

DESIGN AND DEVELOPMENT OF A BREATH-BY-  
BREATH ANALYSIS SYSTEM FOR  
USE AT LOW ALTITUDES

By

CHARLES T. PATTERSON

Bachelor of Science in Electrical Engineering

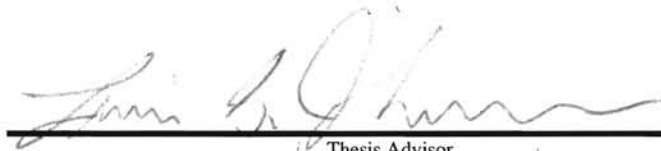
Stillwater, Oklahoma

1993

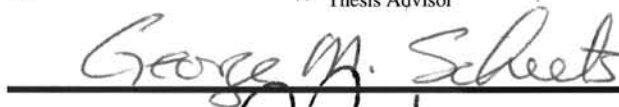
Submitted to the Faculty of the  
Graduate College of  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
December, 1996

DESIGN AND DEVELOPMENT OF A BREATH-BY-  
BREATH ANALYSIS SYSTEM FOR  
USE AT LOW ALTITUDES

Thesis Approved:



Thesis Advisor



Dean of Graduate College

## PREFACE

I found designing and developing a breath-by-breath analysis system to be very rewarding. The challenges and triumphs I experienced, will help me throughout my engineering career. I received a great deal of help on this project and would like to thank all of the people who helped make this project possible.

First, I would like to thank the Federal Aviation Association for funding this project and making the facilities available for its development. A big thanks is given to Bob Garner, the team leader of the project, for his support, advice, and the opportunity he gave me to work on this project. I owe a deep gratitude to Frontier Engineering, my current employer, and Scott Phillips, my manager, for permitting me to use Frontier's facilities for this project and for tolerating my hectic schedule so that I could complete this last step of my Masters.

I would also like to thank the members of my advisory committee, Dr. Louis Johnson, Dr. George Scheets, and Dr. Ramakumar. A special thank you is extended to Dr. Louis Johnson for his guidance and patience during this project and previous semesters.

My parents, Tom and Marsha Patterson, deserve the greatest thanks of all, without their love and support I never could have reached this far.

## TABLE OF CONTENTS

Figure	Page
I. INTRODUCTION.....	1
Purpose and Motivation.....	1
Overview of Procedure.....	3
II. BACKGROUND.....	4
Review of Apparatus.....	4
Inhale/Exhale Flow.....	6
Gas Composition of Breaths.....	7
Pressure.....	8
Temperature.....	8
Relative Humidity.....	8
Data Acquisition.....	8
Computer System.....	9
Selection of Software Development Tool.....	10
Introduction to Graphical Programming.....	10
Introduction to LabVIEW.....	11
Benefits of Using LabVIEW.....	11
III. DESIGN AND DEVELOPMENT.....	14
Operation in Simulated Mode.....	14
Acquiring Data at Desired Rate.....	16
Error Handling.....	18
Calibration.....	19

Circular Buffering of Data .....	21
Circular Buffer w/ 2D Data Write.vi.....	21
Convert Circular Index into Linear One.vi.....	23
Convert Linear Index into Circular One.vi.....	24
Find Signal Index and Channel.vi .....	24
Recognition of Flow Signal.....	24
Find Inhale.vi .....	25
Find Exhale.vi.....	27
Adjusting for Drifting Baseline.....	27
Recognition of Inhale/Exhale Gas Fraction Signals .....	28
Aligning Signals.....	28
Processing Breath Data .....	30
Displaying Data.....	35
Troubleshooting Display .....	35
Main Display.....	36
 IV. CONCLUSIONS .....	 38
Verification of Program Results.....	39
Future Work.....	40
 BIBLIOGRAPHY.....	 41
 APPENDIX A - CODE LISTING.....	 42

## LIST OF TABLES

Figure	Page
I. Linear Conversion Factors .....	32
II. Curvilinear Coefficients.....	32

## TABLE OF FIGURES

Figure	Page
1. Schematic of instrumentation .....	5
2. Schematic of pneumotachograph.....	7
3. PC-Based Test Market Insight Study .....	12
4. Example VI for buffered data acquisition .....	17
5. Pneumotachograph calibration setup .....	20
6. Circular buffer VI input/output definitions.....	21
7. Example of circular buffer initialization .....	22
8. Example of circular buffer write.....	23
9. Double threshold detection .....	26

## CHAPTER I

### INTRODUCTION

This project encompasses the acquisition and analysis of human gas exchange at different altitudes for the purpose of analyzing metabolic responses and breathing kinetics during acute exposure to altitude. All work has been performed in conjunction with the Environmental Physiology Research team within the Protection and Survival Laboratory. These facilities are located at the Federal Aviation Association's Mike Monroney Aeronautical Center in Oklahoma City, Oklahoma. This thesis describes the depth and scope of the electrical and software engineering aspects of performing and displaying breath-by-breath analysis in real-time at altitude.

#### Purpose and Motivation

The primary reason for developing this system is that current systems that perform breath analysis provide a canned solution that is not modifiable. These systems are limiting in that they do not permit the scientist to control the methods used to process the breathing data, and they do not permit additional signals to be monitored and stored along with the breathing data. This system was designed in a manner that is easily modifiable and that allows additional variables, specifically altitude, to be accounted for in processing the breathing data. In addition this newly developed system provides the user with the ability to modify the manner in which the data is displayed and logged.

Past attempts at collecting breath-by-breath analysis have been developed, but assumptions were made which caused results to be accurate only over a group of breaths. In addition, the assumptions made about gas exchange in the lungs precluded these studies from being performed at altitude.

With the data acquisition and software products available today and the high processing speeds of the current computer technologies more data can be collected so that more accurate results can be obtained at different altitudes.

Although the readily available technology has reduced development time from past attempts at this type of study, the same obstacles are ever present. Finding a reliable and efficient form of calibration of the equipment is an ongoing challenge with new ideas and improvements surfacing every step of the way. Another formidable task was aligning the data collected from different sources such that calculations could be performed and results displayed on a breath-by-breath basis.

Since most of the apparatus described in this project was already present, the primary topic of this thesis is the software program written to perform the data acquisition and display of data. The use of a graphical programming language specifically designed for instrumentation and data acquisition dramatically reduced the development cycle of this project and will be described in detail. Graphical programming and other innovative techniques and solutions will be covered in the body of this thesis.



## Overview of Procedure

Breath by breath measurement of gas exchange consists of the following quantities: inhale and exhale volumes, gas composition of inhale and exhale volumes, pressure, temperature, and relative humidity. The flow samples are integrated to calculate the inspired and expired volumes of Oxygen ( $O_2$ ), Carbon Dioxide ( $CO_2$ ), Nitrogen ( $N_2$ ), Argon (A), and Water ( $H_2O$ ). Atmospheric conditions such as temperature, pressure, and relative humidity of the air are used in calculations to normalize volumes to a standard atmospheric condition of air referred to as Standard Temperature and Pressure Dry (STPD), and to a standard condition of bodily measurement termed Body Temperature Pressure Saturated with water (BTPS). Cardiopulmonary measurements are used to determine the physical activity of the subject during the test.

## CHAPTER II

### BACKGROUND

The Mike Monroney Areonautical Center houses an altitude chamber which was very well equipped for this project. Prior to the software development, the hardware required to perform breath-by-breath analysis had already been assembled by Bob Garner. This section addresses the existing hardware and the software tools selected for the development.

#### Review of Apparatus

The physical apparatus required for breath-by-breath analysis included a pentium-based personal computer; a data acquisition card; a breathing tube; a two-way breathing valve; and several specialized instruments to collect inhale and exhale flow, gas composition of the air, temperature, relative humidity, and pressure. All of the instruments involved produced voltages to represent the physical quantity being measured and these voltages were wired directly to the data acquisition card housed in the computer. Since many of the instruments either had probes or tubes that need to be hooked up within the altitude chamber, the associated wires and tubes were fed into the chamber through access ports which were sealed with clay during operation at altitude. Figure 1 provides a system-level schematic of the apparatus. The following sections describe each of the necessary devices.

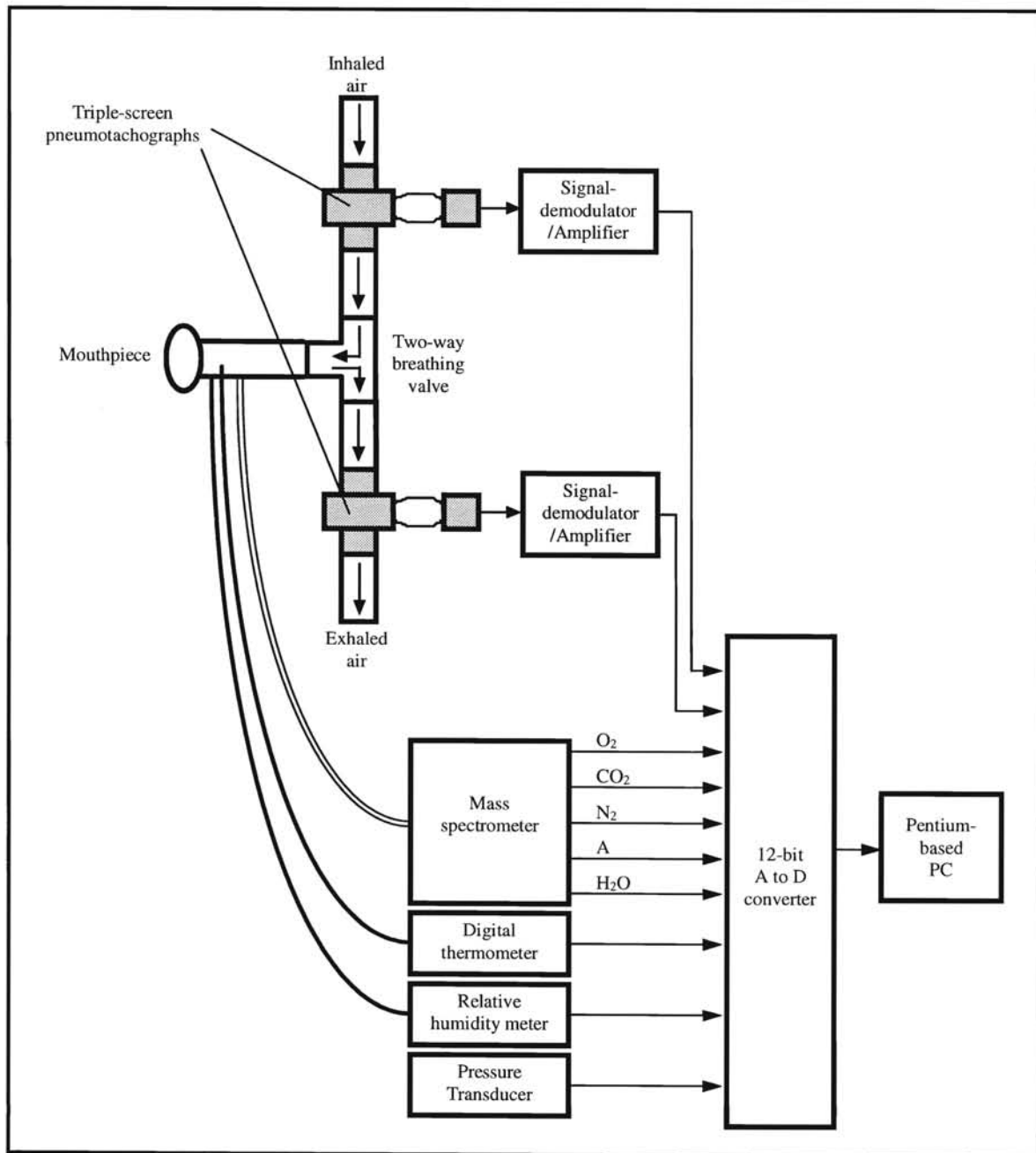


Figure 1: Schematic of instrumentation

## Inhale/Exhale Flow

The inhale and exhale flow signals were measured using pneumotachographs. Pneumotachographs are devices that measure airflow against a fixed resistance and may be found in respirometers, mechanical ventilators, incentive breathing devices, diagnostic spirometers, and other pulmonary function testing instruments. Although several designs exist, a triple screen type pneumotachograph made by Hans Rudolph, Incorporated, was selected for this system. This design was selected because it has slightly better frequency response, less dead space, and is easier to disassemble for cleaning than other designs.

The device consists of an air flow tube, three metal screens with mesh sizes of 400 wires per inch, a heating element, two pressure tubes, and a differential pressure transducer (see figure 2). The heating element heats the metal screens to prevent water condensation on the screens. The three screens form two chambers and pressure tubes from each chamber are connected to a differential pressure transducer. A signal demodulator and amplifier are connected to the differential pressure transducer in order to convert the signal into a voltage linearly dependent on the flow.

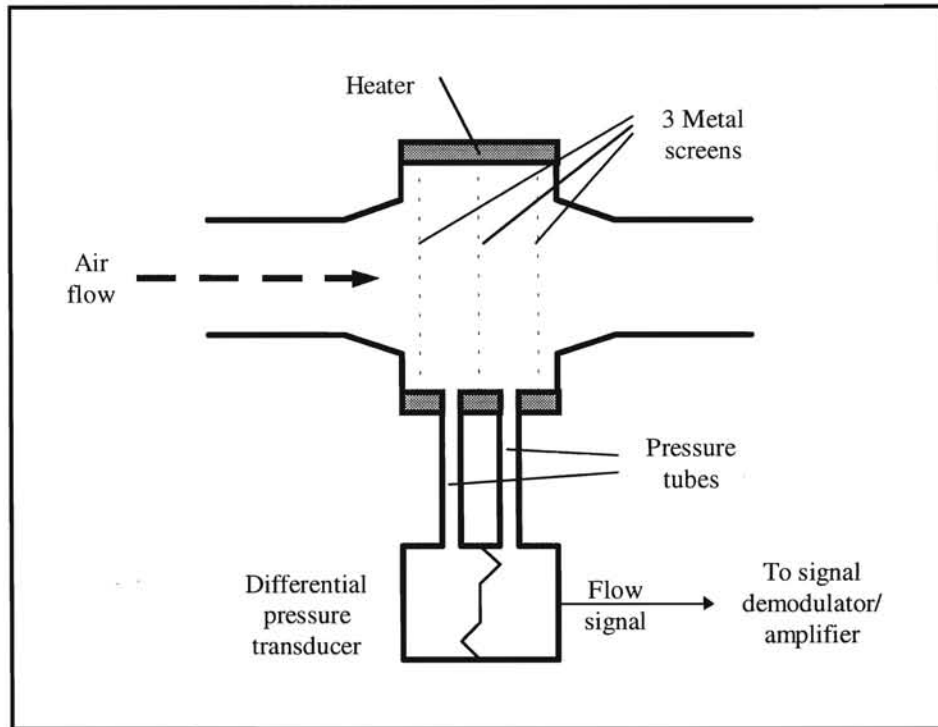


Figure 2: Schematic of pneumotachograph

Two triple screen pneumotachographs were used to measure the flow signals needed for this system. One was placed on the input side of the two-way breathing valve, for inhale flow, and the other was placed on the output side of the valve, for exhale flow.

### Gas Composition of Breaths

A thin capillary tube carries a small sample of air from the breathing mask to a device called a mass spectrometer where the gas composition is determined. The mass spectrometer continuously samples the air carried through the tube and determines the fractions of  $O_2$ ,  $CO_2$ ,  $N_2$ , A, and  $H_2O$ . Each gas fraction has a corresponding voltage output which is proportional to the reading.

## Pressure

The altitude chamber located on the Mike Monroney Aeronautical Center simulates different altitudes by controlling the air-pressure inside the chamber. The relationship between air-pressure and altitude is relatively linear, and thus a simulated altitude can be determined from an air-pressure reading. Several pressure gauges are present in the chamber, but in order to make this reading automatically, an Omega brand pressure transducer was placed in the chamber and connected to the data acquisition card.

## Temperature

Two different thermometers are used to acquire the temperature readings necessary for breath-by-breath analysis. An Omega 871A digital thermometer measures the temperature of the air being breathed in and out, while a Hewlett Packard thermometer measures the ambient temperature of the chamber.

## Relative Humidity

An Omega Relative Humidity meter (DP 205-E) performed the relative humidity measurements for the system. The voltage output from this device was wired directly to the data acquisition card.

## Data Acquisition

All of the instruments needed for this system are wired into a National Instruments data acquisition card, the AT-MIO-64F-5. This card's selection was based on the

following criteria: it is capable of a fast sample rate, in case higher frequency signals are added to the analysis; it contains plenty of channels for single-ended or differential measurements; and it is easy to interface with the selected software language.

The AT-MIO-64F-5 performs a 12-bit analog-to-digital conversion on the present voltage signals, can sample up to 1000 samples a second per channel, and provides 64 single-ended A/D channels where pairs of these channels may be used to perform differential measurements. The pneumotachograph signal is wired into a signal conditioner which provides a single ended output with a common ground with the data acquisition board. In order to reduce signal noise introduced by long cabling, the mass spectrometer, pressure transducer, relative humidity, and temperature signals are all connected in differential mode. All signals present on this board range from 0 to 10 volts. The signals are sampled simultaneously at 200 Hz and are buffered to the computer's memory using a Direct Memory Access (DMA) channel.

### Computer System

A Pentium-based personal computer with a 166 Mega Hertz processor, 64 Mega bytes of Random Access Memory (RAM), and a two Giga-byte hard-drive was selected for the acquisition system so that plenty of processing speed and storage room were available. Microsoft Windows 95 was selected as the operating system to take advantage of 32-bit processing and the plug-n-play technology existing in the hardware add-in cards selected for this and future projects.

## Selection of Software Development Tool

Choosing a good software development tool for this system proved to be critical to the project's successful completion. This section addresses the decision to use the graphical programming language known as LabVIEW as opposed to a text based language like C, Basic, or FORTRAN.

### Introduction to Graphical Programming

Graphical programming is a relatively new type of programming language which dramatically reduces software development time. This is possible through the use of diagrammatic symbols rather than syntax-intensive text-based command sets. A paradigm shift is required when learning this new form of programming, but the learning curve is considerably less steep than those for other text based languages such as C, Basic, and FORTRAN. Many engineers prefer to think in diagrams and visual images rather than words so this new language seems to be a natural choice.

In the beginning phase of any software development, block diagrams and flow charts are used to represent the structure and design of a program, and a well designed graphical language permits the program to simply become an expanded flow chart. The key advantage of graphical programming is that commands and operations are represented by icons which are easier to remember than a set of text commands.



## Introduction to LabVIEW

National Instruments developed one of the first graphical programming languages called LabVIEW. LabVIEW was originally developed as a tool to support instrumentation in an effort to make interfacing with hardware a simpler task, but now after a number of revisions, LabVIEW has evolved into a robust language comparable to the graphical equivalent of C. LabVIEW has replaced the complex driver sets required for instrumentation with icons that could be wired together to direct data flow.

The LabVIEW User Manual states, “LabVIEW programs are called virtual instruments because their appearance and operation imitate actual instruments,” but in most cases the finished product goes far beyond the instrument level. Virtual Instruments, (from here on referred to as VIs) are comprised of a *front panel* that provides an interactive user interface with knobs, push buttons, graphs and other controls and indicators, and a *block diagram* that acts as the source code constructed by wiring icons together. VIs can be broken down into sub-programs called sub-VIs. Sub-VIs are represented as icons in the block diagram and can be passed data and return data through connectors created on the sub-VI’s icon.

## Benefits of Using LabVIEW

A survey performed last year showed that LabVIEW’s usage among Test Equipment Developers using PCs has surpassed BASIC and is rapidly gaining on C and C++. (Test & Measurement World, September 1995)

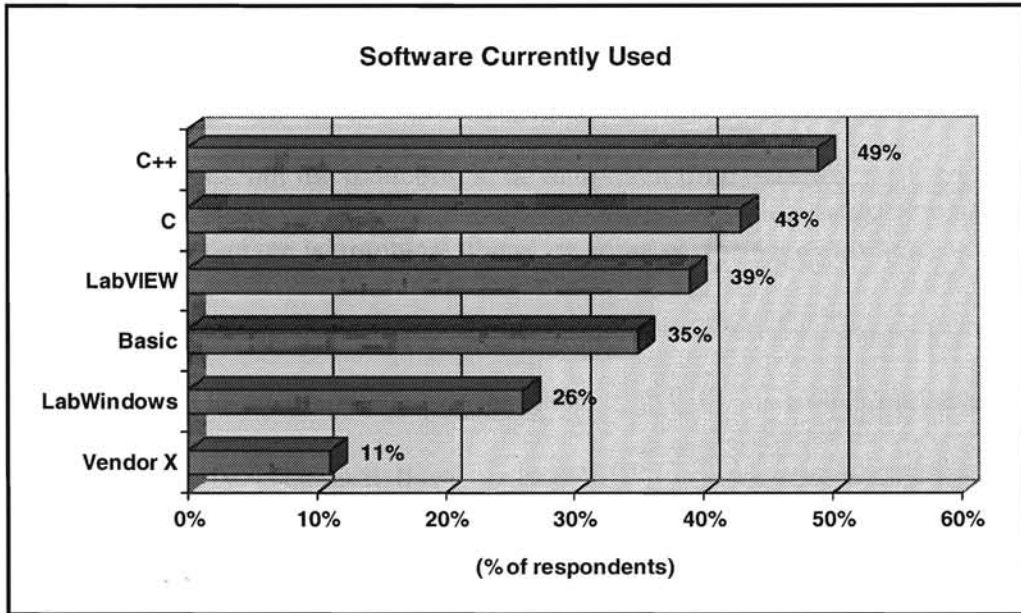


Figure 3: 1995 PC-Based Test Market Insight Study

This trend can be attributed to the many benefits of using LabVIEW. Development times associated with LabVIEW are simply smaller. The graphical user interface is built in, and is simple to modify for a great number of displays and controls. LabVIEW is a real programming language and has few intrinsic limitations. LabVIEW provides convenient VI libraries for performing many high-level operations. The debugging tools provided with LabVIEW permit the user to watch the data flow at all levels. Complex problems can be prototyped in a very short amount of time, and the hideous syntax errors which many programmers find so frustrating have been eliminated. The learning curve for becoming a proficient LabVIEW programmer is not steep, because of the many examples provided and the tutorial. Memorizing syntax and a list of commands is not necessary because everything is visual and is provided at the fingertips of the programmer. Overall, the most commonly heard reason for choosing LabVIEW is that it is fun to use.

As with any programming language, LabVIEW does have its drawbacks. The software package is priced at about three times higher than most full blown C development packages, but the reduction in development time makes it well worth it. Due to the fact that the language is graphical, there are some performance limitations in that the processing speeds are a little slower than programs written in C. These limitations can mostly be attributed to the graphical user interface. LabVIEW is a one vendor product so thus far there is no ANSI standard, however LabVIEW has been accepted by the Department of Defense (DoD) for many automatic test equipment designs. This is due in part to the paper “Diagrammatic-Graphical Programming Languages and DoD STD-2167A” written by two engineers at Frontier Engineering, Steve Bragg and Carl Driskill. The paper establishes processes for developing programs with Graphical Languages and making those developments meet standards written when text-based languages were the only ones in existence. DoD STD-2167A was the all encompassing software development standard for military products and has sub-sequently been replaced with a similar standard.

## CHAPTER III

### DESIGN AND DEVELOPMENT

The software program for this project was developed using an iterative prototyping technique where individual quantities were measured and tested for accuracy. After measuring each quantity successfully, the prototype was incorporated into the overall system. The following paragraphs discuss the key areas of interest in designing and developing this software application.

This program was designed in a modular fashion such that individual parts could be tested as they were developed. All setup parameters and constants are stored in a global variable VI. Two other global variables VIs are used to update the main display and a troubleshooting display. The main user interface is menu-based and permits the operator to modify setup parameters and necessary file locations. The operator may also begin a calibration and a test from this menu. After the system is configured, all acquisition and processing takes place within a calibration loop and then a main test loop.

#### Operation in Simulated Mode

Since testing the code often involves troubleshooting, it is a good practice to leave print statements or indicators in the code and control them with a global Boolean variable or compile flag. Setting this variable true enables the troubleshooting statements or displays. In addition to simple displays and indicators, support of a simulated mode may expedite the troubleshooting process. A simulation mode permits a large portion of the

code proofing and troubleshooting to be performed on a development station rather than requiring the entire system by mimicking hardware or apparatus dependent interfaces with a simulated data file. Values in the simulated data file may be changed to meet the various test cases. Turning on the simulated mode simply requires setting a global variable or compile flag true.

The apparatus for this project resides on a government facility with limited access therefore much of the development was performed away from the hardware. In order to eliminate the hardware dependency from the development stage of this project, a simulated mode was developed. The first step was to create a VI that simply collected raw data for all the channels at the desired rate and saved the data to a spreadsheet file. Subsystems for the program were designed by using the associated channels of data, or columns of data. Once a concept for the data processing had been proved with the data file, the process was converted to operate in a real-time mode.

When attempting to develop a simulated mode a VI was created to operate in a manner that duplicates this buffered acquisition, but the data is acquired from a file rather than a data acquisition board. The VI designed to accomplish this uses the number of scans to read at a time to determine how many points in the file to read for each iteration of the main loop. To run the program in a simulated mode the user must only set a simulation flag to true and specify the calibration data and the test data files to be used for the simulation.

## Acquiring Data at Desired Rate

A majority of the signals needed to perform breath-by-breath analysis must be collected at a rate of at least 200 samples per second while the rest must be collected at least once per breath (approximately once every 3-5 seconds). The fastest observed rise times of flow and mass spectrometer signals were on the order of 20 milliseconds, this translates into a frequency of 50 Hz. The selected sample rate is four times this highest anticipated frequency, so the Sampling Theorem is satisfied. The AT-MIO-64F-5 is capable of sampling data at well above 200 Hz and buffering data to a circular buffer automatically through a DMA channel. This buffered data can be accessed while still collecting new samples. The blocks of data are transferred from the buffer by an Acquire Data Routine which loads the data into a second circular buffer for processing. The second circular buffer was created in LabVIEW and will be discussed later in this thesis.

The VI designed for collecting raw data was based on an example VI provided with LabVIEW shown in Figure 4. This VI uses AI Config.vi to set up a circular buffer in memory where the AT-MIO-64F-5 board can dump data and configures the data acquisition with the following parameters: device to scan (which data acquisition board if multiple boards exist), input high and low limits for each channel, channels to scan, and buffer size. AI Config.vi configures the data acquisition and assigns a task ID that other VIs use to access the associated resources.

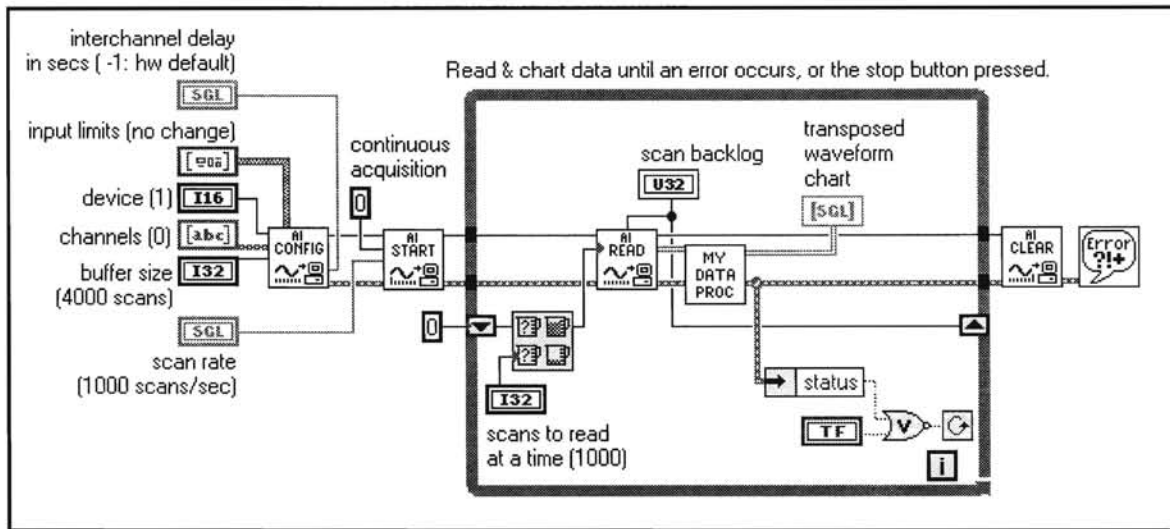


Figure 4: Example VI for buffered data acquisition

In this example, the acquisition is started with a call to AI Start.vi and passing this VI the task ID, the mode of operation (continuous acquisition in this case), and the desired scan rate. Once the acquisition is started a loop continually grabs blocks of data from the circular buffer by calling AI Read.vi and passing it the task ID and the number of scans to read during the current iteration. The number of samples read may differ from iteration to iteration depending on the backlog in the buffer. If processing or displaying the data takes longer than it does to access the desired number of samples then a backlog will be present in the buffer and on the next iteration the number of points to read will be the larger of the backlog and the number of points specified. After a block of data is accessed from the circular buffer the data is passed to a dummy VI (look for MY DATA PROC in figure 4) where any processing may be performed.

To create a VI to store raw data, a VI to store data to a spreadsheet file is substituted for the dummy VI. The loop continues to collect data and only stops iterating when the operator presses a stop button located on the front panel or if a critical error is encountered. After the loop stops executing, AI Clear.vi is called and passed the task ID to stop the data acquisition and release the associated resources.

Accessing the data in this fashion provides a pseudo real-time acquisition where data can be acquired continuously even if process times differ from iteration to iteration. It is especially important to keep this in mind when considering the overall system where breaths are processed after they have been detected. The time necessary for processing a breath is far greater than that required to perform the detection since the detection only considers one block of data from one channel. For this reason this VI was also used as the foundation for the complete breath-by-breath system. The VI used to write data to a file was later replaced with the logic to detect a breath and then process and display the results after detection.

### Error Handling

The example VI also provides a convenient demonstration of the error handling scheme implemented in the final system. An error cluster made up of *status*, a Boolean variable indicating whether an error was encountered; *code*, an integer variable containing the error code associated with the encountered error, and *source*, a string variable specifying the VI where the error was encountered and the call chain for this VI. The error cluster is passed from one VI to the next as they are called in succession. If a VI



encounters an error the VI updates the cluster to notify all successive VIs of this condition. If the error is critical, successive VIs will not execute and the error will eventually be passed to the error handler where the error is reported to the operator. The AI Clear.vi operates regardless of an error condition, so that the card can be reinitialized.

## Calibration

All of the instruments involved in this project need only be calibrated once every few months, except for the pneumotachographs. The pneumotachographs must be calibrated before every test run because the carrier demodulator's zero and gain tend to drift between tests. For this reason, the calibration was incorporated into the main test program.

Several papers address the complexity of calibrating pneumotachographs, but the approach used in this development proves to be relatively simple and quick. Calibrating the pneumotachograph requires the determination of four factors: offset voltage for inhale, offset voltage for exhale, inhale voltage to flow conversion factor, and exhale voltage to flow conversion factor. Determining the offset merely requires that the signal be read with zero flow, but the scale factors must be determined by measuring known volumes.

A calibrated syringe was connected to the mouthpiece of the breathing apparatus to provide simulated breaths of a known volume (see Figure 5). To find the scale factors, five simulated breaths are collected and then the scale factors for the inhale and exhale pneumotachographs are computed by dividing the known volume by the average of the

inhales and exhales. The collected volumes are displayed along with their standard deviation so the operator may observe the accuracy of the calibration. This process can be repeated multiple times and when the values appear acceptable a test run may be performed. The test run collects five more breaths and applies the scale factors. The volumes are displayed along with the percentage of error of the average of these volumes. When the operator is pleased with the calibration, the program returns to the main menu so that an actual test may be performed.

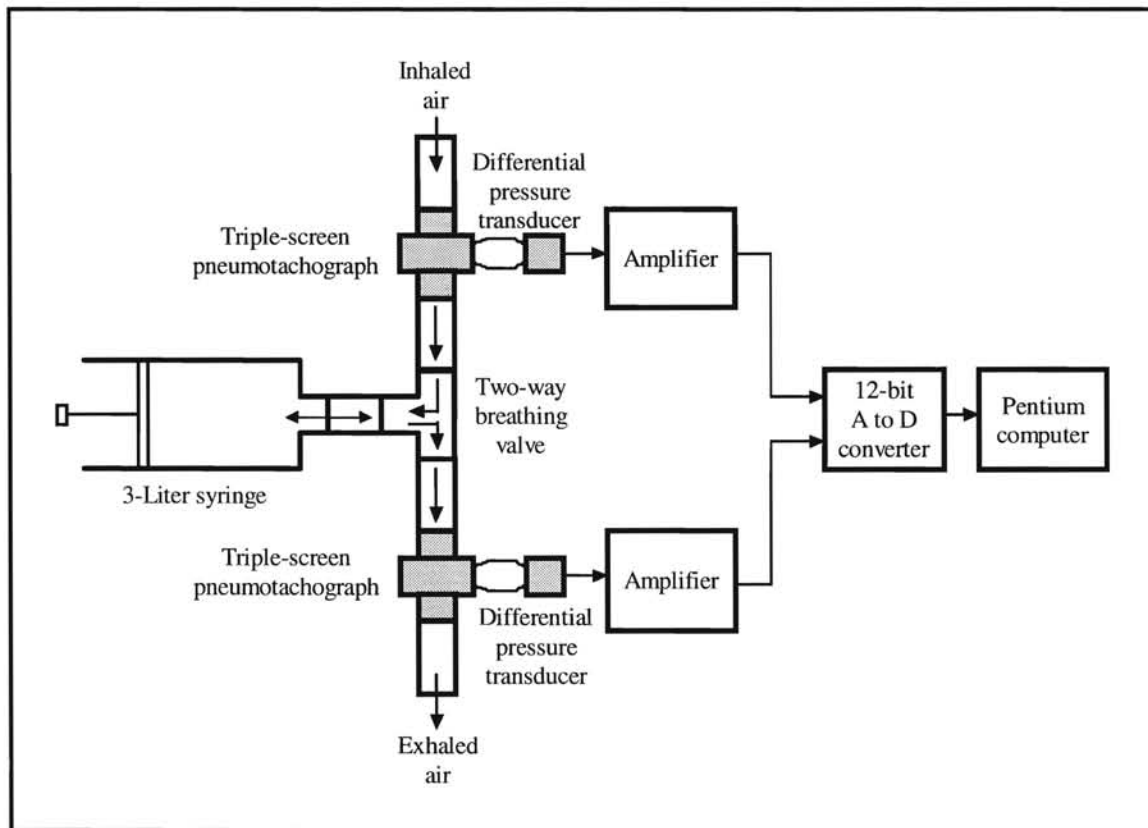


Figure 5: Pneumotachograph calibration setup

## Circular Buffering of Data

The data storage requirements for this program demand that a double buffering system be used such that each time a new respiratory signal is recognized all previous points in that breath are readily available. The circular buffer for this program needed to store and retrieve data quickly. The length of the buffer must be adjustable so that an optimal length could be determined. The buffer must be long enough to hold at least two breaths, and since the length of a breath is variable, an educated guess must be made. Four sub-VIs work in conjunction to implement the circular buffer. One VI controls access to the two dimensional circular buffer, two VIs convert indexes back and forth from linear to circular, and one VI converts channel assignments into indexes. Details for these four VIs follow.

### Circular Buffer w/ 2D Data Write.vi

This VI contains creates a circular buffer and controls access to it through three main operations: *Initialization*, *Write 2D Data*, and *Read Single Channel Subset*.

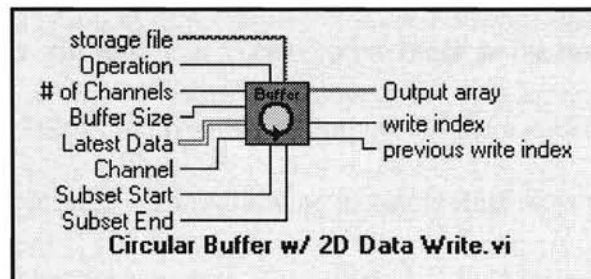


Figure 6: Circular buffer VI input/output definitions

## Initialization

Using input parameters for the buffer dimensions, *Initialization* allocates memory for the buffer and clears it. The dimensions of the buffer correspond to the number of A/D channels (number of columns) and the total number of data points to store (number of rows). The following example illustrates how this VI is called to initialize a circular buffer for 32 channels and 10,000 samples per channel.

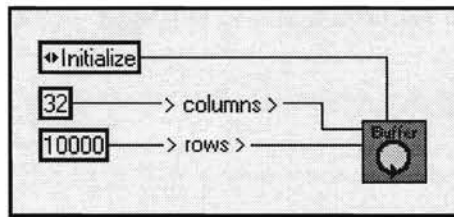


Figure 7: Example of circular buffer initialization

## Write 2D Data

*Write 2D Data* copies the two-dimensional input array of data into the next available block in the buffer. When the end of the buffer is reached, this operation starts writing at the beginning again overwriting old data. After an initialization, the write index is set to zero. After each write operation, the write index is updated and stored such that the next write operation will follow in a consecutive block in the buffer. The write index and the previous write index are provided as outputs so that a subsequent read operation can access the most recent data. The following example illustrates the circular buffer write operation.

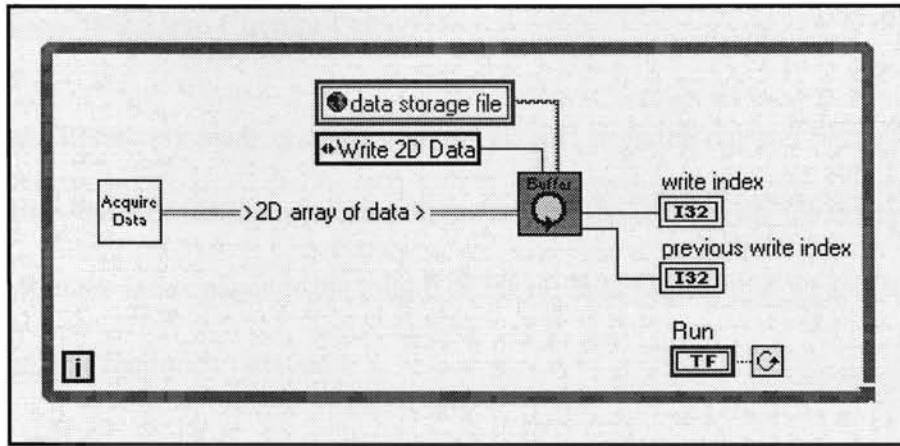


Figure 8: Example of circular buffer write

### Read Single Channel Subset

Using input parameters for channel column index and start and end row indexes, *Read Single Channel Subset* copies a subset of one of the columns within the buffer to a one dimensional output array.

### Convert Circular Index into Linear One.vi

While searching for the start and end points of a signal, subsets of a column are searched. To the search routine the data appears linear, so this VI must insure that the indexes account for the event when a data block is split with a portion at the end of the buffer and a portion at the beginning. This VI may also be used to find the linear difference between two circular indexes.

### Convert Linear Index into Circular One.vi

When iteratively reading single channel subsets from the circular buffer for a search routine, the indexes are incremented linearly by the calling VI. To cover the occasion when the index exceeds the length of the circular buffer, this VI acts like a car odometer rolling the index past zero.

### Find Signal Index and Channel.vi

This VI permits the programmer to assign names to signals and use these names to index the circular buffer. A configuration table contains a mapping of signal names to channel numbers and buffer indexes. When passed a signal name, this VI retrieves the channel number and buffer index for that signal. While signal names must remain permanent in the code, this permits a signal's channel and buffer index assignments to be modified by simply updating the table, without the need to modify the code.

### Recognition of Flow Signal

All of the breath calculations are triggered by the determination of an inhale, and therefore this search must be performed on each block as it arrives. During each iteration of the main loop, the most recently acquired block of inhale data is acquired from the circular buffer and passed to Find Inhale.vi which searches for the start and end points of an inhale. It is possible that the start and end points for the inhale signal will not be contained in the same block of data, and for this reason, the search status must be stored when the end of a block is reached and an entire inhale has not been found. These

intermediate index values are passed on to the main program so the search may resume on the next block of data. If the search routine reaches the end of the block before finding an inhale, then it sets an out of data flag to indicate this condition. A successful inhale search is indicated when this routine completes execution and the out of data flag is false.

#### Find Inhale.vi

The inhale search uses the following state transitions: *Find Start*, *Find End*, and *Start and End Found*. A description of each of these states follows.

#### Find Start

The start point is determined by using a double threshold detection method. The first threshold is set well above the noise level and the second is set closer to zero. The first threshold detection searches the array from the beginning until an element is found that exceeds this first threshold. After this comparison is satisfied, the second threshold detection backtracks in the array until an element is found that falls below the second threshold. The index found by the second threshold detection is stored as the inhale start index, and the search moves to the next state, *search for end*.

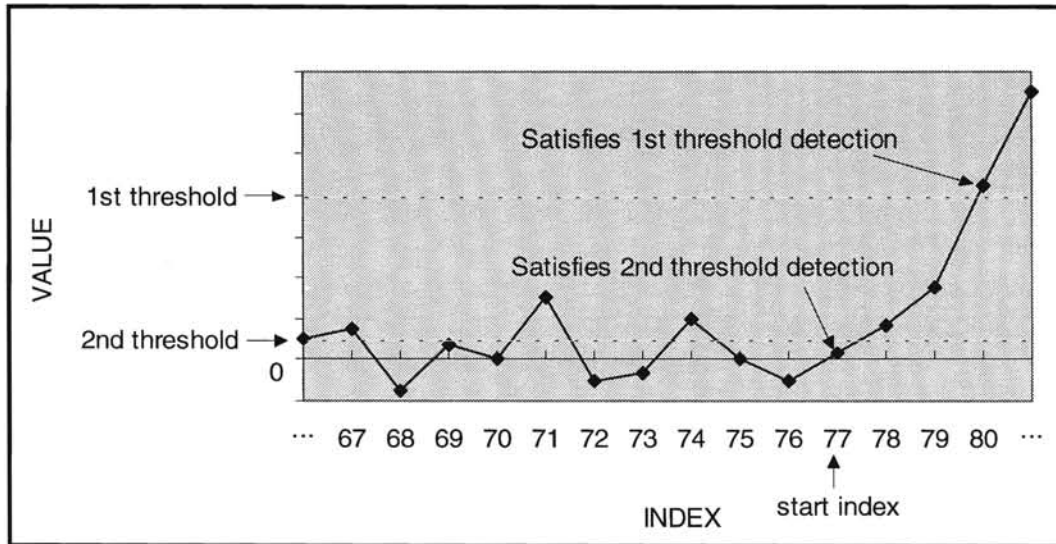


Figure 9: Double threshold detection

Find End

The search for the inhale's endpoint merely requires a single threshold detection where the threshold is set close to the zero. When an element falls below the end threshold, the index is stored as the end, and the search moves to the next state, *start and end found*.

Start and End Found

This state of the inhale search performs a validity check on the detected inhale by integrating the signal and comparing it to a threshold volume. If the signal volume is above the minimum, the search routine stops execution and returns the start and end indexes, otherwise the search continues on the current block of data. Regardless of the validity of the inhale signal the state is set back to *search for start*.



## Find Exhale.vi

The search for an exhale behaves similarly to the inhale search, but does not need to operate on a block by block basis. The start point for the previous inhale and the end point for the next inhale are used to index the exhale block in the buffer so the exhale search can be performed on one contiguous block of data.

## Adjusting for Drifting Baseline

Drift in the pneumotachographs during tests occasionally caused the inhale and exhale signals to rise above the set thresholds thus diminishing the programs ability to detect the breathing flow signals, and for this reason the baseline needed to be readjusted for each channel during operation. This was accomplished by viewing the signals during their zero flow states, namely looking at the inhale signal during an exhale and looking at the exhale signal during an inhale.

A simple average of the zero flow signals did not suffice since flutter from the valve caused positive deflections in the signals which in turn placed the average well above the zero flow level. With the exception of these occasional deflections a majority of the samples from the inhale and exhale zero flow signals remained at the same level. In order to find this level, a histogram of the signal was taken to determine where the majority of the points fell. A weighted average was performed on the two points with the highest concentration of samples, and this value was then added to the offset that is subtracted from the flow signal when it is collected. In addition to subtracting the new offsets from the newly acquired flow signals, the offset was subtracted from the data being processed

for the current breath. This “on the fly” offset adjustment proved to work very well in eliminating the drift caused by the pneumotachographs.

### Recognition of Inhale/Exhale Gas Fraction Signals

Since concentration levels reached by the  $O_2$  and  $CO_2$  signals vary from breath to breath a simple threshold detector cannot be used detect the start and end points within an inhale or exhale. The slopes of the  $O_2$  and  $CO_2$  signals show dramatic changes near the start and end points corresponding to the inhale and exhale stages. Instead of searching for a signal threshold, the transitions are found by first taking the derivative and then applying threshold detection to this derivative.

### Aligning Signals

The length of capillary tube carrying samples from the breathing mask to the mass spectrometer introduces a lag time between the gas fraction signals and the breathing flow signals. For accurate calculations of gas volumes these signals must be aligned.

In previous attempts at breath-by-breath measurements the lag time was corrected by assuming a constant delay. This assumption does not hold in operation at altitude since different lengths of tubes are used at different altitudes. In addition, the pressure at different altitudes affects the flow of the air sample to the mass spectrometer. Once a complete breath has been collected the gas composition signals must be aligned with all other signals such that point-by-point operations can be performed.

In breath-by-breath analysis literature, a breath begins with an inhale and ends after an exhale, and for this reason, the inhale signal was chosen for determining when to begin the gas volume calculations. Due to the lag time introduced by the mass spectrometer, two inhales must be obtained before the gas volumes may be calculated. This assumes that the end of lagging exhale gas concentration data will occur during the next inhale. In practice this has proven to be a sound assumption. Lag times range from half a second to one second, and the shortest breath cycle encountered to date is 1.5 seconds.

Each iteration of the main loop of the program acquires a block of data and stores it in a circular buffer then searches this block for the inhale signal. The very first breath is analyzed after two full inhales are found. The first inhale is used in the current calculation while the second one is processed after the next inhale is found. This procedure is iterated throughout a test cycle. When the next inhale has been found the exhale signal and gas signals are found by searching from the start index of the previous inhale to the end index of the current inhale.

All signals from the mass spectrometer experience the same lag time, but the  $O_2$  and  $CO_2$  are the only signals that change dramatically during a breath and thus provide transitions for determining their placement in the breath cycle. The  $O_2$  and  $CO_2$  signals remain at the constant ambient levels during an inhale, but since these gas signals reflect a drop in  $O_2$  and rise in  $CO_2$  during an exhale, the lag time computed for the exhale signals is applied to the inhale signals for that same breath. Although either signal will suffice, the  $CO_2$  signal was selected to determine the exhale gas signal end point.

The last portion of air inhaled occupies the dead space in a person's mouth, throat, and lungs and does not experience a gas exchange, therefore, the first portion of air exhaled has the same gas concentrations as the ambient air. For this reason, the end of the CO<sub>2</sub> exhale signal is used as the point of alignment. To find the CO<sub>2</sub> start index, the previously described VI, Convert Circular Index into Linear One.vi, calculates the length of the exhale signal from its start and end indexes, and then this length is subtracted from the CO<sub>2</sub> end index. The VI, Convert Linear Index into Circular One.vi, converts the result into a valid circular buffer index. Once found, the start and end indices for the CO<sub>2</sub> exhale signal are used to retrieve all of the mass spectrometer signals from the circular buffer for the current exhale.

The difference between the exhale flow and gas concentration signal end indexes translates into the lag time for the current breath, and is used to perform the alignment of the inhale signals. The inhale gas concentration signal indices are found by simply adding the lag time to the inhale flow signal indices. With the indices for the inhale and exhale gas concentration signals, these signals can be retrieved from the circular buffer and stored in temporary arrays for point by point calculations.

### Processing Breath Data

Once the inhale and exhale signals have been placed in the temporary arrays a series of equations are applied to the arrays to determine the gas volumes required for the analysis. Each of the signals must first be converted from a voltage to a meaningful quantity. With the exception of pressure and altitude, all signals were assumed to have a

linear conversion. Table 1 provides a table of all the linear conversion factors, and Table 2 provides the curvilinear coefficients for pressure and altitude. After applying the conversion factors to the signals, the calculations for determining the volumes may be conducted.

RESEARCH CENTER FOR ENVIRONMENTAL HEALTH

Quantity	Scale factor	Units
O <sub>2</sub>	0.100	%
CO <sub>2</sub>	0.020	%
N <sub>2</sub>	0.100	%
A	0.002	%
H <sub>2</sub> O	0.010	%
Temperature	10.000	°C
Relative Humidity	0.200	%

Table 1: Linear Conversion Factors

Quantity	intercept	x coefficient	x <sup>2</sup> coefficient	Units
Pressure	-421.315	865.659	102.225	mm of Hg
Altitude	96990.812	-123050.700	35827.398	ft

Table 2: Curvilinear Coefficients

The two most accepted standardized methods of reporting breathing data are STPD and BTPS. In order to represent the breath-by-breath data in these two standardized manners, correction factors must be applied to the flow signal to obtain each standard. The correction factors are represented by the following equations:

$$STPD = \left( \frac{T_0}{T_0 + T_{air}} \right) \cdot \left( \frac{P_{bar} - H_{rel} \cdot P_{swv}}{P_0} \right) \quad (1)$$

$$BTPS = \left( \frac{T_{body}}{T_0 + T_{air}} \right) \cdot \left( \frac{P_{bar} - H_{rel} \cdot P_{swv}}{P_0} \right) \quad (2)$$

where  $T_0 = 273 \text{ K}$  (standard temperature)

$T_{body} = 310 \text{ K}$  (body temperature)

$T_{air}$  = measured air temperature

$P_0 = 760 \text{ mmHg}$  (standard pressure)

$P_{bar}$  = measured barometric pressure

$P_{swv}$  = saturated water vapor pressure at measured temperature

$H_{rel}$  = relative humidity

Air temperature, relative humidity, and barometric pressure are sampled at the same rate as the flow and gas concentration signals so correction factors may be applied to every data point. The program must apply the correction factors to every sample because air temperature and relative humidity fluctuate during the course of a breath and because barometric pressure varies with changes in altitude.

The inhale and exhale signal volumes must be determined separately and are then subtracted from one another to determine the volume of gas consumed or produced. Each

point in the gas concentration array is multiplied by its corresponding point in the array of flow values. Breath-by-breath analysis is concerned with the consumption of O<sub>2</sub> and the production of CO<sub>2</sub>, N<sub>2</sub>, A, and H<sub>2</sub>O. The equations for determining these volumes follow. Note that these equations include the correction factors for STPD. BTPS volumes use the same equations but with a different correction factor.

O<sub>2</sub> volume equations:

$$V_{Gas\ consumed} = V_{Gas\ inhaled} - V_{Gas\ exhaled} \quad (3)$$

$$V_{Gas\ inhaled} = \int_{t_{start\ of\ inhale}}^{t_{end\ of\ inhale}} f_{STPD}(t) \cdot f_{Gas}(t) \cdot f_{inhale}(t) \cdot dt \quad (4)$$

$$V_{Gas\ exhaled} = \int_{t_{start\ of\ exhale}}^{t_{end\ of\ exhale}} f_{STPD}(t) \cdot f_{Gas}(t) \cdot f_{exhale}(t) \cdot dt \quad (5)$$

CO<sub>2</sub>, N<sub>2</sub>, A, and H<sub>2</sub>O volume equations:

$$V_{Gas\ produced} = V_{Gas\ exhaled} - V_{Gas\ inhaled} \quad (6)$$

A series of VIs accomplish these operations. The corrections factors are computed first and then multiplied by the flow signals. These adjusted flow signals are integrated to determine the total inhale and exhale volumes and are then fed to separate VIs for each gas volume to be calculated. In each of these VIs, the gas concentration arrays are multiplied by the flow signal, and the integral of the array is computed. An analysis VI, supplied with LabVIEW performs all of the integrations.



## Displaying Data

One of the unique aspects of this program compared to previous endeavors is that the results from each breath are computed and displayed after each breath rather than being collected and stored for later analysis. The start and end points for a breath must be determined as the data is collected and once a complete breath is acquired the appropriate calculations must immediately be performed to convert the data into usable quantities for display. Two methods of display were developed for this system, one for troubleshooting during the software development and one for the results display during test operation.

### Troubleshooting Display

The first display method permits the operator/developer to view all of the data associated with one breath as it is acquired. The data on this display was used during the integration of the individual sections of the program to determine if various processes were working properly.

Two of the indicators of the troubleshooting display are an inhale signals graph and an exhale signals graph that display all of the aligned data after application of conversion factors from the pneumotachographs, mass spectrometer, pressure transducer, relative humidity meter, and thermometer. These graphs permitted the developer to view the shape of these signals to determine if the signals were being detected and aligned properly. The flow signals for inhale and exhale should be hump shaped signals that start near zero and then return to zero. The O<sub>2</sub> signal should rise exponentially during the inhale and decrease exponentially for the exhale. The inverse is true for CO<sub>2</sub>. The N<sub>2</sub> and A signals

should remain relatively constant for both inhale and exhale, and the H<sub>2</sub>O signal should be slightly greater for the exhale. These displays were extremely useful in determining the flow and gas fraction thresholds for the detection processes.

A separate graph for each type of gas was included to display the inhale and exhale volumes, the volume of gas inhaled, the volume of gas exhaled, and the consumption or production of the gas. This display permitted the developer to monitor the correction factor and gas volume calculations. In addition, running totals of the consumption or production of volumes for each gas were provided to determine if the overall averages agreed with previous tests of this nature.

This display was accomplished through the use of a global variable containing all of the previously described indicators. This troubleshooting display is shown by simply setting a variable, *show troubleshooting display*, true. A button located on this display labeled "ABORT" permits the user to terminate test operation and returns the operator to the main menu.

### Main Display

The main test results screen displayed all of the per breath data along with graphs of the volumes plotted in time for trend analysis. A series of numeric indicators display the following data for each breath: inhale volume (VI), exhale volume (VE), breath frequency (freq = 1/(time for breath)), volume of O<sub>2</sub> consumed (VO<sub>2</sub>), volume of CO<sub>2</sub> produced (VCO<sub>2</sub>), ratio of CO<sub>2</sub> to O<sub>2</sub> (R), volume of N<sub>2</sub> produced (N<sub>2</sub>), volume of A produced (A), average of pressure within the chamber (Patm), average relative humidity

of air ( $RH_{atm}$ ), average temperature of air ( $T$ ), simulated altitude ( $Alt$ ), and time stamp relative to start of test ( $Time$ ). There are three separate graphs that display volumes of inhale and exhale versus time, volumes of  $O_2$  consumed and  $CO_2$  produced versus time, and volumes of  $N_2$  and  $A$  produced versus time.

Breath-by-breath analysis is performed in timed stages so indicators were placed on this main display to notify the operator what the current stage is and how much time is left until the next stage. The valid stages and stage times are contained in a global variable and may be set at the beginning of a test.

One of the methods of validating results obtained from this system is to collect the exhaled air in a bag for a set period of time and compare the volume and gas compositions with those reported by this system. To facilitate this verification a second series of numeric indicators displays running totals for  $V_I$ ,  $V_E$ ,  $VO_2$ ,  $VCO_2$ ,  $N_2$ , and  $A$ . The running totals accrue for a set period of time and then start over. The length of time between totals defaults to one minute but this time is variable and is set at the beginning of a test. To help synchronizing this procedure a button on this display labeled "Start Collection" permits the user to control when the test actually begins. Data is displayed but not stored for breaths occurring before the "Start Collection" is pressed, and when the button is pressed the displays are cleared and all subsequent breaths are displayed and stored.

## CHAPTER IV

### CONCLUSIONS

The working breath-by-breath analysis system collects, processes, and displays each breath's data in psuedo real time. Subject information can be entered for unique identification of data files. The setup parameters are easily modified for each test. The calibration is easy to operate and proved very accurate for syringe tests. As anticipated use of the graphical programming language made the development very straightforward and facilitated necessary modifications. Overall, the system meets the requirements for a prototype breath-by-breath analysis system.

There are several distinct differences between this system and previous attempts at breath-by-breath analysis. Both inhale and exhale volumes are computed, whereas previous systems only measured the exhale and used a normalization to estimate the inhale volume. Data is displayed and processed as it arrives so test results can be viewed during the test rather than post test. The modular design of this system is modifiable and permits other signals to be easily integrated for display.

Although the system is functional, results are not ready for verification due to errors introduced by differences in air quality between inhale and exhaled air by human subjects. This became evident when attempting to verify the results.

## Verification of Program Results

Three methods were used to verify the results obtained by this breath-by-breath analysis system. The first is very similar to the calibration process used for the pneumotachographs, the second involves collecting the exhaled air in bags over set intervals of time, and the third involved viewing totals of inhaled and exhaled air over a long period of time.

The first method is used to verify the flow and total air volume calculations. By running a test with calibrated syringes of different sizes, the volumes measured by this system could be compared to the actual syringe volume. This verification proved that the pneumotrachograph calibration and volume calculations were accurate within 2% for all tests. Bear in mind that the air being inhaled and exhaled has the exact same gas and moisture composition.

The second verification tested the volume and gas fraction calculations performed on the exhaled air of a human subject. The measured composition of the exhaled air and the composition of the air in the bag agreed, but the total volume of air measured by the system was nearly 5% over that measured in the bag.

The third method checked if the volume calculations for the inhale and exhale of a human subject matched. Inhales and exhales do not typically match on a per breath basis, but over a long period of time the total amount of air inhaled and the total exhaled should be close to the same, since only a fraction of the air can be stored in the lungs or apparatus. This verification showed results that were up to 8% off from each other.

These verifications suggest that an additional compensation needs to be added when dealing with human subject air flow. This seems to point to differences in the air quality between the inhale and exhale. The biggest difference in the gas composition of inhales and exhales is the water content (21% in ambient air and 100% in exhaled air). This suggests that the pneumotachographs respond differently to dry and wet air. This project is on going and the system will continue to be refined until the above verifications reveal acceptable results.

### Future Work

The following tasks still need to be completed:

- Test the pneumotachographs response to dry and wet air, so that a compensation factor may be applied to the pneumotachograph measurements.
- Control needs to be added so that the valve for bag collection can be controlled from the computer so an additional person is not needed in the chamber to perform this task.
- Processing needs to be added to display data collected from the following cardio-pulmonary behavior measurement devices : Novamatrix 800 PO<sub>2</sub>/PCO<sub>2</sub> monitor, Marquette ECG Cart (MAC-6), and Nellcor N200 Pulse Oximeter.

## BIBLIOGRAPHY

- Beaver, W. L., Lamarra, N., & Wasserman, K. (1981). Breath-by-breath measurement of true alveolar gas exchange. Journal of Applied Physiology, *51*(6), 1662-1675.
- Beaver, W. L., Wasserman, K., & Whipp, B. J. (1973). On-line computer analysis and breath-by-breath graphical display of exercise function tests. Journal of Applied Physiology, *34*(1), 128-132.
- Bragg, S. D., Driskill, C. G. (1994). Diagrammatic-Graphical Programming Languages and DoD STD-2167A. IEEE AUTOTESTCON, System Readiness Technology Conference, Atlanta, Georgia
- Hughson, R. L., Northey, D. R., Xing, H. C., Dietrich, B. H., & Cochrane, J. E. (1991). Alignment of ventilation and gas fraction for breath-by-breath respiratory gas exchange calculations in exercise. Computers and Biomedical Research, *24*, 118-128.
- Johnson, Gary W. (1994). LabVIEW Graphical Programming: Practical Applications in Instrumentation and Control. New York, New York: McGraw-Hill, Inc.
- National Instruments, Incorporated (1996). LabVIEW User Manual. Austin, Texas.
- Sullivan, W. J. (1984). Pneumotachographs: Theory and Clinical Application. Respiratory Care, *29*, 736-749.
- Test & Measurement World. 1995 PC-Based Test Market Insight Study. Test & Measurement World. September (1995).
- Winslow, R. M., & McKneally, S. S. (1986). Analysis of breath-by-breath exercise data from field studies. International Journal of Clinical Monitoring and Computing, *2*, 167-180.

APPENDIX A - CODE LISTING





### **MAIN MENU.vi**

This VI is the main menu for the system.  
The operator is afforded the following options:

- 1 Enter Subject Info
- 2 Operation Parameters
- 3 Select Storage Files
- 4 Simulation Options
- 5 Run Calibration
- 6 Run Test
- 7 View Results In Excel
- 8 Exit Program

1,2, and 3 must be performed before 5,  
5 before 6, and 6 before 7.  
8 may be performed at any time.

## **Low Altitude Hypoxia Testing Study**

### **MAIN MENU**

- Enter Subject Info
- Operation Parameters
- Select Storage Files
- Simulation Options
- Run Calibration
- Run Test
- View Results in Excel
- Exit Program

Initialize  
Main  
Display

### **Initialize main display variables.vi**

This VI performs initialization on several global variables.



Text Header

### Header File.vi

This VI permits the operator to enter the subject information to make headers for the data files.

Last Name

First Name

Age

Weight

ID #

Select  
Files

### Select Data Files.vi

This VI permits the operator to select the file names and paths used for storing and reading data.

breath-by-breath file  
C:\FAA\DATA\TEST.TXT Browse

timed totals file  
C:\FAA\DATA\TEST.TXT Browse

calibration storage file  
Browse

data storage file  
Browse

NOTE: These files are used to store raw data collected during the test and get very large. Therefore, these file controls should be left empty unless the current test will be used to create a simulated run.

scale factor storage file  
C:\FAA\DATA\SCALE.TXT Browse

NOTE: DO NOT CHANGE. This file's name and location should not change, unless the directory structure of the program has changed.

EXCEL location  
Browse

NOTE: DO NOT CHANGE. This file's name and location should not change, unless the directory structure of the program has changed.

Press Here  
When Finished

Setup  
Param

### Setup Parameters.vi

This VI permits the operator to modify the system setup parameters.

Calibration Syringe Size <input type="text" value="3.000"/>	Use Halidayne Transfer? <input type="button" value="YES"/>
1st Start FlowThreshold <input type="text" value="5.000"/> L/min	Test Type <input type="button" value="Human Subject"/>
2nd Start Flow Threshold <input type="text" value="2.500"/> L/min	Control: Is Subject or Syringe Data Being Collected
End Flow Threshold <input type="text" value="2.500"/> L/min	troubleshoot? <input type="button" value="OFF"/>
Volume Threshold <input type="text" value="0.100"/> L	one pneumotac? <input type="button" value="NO"/>
time between totals <input type="text" value="60"/> sec	<input type="button" value="Press Here to View Global Variable"/>
<input type="button" value="Press Here&lt;br/&gt;When Finished"/>	

Simul.  
Options

### Simulated Options.vi

This VI permits the operator to modify the simulation parameters.

**Simulation Options**

simulate?

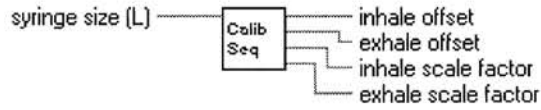
NO

time scale  
1.00

simulated calibration file  
C:\FAA\DATA\M0405\_GL\_cal\_1001a.txt Browse

simulated data file  
C:\FAA\DATA\M0405\_GL\_dat\_1001a.txt Browse

Press Here  
When Finished



### Calibration sequence.vi

This VI performs the pneumotachograph calibration.

inhale scale factor (L/V)			inhale scale factor mean			exhale scale factor (L/V)			exhale scale factor mean								
0.00	0.00	0.00	0.00			0.00	0.00	0.00	0.00								
inhale volumes (L)			inhale row means			inhale test volumes			exhale volumes (L)			exhale row means			exhale test volumes		
0.000	0.000	0.000	0.000			0.000			0.000	0.000	0.000	0.000			0.000		
0.000	0.000	0.000	0.000			0.000			0.000	0.000	0.000	0.000			0.000		
0.000	0.000	0.000	0.000			0.000			0.000	0.000	0.000	0.000			0.000		
0.000	0.000	0.000	0.000			0.000			0.000	0.000	0.000	0.000			0.000		
inhale std dev (L)			inhale row mean			inhale error			exhale std dev (L)			exhale row mean			exhale error		
0.000	0.000	0.000	0.0000			0.0000			0.000	0.000	0.000	0.0000			0.0000		

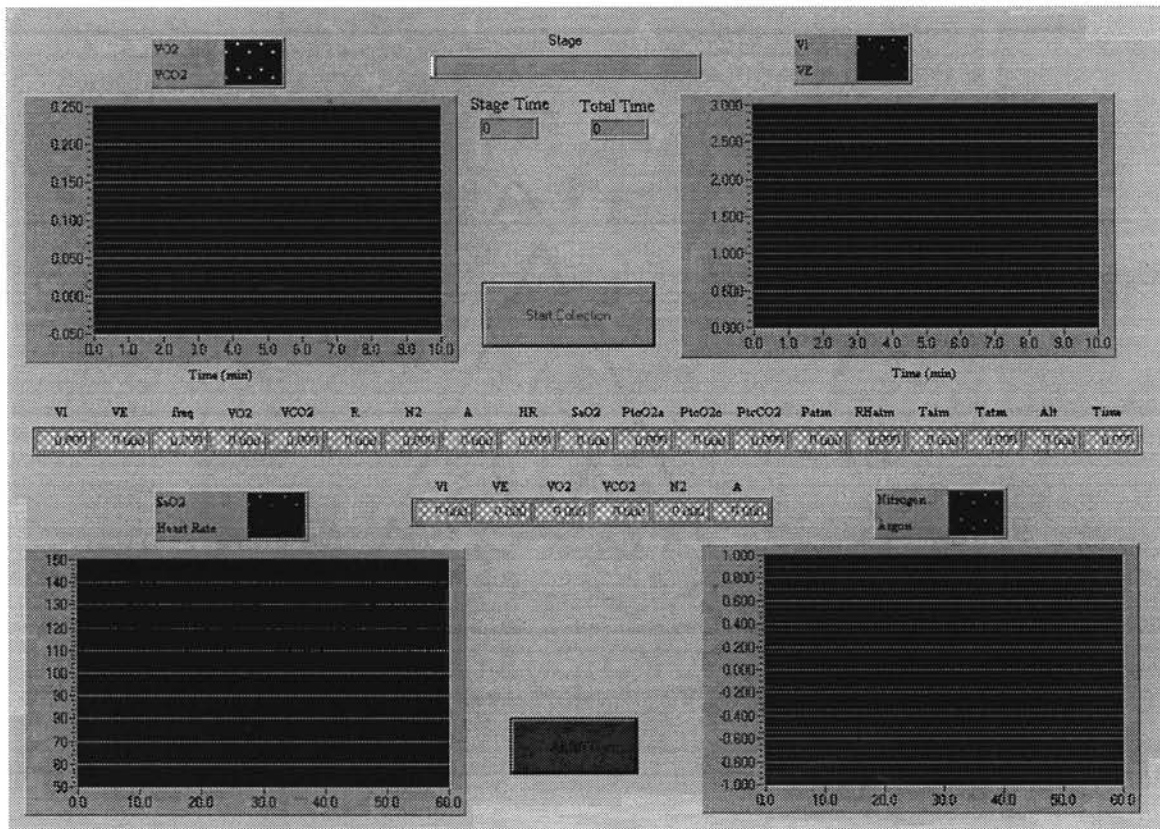
  

<b>Run Calibration</b>	<b>Test Calibration</b>	<b>Return To Main Menu</b>
------------------------	-------------------------	----------------------------

Main Display ..... abort out

### Main display.vi

This VI displays all breath analysis data during test operation.

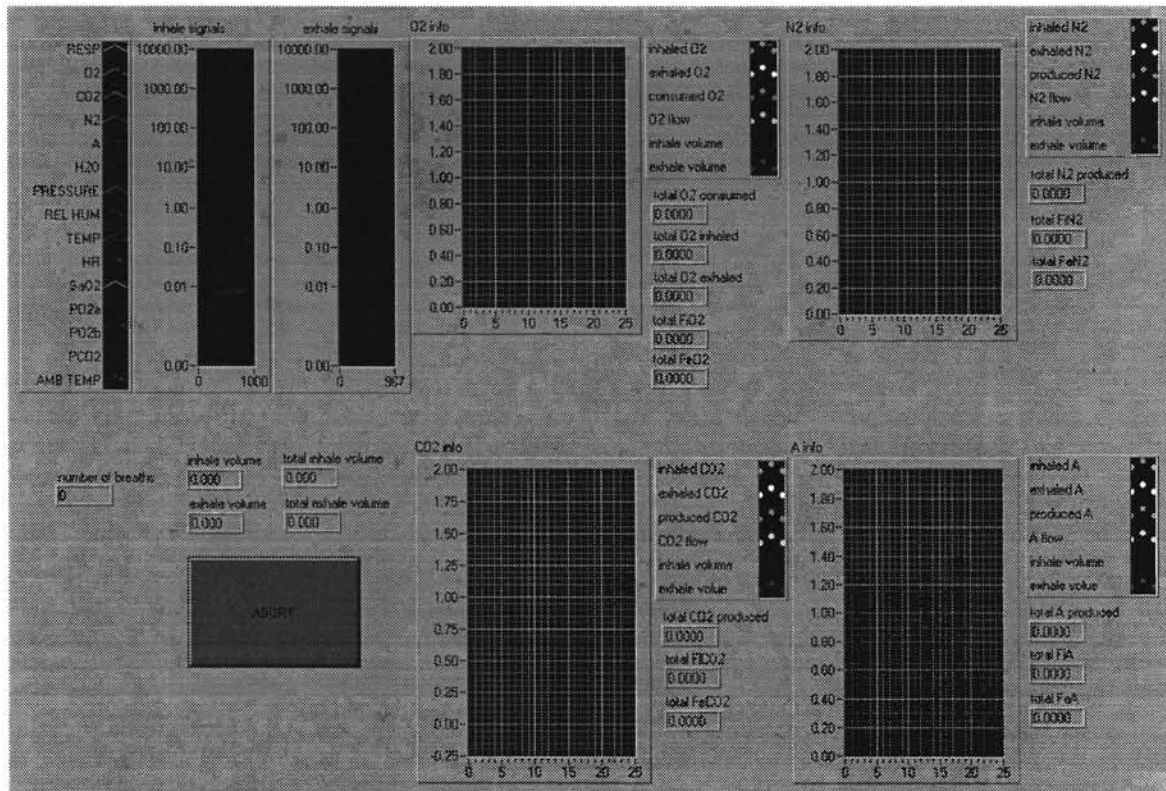






## Troubleshooting display.vi

This Global VI displays breath analysis data for troubleshooting purposes.

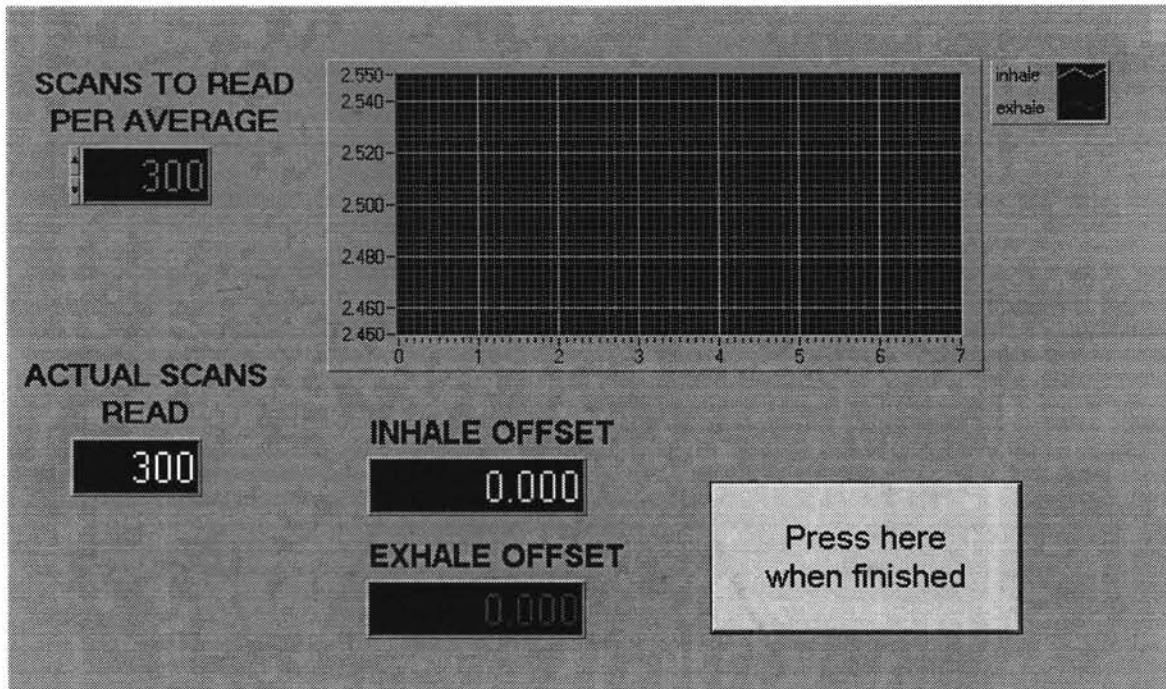


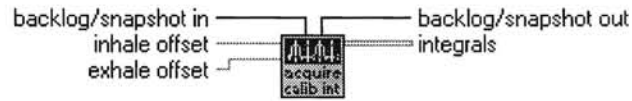


INHALE OFFSET  
EXHALE OFFSET

### Acquire inhale & exhale offsets.vi

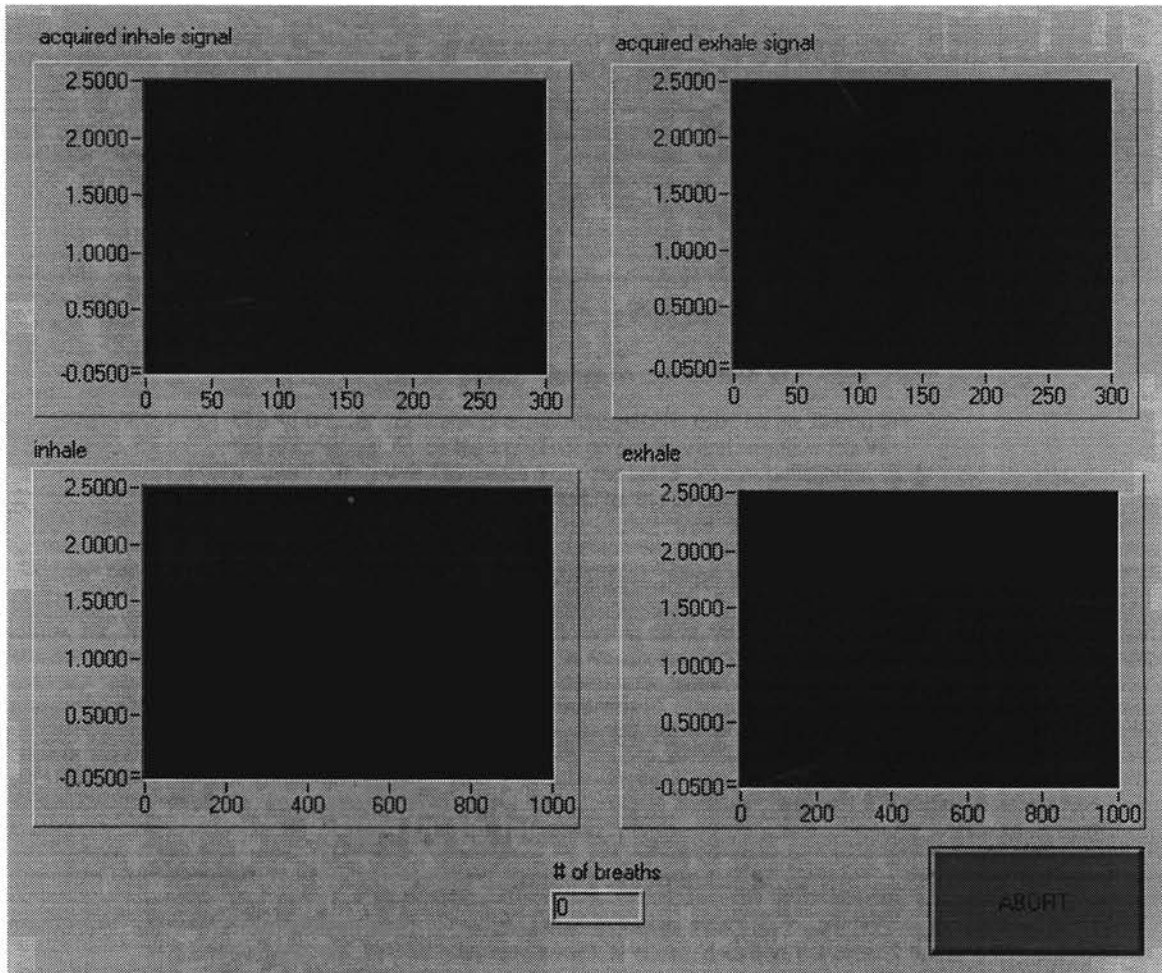
This VI acquires and averages the flow signals to obtain zero flow voltage offsets for the calibration procedure.

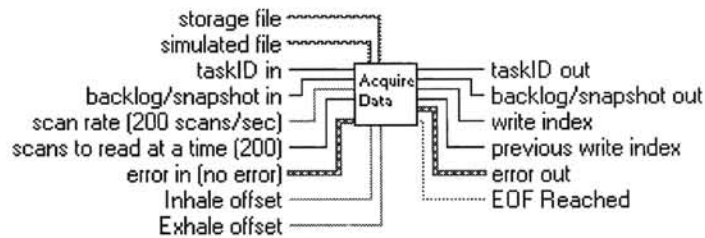




**Acquire inhale & exhale integrals for calibration.vi**

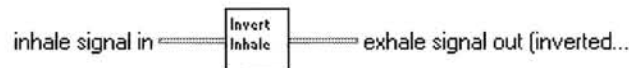
This VI collects data for a fixed number of calibrated syringe pumps.





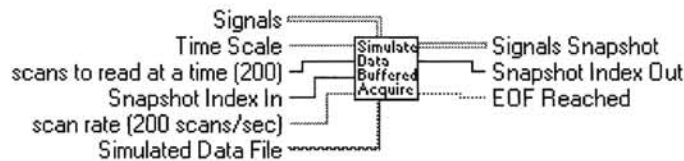
### Acquire Data.vi

This VI acquires data from the data acquisition card and loads it into a circular buffer. If in simulated mode the data is collected from a file instead of the card.



### Invert inhale signal to get exhale.vi

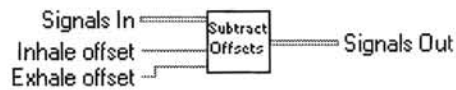
This VI is used if only one pneumotachograph is used for measuring the breathing flows. Since the signal will go positive and negative this VI inverts the flow signal to create a second set of positive deflections. Only the positive deflections are considered for the inhale and exhale detection.



### Simulate Buffered Data Acquisition.vi

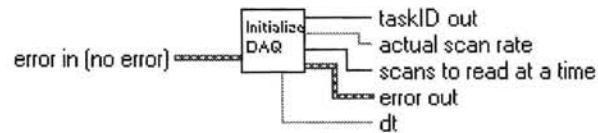
This VI retrieves snapshots of data from a simulated data file created during an actual test run. This VI is setup to mimick acquiring data from the circular buffer of a data acquisition card.

To make the operation even more realistic this VI waits the proper amount of time to simulate the real acquisition, but this can be sped up by making the time scale less than one (e.g. 1 sec = 0.5 simulated seconds).



### Subtract Offsets From Inhale & Exhale.vi

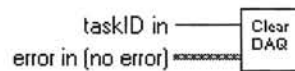
This VI subtracts the zero flow offsets from the inhale and exhale flow signals.



### Initialize DAQ for Cont Acq (Buffered).vi

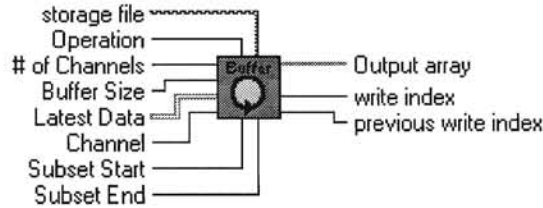
This VI configures the data acquisition card using parameters found in the global variable, starts the acquisition, and initializes the second circular buffer. Specially this configures each channel's voltage range and coupling and sets up the cards circular buffer.

If in simulated mode this VI just initializes the circular buffer.



### Clear DAQ & Report Errors.vi

This VI clears the data acquisition associated with the task ID specified by releasing all resources used for that session. After clearing this card, this VI reports all errors with a dialog box.



### Circular Buffer with 2D Data Write.vi

This VI controls all access to the secondary circular buffer through three available operations:

#### Initialize

Space in memory is cleared and reserved for the two dimensional array with dimensions of # of channels (# of columns) by buffer size (# of rows).

#### Write 2D data

This mode updates the circular buffer with the block of data being input. The write index is updated and stored for the next write operation. When the end of the buffer is reached the data is written at the beginning of the buffer and continues overwriting old data with subsequent write operations. If a storage file is specified this mode also performs a write to file where the data is appended to the specified storage file.

#### Read Single Channel Subset

This mode reads a subset (from subset start to subset end) of the specified column (Channel).



### Find Signal Index & Channel.vi

This VI uses a signal name (e.g. INHALE, EXHALE, O2, CO2, etc.) to retrieve a buffer column index and data acquisition card channel from the global variable setup tables. The column index is used to index the circular buffer for a read operation.



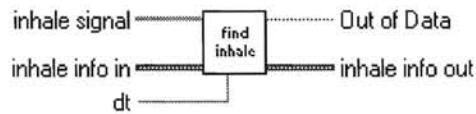
### Convert circular index into linear one.vi

Given two circular index values this VI computes the linear difference between them. This VI accommodates for the condition when the buffer has rolled over to the beginning where a start index is at the end of the buffer and the end index is towards the beginning.



### Convert linear index into circular one.vi

This VI converts a linear index into a circular one. This VI accommodates for the condition where a linear index is greater than the buffer size by wrapping it to the beginning of the buffer. Similarly a negative linear index wraps back from the end of the buffer.



### find inhale.vi

This VI searches for the inhale start and end points on blocks of acquired data, and when an inhale is found, it checks it for validity. The search is broken into three states so that it can remember its states from call to call. These states are:

#### search for start

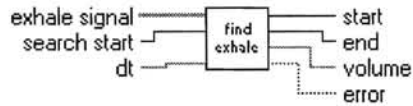
The current block of data is searched until a point is above a 1st threshold. When the 1st threshold is satisfied the search reverses to find a point below a 2nd threshold. If the 1st threshold is not satisfied before the end of the block of data the state remains the same and the out of data flag is set TRUE, otherwise it is advanced to search for end.

#### search for end

The block of data is searched until a point falls below the end threshold. If the end threshold is not satisfied before the end of the block of data the state remains the same and the out of data flag is set TRUE, otherwise it is advanced to start and end found.

#### start and stop found

Once start and end indexes have been found the integral of the detected inhale is calculated and compared to the volume threshold. If above the threshold then the state is set back to search for start and this VI stops execution and returns to the calling VI with the out of data flag set FALSE. If the volume threshold is not met the state is set to search for start and the search continues until the end of a block. If this VI returns with the out of data flag FALSE then an inhale has been found.



**find exhale.vi**

This VI searches for the exhale start and end points, and checks it for validity. The search is broken into three states:

**search for start**

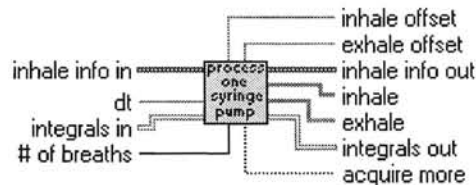
The data is searched until a point is above a the 1st threshold. When the 1st threshold is satisfied the search reverses to find a point below a 2nd threshold.

**search for end**

The data is searched until a point falls below the end threshold.

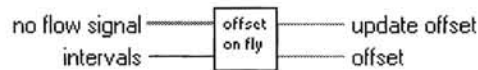
**start and stop found**

Once start and end indexes have been found the integral of the detected inhale is calculated and compared to the volume threshold. This VI stops execution if the volume is above the threshold, otherwise the state is set back to search for start and the search continues.



**Process one syringe pump for calibration.vi**

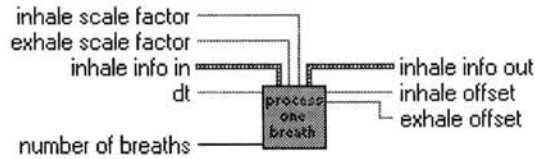
This VI is called after an inhale has been found in the calibration procedure. It calls the search for an exhale, updates the array of calibration integrals, and adjusts the integrals for drifting baseline.



**Calculate offset on the fly.vi**

This VI computes the on the fly baseline offset for a flow signal. It does this by performing a histogram on the zero flow signal and performing a weighted average on the two values with the most occurrences.

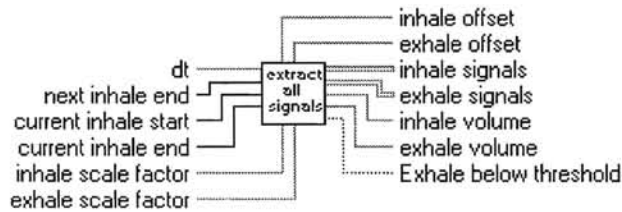




### Process One Breath.vi

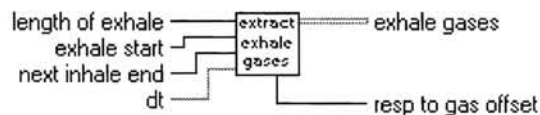
Once an inhale is found this VI coordinates finding and aligning the corresponding exhale, the inhale and exhale gas signals, and the atmospheric signals (temperature, pressure, and relative humidity).

The computations necessary for analysis are performed by calling other VIs from this VI. The main display global variables are also updated by this VI.



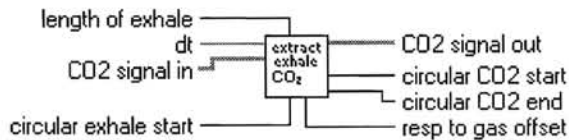
### Extract all signals.vi

This VI aligns all of the breath signals for an inhale and an exhale. It accommodates the baseline drift of the flow signals by calling the subtracting the "on the fly offset" from the inhale and exhale flow signals. This VI also applies the STPD correction factor to the flow signals. The aligned signals are stored in separate two-dimensional arrays for inhale signals and exhale signals.



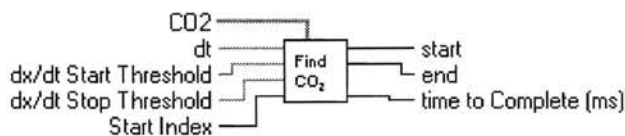
### Extract exhale gas signals.vi

This VI aligns all of the exhale signals by finding the CO<sub>2</sub> gas concentration signal and applying the same lag time to the other gas signals. When the start and end points are determined all of the gas signals are indexed and collected in a two dimensional array.



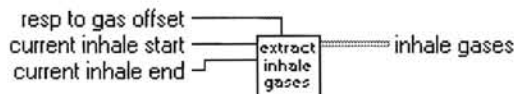
### Extract exhale CO2 signal.vi

This VI decimates the CO2 signal to smooth out the noise so that the start and end points may be found. Once found the start and end indexes are used to retrieve the CO2 signal for the current exhale.



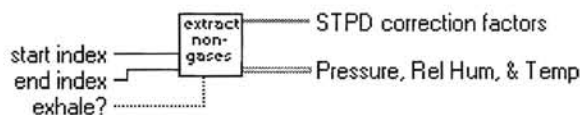
### Find CO2 Start-Stop dx-dt based.vi

This VI finds the start end points of the CO2 signal for the current exhale. This operation is timed for testing system speed.



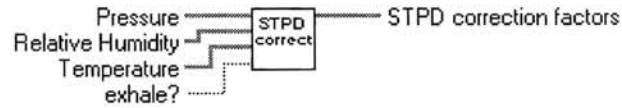
### Extract inhale gas signals.vi

This VI aligns all of the inhale signals by applying the lag time found for the CO2 signal in the exhale to the inhale gas signals. When the start and end points are determined all of the gas signals are indexed and collected in a two dimensional array.



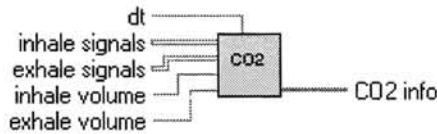
### Extract Pressure, Rel Hum, & Temp.vi

This VI aligns all of the non-gas signals and places them in two dimensional arrays for inhale signals and exhale signals. The atmospheric signals are also used to compute the STPD correction factors.



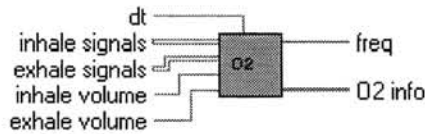
**Calculate STPD correction factors.vi**

This VI uses the pressure, temperature, and relative humidity signals to compute the STPD correction factors. If a human subject is being tested 100% humidity is assumed for an exhale.



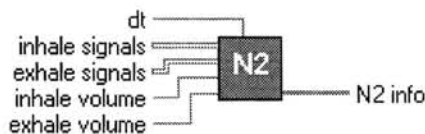
**Calculate CO2 info.vi**

This VI computes the inhaled CO2, exhaled CO2, and the CO2 produced.



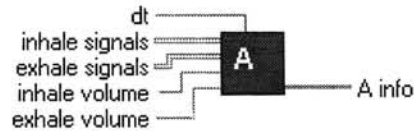
**Calculate O2 info.vi**

This VI computes the inhaled O2, exhaled O2, and the O2 consumed.



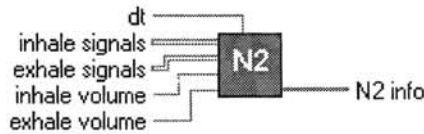
**Calculate N2 info.vi**

This VI computes the inhaled N2, exhaled N2, and the N2 produced.



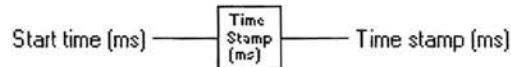
**Calculate A info.vi**

This VI computes the inhaled A, exhaled A, and the A produced.



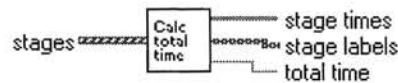
**Calculate N2 info.vi**

This VI computes the inhaled N2, exhaled N2, and the N2 produced.



**Millisecond Time Stamp.vi**

This VI can be used along with millisecond time to create a stop watch. It accomodates for the roll over of the millisecond timer at  $2^{32}$ . The start time can be retrieved from the 32 bit counter and then this VI will retrieve a stop time and compute the total time elapsed.



**Calculate total time.vi**

This VI accumulates all of the stage times and determines the total time for the test.



### Global.vi

This Global VI contains configuration parameters for the entire breath analysis system.

**AT-MID-64F**

device	coupling & input config (no change.0)	scan rate (200 scans/sec)
1	no change non-ref.	300.00
INITIAL	no change non-ref.	buffer size (2000 scans)
EXHALE	no change differential	3000
O2	no change differential	scans to read at a time (200)
CO2	no change differential	300
N2	no change differential	hardware settings
A	no change differential	input limits (no change)
H2O	no change differential	high limit
HRV	no change differential	10.00
FLOW	no change differential	low limit
PRESSURE	no change differential	0.00
REL HUM	no change differential	interchannel delay in secs [-1: hw default]
TEMP	no change non-ref.	-1.00E+0
HR	no change differential	
SaO2	no change differential	
PO2a	no change differential	
PO2b	no change differential	
PCO2	no change differential	
AMB TEMP	no change non-ref.	

calibration syringe size (L)	time between totals	stages
3.00	60.00	stage
user info	time scale	GL Baseline
	1.00	duration (sec)
		180.00

<b>Inhale/Exhale Thresholds (Liters)</b> 1st Start Threshold: 6.000000 2nd Start Threshold: 2.500000 End Threshold: 2.500000 Volume Threshold: 6.1000		<b>scale factors</b> O2: 0.100 CO2: 0.020 N2: 0.100 A: 0.002 H2O: 0.010 PRESSURE intercept: 1421.315454475619 PRESSURE x coeff: 865.656206110273 PRESSURE x^2 coeff: 802.225054912762 ALTITUDE intercept: 186390.6123702066 ALTITUDE x coeff: 1123060.7000635640 ALTITUDE x^2 coeff: 35027.3982432961 REL HUM: SaO2: 100.000 TEMP: PO2a: 100.000 AMB TEMP: PO2b: 100.000 HR: PCO2: 100.000		<b>simulated calibration file</b> %C:\FAANDATA\M0405_GL_cal_1001a.txt <b>simulated data file</b> %C:\FAANDATA\M0405_GL_dat_1001a.txt <b>EXCEL Location</b> %C:\MSOFFICE\EXCEL\EXCEL.EXE <b>max file totals</b> %C:\FAANDATA\TEST_MIN.TXT <b>breath-by-breath file</b> %C:\FAANDATA\TEST.TXT <b>calibration storage file</b> %C:\FAANDATA\SCALE.TXT <b>data storage file</b> %C:\FAANDATA\SCALE.TXT <b>scale factor storage file</b> %C:\FAANDATA\SCALE.TXT		<b>breathing valve volume</b> 108.20E-3 <b>test time (ms)</b> 4.500E+6 <b>maximum offset</b> 12000 <b>circular buffer size</b> 12000 <b>start breath #</b> 1 <b>start calib breath #</b> 2 <b># of breaths for calib</b> 5 <b>calibration threshold (% of spring volume)</b> 0.50	
<b>Inhale thresholds (Volts)</b> 1st Start Threshold: 0.010000 2nd Start Threshold: 0.005000 End Threshold: 0.005000 Volume Threshold: 0.0002		<b>temp status</b> Po (mmHg): 750.00 To (K): 273.00 declination factor: 10 temperature: 0 water vapor pressure: 4.58					
<b>Exhale thresholds (Volts)</b> 1st Start Threshold: 0.010000 2nd Start Threshold: 0.005000 End Threshold: 0.005000 Volume Threshold: 0.0002							
<b>O2 thresholds</b> da/dt Start Threshold: 25.00 da/dt Stop Threshold: 25.00							
<b>CO2 thresholds</b> da/dt Start Threshold: 100.00 da/dt Stop Threshold: 100.00							
		<b>simulate?</b> <input type="button" value="OFF"/> Use Nitrogen Assumption: <input type="button" value="ON"/> troubleshooting?: <input type="button" value="OFF"/> breathing?: <input type="button" value="ON"/> end pneumatics?: <input type="button" value="NO"/>					

Science  
 Oklahoma  
 Motors Division,  
 Technician, Frontier  
 to July, 1993;  
 and Computer Engineering  
 Software/Systems Engineer.  
 Present.  
 Oklahoma Society

VITA

Charles T. Patterson

Candidate for the Degree of

Master of Science

Thesis: DESIGN AND DEVELOPMENT OF A BREATH-BY-BREATH ANALYSIS SYSTEM FOR USE AT LOW ALTITUDES

Major Field: Electrical and Computer Engineering

Biographical:

Personal Data: Born in Fort Riley, Kansas, October 9, 1970, son of Marsha and Tom Patterson.

Education: Graduated from Kickapoo High School, Springfield, Missouri, in May, 1988; received Bachelor of Science Degree in Electrical and Computer Engineering from Oklahoma State University, Stillwater, Oklahoma in July, 1993; completed requirements for the Master of Science Degree with a major in Electrical and Computer Engineering at Oklahoma State University in December, 1996.

Professional Experience: Engineering Intern, General Electric Motors Division, Springfield, Missouri, Summer, 1991; Engineering Technician, Frontier Engineering, Stillwater, Oklahoma, January, 1993 to July, 1993; Laboratory Assistant, Department of Electrical and Computer Engineering, Oklahoma State University, Fall, 1993; Software/Systems Engineer, Frontier Engineering, August, 1993 to present.

Professional Memberships: Eta Kappa Nu, Oklahoma Society of Professional Engineers.