window. Active palette is a component on the software GUI that shows the color ramps of all the image windows and the image area according to the selection of the user.

5.5.19 Export - value change event

In this event, the states of the export leds are read, and according to these the images are exported to a specified Matlab mat file.

5.5.20 Export file path - value change event

The path inputted by the user to define the location of the export file is checked, and necessary file extension is added if it is not done by the user.

5.5.21 Window titles - value change event

If the user modifies a window title on the table defining the window titles on the GUI, this event is generated. In this event window titles are updated according to the user input. Also if the user tries to modify a non valid window's title, that title is not accepted and deleted.

5.5.22 Variable names - value change event

This event is generated when the user modifies the export variable name of an image. The modified export variable names are checked to ensure that they comply with Matlab naming conventions. Also if the user tries to modify the export variable name of a non valid image, that name is not accepted and deleted.

5.5.23 Batch export - value change event

Batch exporting is a useful functionality, especially when exporting a large set of data files that are named following a naming convention. In this event a data file start string, a variable start string and three string arrays are inputted by the user. The data file paths and their corresponding export variable names are created by starting with the start strings, and appending them the strings that are contained in the string arrays for all possible combinations. After that all data files are loaded

according to the paths created, and the k-space data stored in them is exported to a specified Matlab mat file under the export variable names that are created.

5.5.24 Preview button - value change event

This event is used to preview the data file paths and export variable names that are created for the batch export process before performing it.

5.5.25 Color table graph - mouse up event and beginning color, final color - value change events

These are duplicate events. They are written to perform the user defined colormap generation operation. The beginning and final colors of the colormap is inputted by the user by specifying the RGB values of the colors. Also an intermediate color is inputted. The colormap is calculated based on these color values and displayed to the user.

5.5.26 Menu selection (user) event

This event handles the menu of the software. Handling is done as explained in Chapter 4. Nearly all of the software functionality can be controlled from the front panel of the software therefore the menu is simple.

The software has a file menu with open data file and exit options. It has a mode menu that defines the operation mode. In file mode, the software displays the functionality related with image reconstruction and processing in its front panel whereas in run-time mode, the front panel of the display current data subVI is displayed on the software's front panel using a subpanel. As explained in Chapter 4, the run-time mode is for displaying the data acquired in the progressive mode of the "Pulse Sequence Design & DAQ" software. The software also has an about menu for displaying version information.

5.5.27 Start data file browsing and data file browse directory - value change events

The software has a data file browsing functionality. The contents of a directory with many data files can be viewed easily by browsing. In this event, firstly the specified directory is analyzed for data files. Filters for specifying the names of files to be analyzed can also be used. After that, the number of data files that meets the filtering criteria is displayed to the user. Also the first one of these files is loaded and displayed to the user along with the information about the pulse sequence used to acquire that data. Naturally, loading and displaying tasks are performed by calling user events that are previously explained in this chapter.

5.5.28 Next and previous - value change events

These two events are used for the data file browsing operation. In these events, the next data file or the previous data file in the specified data file directory is loaded and displayed to the user.

5.5.29 Events that call (generate) user events to perform the requested operations

As explained previously, the software has four user events which are called from other events in order to centralize and standardize the software code. As a result there are many events that only call user event such as the value change events for the buttons of the processing operations which call process - user event. Also an event may call more than one user event such as display slice - value change event which calls both the load and extract user events.

CHAPTER 6

MEASUREMENT OF SECONDARY MAGNETIC FIELD INDUCED UTILIZING GRADIENT COILS

6.1 Introduction

In this chapter, a new method is proposed for inducing current in a conductor object to be imaged. The chapter starts with explaining the magnetic resonance - current density imaging technique in which the proposed method might be utilized. Induced current - current density imaging research is mentioned next which is followed by the objective of this study. The experimental setup and the pulse sequence used during experimentation are explained next, followed by the methodology. The discussion of the results of the experiments is presented next, and the chapter ends with the conclusion.

6.2 Magnetic resonance - current density imaging

In magnetic resonance – current density imaging (MR-CDI), distribution of the current density inside a conductor object that contains nuclear magnetic resonance active nuclei is imaged [16,17,18]. In this modality the current is injected to the object to be imaged.

Several groups conduct research on MR-CDI. Different levels and frequencies for uniform and nonuniform current are used under various magnetic field strengths by these groups. Also several experiments are performed using phantoms or in-

vivo. In some works just one component of the current density is imaged whereas in some all three components are imaged. A good summary about the research in this field can be found in [9].

MR-CDI experiments can be classified into three main groups as DC, RF and AC according to the frequency range of the current applied.

In DC current density imaging, bipolar current pulses are applied to the object to be imaged synchronously with a MRI pulse sequence. Applied current will act like a local gradient field, and dephases the spins residing in the volume that the current passes through, therefore generating a current encoded phase in the complex image. Current induced magnetic flux density can be extracted from this image, and the current density can be calculated utilizing Biot-Savart law [17].

In RF current density imaging, a RF current is applied to the object. This current has a frequency that is equal to the resonant frequency of the spins. A RF magnetic field is created by this RF current, and since the resonance condition is satisfied, the spins flip towards the transverse plane proportionally with the strength and the duration of the RF current. Magnetic flux density resulting from this current is calculated from the flip angle, and finally current density is found using Biot-Savart law [19].

In AC current density imaging, an AC current is applied to the medium to be imaged. Therefore a current induced phase resulting from the applied current is added to the complex image. Two data sets are collected; one with current application and another one with no current application. Difference of the phase images of these two data is computed to find the current induced phase. Finally, current density can be calculated similarly to the DC-CDI technique [20].

6.3 Induced current - current density imaging

Nearly in all of the previous studies, the application of the current is performed by injecting it via electrodes that are attached to the boundary of the conductor object. Due to the presence of current injection electrodes, these applications suffer from susceptibility artifacts and denser currents near the electrodes [21].

However, the current can also be applied by induction. Only few studies exists that utilize this method. One of these studies is performed for EIT (electrical impedance tomography) by Gençer *et al* in 1994 [22]. Another one is performed for MR-EIT (magnetic resonance - electrical impedance tomography) by Özparlak and İder in 2005 [21]. Their work is a simulation work. In their method, the current density inside the object is computed first, and the conductivity distribution is reconstructed based on the current density.

In these studies external coils surrounding the imaging volume are used to generate an AC magnetic field inside the object. Eddy currents inside the object are generated as a result of this primary magnetic field. These eddy currents then generates a secondary magnetic field inside the object. This secondary magnetic flux density is the quantity that is used to reconstruct images of conductivity distribution.

6.4 Objective of the study

In this study, a new method for inducing current in conductor objects that reside in a magnetic resonance scanner is proposed. In the proposed method, one of the gradient coils of the magnetic resonance imaging system is used to induce an AC current inside the object instead of an external coil. In this study, the gradient coil chosen for this purpose is the x-gradient coil. Since this gradient coil is already present in the system, there is no need for any additional hardware.

In this study, it is shown that an AC current can be induced in the object to be imaged by using the readily installed gradient hardware of the scanner. For this purpose a special form of a spin echo pulse sequence is designed, and k-space data is collected with and without inducing current using this sequence. Current induced magnetic fields are extracted from the phase difference images of current inducing and non current inducing datasets for different amplitudes of AC current passing through the gradient coil. Also k-space data is analyzed for different amplitude levels.

Pulse sequence design, data acquisition and image reconstruction tasks performed throughout this study is made utilizing the functionalities of the new software suit and the new DAQ hardware of the METU MRI System.

6.5 Experimental setup

A rectangular prism plexi-glass phantom is fabricated and is used for the experimentation. It has the dimensions of 9.4 cm x 4.8 cm x 4.3 cm. The phantom is placed at the center of the scanner, and is filled with 1 g/l $CuSO_4.5H_20$ solution in water. $CuSO_4$ is used for reducing the T_1 relaxation time.

The positioning of the object in the scanner is illustrated in Figure 6.1. Since x gradient coil is used for inducing current, the field created by this coil is always in the same or opposite direction of the main magnetic field.

However the magnitudes of the field vary as a function of position along x direction. Therefore when moving along x direction from the sides, the induced current will decrease towards the center of the slice since the strength of the gradient field decreases.

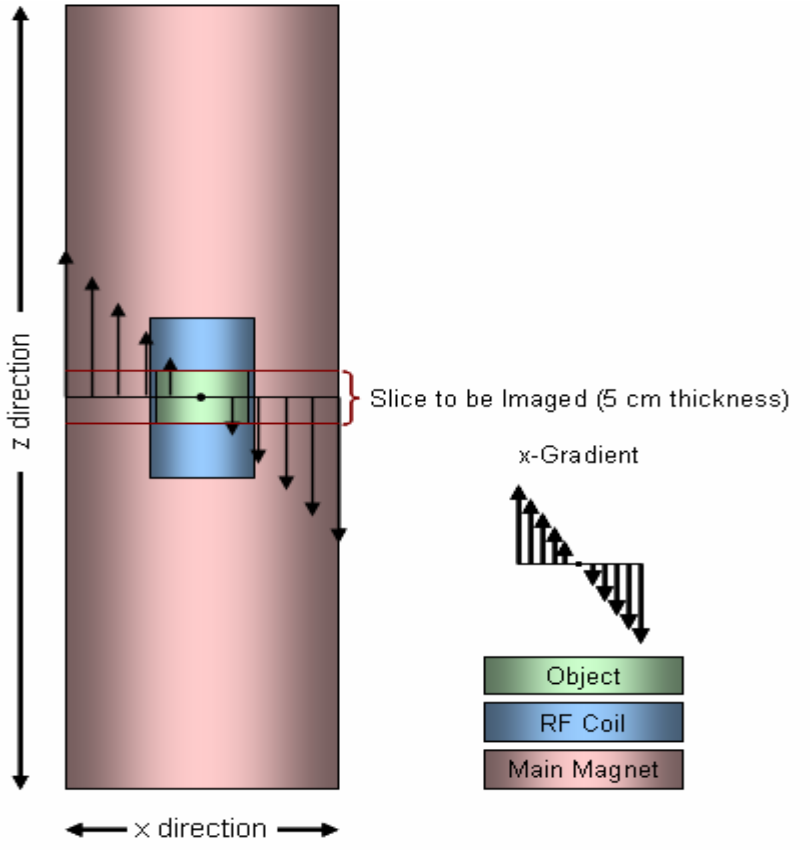


Figure 6.1: Object positioning in the scanner.

6.6 The pulse sequence used

In this study a similar pulse sequence to the ones that are used by Mikac *et al* [20] in 2001 and Özparlak and İder [21] in 2005 is used. The pulse sequence used is given in Figure 6.2.

This pulse sequence is a modified version of a standard spin echo pulse sequence. In the pulse sequence; z gradient is used for slice selection, y gradient is used for phase encoding, and x gradient is used for frequency encoding purposes.

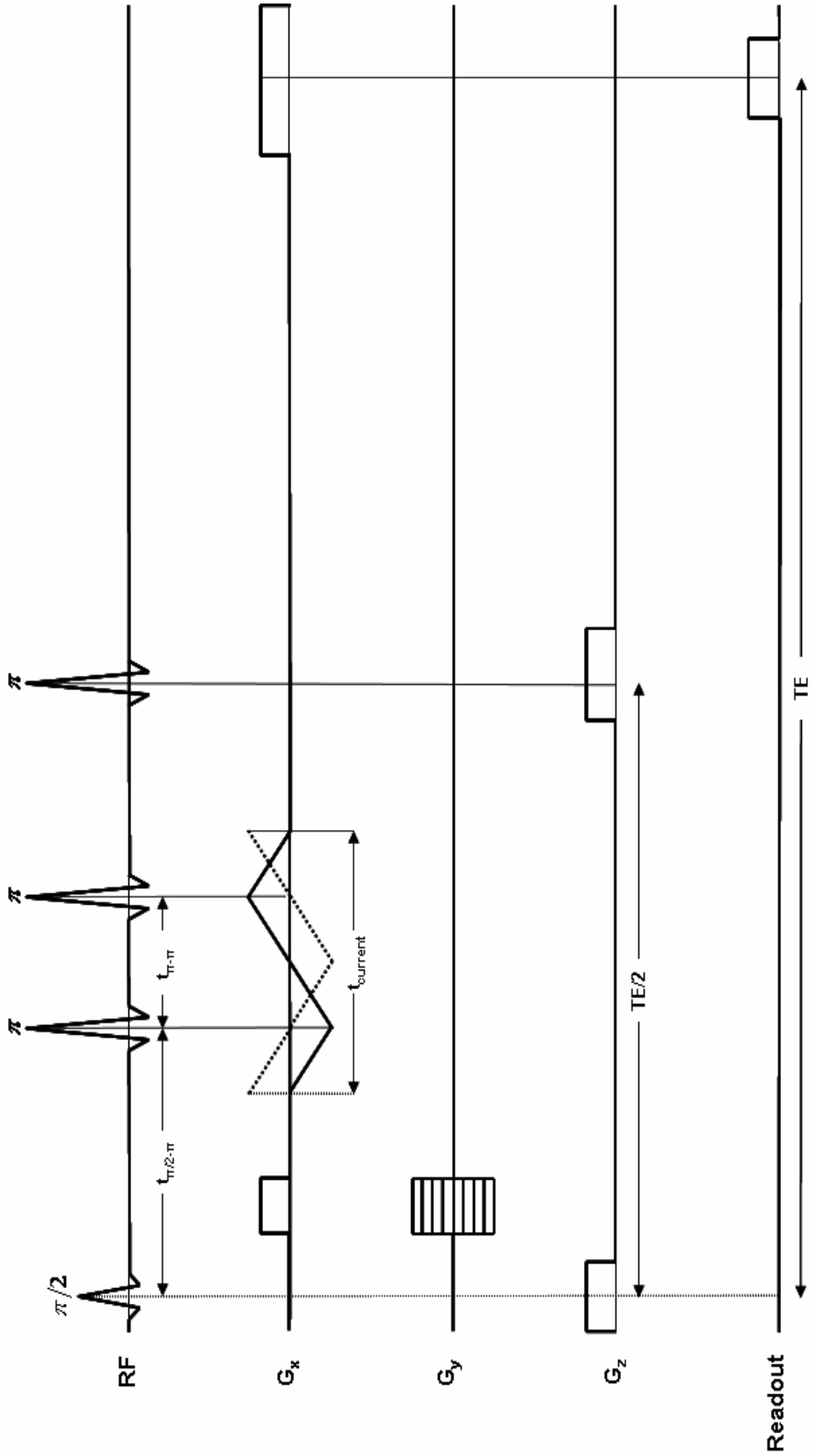


Figure 6.2: Pulse sequence used for inducing current in the object.

In contrast with a spin echo sequence, the x gradient is also used for inducing current in the medium. This is achieved by applying a triangular shaped AC current to the x gradient coil. One full cycle of this AC current is applied to the x gradient coil as shown in Figure 6.2 by solid lines.

As a result of the AC current flowing in the x gradient coil, a primary AC magnetic field $\mathbf{B}_p$ which is in-phase with this current is created in the imaging volume. This magnetic field induces eddy currents inside the conductor object to be imaged. These create a secondary ac magnetic field $\mathbf{B}_s$, shown in Figure 6.2 by broken lines. This newly created field is out-of-phase by 90° from the $\mathbf{B}_p$.

One important point to mention is the application of two 180° RF pulses at the peaks of the $\mathbf{B}_p$ which reverses the spin dephasing process that occurs during the first half of the half cycle of AC current. Therefore the spins are rephased at the second half of the half cycle. As a result total phase accumulation due to $\mathbf{B}_p$ is zero.

As mentioned previously, $\mathbf{B}_s$ is out-of-phase by 90° from the $\mathbf{B}_p$. Therefore, 180° RF pulses are positioned at the zero crossings of $\mathbf{B}_s$. As a result, this time the phases accumulating due to $\mathbf{B}_s$ will add up in the two half cycles.

Therefore phase is only accumulated due to $\mathbf{B}_s$. This accumulated phase due to induced current is given in Equation 6.1 where $\langle B_{sz}(x,y)\rangle$ is the time average value of the Bsz which is the z component of $\mathbf{B}_s$, $t_{current}$ is the current application duration, γ is the gyromagnetic ratio and B_{sz}^{peak} is the peak value during positive half cycle.

$$\begin{aligned} \varphi_s(x,y) &= \gamma\langle B_{sz}(x,y)\rangle t_{current} \\ \langle B_{sz}(x,y)\rangle &= B_{sz}^{peak}/2 \end{aligned} \tag{6.1}$$

The pulse sequence has a TR of 350 ms. and a TE of 35.5 ms. A center slice with 5 cm thickness that covers all the length of the object is excited. The $t_{\pi/2\text{-}\pi}$, $t_{\pi\text{-}\pi}$ and $t_{current}$ times that are shown in Figure 6.2 are 8.5 ms., 4 ms. and 8 ms. respectively.

6.7 Methodology

A diagram that illustrates the steps followed in order to obtain the time average of the magnetic flux density - z component due to induced current ($\langle B_{sz}(x,y)\rangle$) is given in Figure 6.3. In this figure subscript c denotes the dataset that is collected with current induced.

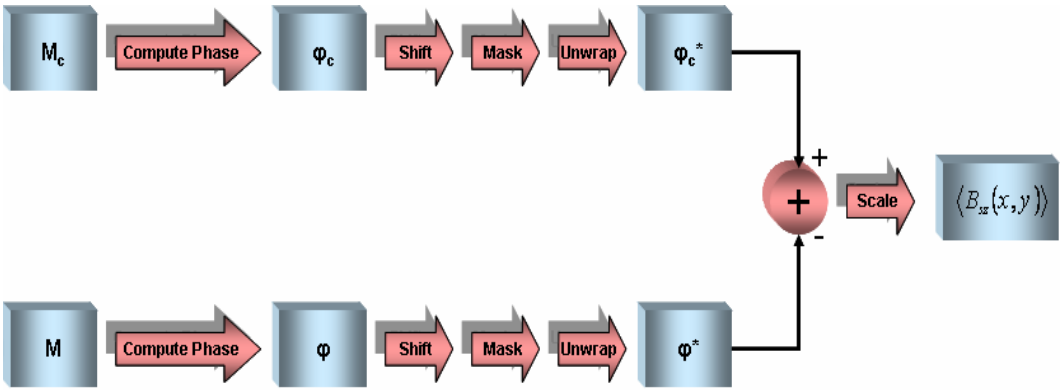


Figure 6.3: Steps followed for computing $\langle B_{sz}(x,y)\rangle$.

Firstly, complex images of magnetization density (M_c and M) are reconstructed with and without applying current. Then the phases of these images (φ_c and φ) are computed. After that these phase images are examined and shifted in frequency encoding direction, in order to compensate the effects of the thermal changes of the main magnet. After shifting the data to the same position in all images, a mask is applied to remove the background noise of the phase images. This step is

performed since the unwrap algorithm that will be used later fails in the above mentioned random phase regions which also contain no information.

The phase of a complex image is in the range [-π, π]. Any phase component that is not in this principal range is wrapped up into this range. Therefore, the previously mentioned phase unwrapping algorithm is used to remove these phase wraps. After this step the current induced and no current induced phase images (φ_c^* and φ^*) are ready to be subtracted from each other, and this form of the phase images are denoted with * superscript in Figure 6.3.

The unwrapped phase images of the datasets with and without current is subtracted, and therefore the phase accumulated that is resulting from only the induced current is found (main magnetic field inhomogeneities are removed by this operation). Following this step, the normalized phase image is scaled with gyromagnetic constant and the total current induction time, to obtain the time average of the magnetic flux density - z component due to induced current.

6.8 Results

The magnitude of the k-space data collected for various peak amplitudes of AC currents that are applied to the x gradient is given in Figure 6.4.

As can be seen from these data, the k-space data is dephased with increasing current up to the 51.3 A. This situation is caused by the induced current which dephases the spins residing in the object by accumulating a phase as explained previously. Since current is induced into the object, the dephasing occurs everywhere in k-space. As a result of this dephasing, significant amount of signal lost is observed since some of the signal components gain an excessive amount of phase and as a result of it shifted out from the k-space.

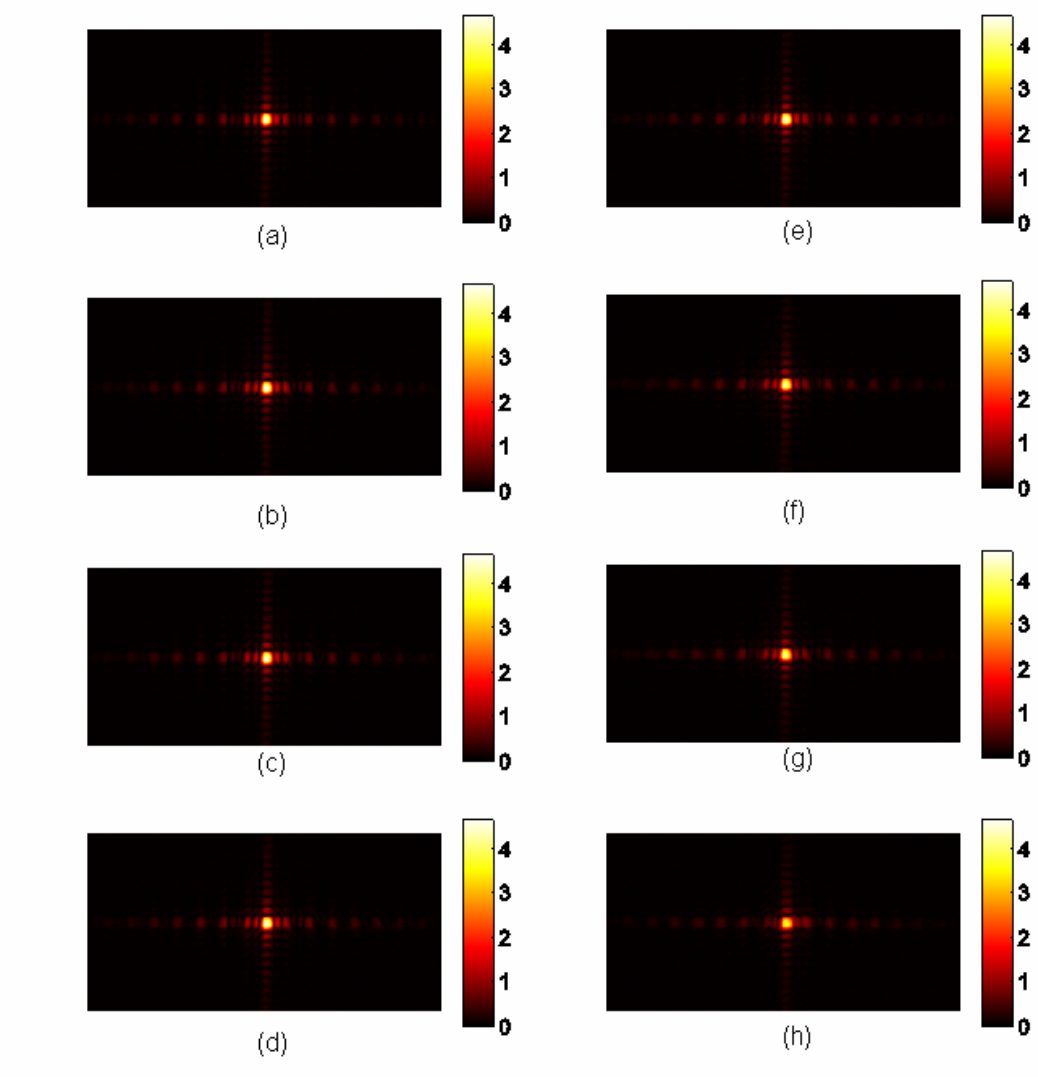


Figure 6.4: Magnitude of the k-space data collected for various peak amplitude AC current levels that are applied to the x gradient coil.

(a) 0 A, (b) 1.71 A, (c) 3.42 A, (d) 5.13 A, (e) 6.84 A, (f) 8.55 A, (g) 10.26 A, (h) 11.97 A, (i) 13.68 A, (j) 15.39 A, (k) 17.1 A, (l) 20.52 A, (m) 23.94 A, (n) 27.36 A, (o) 30.78 A, (p) 34.2 A, (q) 37.62 A, (r) 41.04 A, (s) 44.46 A, (t) 47.88 A, (u) 51.3 A, (v) 54.72 A, (w) 61.56 A, (x) 68.4 A, (y) 76.95 A, (z) 85.5 A, (aa) 94.05 A, (ab) 102.6 A, (ac) 119.7 A, (ad) 136.8 A, (ae) 153.9 A, (af) 168 A.

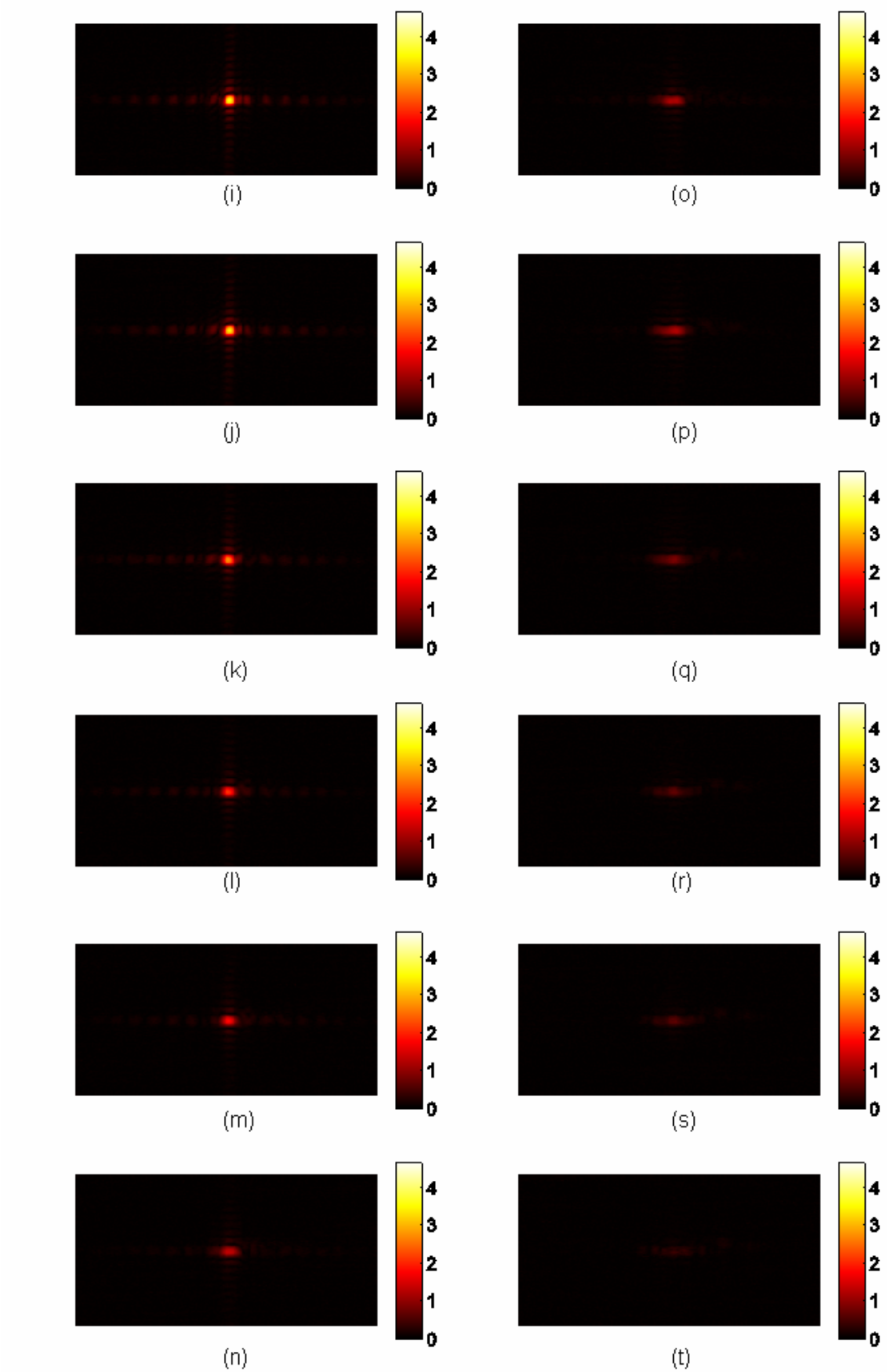


Figure 6.4 (continued)

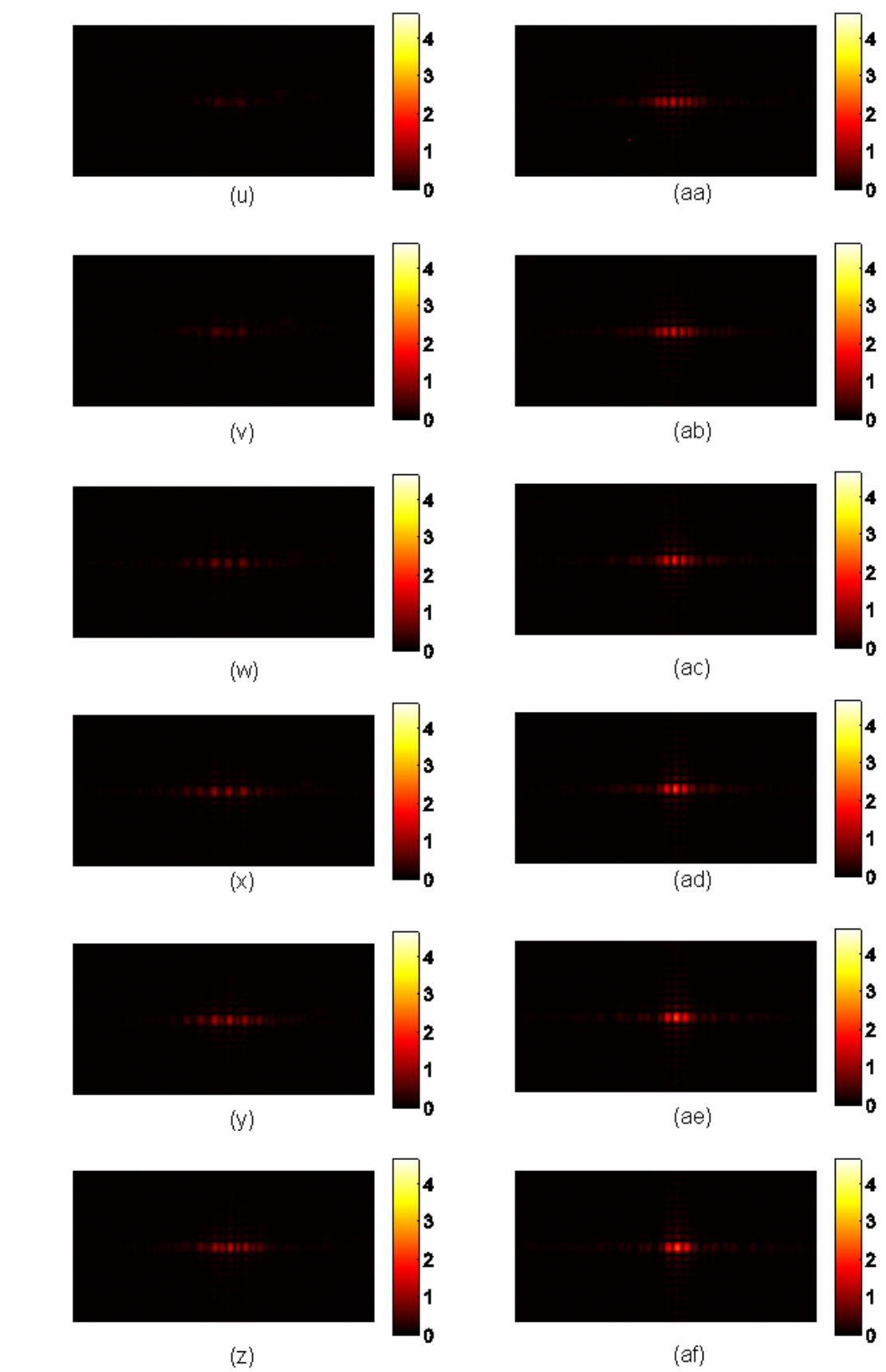


Figure 6.4 (continued)

If the current is continued to increase after 51.3 A, it is observed that the lost signal begins to reappear. This situation means that at those points in the k-space where the signal begins to reappear, the accumulated phase resulting from the induced current has passed beyond 360°. Therefore k-space data is going to rephase with increasing current. This situation is observed until the maximum output current limitation (168 A) of the gradient amplifier unit is reached. At this point it is observed that a significant amount of k-space data is rephased.

This situation can be understood more clearly by examining Figure 6.5. In this figure the magnitude of the k-space data collected for all amplitudes are integrated and are plotted.

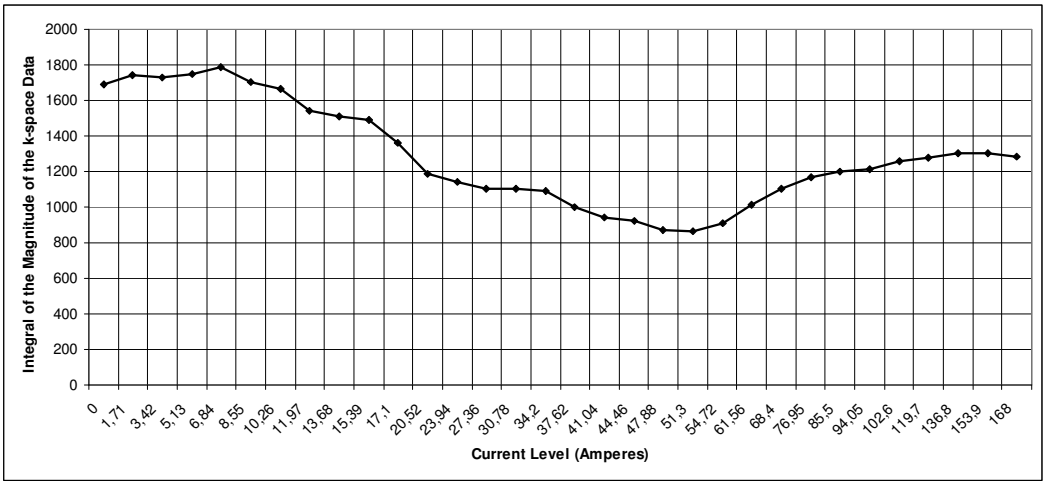


Figure 6.5: Integral of the magnitude under magnitude of the k-space data.

In Figure 6.6 the time average of the magnetic flux density - z component due to induced current ($\langle B_{sz}(x,y)\rangle$) is given. This magnetic flux density is computed according to the Sections 6.6 and 6.7.

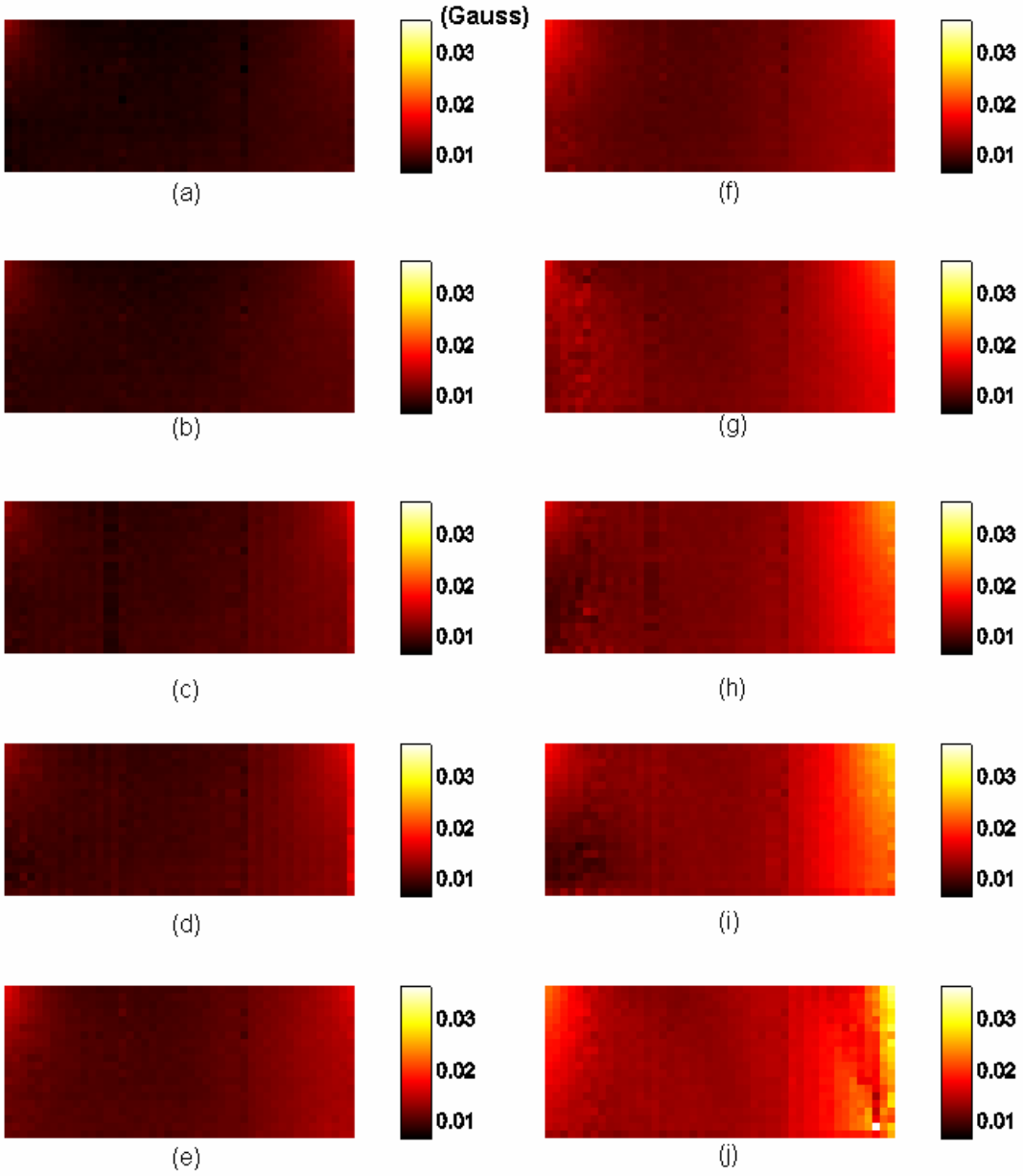


Figure 6.6: Time average of the magnetic flux density - z component due to induced current for various peak amplitude AC current levels that are applied to the x gradient coil.

(a) 1.71 A, (b) 3.42 A, (c) 5.13 A, (d) 6.84 A, (e) 8.55 A, (f) 10.26 A, (g) 11.97 A, (h) 13.68 A, (i) 15.39 A, (j) 17.1 A.

Signal loss has occurred in the cases with current application greater than 17.1 A to the x gradient coil. Therefore magnetic flux densities are computed up to the 17.1 A case.

As can be seen from the results, fields with around 30 mG strengths can be generated on the sides of the object where the x gradient strength is higher (therefore more current is induced).

The decreasing behavior of the magnetic flux density when moving towards the center along x direction is also expected since the x gradient strength is reduced more and more as coming close to the center.

The x gradient coil in METU MRI System has 14 turns. Therefore the effective current that passes through the coil may be considered 14 times higher than the amplitude values specified in Figures 6.4, 6.5, and 6.6. Therefore in order to measure a noticeable amount of magnetic field due to induced current (its strength is around 20 mG at the sides) in the phantom, the effective current that are flowing through the gradient coil is found to be around 70 A (5 A x 14).

This measured value is significantly lower than the 1000 A simulation value that is specified in Özparlak and İder [21] to obtain a maximum current density of 7500 mA m^{-2} in a phantom for the single turn coil case.

One thing that needs further examination is the polarity of the magnetic field close to the sides of the phantom. Since the direction of the x gradient field differs from the left side to the right side, it is expected that the measured field to have different directions in each side. However this is not the case observed, as the direction of the field is found the same in both sides. This observation is consistent in repeated experiments and needs to be investigated.

6.9 Conclusion

In this study, it is shown that the readily built gradient hardware of an MRI scanner can be used to induce currents inside an object. Current density imaging algorithms that are based on this scheme can be created.

The main advantage of this approach lies in the fact that it does not requires any additional hardware, and it can be integrated to conventional MRI scanners by just adding new pulse sequences to their databases.

CHAPTER 7

CONCLUSION

In this study, a software suit for designing pulse sequences, performing data acquisition tasks, reconstructing and processing images is developed. The software suit is designed and implemented in a way that is easy to understand for the future programmers of the METU MRI System. Therefore the code enables easy and less time consuming modification. The code can also be scaled easily therefore adding new channels etc. can be performed without excessive programming.

The designed and implemented “Pulse Sequence Design and DAQ” software provides an interactive environment to the user to design pulse sequences efficiently. All of the parameters related with pulse sequence design can be set using this software. Designed pulse sequence waveforms can be examined closely from graphs that have zooming features and from information tables. There are pulse sequence design utilities that help the user in the design process such as the automatic gradient strength calculation to obtain the necessary field of view, area under gradients calculators etcetera.

This software also performs error checking on the specified pulse sequences according to the critical system limits that can be changed by the user in order not to damage the system hardware.

The software has some new capabilities such as remote frequency control, automatic center frequency adjustment, automatic tip angle adjustment, on the fly

pulse sequence modification, channel disabling features etcetera. It has pulse sequence browsing capabilities that makes it easier to work with directories that contain large amounts of data files. Also this program can control the current source that is used in injected current density imaging experiments.

The “Image Reconstruction” software can reconstruct several magnetic resonance related images. Following this, image processing operations can be performed easily on the images reconstructed such as phase unwrapping, automatic mask generation, masking, rotating, shifting, thresholding etcetera. Profiles and histograms of the images can also be displayed according to the ROIs that are drawn on the images.

Also this software has browsing capabilities for enabling the easy browsing of data files in a directory. The software has a very strong interaction mechanism with Matlab. All images that are reconstructed or processed by this software can be very easily exported to the Matlab environment. The software provides auto naming functionality for further simplifying the export process, and creating a convenient way of recognizing images in the software environment. Another export functionality is the batch export. By using this functionality, a dataset containing multiple data which is named following a naming convention can be exported to Matlab in seconds. The software has also some functionality to display the echo signals collected for each pulse sequence cycle.

Also in this study, a new method for inducing current in a conductor object that can be utilized for current density imaging is proposed and investigated experimentally. In this method readily built x gradient coil of the MRI System is used to induce the current. Therefore no additional hardware is required. The induced current is an AC current. This is achieved by applying an AC current to the x gradient coil which induces a secondary AC current in the object.

The dephasing due to this induced current is measured from two datasets that are acquired with and without current. From the phase accumulation that is caused by

the induced current, the time average of the magnetic flux density - z component due to induced current is computed for varying peak current amplitude levels that are applied to the x gradient coil. These results and the volumes under the k-space magnitude are examined, and it is shown that the current induction by this method is possible.

Starting from around 5 A of peak amplitude AC current that flows through the x gradient coil, magnetic field strengths around 20 mG can be measured due to induce current. This 5 A value corresponds to a 70 A effective value since the x gradient coil used in the experimentation has 14 turns. Also signal loss due to dephasing has occured starting from 17 A (238 A effective) case.

7.1 Future work

There are other components in the METU MRI system that are not currently fully automated now. Systems such as the cooling system and the main magnet power supply can be connected to the DAQ cards and can be controlled and monitored from the software. Also continuous automated data logging of temperature sensors from various parts of the system can be implemented for diagnostic purposes.

If further and more advanced digital I/O is requested or needed, a new digital I/O or multifunction card that supports triggering and synchronization capabilities can be integrated into the system.

Induced current MRCDI study using gradient coils can be extended to MREIT. Utilizing all three gradient coils should be also considered.

REFERENCES

[1] F. Bloch, “Nuclear induction”, *Physical Review*, vol. 70, pp. 460-474, 1946.

[2] F. Bloch, W. W. Hansen, and M. Packard, “The nuclear induction experiment”, *Physical Review*, vol. 70, pp. 474-485, 1946.

[3] 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, 1946.

[4] P. C. Lauterbur, “Image formation by induced local interactions - Examples employing nuclear magnetic-resonance”, *Nature*, vol. 242, pp. 190-191, 1973.

[5] P. Mansfield and P. K. Grannell, “NMR 'diffraction' in solids”, *Journal of Physics C: Solid State Physics*, vol. 6, pp. 422-426, 1973.

[6] R. Damadian, M. Goldsmith, and L. Minkoff, “NMR in cancer: XVI. FONAR image of the live human body”, *Physiological Chemistry and Physics*, vol. 9, pp. 97-100, 1977.

[7] L. T. Müftüler, “Measurement of magnetic field generated by non-uniform AC current density using magnetic resonance”, PhD thesis, Middle East Technical University, Dept. of EE, 1996.

[8] E. Turan, “Implementation of 2DFT spin echo NMR pulse sequence on a PC based system”, Master’s thesis, Middle East Technical University, Dept. of EE, 1992.

[9] O. Özbek, “Imaging current density distribution using 0.15 T magnetic resonanc tomography”, Master’s thesis, Middle East Technical University, Dept. of EE, 2002.

[10] National Instruments , PCI-6713 Card, User Manual.

[11] Texas Instruments, CD4094BE Register, Datasheet.

[12] Ö. Birgül, "Developement of reconstruction algorithms for magnetic resonance – electrical impedance tomography and experimental realization", PhD thesis, Middle East Technical University, Dept. of EE, 2002.

[13] Z. P. Liang and P. C. Lauterbur, "Principles of magnetic resonance imaging: A signal processing perspective", *SPIE Optical Engineering Press*, 2000.

[14] Z. P. Liang, "A model-based method for phase unwrapping", *IEEE Transactions on Medical Imaging*, vol. 15, pp. 893-897, 1996.

[15] S. Liberman, "Can I Import Data from Matlab to LabVIEW or Vice Versa?", *http://digital.ni.com/ public.nsf/websearch/2F8ED0F588E06BE1 862565A90066E9BA?OpenDocument* , 10-08-2005.

[16] M. L. G. Joy, G. C. Scott, and R. M. Hankelman, "In vivo detection of applied electric currents by magnetic resonance imaging", *Magnetic Resonance Imaging*, vol. 7, pp. 89-94, 1989.

[17] G. C. Scott, M. L. G. Joy, R. L. Armstrong, and R. M. Hankelman, "Measurement of non-uniform current density by magnetic resonance", *IEEE Transactions on Medical Imaging*, vol. 10, no. 3, pp. 362-374, 1991.

[18] G. C. Scott, M. L. G. Joy, R. L. Armstrong, and R. M. Henkelman, "Sensitivity of magnetic resonance current density imaging", *Journal of Magnetic Resonance*, vol. 97, pp. 235-254, 1992.

[19] G. C. Scott, M. L. G. Joy, R. L. Armstrong, and R. M. Hankelman, "RF current density imaging in homogeneous media", *Magnetic Resonance in Medicine*, vol. 28, pp. 186-201, 1992.

[20] U. Mikac, F. Demsar, K. Beravs, and I. Sersa, "Magnetic resonance imaging of alternating electric currents", *Magnetic Resonance Imaging*, vol. 19, pp. 845-856, 2001.

[21] L. Özparlak and Y. Z. İder, “Induced current magnetic resonance - electrical impedance tomography”, *Physiol. Meas.*, vol. 26, pp. 289-305, 2005.

[22] N. G. Gençer, M. Kuzuoğlu, and Y. Z. İder, “Electrical impedance tomography using induced currents”, *IEEE Transactions on Medical Imaging*, vol. 13, pp. 338-350, 1994.

[23] National Instruments , PCI-6110E Card, User Manual.

APPENDIX A

GRAPHICAL USER INTERFACE OF THE PULSE SEQUENCE DESIGN & DAQ SOFTWARE

A.1 Notation used when explaining the GUI of the software

The graphical user interface (GUI) of the software is explained by using the screenshots of it in which the sections to be explained are enclosed in rectangles that are drawn with broken lines. These rectangles are color coded, and also have a reference number. From now on, a section that is going to be explained which is enclosed in a rectangle with reference number x in Figure A.y will be referred as Ay.x shortly.

Also throughout this appendix the “Pulse Sequence Design & DAQ” software will be referred as software, shortly.

In this appendix, the software is explained in a user interface perspective, the detailed working principles of the functionality provided were given in Chapter 4.

A.2 Front panel of the software

A screenshot of the front panel of the software is given in Figure A.1.

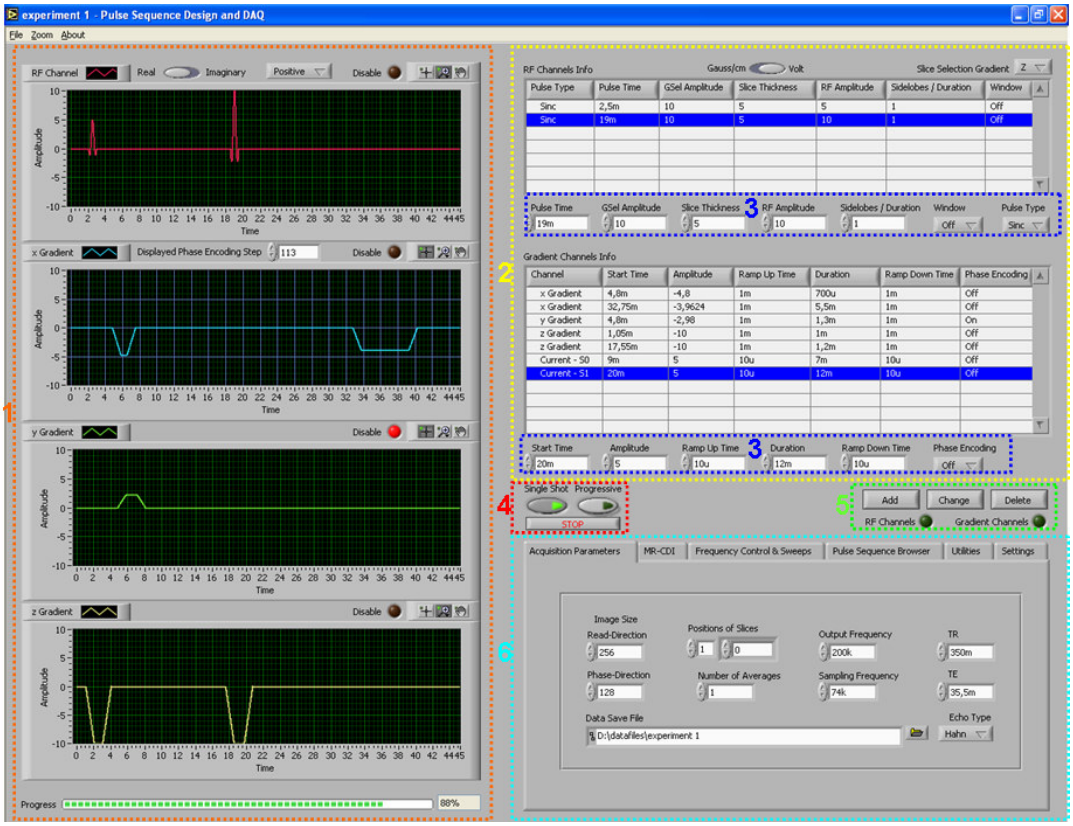


Figure A.1: Front panel of the Pulse Sequence Design & DAQ software.

In this figure, A1.1 contains four waveforms graphs. These are used to display the RF envelope, x gradient, y gradient and z gradient waveforms from top to bottom. The RF envelope waveform have also a real/imaginary channel selector and a DC offset elimination step selector for choosing the channel to be displayed and the DC offset elimination step to be displayed, respectively. Also there is a control for changing the displayed phase encoding step.

All of these waveforms has their own disable buttons for preventing the generation of the waveform samples from DAQ hardware. This functionality is useful for not manually turning off the gradient amplifiers or RF power amplifier during signal adjustment procedures.

These graphs also have zooming palettes located on the top right corner of them for inspecting particular sections of pulse sequences more closely. At the bottom of the four graphs, a progress bar is located for displaying the current generation progress of a pulse sequence.

A1.2 contains the pulse sequence information tables. The top table displays information about the RF envelope channel whereas the bottom one is for gradient channels information. There is a selector to choose between the gauss/cm unit and the volt unit. Also the slice selection gradient can be specified here.

A1.3 has the controls for inputting the data such as start time, duration, amplitude etc. from which the RF envelope blocks and the gradient blocks are created.

A1.4 contains the buttons that initiate the waveform generation process. This section also contains the stop button that is used to terminate a pulse sequence generation process upon request.

A1.5 contains buttons for adding new blocks to a pulse sequence, changing the properties of an existing block of a pulse sequence and deleting blocks of a pulse sequence. This section also contains a selector which is used by some functionality of the software that is used for choosing between RF envelope and gradient channels.

A1.6 is the tab selector that contains many functionality of the software in its tabs. In the screenshot given in Figure A.1, the “Acquisition Parameters” tab is shown.

The “Acquisition Parameters” tab contains the most critical pulse sequence parameters such as TR, TE, no of phase encoding steps, positions of slices etcetera.

A.3 Progressive mode

A screenshot that is taken during the progressive mode is given in Figure A.2.

In Figure A.2, A2.1 contains the controls that are used for progressively changing the pulse sequence parameters on the fly. As one may notice, the controls that are displayed previously in that area (A1.3) are now hidden. From the arrow buttons in A2.1 the user can choose the entry in the information table that he/she wants it to be changed. There is also a selector for moving between the RF envelope channels information table and the gradient channels information table. Also a button is provided to change the amplitudes of the RF envelope pulses simultaneously as explained in Chapter 4.

The currently modified table entry is displayed in red color, as shown in A2.2.

If the user gives an erroneous input, a dialog box that informs the user about the place and the type of the error is displayed, as shown in A2.3.

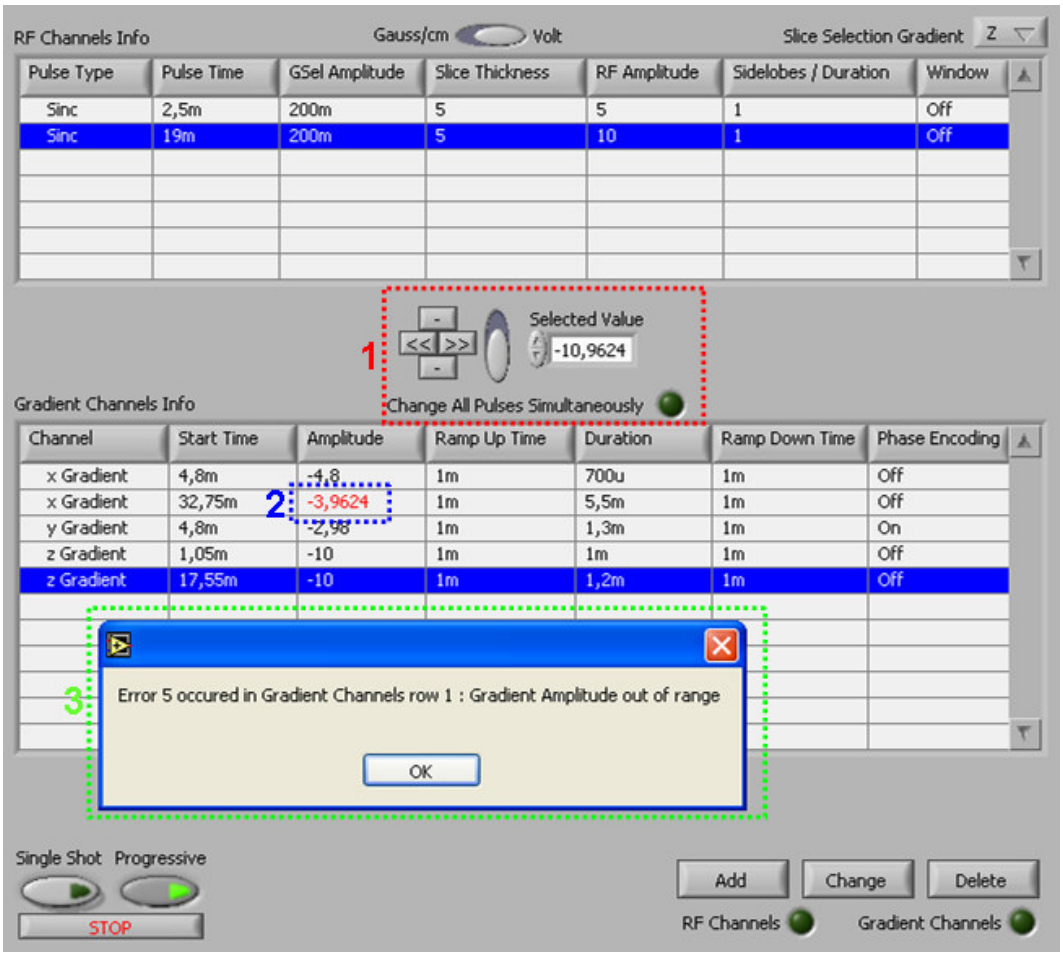


Figure A.2: Software front panel during the progressive mode is engaged.

A.4 MR-CDI tab

A screenshot of the MR-CDI tab is given in Figure A.3.

This tab is used to display the current channel during MR-CDI experimentation. The current source control S0 and S1 channels are displayed as single channel in this tab with S0 corresponding to the positive polarity and S1 corresponding to the negative polarity.

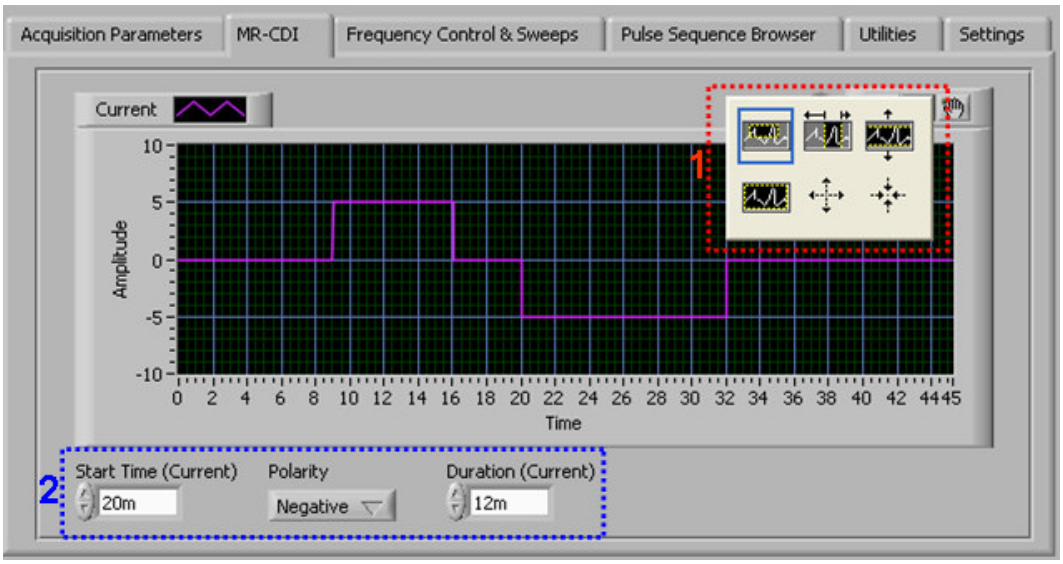


Figure A.3: MR-CDI tab.

A3.1 is the menu of the zoom palette that contains functionality to zoom in, zoom out, boxed zoom etcetera.

A3.2 is the controls that are used for the specification of the current source control signals.

A.5 Frequency control & sweeps tab

A screenshot of the frequency control & sweeps tab is given in Figure A.4.

In A4.1 frequency setting control knobs are shown. The leds on the top represent the frequency control signal Data that will be driven into the frequency synthesizer control card.

In A4.4 a voltage sweep or a frequency sweep can be selected to be performed. Also the start, increment and end values of the sweep are specified.

In A4.2 there is a button for switching between the local and remote modes, and there are two indicators for displaying the frequency value and the current value of the sweep.

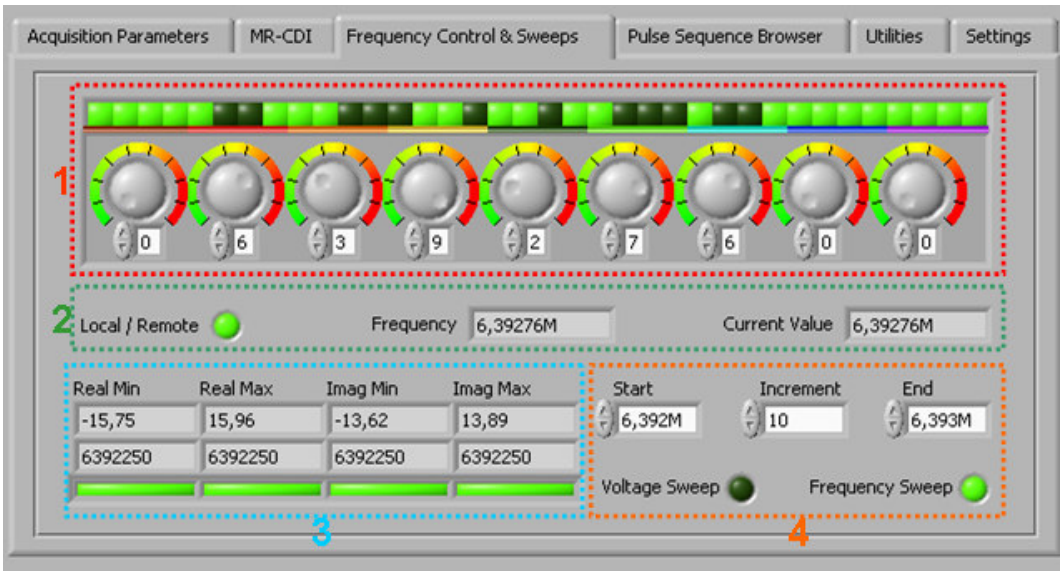


Figure A.4: Frequency control & sweeps tab.

A4.3 has indicators to display the peak values along with the value of the sweep variable for the peaks found. Obviously, these peaks are detected for the real and imaginary echo channels. There are four leds at the bottom showing that a peak is detected by the software.

A.6 Simultaneous display of data during progressive mode

In progressive mode, all the parameters of a pulse sequence can be changed on the fly, sweeps are performed, and the results are displayed simultaneously. This simultaneous display is performed on the front panel of the "Image Reconstruction" software. The simultaneous display data is shown in Figure A.5.

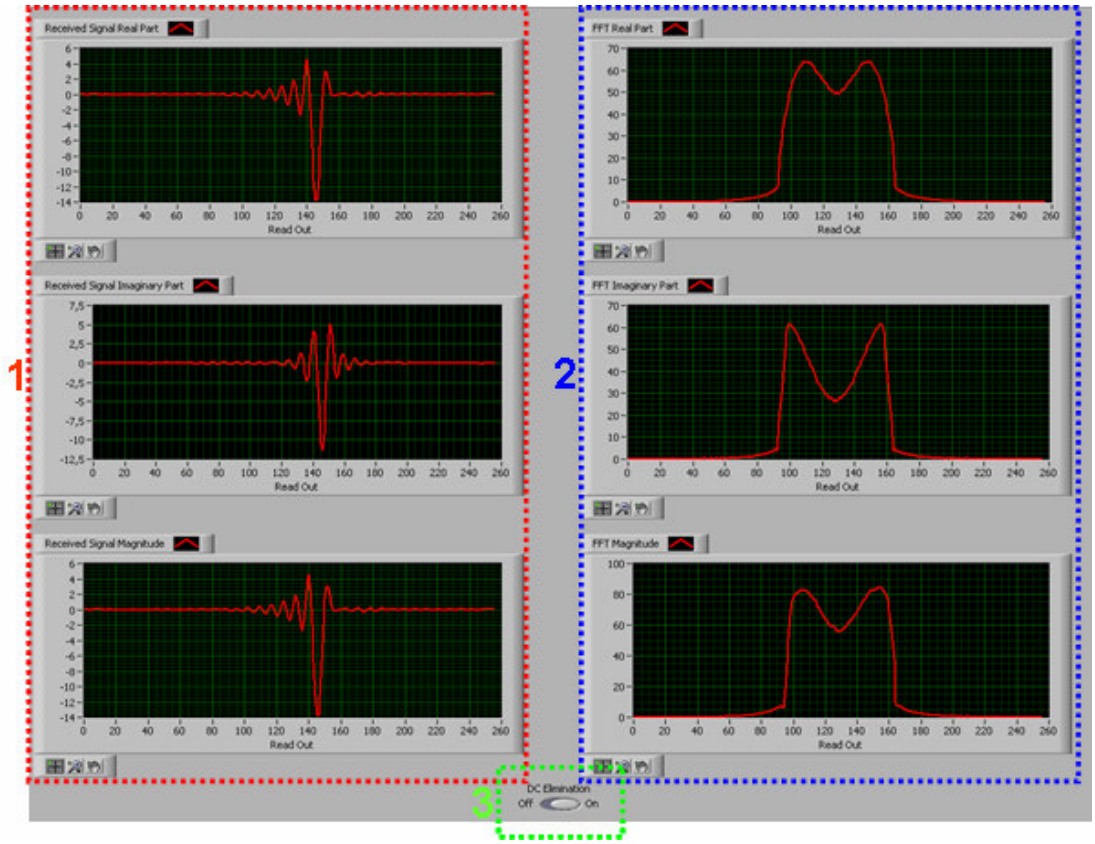


Figure A.5: Simultaneous data display.

In A5.1 the received signals are displayed to the user whereas in A5.2 their corressponding FFTs are displayed. In A5.3 there is an option for performing DC offset elimination.

A.7 Pulse sequence browser tab

A screenshot of the pulse sequence browser tab is given in Figure A.6.

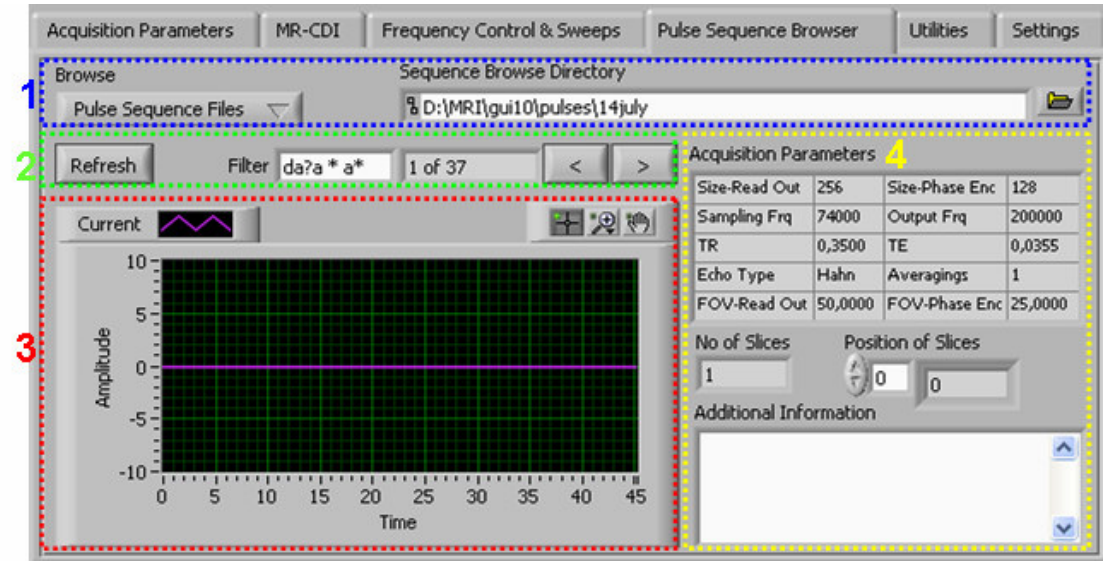


Figure A.6: Pulse sequence browser tab.

In A6.1 the browsing directory, and the types of files to be detected (pulse sequence files, data files or both) is specified.

In A6.2 filtering can be applied to the file names, there is an indicator for displaying the currently browsed file, there are controls to refresh the browse operation according to the new values entered, browse the previous file and browse the next file.

The current channel of the browsed file is displayed in A6.3. Also all critical pulse sequence information for the browsed file is displayed in A6.4.

A.8 Utilities tab

A screenshot of the utilities tab is given in Figure A.7.

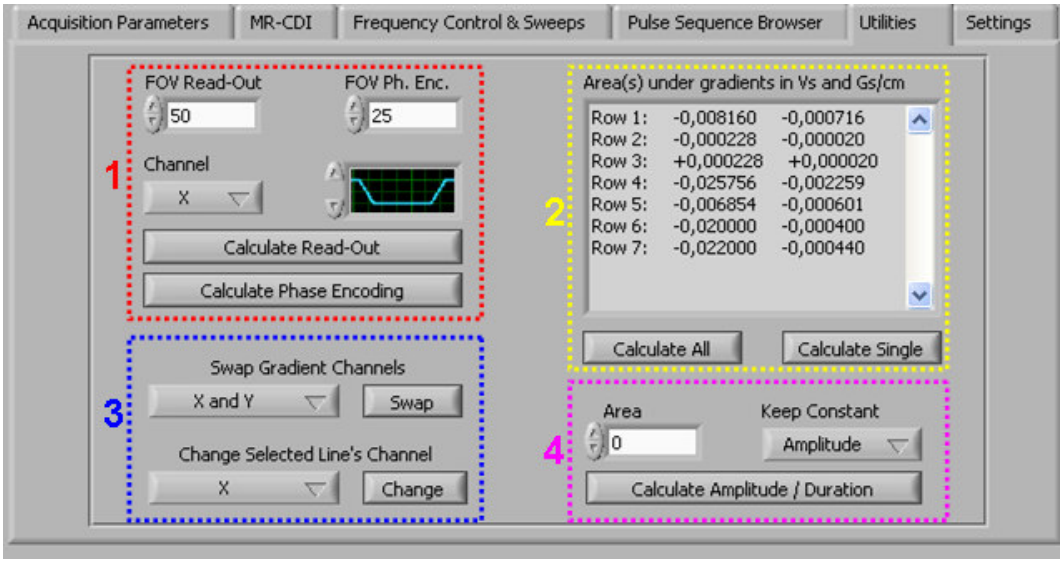


Figure A.7: Utilities tab.

The operations performed in this tab are previously mentioned while explaining the events of the software in Chapter 4.

In A7.1, the necessary field strengths to generate the specified amount of field of view are calculated. In A7.2 area(s) under gradient(s) can be calculated. In A7.3 gradients can be swapped or a specified gradient's channel can be changed. In A7.4 a gradient block can be designed based on the area under it.

A.9 Settings tab

A screenshot of the settings tab is given in Figure A.8.

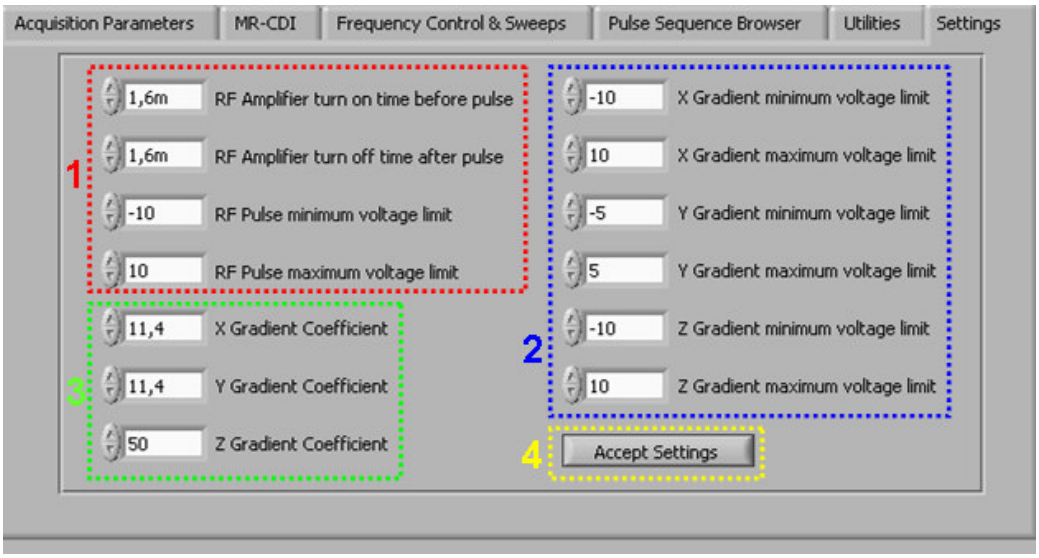


Figure A.8: Settings tab.

The parameters in this tab were explained in detail in Chapter 4, previously. In A8.1 and in A8.2 there are controls for setting the critical limits. In A8.3 the gradient coefficients can be specified. A8.4 is for accepting the settings and saving the changes to the configuration settings file.

APPENDIX B

GRAPHICAL USER INTERFACE OF THE IMAGE RECONSTRUCTION SOFTWARE

B.1 Notation used when explaining the GUI of the software

The same notation that is explained in Section A.1 is used in this section. Also throughout this section the “Image Reconstruction” software will be referred as software, shortly.

B.2 Front panel of the software

A screenshot of the front panel of the software is given in Figure B.1.

In B1.1 tool palettes for drawing ROIs (region of interest) on the image area and image windows are given.

In B1.2 two sample image windows are displayed with different colormaps. In B1.3 an image in image area is shown along with a ROI drawn on the image. B1.4 gives information about the pixel coordinates and intensity.

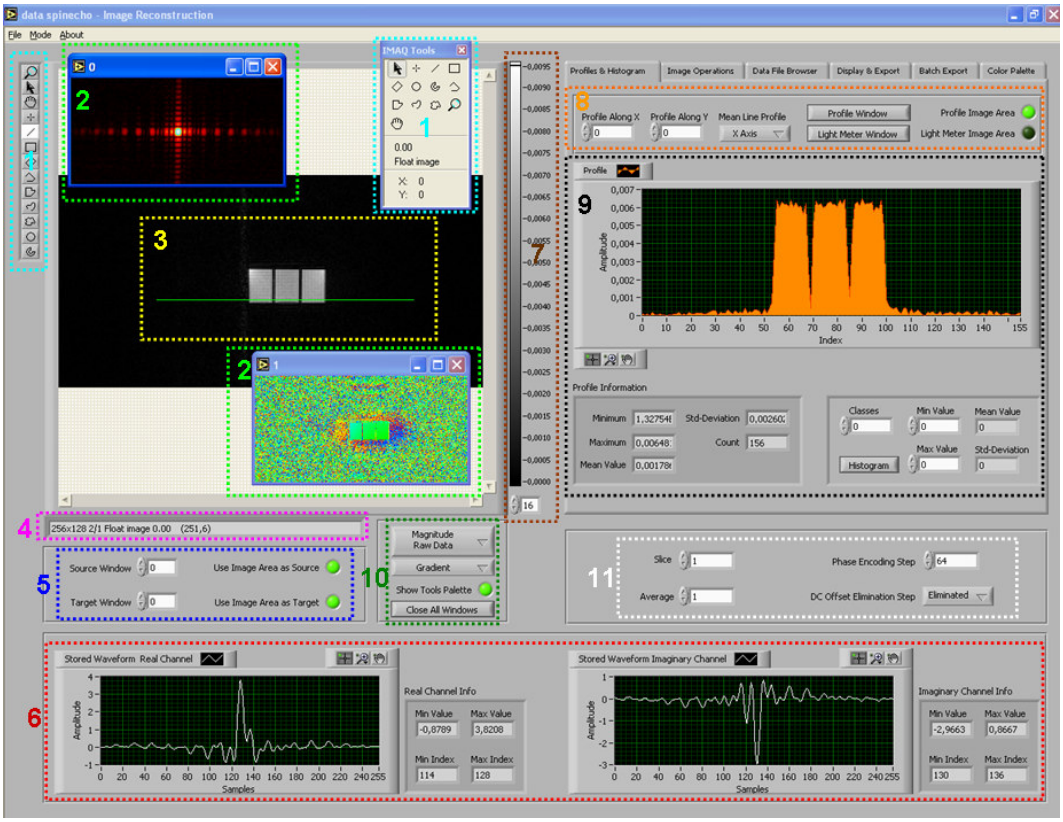


Figure B.1: Front panel of the Image Reconstruction software.

In B1.5 there are controls for specifying the source and target images that the operations will be performed on.

B1.6 has two waveform graphs displaying the received echo signals (real and imaginary channels) for a specified pulse sequence cycle. Also peak values of these signals are shown.

The color ramp used for the images is shown in B1.7. Also there is a controller to select the color ramp of which image is to be displayed.

In the screenshot given in Figure B.1, the profiles & histogram tab of the tab selector is displayed. This tab selector contains many functionality of the software in its tabs.

In B1.8 there are tools for profiling, linear averaging and performing the light meter operations that are previously explained in Chapter 5.

The results of the above mentioned operations are displayed in the waveform graph located in B1.9. The histogram operation is also performed here. Also there are indicators to display profile information.

In B1.10 there are buttons to specify the image to be reconstructed, to set the colormap of the image, to close all open image windows and to hide or show the tools palette for window operations.

By using the controls in B1.11, the user can specify the cycle of the pulse sequence that he/she wants to examine. Also images are reconstructed based on the slice specified in these fields.

B.3 Image operations tab

A screenshot of the image operations tab is given in Figure B.2. In this tab various image operations that are explained in Chapter 5 are performed.

B2.1 is the magic wand mask creation operation, using the controls in B2.2 a source image can be masked or a part of it is extracted based on a mask. B2.3 is for converting a ROI to a mask. B2.4 is used to convert a mask into a ROI and to copy ROI of an image to another one. B2.5 contains the controls related with the shift operation. B2.6 is for thresholding and B2.7 is for three dimensional viewing. Images can be rotated using the controls in B2.8. In B2.9 there are buttons for copying images, saving images and unwrapping images. Difference operations can be performed using B2.10.

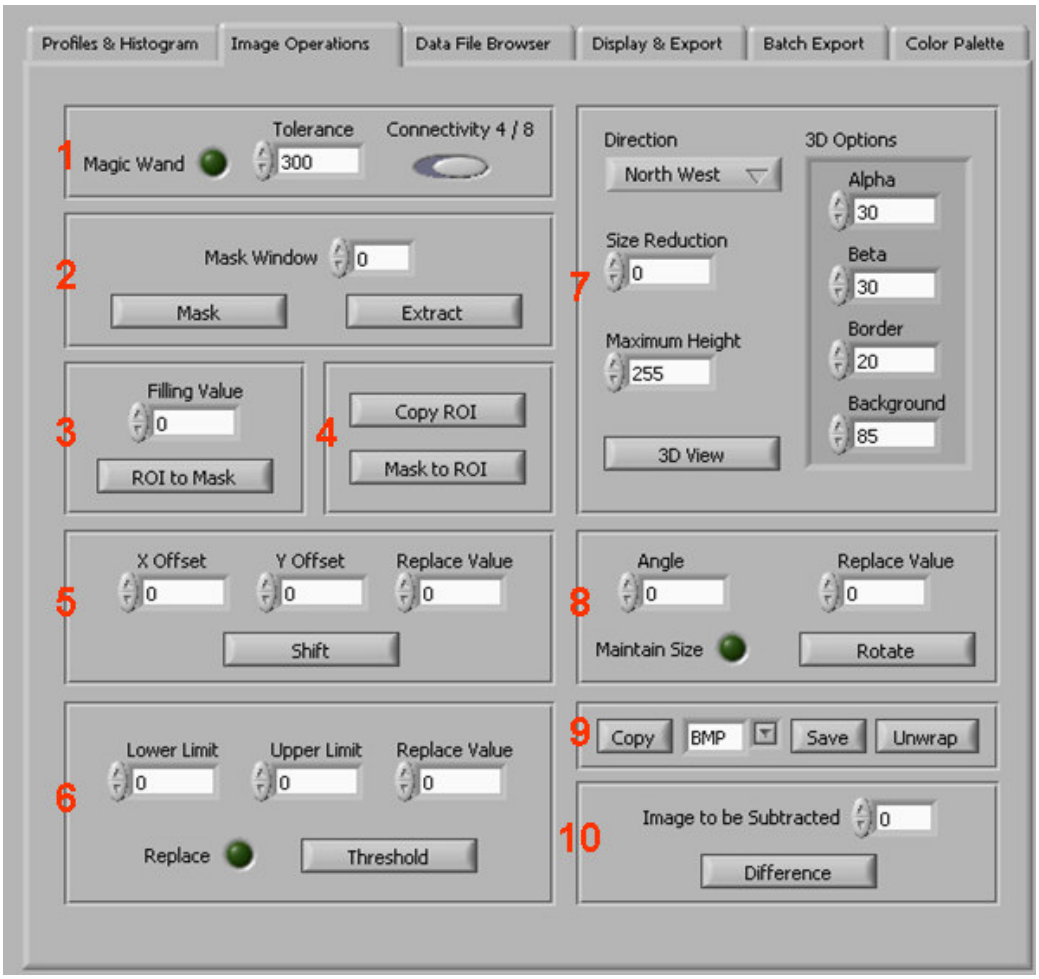


Figure B.2: Image operations tab.

B.4 Data file browser tab

A screenshot of data file browser tab is given in Figure B.3. Same functionalities with the pulse sequence file browser is provided in this tab. Obviously, this time data files are loaded, and their images are reconstructed during browsing.

The control of the browsing operation can be done in B3.1 and the critical pulse sequence information can be viewed from B3.2.

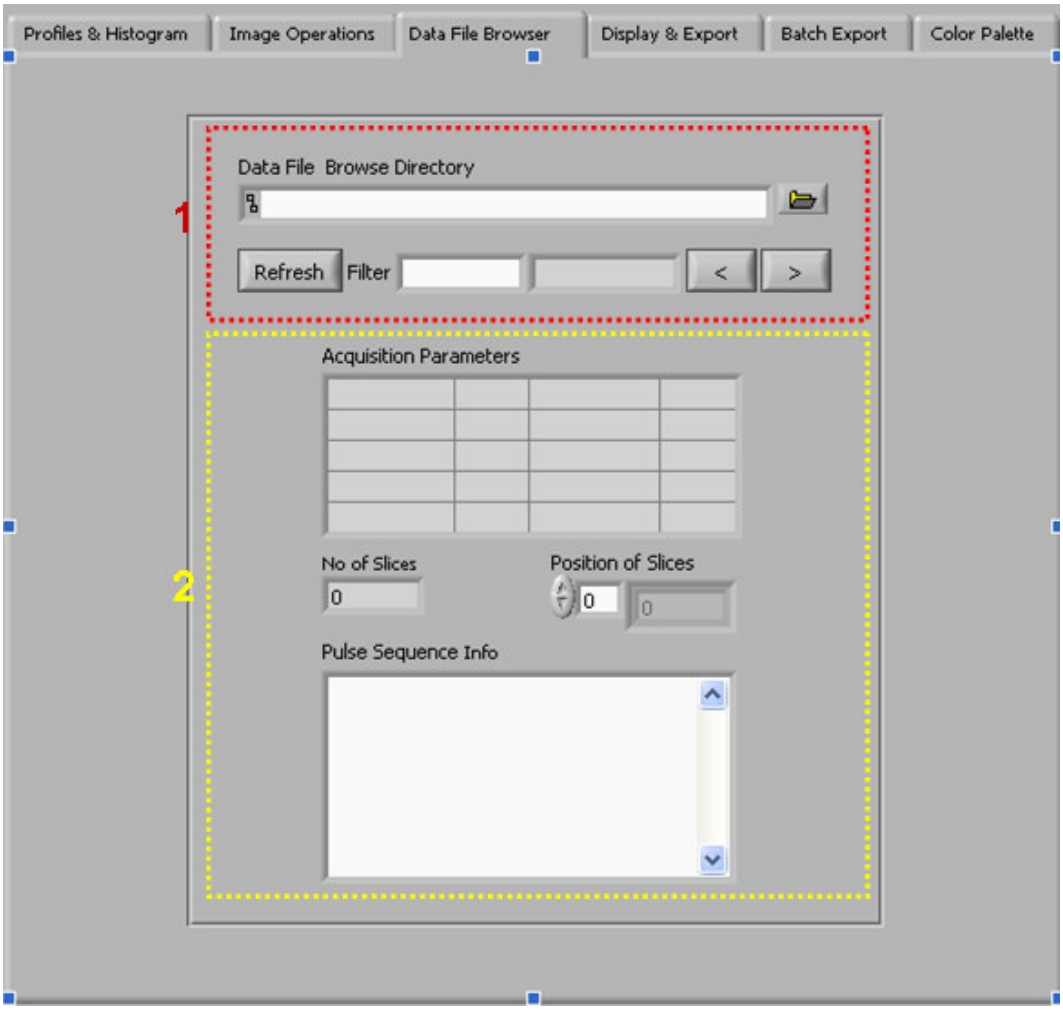


Figure B.3: Data file browser tab.

B.5 Display & export tab

A screenshot of display & export tab is given in Figure B.4. Using the leds in B4.1 images can be cleared permanently form memory whereas the leds in B4.2 are used for displaying or hiding requested images.

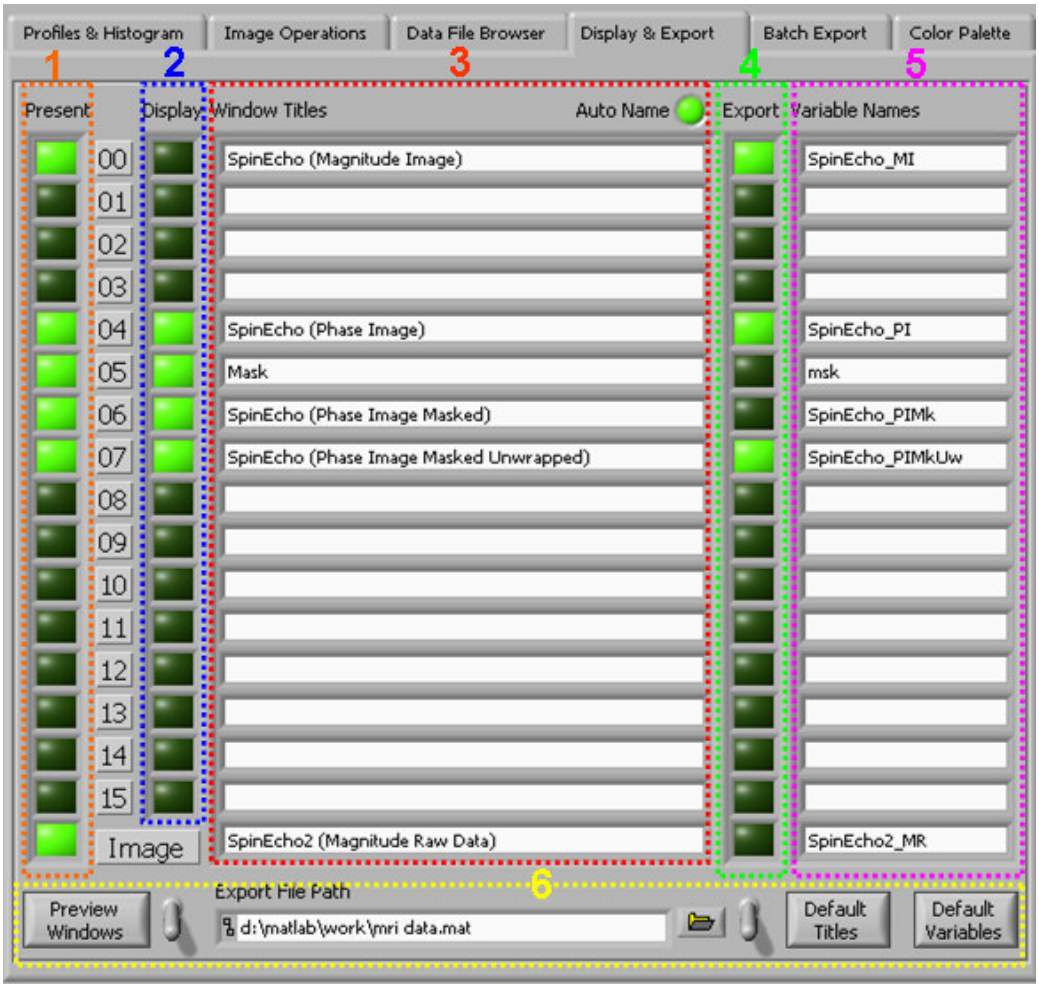


Figure B.4: Display & export tab.

Image window titles can be entered from B4.3, and there is also an auto naming functionality to handle the naming process of the window titles and variable

names automatically. Using the leds in B4.4 requested images can be exported to Matlab. B4.5 is for entering the Matlab variable names. In B4.6 multiple windows can be viewed in a single image, the path of the export file can be specified and titles and variable names can be defaulted.

B.6 Batch export tab

A screenshot of the batch export tab is given in Figure B.5. Batch export functionality is explained in Chapter 5.

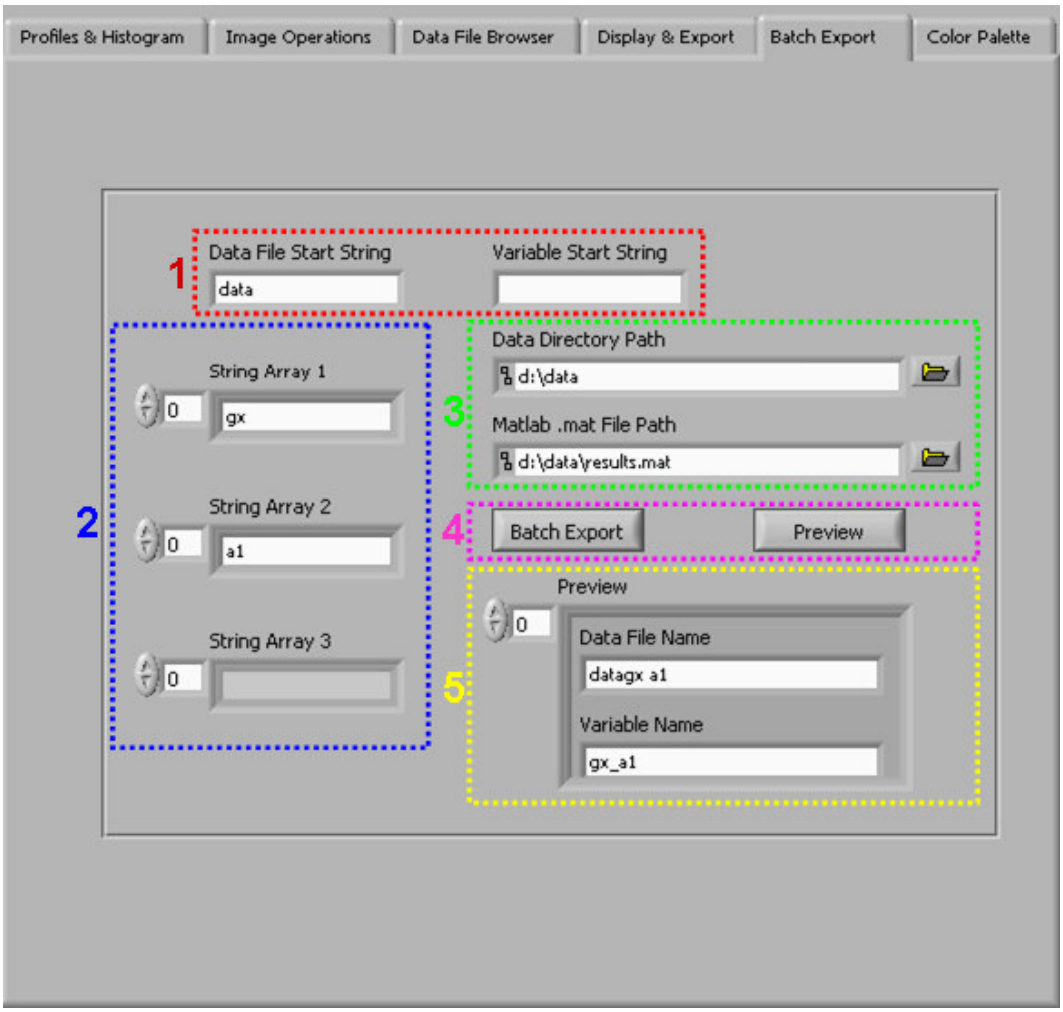


Figure B.5: Batch export tab.

In B5.1 the data file start string and the variable start string can be inputted. In B5.2 the three string arrays can be entered. The data file directory and the save file path can be entered from B5.3. The generated data file names and variable names can be previewed from B5.5 by pressing the preview button located in B5.4. Finally, the requested dataset can be batch exported to Matlab by pressing the batch export button in B5.4.

B.7 Color palette tab

A screenshot of the color palette tab is given in Figure B.6.

In B6.1, using the beginning color, the end color, and by specifying an intermediate color from the graph, custom colormaps can be generated. These colormaps are displayed in B6.2.

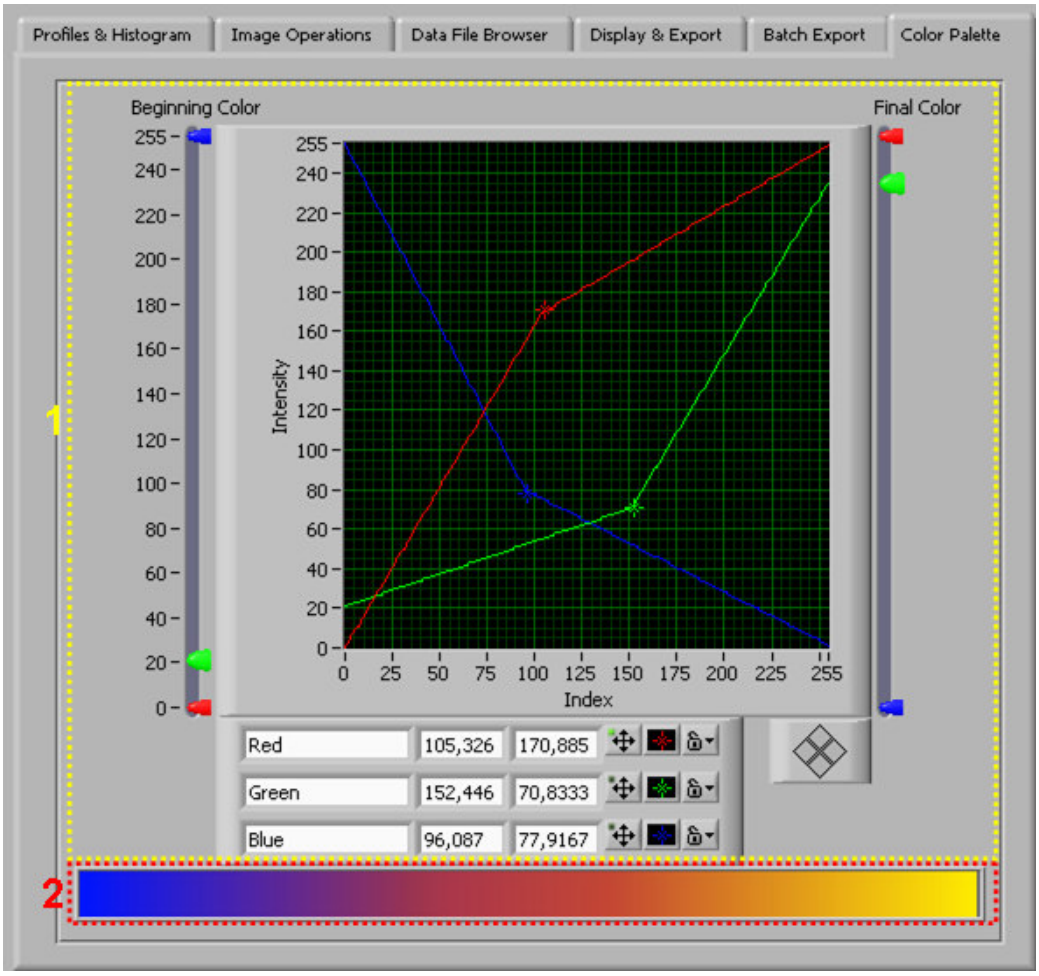


Figure B.6: Color palette tab.

APPENDIX C

CATEGORIZED LISTING OF SUBVIS THAT ARE USED IN THE SOFTWARE SUIT

SubVIs of the “Pulse Sequence Design & DAQ” software and the “Image Reconstruction” software are given in Table C.1 and Table C.2 respectively. The ones that are shown in italics are used from LabVIEW libraries whereas the ones that are shown in regular style are written during this thesis.

Table C.1: subVIs used in the pulse sequence design and DAQ software.

subVI Name	**Category**
Calculate RF Waveform.vi	Waveform design
Calculate Gradient Waveforms MR-CDI.vi	
Set Plot Properties.vi	Waveform display
Sort RF Pulses.vi	
Sort Gradients.vi	
Convert to Listbox.vi	
Convert V to G per cm.vi	
Check Errors.vi	Error checking
General Error Handler.vi	
Merge Errors.vi	
Progressive RF Change.vi	Progressive mode
Progressive Gradients Change.vi	
Display Current Data.vi	
Frequency to Knobs.vi	Frequency control
Knobs to BCD & Numeric.vi	
Load Limits.vi	File operations
Save Pulse Sequence.vi	
Load Pulse Sequence.vi	
Open/Create/Replace File.vi	
Write Key.vi	
Write Key (Double).vi	
Check Single File.vi	
Check Browse Files.vi	
Open Config Data.vi	
Close Config Data.vi	
Not A Config Data Refnum.vi	
Set GUI Object(s) Mode.vi	Miscellaneous
Disable Channels.vi	

Table C.1 (continued)

subVI Name	**Category**
DAQmx Timing.vi	Data acquisition
DAQmx Timing (Implicit).vi	
DAQmx Create Virtual Channel.vi	
DAQmx Create Channel (CO-Pulse Generation-Frequency).vi	
DAQmx Start Task.vi	
DAQmx Trigger.vi	
DAQmx Start Trigger (Digital Edge).vi	
DAQmx Timing (Sample Clock).vi	
DAQmx Write.vi	
DAQmx Read.vi	
DAQmx Create Channel (AO-Voltage-Basic).vi	
DAQmx Create Channel (AI-Voltage-Basic).vi	
DAQmx Write (Analog 2D DBL NChan NSamp).vi	
DAQmx Clear Task.vi	
DAQmx Stop Task.vi	
DAQmx Wait Until Done.vi	
DAQmx Read (Analog 2D DBL NChan NSamp).vi	
DAQmx Create Channel (DO-Digital Output).vi	
DAQmx Write (Digital Bool 1Line 1Point).vi	
DAQmx Configure Output Buffer.vi	
DAQmx Control Task.vi	

Table C.2: subVIs used in the image reconstruction software.

subVI Name	**Category**
Display Image.vi	Image display
Display Requested Images.vi	
Scale Image to 8 Bits.vi	
Set Color Ramp Palette Table.vi	
Color Table Generator.vi	
Convert Color Palette to Color Table.vi	
Check Variable Name.vi	Matlab export
Check All Variable Names.vi	
Export Requested Images.vi	
Change Variable Names.vi	
Check Auto Variable Name.vi	
Load Pulse Sequence.vi	File operations
Open/Create/Replace File.vi	
Check Single File.vi	
Find Load Offset.vi	
Check Browse Files.vi	
Extract Raw Data.vi	
Save to MAT	
Convert Little Endian to Big Endian.vi	
Check Valid Window.vi	Image window handling
Change Window Titles.vi	
Copy Keys.vi	
Check Auto Titles.vi	
Trim Whitespace.vi	
General Error Handler.vi	Error checking
Merge Errors.vi	
System Exec.vi	Miscellaneous
Set GUI Objects Continuous.vi	

Table C.2 (continued)

subVI Name	**Category**
IMAQ Create	Image processing
IMAQ MagicWand	
IMAQ MaskToROI	
IMAQ ArrayToComplexImage	
IMAQ ComplexPlaneToImage	
IMAQ Subtract	
IMAQ ComplexFlipFrequency	
IMAQ WindToolsShow	
IMAQ WindClose	
IMAQ ROIProfile	
IMAQ WindToolsSetup	
IMAQ Mask	
IMAQ Get Custom Keys	
IMAQ WindShow	
IMAQ Dispose	
IMAQ 3DView	
IMAQ Read Custom Data	
IMAQ Write Custom Data	
IMAQ Rotate	
IMAQ Threshold	
IMAQ LinearAverages	
IMAQ Histograph	
IMAQ WindGetROI	
IMAQ ImageToArray	
IMAQ ArrayToImage	
IMAQ GetImageInfo	
IMAQ ROIToMask	
IMAQ WindSetROI	
IMAQ GetRowCol	
IMAQ Convert Line to ROI	
IMAQ GetImageSize	
IMAQ Extract	
IMAQ Light Meter (Rectangle)	

Table C.2 (continued)

subVI Name	**Category**
IMAQ Convert ROI to Rectangle	Image processing
IMAQ Shift	
IMAQ Browser Setup	
IMAQ Browser Replace	
IMAQ WriteFile	
IMAQ Cast Image	
IMAQ ColorUserLookup	
IMAQ Copy	
IMAQ Browser Delete	
IMAQ InverseFFT	

APPENDIX D

CONFERENCE PRESENTATIONS PERFORMED DURING THE THESIS

[1] Murat Eşref Özsüt and B. Murat Eyüboğlu, "Design and implementation of LabVIEW© based data acquisition and image reconstruction environment for METU magnetic resonance imaging system", *Biomed 11th International Conference*, *Ankara* – Turkey, 2004.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name : Murat Eşref ÖZSÜT

Nationality : Turkish (TC)

Date and Place of Birth : 7 October 1980, Ankara

Marital Status : Single

Phone : +90 532 6631010 (gsm), +90 312 4365607 (home)

Email : mozsut@superonline.com

EDUCATION

Degree	Institution	Year of Graduation
MS	METU Electrical & Electronics Engineering	2005
BS	Başkent University Electrical & Electronics Engineering	2002
High School	TED Ankara College	1998

WORK EXPERIENCE

Year	Place	Enrollment
Sep. 2002 - Sep. 2005	METU, Department of Electrical & Electronics Engineering	Researcher in Biomedical Group

Jun. 2001 - Oct. 2001	Stanford University, Ginzton Laboratory	Visiting researcher in Khuri-Yakub Ultrasonics Group
Jul. 2000 - Oct. 2000	Karel Electronics	Intern Student in Research & Development Dept.

ACADEMIC HONORS

2003-2004 Turkish Scientific and Technical Research Council (TUBITAK) – Munir Birsel Foundation, Graduate Scholarship

1998-2002 Baskent University, Fellowship

FOREIGN LANGUAGES

Advanced English

CONFERENCE PRESENTATIONS

1. Murat Eşref Özsüt and B. Murat Eyüboğlu, "Design and implementation of LabVIEW© based data acquisition and image reconstruction environment for METU magnetic resonance imaging system", *Biomed 11th International Conference*, *Ankara* – Turkey, 2004.

HOBBIES

Skiing, Tennis, Guitar, Latin dancing, Sailing