Development and implementation of a DSP based air detector system to prevent embolism during hemodialysis therapy

Nhat Nguyen
University of South Florida

Follow this and additional works at: http://scholarcommons.usf.edu/etd
Part of the American Studies Commons

Scholar Commons Citation
Nguyen, Nhat, "Development and implementation of a DSP based air detector system to prevent embolism during hemodialysis therapy" (2005). Graduate Theses and Dissertations.
http://scholarcommons.usf.edu/etd/792

This Thesis is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
Development and Implementation of a DSP Based Air Detector System to Prevent Embolism During Hemodialysis Therapy

by

Nhat Nguyen

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Wilfrido Moreno, Ph.D.
James T. Leffew, Ph.D.
Paris Wiley, Ph.D.

Date of Approval
November 4th, 2005

Keywords: CodeWarrior, Timer, Interrupt, Ultrasound, Piezoelectric Effect

© Copyright 2005, Nhat Nguyen
DEDICATION

I dedicate this thesis to my father, Nguyen Chi, and my mother, Nguyen Thi Kinh.
ACKNOWLEDGEMENTS

I would like to take the opportunity to thank my major professor Dr. Wilfrido Moreno for supporting me throughout my undergraduate and graduate studies.

I would like to thank the entire faculty and staff of the Department of Electrical Engineering at the University of South Florida. Particular mention goes to Dr. Wiley, Dr. Leffew and Prof. Ezurek for their teachings and education.

I would like to thank Angel Lasso, Director of Engineering at Baxter Tampa Bay for giving me the opportunity to intern at Baxter and for supporting the development of my thesis.

I would like to thank all the engineers and employees at Baxter Tampa Bay. Particular mention goes to Dr. Alex Yu, East Lee, Joel Tejedor, Hector Caro and Derrick Benton for improving my engineering skills and making my internship such an incredible learning experience.

Last but not least, I would like to thank Yohan Prevot for his continuous support and friendship.
# TABLE OF CONTENTS

LIST OF TABLES iii

LIST OF FIGURES v

LIST OF CODES viii

ABSTRACT ix

CHAPTER 1 INTRODUCTION 1

1.1 Problem Statement 1
1.2 Research Objective 2
1.3 Thesis Organization 3

CHAPTER 2 BIO-MEDICAL OVERVIEW 5

2.1 Kidney Function and Kidney Failure 6
2.2 Medical Treatments 8
2.3 Hemodialysis 11
2.4 Air Embolism 14

CHAPTER 3 SYSTEM OVERVIEW 15

3.1 Air Detector Module 15
   3.1.1 Piezo Electric Ceramic and Piezo Electric Effect 16
   3.1.2 Ultrasonic Transmitter and Receiver 19
   3.1.3 Acoustic Impedance and Impedance Mismatch between Air and a Fluid 20
3.2 Blood Line Clamp 24
3.3 By Pass Loop for Air Bubble Injection 25
3.4 Visual and Audio Alarms 27

CHAPTER 4 HARDWARE OVERVIEW 28

4.1 Interface Circuit Board 28
   4.1.1 DC-DC Converter 30
4.1.2 556 Timer Circuit Driving the Audio Buzzer Alarm 32
4.1.3 3.3V Voltage Regulator 36
4.1.4 Diode Clamp Circuit 37
4.1.5 LEDs 38
4.2 Motorola DSP 56824EVM 39
   4.2.1 Port B: General Purpose Input/Output, (GPIO), Port 41
   4.2.2 Clock Synthesis 46
   4.2.3 Timers 49
   4.2.4 Generation of Interrupts 52

CHAPTER 5 SOFTWARE OVERVIEW 58
   5.1 Metrowerks CodeWarrior Development Environments 58
   5.2 Air Detection Algorithm 59
      5.2.1 Registers Settings 64
      5.2.2 GPIO Interrupt Service Routine 66
      5.2.3 Air Detector Input Signal Detection 68

CHAPTER 6 EXPERIMENTAL RESULTS 74
   6.1 System Validation Test 75
   6.2 System Performance Test 76

CHAPTER 7: CONCLUSIONS AND FUTURE WORK 83
   7.1 Conclusions 83
   7.2 Future Work 84

REFERENCES 85

APPENDICES 87

   Appendix A  Air Detector DSP Code 88
   Appendix B  “56824Register.h” Code 92
   Appendix C  System Performance Test Data 93
LIST OF TABLES

Table 3.1: Typical Piezo Electric Materials and their d and K Values 19
Table 3.2: Impact of Temperature on Acoustic Impedance 21
Table 3.3: Acoustic Impedance in Different Media 22
Table 4.1: Truth Table for the 74ACQ244 Chip 31
Table 4.2: Limiting Resistances 39
Table 4.3: Port B: GPIO and PBDDR Register Formats 43
Table 4.4: PBDR Register When No Air was Detected in the Blood 44
Table 4.5: PBDR Register When Air was Detected in the Blood 44
Table 4.6: PBINT Register for Detection of Rising Edge Transition on PB0 45
Table 4.7: PBINT Register for Detection of Falling Edge Transition on PB0 46
Table 4.8: PCR0 Register for the 70MHz Clock Signal 47
Table 4.9: PCR1 Register for the 70MHz Clock Signal 48
Table 4.10: Clock Source/Event Select Bits Association 51
Table 4.11: TCR01 Register Format 51
Table 4.12: TCR2 Register Format 51
Table 4.13: TCR01 Register for Activation of the Timer1 & Timer0 Timing Modules 52
Table 4.14: Interrupt Priority Structure; Level 1 54
Table 4.15: Interrupt Priority Structure; Level 0 54
LIST OF FIGURES

Figure 1.1: DSP Based Air Detector System 4
Figure 2.1: ESRD Worldwide Patient Population 6
Figure 2.2: Front View of Urinary Tract 7
Figure 2.3: Right Kidney Sectioned 8
Figure 2.4: Kidney’s Vasculature 8
Figure 2.5: PD Process 10
Figure 2.6: Hemodialysis Process 12
Figure 2.7: Central Venous Catheter Assess 12
Figure 2.8: Fistula Access 13
Figure 2.9: Graft Access 13
Figure 3.1: Air Detector Module 15
Figure 3.2: Induced Polarization Due to an Applied Mechanical Strain 16
Figure 3.3: Induced Polarization Resulting in an Electric Field 17
Figure 3.4: Mechanical Strain Due to an Applied Electric Field 18
Figure 3.5: Ultrasonic Transmitter and Receiver System 20
Figure 3.6: Sound Pressure in Different Media 23
Figure 3.7: Blood Line Clamp 24
Figure 3.8: Blood Line Tubing 25
Figure 3.9: Peristaltic Pump 26
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.10</td>
<td>Visual and Audio Alarms</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Interface Circuit Board</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>Schematic Diagram of the Interface Circuit Board</td>
<td>29</td>
</tr>
<tr>
<td>4.3</td>
<td>IC Connection Diagram for the 74ACQ244 Chip</td>
<td>31</td>
</tr>
<tr>
<td>4.4</td>
<td>Schematic Diagram of the DC-DC Converter</td>
<td>32</td>
</tr>
<tr>
<td>4.5</td>
<td>Schematic Diagram of the 556 Astable Timer</td>
<td>33</td>
</tr>
<tr>
<td>4.6</td>
<td>The 556 Astable Timer and Capacitor Output Plots</td>
<td>34</td>
</tr>
<tr>
<td>4.7</td>
<td>Block Diagram of the 556 Timer</td>
<td>34</td>
</tr>
<tr>
<td>4.8</td>
<td>Schematic Diagram of the LM317 Voltage Regulator</td>
<td>36</td>
</tr>
<tr>
<td>4.9</td>
<td>Schematic Diagram of the Diode Clamp Circuit</td>
<td>37</td>
</tr>
<tr>
<td>4.10</td>
<td>Output of Diode Clamp Circuit</td>
<td>38</td>
</tr>
<tr>
<td>4.11</td>
<td>Schematic Diagram of the Power LEDs</td>
<td>39</td>
</tr>
<tr>
<td>4.12</td>
<td>DSP56824 Chip Architecture</td>
<td>40</td>
</tr>
<tr>
<td>4.13</td>
<td>Connection Diagram for the DSP GPIO Board</td>
<td>43</td>
</tr>
<tr>
<td>4.14</td>
<td>Block Diagram of On-Chip Clock Synthesis Module</td>
<td>47</td>
</tr>
<tr>
<td>4.15</td>
<td>Block Diagram of the Timer Module</td>
<td>50</td>
</tr>
<tr>
<td>4.16</td>
<td>Interrupt Service Routine Diagram</td>
<td>53</td>
</tr>
<tr>
<td>5.1</td>
<td>CodeWarrior IDE Debugging Window</td>
<td>59</td>
</tr>
<tr>
<td>5.2</td>
<td>Main Program Flow Chart</td>
<td>60</td>
</tr>
<tr>
<td>5.3</td>
<td>Port B Interrupt Service Routine Flow Chart</td>
<td>62</td>
</tr>
<tr>
<td>5.4</td>
<td>IRQB Push-button Interrupt Service Routine Flow Chart</td>
<td>63</td>
</tr>
<tr>
<td>5.5</td>
<td>State Diagram for Air Detector Input Signal Detection Method</td>
<td>68</td>
</tr>
<tr>
<td>6.1</td>
<td>Bovine Blood Used During Testing</td>
<td>75</td>
</tr>
</tbody>
</table>
Figure 6.2:  XY-Scatter Plot of System Performance  78
Figure 6.3:  3D Column Plot of System Performance  79
Figure 6.4:  Average and Standard Deviation Plots of System Performance  80
Figure 6.5:  Line Plot of System Performance  81
Figure 6.6:  Detection Percentage Plot of System Performance  82
LIST OF CODES

Code 5.1: Simplified Version of Main Program 65
Code 5.2: Setup Interrupt Service Routine 67
Code 5.3: Output of State 2 70
Code 5.4: Output of State 3 71
Code 5.5: Behavior of State 4 72
Code 5.6: Behavior of State 5 73
DEVELOPMENT AND IMPLEMENTATION OF A DSP BASED AIR DETECTOR SYSTEM TO PREVENT EMBOLISM DURING HEMODIALYSIS THERAPY

Nhat Nguyen

ABSTRACT

This thesis describes the design of a DSP based air detector system to prevent air embolism during Hemodialysis, which is a treatment option for kidney failure disease. Hemodialysis consists of removing blood from the body, filtering and treating the blood to remove toxic substances such as wastes and fluids, reestablishing proper chemical levels in the blood and returning the processed blood to the body. The functions of hemodialysis are performed through the use of a dialyzer, which is also known as an artificial kidney. During hemodialysis small air bubbles may infiltrate the tubing used during the therapy and combine to form larger air bubbles that are harmful to the patient. If an air bubble is large enough and enters the patient’s circulatory system, the blood flow can be blocked and the patient can die by embolism.

Most of the hemodialysis instruments in use today are equipped with air detection systems, which are based on analog design and digital microcontrollers. This thesis presents a design method based strictly on DSP technology. The Motorola DSP 56824EVM was considered suitable for this biomedical application since its performance
parameters include high-speed, multi-signal control capability, reliability and stability. These performance parameters are considered to be the most important when designing biomedical instruments dealing with human beings’ life and safety. The objective of this research was the development and implementation of a DSP algorithm for the detection and measurement of the sizes of air bubbles in a fluid. In addition the algorithm had to possess the capability, when appropriate, to initiate protective and awareness measures such as triggering a tube clamp as well as activating visual and audio alarms. The air detection was accomplished by means of a commercial air detector module, which was based on piezo ceramic and ultrasound sensing. The function of the tubing clamp was to stop the fluid flow in the tubing and prevent an air bubble from entering the patient’s circulatory system. A secondary goal of this research was to exploit the capability of the DSP 56824EVM and demonstrate its suitability for biomedical applications.
CHAPTER 1

INTRODUCTION

1.1 Problem Statement

The kidneys have the physiological function of cleaning the blood by removing wastes, excess fluids and minerals. Additionally, the kidneys regulate the level of certain chemicals such as sodium, phosphorus and potassium in order to maintain an adequate chemical balance in the body. When the kidneys’ functionality is greatly reduced or the kidneys completely cease to work, kidney failure occurs. A person suffering renal disfunction usually undergoes medical treatments to sustain life. A common medical treatment for kidney failure is Hemodialysis, which uses an artificial kidney to clean the blood. A rare but possible problem associated with hemodialysis is the infiltration of small air bubbles inside the tubing used during the therapy. The infiltrated air gases, inside the blood, can combine together and form large air bubbles, which are called air emboli. These air bubbles are very dangerous and harmful to the patient. If an air bubble enters the patient’s circulatory system the blood flow can be blocked and the patient can die as a result of the air embolism. Depending on the size of the air bubble and the location where the blood stream is obstructed, air embolism can cause strokes, brain damage and cardiovascular arrest.
1.2 Research Objective

This research focused on the development and implementation of an air detector system to prevent air embolism during hemodialysis therapy. Most of the new hemodialysis instruments in use today are equipped with air detection systems based on analog designs and the use of digital microcontrollers. This research was directed toward the development and implementation of an air detection system based on Digital Signal Processing, (DSP), technology. Any DSP, which is adequate for such a biomedical application, must possess performance characteristics that include high-speed, multi-signal control capability, reliability and stability. Such performance parameters are considered to be the most important requirements in designing biomedical instruments, which deal with human beings’ life and safety. The Motorola DSP56824EVM was considered to be suitable for this biomedical application. The detection of the entrapped air was accomplished through the use of a commercial air detector module, which utilized piezo ceramic and ultrasound sensing. A DSP algorithm was developed and implemented to recognize the signal generated by the air detector module. Measurement of the sizes of the air bubbles was accomplished by examining the pulse width of the electrical signals received from the air detector module. Additionally, the DSP algorithm was used to trigger a tubing clamp as well as a visual and audio alarm when the presence of 60µL air bubbles was detected. The function of the tubing clamp was to stop the blood flow in the tubing and prevent the air bubbles from re-entering the patient’s circulatory system. The DSP algorithm was also structured so that if air bubbles smaller than 60µL were detected neither the tubing clamp nor any of the alarms triggered. The ultimate goal
of this research was to exploit the capability of the DSP56824EVM and demonstrate its suitability for biomedical applications.

1.3 Thesis Organization

This thesis consists of 7 chapters. Chapter 2 provides a biomedical overview, which presents the physiology of kidneys and general medical information about kidney failure, hemodialysis therapy and problems associated with air embolism. Chapter 3 provides an overview of the DSP based Air Detector System from the system level point of view. Chapter 3 illustrates the physics behind the detection of air and describes the various components in the system, their characteristics and their functional properties. The discussion continues in Chapter 4, which illustrates the electrical hardware employed by the system. Chapter 4 emphasizes the design of the Interface Circuit Board and the hardware characteristics of the Motorola DSP56824EVM board. Chapter 5 describes the software algorithm and the codes that were developed and implemented on the DSP board in order to detect and measure the pulse width of the signal generated by the Ultrasonic Air Detector module. A System Validation Test and the System Performance Data are presented in Chapter 6. The System Validation Test was conducted to ensure that the Dsp based Air Detector System consistently, reliably and accurately detected air bubbles in the blood. The System Performance Data were collected using bovine blood to analyze the system performance with different sizes of air bubbles and to verify that air bubbles larger than 60µL were successfully stopped via the blood line clamp. Conclusions and future work recommendations are discussed in Chapter 7. Figure 1.1 provides a picture of the system developed during this research.
Figure 1.1: Dsp Based Air Detector System
CHAPTER 2

BIO-MEDICAL OVERVIEW

Throughout the United States in excess of 450,000 people are undergoing medical treatment for end-stage renal disease, (ESRD), which is irreversible and lethal if untreated. Based on data published by the Centers for Medicare and Medicaid Services, (CMS), the approximate number of ESRD patients under Medicare or Medicaid that require medical treatments has grown from 66,000 in 1982 to 260,000 at the end of 2000. This population is estimated to be growing at an annual rate of 8%. The most recent and complete data for the total ESRD population in the United States come from the 2004 Annual Data Report of the United States Renal Data System, (USRDS). The USRDS is a national data system that collects, analyzes and distributes information about ESRD in the United States in conjunction with the CMS. According to this annual report the number of patients receiving ESRD therapy at the end of 2002 was 431,284. This implies that approximately one of every 3500 people in the United States is undergoing treatment for ESRD. However, this problem is not isolated to the United States. In 1984 the number of ESRD patients worldwide was estimated to be 300,000. Today there are approximately 1,500,000 people with renal failure. Figure 2.1 illustrates the worldwide ESRD patient population.
ESRD is not only a serious medical problem for the United States. ESRD is a public health concern throughout the world. Therefore, an understanding of the problem from the medical prospective is essential for optimizing treatment and medical devices, which sustain life for those affected by ESRD.

2.1 Kidney Function and Kidney Failure

The kidneys are organs located just below the rib cage near the middle of the lower back. The physiological function of healthy kidneys is to clean the blood by removing excess fluid, minerals and wastes. Each day a pair of kidneys processes about 200 quarts of blood. Approximately 2 quarts of waste products and extra water, which become urine, are filtered from the blood by the kidneys. Urine is collected in the bladder after it flows through tubes, called ureters. Wastes in the blood originate from the normal breakdown of active tissues and from the digestive process of food. The body uses the nutritious elements from food for energy and self-repairs, while the waste is sent to the blood. Figure 2.2 illustrates the front view of the urinary tract.
The filtering process in the kidneys takes place in tiny elements called nephrons. A healthy kidney has about a million nephrons. In a nephron, chemical exchange occurs in small blood vessels called glomerulus, which are connected to tiny urine collecting tubes or tubule. During the chemical exchange waste materials and excess water leave the blood and enter the urinary system. Simultaneously, the kidneys regulate the level of chemicals such as sodium, phosphorus and potassium in order to maintain the proper chemical balance necessary for life. If the body has insufficient amounts of these substances, the kidneys release them back to the blood. On the contrary, if these chemicals are excessive they are purged from the blood via the urinary system. In addition to removing wastes and balancing chemicals in the body, the kidneys produce hormones that keep bones strong and maintain healthy blood. Figure 2.3 and 2.4 picture respectively the internal components of a healthy kidney and the typical appearance of kidney’s blood vessels, (vasculature).
When the kidneys stop operating at their full potential or lose completely their physiological functions ESRD occurs. Harmful wastes build up in the body causing blood pressure to rise and excess fluid to be retained in the blood. These conditions adversely affect the synthesis of red blood cells, which leads to anemia and medical conditions that affect bones, (Renal Osteodystrophy), nerves and skin (Pruritus due to uremic toxins).

2.2 Medical Treatments

Two options are available to an individual that has been diagnosed with ESRD. In the United States a person can choose to undergo medical treatments to sustain life or refuse and/or withdraw from medical treatment if that individual feels that such treatment is a burden that will only prolong suffering. This second, extreme, choice leads to death within weeks due to the fatal nature of the disease. Individuals who choose life sustaining treatments have a choice of three medical treatment options, which are:
- Hemodialysis,
- Peritoneal Dialysis,
- Kidney Transplantation.

Hemodialysis, (HD), is a therapy that cleans and filters blood by using a machine to temporarily remove harmful wastes, extra salt and extra water. During HD blood is pumped out of the body and is run through an artificial kidney called a dialyzer. Wastes and extra water are removed while chemicals such as potassium, sodium, calcium and bicarbonate are brought to physiological levels. Hemodialysis is usually required three times a week. Each treatment lasts from 3 to 5 or more hours. Usually, HD is performed in specialized clinics. However, a new generation of hemodialysis machines allows hemodialysis to be performed comfortably at home.

Peritoneal dialysis, (PD), uses the lining of the abdomen to remove wastes. This lining is called the peritoneal membrane and acts as the artificial kidney. During PD the peritoneal cavity is filled, through a soft tube, with a dialysis solution that contains a mixture of minerals and sugar. The sugar, called dextrose, draws wastes, chemicals and extra water from the tiny blood vessels inside the peritoneal membrane into the dialysis solution. The filtering process takes several hours. Once it is completed, the used solution is drained from the peritoneal cavity and collected in a disposable bag. There are two types of Peritoneal Dialysis. Continuous Cycler-Assisted Peritoneal Dialysis, (CCPD), uses a machine called a cycler to fill and empty the peritoneal cavity with the dialysis solution three to five times at night, while the individual is sleeping. In the morning the patient is required to perform only one exchange that lasts the entire day. A
drain is performed at bedtime when the cycler is reconnected. The second type of PD is called Continuous Ambulatory Peritoneal Dialysis, (CAPD), which does not require the use of a cycler machine. CAPD requires the dialysis solution to always be present in the abdomen. The dialysis solution is exchanged 4 to 5 times a day by manually draining and refilling the abdomen. At night the patient fills the peritoneal cavity with the solution and drains it in the morning. Both PF therapies require strict aseptic techniques. Figure 2.6 illustrates the PD process.

Hemodialysis and peritoneal dialysis are medical treatments that replace the kidney functions. These treatments help patients to feel better and live longer. However, they do not cure kidney failure. Surgical kidney transplantation is considered to be the only definitive cure for ESRD. Kidney transplantation consists of surgically placing a healthy kidney from a donor into the patient’s body. The replacement kidney can be donated by a family member of the patient, (living related donor), by a person who has recently died, (deceased donor), or from a spouse or a very close friend of the patient, (living unrelated donor). If an ESRD individual does not have a living donor, the patient is placed on a waiting list for a deceased donor kidney. The wait for a deceased donor kidney can be several years. Therefore, patients awaiting a replacement kidney, from a
deceased donor, will be required to receive dialysis treatments. There are three main factors in matching available kidneys with potential recipients. These factors help to predict whether the patient’s immune system will accept or reject the new kidney. The three factors are:

- Blood type,
- Human leukocyte antigens, (HLAs),
- Cross-matching antigens.

Blood type is the most important matching factor. The blood type of the donor has to match the blood type of the recipient. In addition the HLAs of the donor and the recipient have to be compatible. Since HLAs are inherited antigens, family members are most likely to possess a complete match. The last factor is cross-matching. This is the last step before implanting the organ. A small sample of the blood of the donor is mixed with the blood of the recipient to see if there is a side-effect reaction. When these three factors are successfully matched the transplant operation can take place.

2.3 Hemodialysis

As described in section 2.2, hemodialysis is a therapy that removes organic wastes and extra water by filtering blood. During hemodialysis a machine pumps blood out of the patient’s body via a system of tubings. The blood is run through an artificial kidney called a dialyzer and then returned to the patient. Waste in the blood is removed by a diffusion process while extra fluids are eliminated by an ultrafiltration method. Heparin is infused in the blood, during the therapy, to prevent the blood from coagulating and clotting. A system of multiple arterial and venous pressure monitors are engaged in order
to ensure the blood pressure is kept within a safe margin. An air trap and air detector are used to prevent air embolism. Figure 2.7 illustrates the hemodialysis process.

![Figure 2.6: Hemodialysis Process](image)

Before starting hemodialysis, a vascular access to the bloodstream has to be created in order to provide a means of extracting and returning blood rapidly from the body to the hemodialysis machine. Typically, there are three main types of accesses. The first type is the Central Venous Catheter, which is a long slender tube placed in a vein either in the chest, neck or leg. Figure 2.7 illustrates a venous catheter.

![Figure 2.7: Central Venous Catheter Assess](image)
The second access is via a fistula. The fistula is a direct surgical connection of an artery to a vein, which is usually located in the forearm. The increased blood flow makes the vein grow larger and stronger, which facilitates its use for repeated needle insertions. Figure 2.9 illustrates a fistula for hemodialysis.

![Fistula Access](image1)

**Figure 2.8: Fistula Access**

The last type of vascular access uses a graft, which connects an artery to a vein through the use of a synthetic tube. Hemodialysis needles are inserted into the synthetic tube instead of the vein. Figure 2.10 illustrates a graft vascular access.

![Graft Vascular Access](image2)

**Figure 2.9: Graft Vascular Access**

Vascular accesses dangerously expose blood and the circulatory system to the environment. If not cleaned regularly and disinfected appropriately, vascular accesses can be subjected to infection. Additionally, if hemodialysis needles are accidentally disconnected, air may infiltrate and travel inside the blood causing air embolism.
2.4 Air Embolism

Embolism occurs when a solid, semi-solid or gaseous substance traveling in the bloodstream obstructs the blood flow. The substance is called an embolus. Common types of embolus are blood clots, crystal or cholesterol, clumps of infected cells, bits of bone marrow and a mix of air gases. If the blood obstruction is caused by air bubbles circulating in the blood an air embolism arises. When an air embolus is present in an artery, it will travel through a system of capillaries that gradually becomes smaller. The embolus will eventually reach a point where it completely blocks a blood vessel and cuts off the blood supply to some area. Absence of blood to an area causes the corresponding tissues and cells to die due to the absence of oxygen. If the embolus blocks an artery that supplies the brain a stroke will ensue and permanent brain damage will occur. However, if the air embolus occurs in a vein it does not cause harm until it reaches the heart since the vein system widens along the direction of the blood flow. In addition, if the air embolus is large enough to block a cardiac artery a cardiovascular collapse may take place.

Air embolism during hemodialysis therapy is a rare but potentially lethal complication. Air can infiltrate inside the hemodialysis circuit from malfunctioning tubing fittings, vascular assess’ needles and loose dialyzer ports. The dialyzer itself may be a source of air if refrigerated dialysate containing dissolved air is used. In fact, due to the pressure developed in the tubing and the blood flow rate, dissolved air may aggregate and form larger air bubbles. Due to the danger of air embolism, new hemodialysis machines are equipped with the double safety features of an air trap chamber and an air detector module.
CHAPTER 3

SYSTEM OVERVIEW

In order to understand how the DSP based Air Detector functions, it is important to comprehend the physics behind the air detection and to learn the various components in the system. This chapter introduces each module and explains their characteristics and functional properties.

3.1 Air Detector Module

The Air Detector Module is a commercial, non-invasive, ultrasonic transducer that is used to detect the presence of air bubbles in fluid flowing through flexible plastic tubing. Two precision piezo electric ceramic plates are secured within a molded protective housing. The plates are positioned so that they face one another. The tubing is placed between the two ceramic plates. Figure 3.1 provides a picture of the Air Detector Module.

Figure 3.1: Air Detector Module
3.1.1 Piezo Electric Ceramic and Piezo Electric Effect

Piezo electric ceramics are materials that have the unique property of producing electrical charges when compressed, twisted or distorted and exhibit mechanical strain or distortion when an electric field is applied. If a piezo electric ceramic is mechanically stressed, charge separation occurs on the surfaces of the ceramic. As a result, one surface of the ceramic becomes positively charged while the other side is negatively charged, which causes a potential difference to develop. Changing the direction of deformation reverses the polarity of the generated voltage as illustrated in Figure 3.2.

![Figure 3.2: Induced Polarization Due to an Applied Mechanical Strain](image)

(a) Piezo Electric Ceramic in the absence of an applied force.
(b) Piezo Electric Ceramic under a compressional force.
(c) Piezo Electric Ceramic under a tensional force.

Generally, an applied stress in one direction produces an induced electrical potential in other directions. Suppose a mechanical force, \( T_j \), along the \( j \)-direction is
applied to a piezo ceramic. An induced polarization, \( P \), will result, which is linearly related to \( T_j \) by:

\[
P_i = d_{ij} T_j, \quad (3.1)
\]

where \( d_{ij} \) is the Piezo Electric Coefficient. For most materials, the resulting electric field and the polarization are related by:

\[
P = \varepsilon_0 \chi_e E \quad (3.2)
\]

where \( \chi_e \) is the Electric Susceptibility and \( \varepsilon_0 \) is the Permittivity of Free Space. Figure 3.3 illustrates how the induced polarization of charge produces an electric field.

On the contrary, the same piezo electric ceramic shows signs of mechanical strain and distortion when the ceramic is placed into an electric field. The direction of mechanical deformation depends on the direction of the applied field and the intensity is proportional to the strength of the field. The induced strain, \( S_j \), along the \( j \)-direction is proportional to the applied electric field, \( E_i \), along the \( i \)-direction. The induced strain and the applied electric field are related by:

\[
S_i = d_{ij} E_j, \quad (3.3)
\]

Figure 3.4 illustrates the relationship of induced strain due to the applied electric field.
Figure 3.4: Mechanical Strain Due to an Applied Electric Field

(a) Piezo Electric Ceramic in the absence of an applied electric field.

(b) Piezo Electric Ceramic under an applied electric field, $V$.

(c) Piezo Electric Ceramic under an applied electric field, $-V$.

These two effects are paired together and define the Piezo Electric Effect, which can be a Direct Effect or a Converse Effect.

Piezo electric ceramics are electromechanical transducers that convert an electrical signal into a mechanical signal and vice versa. They are widely used in many engineering applications such as crystal oscillators, ultrasonic transducers, accelerometers and microphones. The relationship between electrical and mechanical energies is given by the Electromechanical Coupling factor, $K$, which is defined in terms of $K^2$ by:

$$K^2 = \frac{\text{Electrical Energy converted to Mechanical Energy}}{\text{Input of Electrical Energy}}.$$  \hspace{1cm} (3.4)

and

$$K^2 = \frac{\text{Mechanical Energy converted to Electrical Energy}}{\text{Input of Mechanical Energy}}.$$  \hspace{1cm} (3.5)

Table 3.1 lists some typical piezo electric materials and the values for their $d$ and $K$ coefficients.
Table 3.1: Typical Piezo Electric Materials and their d and K Values

<table>
<thead>
<tr>
<th>Piezo Electric Material</th>
<th>Piezo Electric Coefficient d (m V⁻¹)</th>
<th>Electromechanical Coupling Factor K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quartz (crystal SiO₂)</td>
<td>2.3 X 10⁻¹²</td>
<td>0.1</td>
</tr>
<tr>
<td>Barium Titanate (BaTiO₃)</td>
<td>190 X 10⁻¹²</td>
<td>0.49</td>
</tr>
<tr>
<td>Lead Zirconate Titanate (PbTi₁₋ₓZrxO₃)</td>
<td>480 X 10⁻¹²</td>
<td>0.72</td>
</tr>
<tr>
<td>Polyvinylidene Fluoride (PVDF)</td>
<td>18 X 10⁻¹²</td>
<td>_</td>
</tr>
</tbody>
</table>

3.1.2 Ultrasonic Transmitter and Receiver

Piezo electric ceramics are widely used to generate ultrasonic waves, (ultrasonic transmitter), and to detect such waves, (ultrasonic receiver). Piezo electric ceramics possess two modes of operation. The first mode is known as the Generator Mode. In the Generator Mode an applied voltage causes the piezo electric ceramic to distort, which causes mechanical strain as described in section 3.1.1 and is known as a Converse Effect. If a sine wave voltage at an ultrasonic frequency is applied to the ceramic, (transmitter), and the ceramic is in contact with a medium it will create a compressional wave that travels through the medium. The generated compressional wave can be longitudinal or transverse according to the ceramic’s cut and shape. The vibration is largest when the electric field stimulates a natural frequency of the piezo electric ceramic. Such a frequency is known as a Resonant Frequency. Generally, the ceramic is cut into a slice
with a thickness equal to one-half of the wavelength of the desired ultrasonic frequency. This construction ensures that most of the energy is emitted at the fundamental frequency. The second mode of operation is called the Motor Mode. When a mechanical wave strikes a ceramic it distorts and creates a voltage, which is viewed as a Direct Effect. Taking advantage of these two modes of operation makes it possible to build an ultrasonic transmitter and receiver system by placing two ceramic plates that face each other and are coupled together. Figure 3.5 illustrates the ultrasonic transceiver system.

3.1.3 Acoustic Impedance and Impedance Mismatch between Air and a Fluid

Ultrasound is defined as high frequency sound waves, which are above the range of human hearing. Normally the ultrasonic frequency range starts at 20 kHz and go up into the megahertz range. As sound waves, ultrasonic waves are simply organized mechanical vibrations traveling through a medium at a specific speed and with a predictable direction of propagation. When the waves encounter a boundary, with a different medium, a portion of their energy will be reflected and a portion will be
transmitted. The amount of reflected energy is related to the acoustic impedances of the two media. Given any two media, the reflection coefficient, \( \Gamma \), as a percentage of incident energy is calculated as:

\[
\Gamma = \frac{Z_2 - Z_1}{Z_2 + Z_1},
\]

where \( Z_1 \) and \( Z_2 \) are the acoustic impedances of medium 1 and medium 2 respectively.

The acoustic impedance is defined as the ratio of the sound pressure, \( p \), to particle velocity, \( v \). The acoustic impedance is also the product of the density, \( \rho \), of the medium and the speed of sound, \( c \), in the medium. The acoustic impedance is measured in Pa·s/m and is given by:

\[
Z = \frac{p}{v} = \frac{J}{v^2} = \frac{p^2}{J} = \rho \ c
\]

where \( J \) is the sound intensity measured in W/m\(^2\). Temperature also has an impact on acoustic impedance. In most cases higher temperatures yield lower acoustic impedances. Table 3.2 presents the impact of temperature on acoustic impedance.

Table 3.2: Impact of Temperature on Acoustic Impedance

<table>
<thead>
<tr>
<th>Temperature (°C)</th>
<th>Speed of Sound in Air C(m/s)</th>
<th>Density of Air ( \rho )(kg/m(^3))</th>
<th>Acoustic Impedance (Pa·s/m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-10</td>
<td>325.4</td>
<td>1.341</td>
<td>436.5</td>
</tr>
<tr>
<td>-5</td>
<td>328.5</td>
<td>1.316</td>
<td>432.4</td>
</tr>
<tr>
<td>0</td>
<td>331.5</td>
<td>1.293</td>
<td>428.3</td>
</tr>
<tr>
<td>5</td>
<td>334.5</td>
<td>1.269</td>
<td>424.5</td>
</tr>
</tbody>
</table>
Air and blood have very different acoustic impedances. A beam of ultrasonic waves, going from blood to air, is almost entirely reflected and only a small portion is transmitted. This effect is due to the fact that the acoustic impedance of blood is several orders of magnitude larger than the acoustic impedance of air, which causes the mismatch between air and blood, as well as the reflection coefficient, to be very high. Table 3.3 lists different medium and the relative acoustic impedance. The acoustic impedances of air and blood are given specifically in Table 3.3.

### Table 3.3: Acoustic Impedance in Different Media

<table>
<thead>
<tr>
<th>Medium</th>
<th>Acoustic Impedance (Pa·s/m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air</td>
<td>429</td>
</tr>
<tr>
<td>Water</td>
<td>1500000</td>
</tr>
<tr>
<td>Blood</td>
<td>1590000</td>
</tr>
<tr>
<td>Fat</td>
<td>1380000</td>
</tr>
<tr>
<td>Muscle</td>
<td>1700000</td>
</tr>
<tr>
<td>Bone</td>
<td>6500000</td>
</tr>
</tbody>
</table>
The enormous acoustic impedance mismatch between air and blood is the key to the ability to detect air bubbles in blood. When the ultrasonic transmitter is excited by a sine wave voltage of a given frequency compressional waves are created. In the present of only blood, the sound pressure amplitude received by the ultrasonic receiver matches closely the amplitude of the transmitted wave. The piezo ceramic on the receiver side gets excited, becomes polarized and creates an electrical signal as illustrated in Figure 3.6.

![Diagram of ultrasonic transmitter and receiver](image)

**Figure 3.6: Sound Pressure in Different Media**

(a) Sound Pressure in the presence of only blood

(b) Sound Pressure in presence of air bubble in blood

When an air bubble passes through the tubing, the compressional waves are greatly reduced in intensity by the reflection phenomenon. The piezo ceramic on the receiver side returns to a neutral electrical state and no electrical signal is produced. The electrical signal is not present for the entire time the air bubble passes through the tubing.
It is possible, with proper electrical signal conditioning, to generate an electric pulse that is logic high when blood is flowing in the tubing and logic low when air is passing. In addition, the width of the pulse can be made proportional to the size of the air bubble. It is this pulsed signal that alerts the DSP board of the presence and the size of an air bubble in the blood.

3.2 Blood Line Clamp

The function of the blood line clamp is to pinch the tubing as soon as the system detects a harmful air bubble in the blood line. By clamping the tubing, the blood flow is stopped and the air bubble cannot enter the patient’s circulatory system. A brushless DC motor was used to accomplish this task. The motor features high energy neodymium magnets for high torque, high speed and extremely smooth and precise motion. It required a +24V input voltage and turned on when +5V was applied to its Clamped Signal. The output torque generated by the motor, when it turned on, was 90oz-in. The current drawn by an active motor was 1.15A. Figure 3.7 presents a picture of the blood line clamp assembly.

![Figure 3.7: Blood Line Clamp](image-url)
3.3 By Pass Loop for Air Bubble Injection

For testing purposes, a By Pass Loop was required in order to inject air bubbles. Due to the flow and the pressure developed inside the tubing an air bubble was likely to break into smaller bubbles if it was injected directly inside the tubing. In the By Pass loop, a Y connector was used to divide the blood flow into two lines. The main line constantly allowed the blood to flow. The second one was manually clamped through the action of a brushless DC motor/clamp, which was identical to the Blood Line Clamp. When the line was clamped, the blood flow was stopped. However, the blood continued to stream into the main line. Since the by pass line was pinched a bubble could be manually injected without being broken by the combined action of pressure and flow. Figure 3.8 presents a diagram of the blood line tubing setup.

Figure 3.8: Blood Line Tubing
(a) Blood flows in both lines
(b) Blood flows only into the By Pass Loop
The air bubble injection was performed using a µL-calibrated syringe via the injection port. Once the air bubble was entered in the tubing, the By Pass Loop clamp was manually turned off, which allowed the blood to flow in both lines and then recombined together though a second Y connector. The air bubble was then pushed by the blood flow and traveled through the tubing eventually reaching the air detector module. The flow speed was controlled by a peristaltic blood pump that sucked the blood from a blood container, ran it through the system’s tubing and returned it to the same container in a continuous cycle. Figure 3.9 presents a picture of the Peristaltic Pump.

![Figure 3.9: Peristaltic Pump](image)
3.4 Visual and Audio Alarms

The system developed in this research was provided with both a visual and an audio alarm. When the blood was free of air a bi-color LED, with a green color, was illuminated and the audio alarm was set to mute. When a harmful air bubble was detected the bi-color LED turned red and the audio alarm emitted a pulsed sound. Figure 3.9 presents a picture of the visual and audio alarm mechanism.

Figure 3.10: Visual and Audio Alarms
CHAPTER 4

HARDWARE OVERVIEW

The main hardware components utilized in this research were the Interface Circuit Board and the Motorola DSP56824. The following sections provide an overview of these two components.

4.1 Interface Circuit Board

The interface circuit board was designed to allow the system components to efficiently interact with each other and to be electrically compatible in terms of input and output voltages. In addition, the interface circuit was required in order to accommodate, on board, all the analog signals utilized by the DSP based Air Detector System. Figure 4.1 provides a picture of the Interface Circuit Board while Figure 4.2 presents the circuit schematic of the Interface Circuit Board.
Figure 4.1: Interface Circuit Board

Figure 4.2: Schematic Diagram of the Interface Circuit Board
The operational characteristics for each sub-circuit of the Interface Circuit Board will be described in detail in the following sections.

### 4.1.1 DC-DC Converter

When the Motorola DSP board detects a large bubble, in the blood, three output pins on the General Purpose Inputs/Outputs are activated. The three pins produce 3.3V logic high outputs that trigger the blood line clamp, the audio alarms and the visual alarm. Both the blood line clamp and the audio buzzer required a 5V control signal. Therefore, a DC-DC converter was required to convert the 3.3V output signals generated by the DSP board to 5V levels. A 74ACQ244 integrated circuit was used to accomplish the change in potential level. The 74ACQ244 is a Quiet Series Octal Buffer/Line Driver with 3-STATE outputs. The 74ACQ244 was designed to be employed as a memory address driver, clock driver and bus oriented transmitter or receiver. During this research the IC was used to simply convert an output voltage of 3.3V to 5V.

The 74ACQ244 integrated circuit has eight buffers, which are grouped in two sets. Each set has 3-STATE Output Enable Inputs called OE₁ and OE₂. When the enable input is set low, the outputs of the buffers follow the state of the corresponding input. That is, if the input of the buffer is low, the output is low as well and if the input is high, the output follows the input and becomes high. When the enable input is set to high, the outputs of the buffers become high-impedance outputs regardless of the inputs. Figure 4.3 presents the connection diagram for the 74ACQ244 chip and Table 4.1 presents the truth table related to the functional capabilities of the 74ACQ244 chip.
Since a DC-DC converter circuit was required, the enable input OE1 was tied to ground in order to force the outputs of the buffers to follow the state of their corresponding input. Also, the VCC pin was connected to the 5V line to guarantee the presence of a 5V potential at the buffer output stage when the inputs were high. In addition, two pull down resistors were added to the outputs of the buffers to prevent them from floating when the buffers’ inputs were low. When the input is low, the buffer output is pulled down to ground, which prevents the output from floating. However, if
the input is high, the output is 5V and very little current flows through the pull down resistor due to its high resistance. Figure 4.4 presents the schematic for the DC-DC converter sub-circuit.

![DC-DC Converter Schematic Diagram](image)

4.1.2 The 556 Timer Circuit for Driving the Audio Buzzer Alarm

A 556 timer operating in an astable mode was used to drive the Audio Buzzer Alarm. The function of the circuit was to turn the buzzer on for approximately half a second and to turn it off for approximately half a second. Therefore, a 556 timer was employed to provide a 5V square wave with a duty cycle of 50% and a period of 1 second. The schematic for the 556 astable timer sub-circuit is presented in Figure 4.5.
The 556 astable circuit uses a capacitor, C, which is connected between the Trigger pin and the Ground pin in order to force the output to switch repeatedly between logic high and logic low levels. As soon as the 556 timer is powered by a 5V Vcc potential capacitor C starts charging. The Threshold and Trigger inputs monitor the capacitors’ voltage level. When the capacitors’ voltage level reaches \( \frac{2}{3} \) of Vcc, also known as the Threshold Voltage, the output becomes logic low and the Discharge pin is connected to ground. With the Discharge pin connected to ground, capacitor C discharges and current flows through a resistor placed between the Discharge pin and the Threshold/Trigger pin. When the voltage across C decreases to \( \frac{1}{3} \) of Vcc, also known as Trigger voltage, the output becomes high again and the discharge pin is disconnected, which allows the capacitor to start charging again. The waveforms for the 556 astable timer and capacitor output are illustrated in the Figure 4.6.
Figure 4.6: The 556 Astable Timer and Capacitor Output Plots

Figure 4.7 presents the pin layout and block diagram of the 556 timer.

The period and frequency of the square wave were calculated using the relationships given by:

\[ T = 0.7(R_{11} + 2R_{12})C_2 \]  \hspace{1cm} (4.1)

\[ f = \frac{1.44}{(R_{11} + R_{12})C_2} \]  \hspace{1cm} (4.2)

The time period is the sum of the output high Mark Time, \( T_m \), and the output low Space Time, \( T_s \), which are defined as:
\[ T_m = 0.7(R_{11} + R_{12})C_2, \]  
\[ T_s = 0.7R_{12}C_2. \]  

In order to produce a timer waveform with a duty cycle of 50% \( T_m \) was set to 1 second and \( T_s \) was set to 0.5 second. In addition, choosing a value of 20\( \mu \)F for \( C_2 \), in equations (4.3) and (4.4) yields simultaneous relations for \( R_{11} \) and \( R_{12} \) as:

\[
\begin{cases}
1 = 0.7(R_{11} + 2R_{12})20 \times 10^{-6} \\
0.5 = 0.7R_{12}20 \times 10^{-6}
\end{cases}
\]  

(4.5)

Solution of the simultaneous relations presented in equation (4.5) yielded the design values for the parameters presented in equation (4.6):

\[ C_2 = 20\mu \text{F} \]  
\[ R_{11} = 3500\Omega \]  
\[ R_{12} = 35000\Omega \]  

(4.6)

Additionally, a 0.1\( \mu \)F capacitor, \( C_3 \), was connected between the control pin and ground in order to reduce electrical noise.
4.1.3 3.3V Voltage Regulator

The air detector module’s output is a 5V DC signal that goes low when an air bubble is detected. Since the Motorola DSP board’s General Purpose Inputs/Outputs are electrically rated so that a Logic High Voltage, \( V_{HH} \), on both inputs and outputs is 3.3V. A DC voltage regulator was required in order to provide a 3.3V regulated voltage for the system. The 3.3V voltage was required as a reference voltage for the diode clamp circuit, which will be described in the next section. The schematic for the 3.3V Voltage Regulator circuit is presented in Figure 4.8.

![3.3 V Voltage Regulator](image)

Figure 4.8: Schematic Diagram of the LM317 Voltage Regulator

An LM317 integrated circuit was used in the design of the voltage regulator. The LM317 is a regulator with no ground terminal. The LM317 adjusts the output voltage to maintain a constant 1.25V potential between the output terminal and the adjustment terminal. A small resistor, \( R_6 \), was used between these two terminals so that the voltage remained constant at 1.25V. The current through \( R_6 \) was chosen to be 5.2mA. Therefore, the value for \( R_6 \) was required to be 240\( \Omega \). The output voltage is given by:

\[
V_{out} = 1.25(1 + R_5/R_6),
\]

(4.7)
where \( R_5 \) is the resistor that controls the output voltage. The value for \( R_5 \) was found to be equal to 394\( \Omega \) for an output voltage of 3.3V. In addition, a loop compensation capacitor was used as for the regulator. The loop compensation capacitor was connected between the output voltage pin and the ground pin.

4.1.4 Diode Clamp Circuit

The diode clamp circuit was designed to prevent the voltage of the \textit{Air Detected} signal, produced by the Air Detector Module, from exceeding 3.3V. Figure 4.9 presents the schematic diagram for the Diode Clamp sub-circuit.

![Diode Clamp Circuit Diagram](image)

Figure 4.9: Schematic Diagram of the Diode Clamp Circuit

A voltage divider with an input voltage of 3.3V was used to obtain a bias voltage of 2.6V across \( R_8 \). The diode becomes forward biased as soon as the anode voltage exceeds the sum of the bias voltage and the 0.7V diode voltage drop. When the diode is forward biased, the anode voltage cannot assume a value larger than the bias voltage +0.7V. Therefore, any voltages that would exceed 3.3V are clamped at the 3.3V level. When the diodes’ anode voltage is lower than 3.3V the diode is reversed biased.
reverse biased diode appears as an open circuit, which forces the anode voltage to maintain a constant voltage level. A 1 KΩ resistor was added between the input signal and the anode of the diode to limit the sink current. Figure 4.10 illustrates how a 5V signal, generated by the Air Detector Module, was clamped to 3.3V.

\begin{figure}[h]
\begin{center}
\includegraphics[width=\textwidth]{diode_clamp_circuit.png}
\caption{Output of the Diode Clamp Circuit}
\end{center}
\end{figure}

4.1.5 LEDs

For troubleshooting purposes a set of LEDs was used to monitor the presence of 3.3V, 5V and 24V potentials in the interface circuit. Each LED required a limiting resistor whose value was determined from the equation:

\[
R_{\text{limiting}} = \frac{V_{\text{supply}} - V_{\text{LED}}}{I_{\text{desired}}},
\]

(4.8)

where \(V_{\text{LED}}\) is the voltage drop across the LED and \(I_{\text{desired}}\) is the desired current through the LED. \(V_{\text{LED}}\) was 2.2V and \(I_{\text{desired}}\) was 15mA. The limiting resistances are listed in Table 4.2.
### Table 4.2: Limiting Resistances

<table>
<thead>
<tr>
<th>V supplied (V)</th>
<th>Limiting Resistance (Ω)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>80</td>
</tr>
<tr>
<td>5</td>
<td>193</td>
</tr>
<tr>
<td>24</td>
<td>1460</td>
</tr>
</tbody>
</table>

The schematic diagram for the power LED’s is presented in Figure 4.11.

![Schematic Diagram of the Power LEDs](image)

**Figure 4.11: Schematic Diagram of the Power LEDs**

### 4.2 Motorola DSP 56824EVM Board

The Motorola DSP56824EVM is a Digital Signal Processor board that provides a hardware tool for the development of applications that use the DSP56824 chip. The configuration flexibility and processing power of this general purpose DSP makes it ideal for signal processing and control functions. The CPU permits as many as six operations per instruction cycle via the parallel operations of three execution units. This capability translates into 35 million instructions per second, (MIPS), with a 70 MHz clock. The DSP56824 consists of the DSP56800 core, program and data memory and peripherals useful for embedded control applications. Figure 4.12 presents the architecture of the DSP core chip.
The main features of the DSP56824EVM board include:

- **On-Chip memory**
  1. 32K, 16-bit program ROM
  2. 128K, 16-bit program RAM
  3. 3.5K, 16-bit RAM for data and applications
  4. 2K, 16-bit Data RAM

- **Off-Chip memory**
  1. As much as 64 K, 16-bit data memory
  2. As much as 64 K, 16-bit program memory
  3. External memory expansion; port programmable
Peripheral Circuits

1. External Memory Interface
2. Sixteen dedicated GPIO pins, (eight pins programmable as interrupts)
3. Programmable Input/Output Port
4. Serial Peripheral Interface, (SPI)
5. Synchronous Serial Interface, (SSI)
6. Three programmable 16-bit timers
7. Two external interrupt-mode control push-buttons
8. One external reset pin for hardware reset
9. JTAG/On-Chip Emulation, (OnCE™)
10. Phase Lock Loop-based, (PLL-based), frequency and clock synthesizer for the DSP core clock
11. Debugging LEDs

4.2.1 Port B: General Purpose Input/Output, (GPIO), Port

The DSP56824EVM offers several General Purpose Input/Output, (GPIO), ports. Port B is a dedicated GPIO that provides 16 programmable I/O pins. Port B can be configured to generate an interrupt on its lower eight pins, (PB7 through PB0), if they are configured as inputs. The upper eight pins, (PB15 through PB8), do not offer such a feature. Port B is controlled by three read/write registers, which are located in the X memory of the processor. The three read/write registers are designated as:
The Port B Data Direction Register (PBDDR) allows each pin to be programmed as an input pin or an output pin. The direction of each pin is controlled by a corresponding control bit in the PBDDR. A pin is configured as an input pin if the corresponding control bit is set to “0”. When a pin is used as an output pin the corresponding control bit must be set to “1”. The PBDDR register is cleared during processor reset, which configures all pins as input pins.

During this research, one input and four outputs were required from the Port B GPIO. The input received the Air Detected signal generated by the air detector module and conditioned it through the diode clamp circuit. The input pin associated with the Air Detected signal was pin 1 and the related control bit in the PBDDR register was PB0. The first output was connected to the visual alarm circuit, which turned on the green LED when air was not present in the blood. This output was associated with GPIO pin 9 and its control bit was PB8. The other three outputs were connected respectively to the audio alarm, (GPIO pin 10), the red LED of the visual alarm circuit, (GPIO pin 11), and the blood line clamp, (GPIO pin 12). The corresponding control bits in the PBDDR register were PB9, PB10 and PB11. These outputs were active when air was detected. Figure 4.13 presents the DSP GPIO pin layout.
Figure 4.13: Connection Diagram for the DSP GPIO Board

Pin 17 and Pin 18 of the GPIO are reference ground signals and are not programmable. Pins 2 through 8 and 13 through 16 were not used. For convenience, unused pins were cleared, even though they could be set with no effects on system performance. Table 4.3 catalogs each pin, the associated control bit in the register and its assigned value.

<table>
<thead>
<tr>
<th>GPIO PIN</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBDDR Control Bit</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The second register used for controlling the Port B GPIO was the PBDR. This register determined the logic level of the pins in use. When a value of “0” was assigned to a PBDR control bit the corresponding pins output logic level was set to “0”. However, if a PBDR control bit was set to “1” the associated pin generated a logic level of “1”. A potential of 3.3V defined a logic level of “1”. This behavior was only true for pins that were configured as outputs. If a pin was configured as an input pin the corresponding bit value reflected the potential value applied to the pin when the register was read. It was
possible to overwrite the PBDR control bit of an input pin. However, the value was only transferred to the PBDR register and not to the input pin.

During this research, the PBDR register was used to control the outputs associated with the green LED visual alarm, the red LED visual alarm, the audio alarm and the blood line clamp. When air was not present in the blood, the only active output was the green LED of the visual alarm. Therefore, only output pin 9 was set to “1” and the PBDR register was assigned bit vector “100000000”. This condition is presented in the register formats presented in Table 4.4.

Table 4.4: PBDR Register When No Air was Detected in the Blood

| GPIO PIN | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| PBDR Control Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Value    | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

When air was detected in the blood the three remaining output pins became active, pin 9 was deactivated and the PBDR register received bit vector “111000000000”. Table 4.5 the register format for the PBDR register when air was detected.

Table 4.5: PBDR Register When Air was Detected in the Blood

| GPIO PIN | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| PBDR Control Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Value    | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

The third register associated with Port B is the Interrupt Register. PBINT is a 16 bit control register used to set the capability of the lower 8 GPIO pins, (PB7 though PB0), in order to trigger an interrupt. An interrupt is a technique used in real-time
programming, which consists of interrupting the normal activity of the processor when a certain event occurs. An interrupt signal forces the processor to execute a specific set of instructions and return, when complete, to the state where it left. The PBINT register configures the lower GPIO inputs in such a way that a rising or falling transition on the pins can trigger an interrupt routine. PBINT control bits are divided into Interrupt Mask bits, (PBINT bit 15 through bit 8), and Interrupt Invert bits, (PBINT bit 7 through bit 0). Each interrupt mask bit is associated with a lower GPIO input pin and can enable or disable the pin to generate an interrupt. A value of “0” assigned to an interrupt mask bit disables the corresponding pin for interrupt generation. A value of “1” enables the pin to generate an interrupt. The interrupt invert bits are used to individually program whether a rising or falling transition is to be detected on each pin. A value of “0” assigned to an interrupt invert bit allows the associated GPIO input pin to generate an interrupt when a rising edge transition, (from logic level 0 to logic level 1), is detected on that pin. Similarly, a value of “1” permits the generation of an interrupt when the corresponding GPIO input during a falling edge transition, (from logic level 1 to logic level 0). This process is crucial for the air detection algorithm and it will be described in detail in Chapter 5. During this research, PB0 was programmed to generate an interrupt for both rising and falling edge transitions of the Air Detected input signal. Tables 4.6 and 4.7 present the values assigned to the PBINT in both cases.

<table>
<thead>
<tr>
<th>GPIO INPUT PIN</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBINT Control Bit</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 4.7: PBINT Register for Detection of the Falling Edge Transition on PB0

<table>
<thead>
<tr>
<th>GPIO INPUT PIN</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBINT Control Bit</td>
<td>Interrupt Mask Bits</td>
<td>Interrupt Invert Bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Tables 4.6 and 4.7 indicate that, in order to generate an interrupt on the rising edge transition of the PB0 input signal, the PBINT has to receive the bit vector “100000000”. Similarly, assignment of the bit vector “100000001” to the PBINT allows the GPIO input pin PB0 to trigger an interrupt on the falling edge transition of its input signal.

4.2.2 Clock Synthesis

Two main clocks are used by the DSP56824EVM board to drive the DSP core and the peripheral circuits. The main clocks are the Oscillator Clock and the Phi Clock. The oscillator clock derives its clock signal from an external crystal and runs at 3.6864MHz. The phi clock generates its clock signal from the oscillator clock. The phi clock can actually produce a higher frequency than the Oscillator Clock due to an on-board Phase Locked Loop, (PLL). The maximum frequency supported by the DSP56824 board is 70MHz, which is obtained by forcing the PLL to generate a clock signal with a frequency 19 times higher than the oscillator clock’s frequency, (19*3.6864MHz=70 MHz). Generally, a higher clock signal translates into higher time resolution, which is a very important feature when dealing with algorithms involving timing such as the one
developed during this research. Figure 4.14 presents the block diagram of the On Chip Clock Synthesis Module.

![Block Diagram of the On-Chip Clock Synthesis Module](image)

The Clock synthesis is controlled by PLL control registers PCR0 and PCR1. The first register is a 16-bit-wide register that holds the binary value of the PLL multiplier + 1. Only 10 bits, PCR0 bit 14 through bit 5, of the register are programmable. The remaining bits are reserved bits for future compatibility and are automatically written with 0s. Since the 70MHz clock signal was required the PLL multiplier was chosen to be 19. Therefore, PCR0 received the binary value of 20, (10100₂), in bits 14 through 5. Table 4.8 presents the PCR0 format for this situation.

<table>
<thead>
<tr>
<th>PCR0 Control Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Reserved Bits for future compatibility.

Theoretically, the PCR0 register bits 14 through 5 can be set to “1111111111”, which yields a PLL multiplier factor of 1023. However, this is not recommended since the DSP processor cannot physically run faster than 70MHz. A PLL multiplier of 1023
would force the DSP board to attempt to run at 3.771GHz, \((1023 \times 3.6864 \text{ MHz}) = 3771 \text{MHz}\).

The second register, PCR1, enables or disables the PLL. To enable the PLL bit 14, (PLL Enable), and bit 13, (PLL Power Down), of the PRC1 register must be set to “1” and “0” respectively. Bits 10 through 8, (PS bits), control the prescaler clock, which is another clock signal that can be generated from the DSP board. For convenience, these three bits were set to “010” even though the prescaler clock was not used. Bit 3, (VCS), allows the Voltage Controlled Oscillator, (VCO), to be optimized for 40 to 70MHz operation when the bit is set to “0”. If bit 3 is set to “1” the VCO is optimized for 10 to 40MHz operation. Since the operating frequency was 70MHz, bit 3 was cleared. The remaining bits were not critical or reserved and were set as depicted in the format presented in Table 4.9.

<table>
<thead>
<tr>
<th>PCR1</th>
<th>Control Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Reserved Bits for future compatibility.

During this research the DSP Board was programmed to synthesize a 70MHz clock signal that provided a time base for the DSP board timers. In reality, the timers received a clock signal whose frequency was \(\frac{1}{4}\) of the phi clock. The following section explains in detail the functionality of the timers used during this research. The topic of timing will be discussed in Chapter 5.
4.2.3 Timers

In order to measure the pulse width of the Air Detected signal it was necessary to keep track of time using the DSP board timer module. The DSP56824 provides three independently programmable 16-bit timer/event counters, which are termed Timer0, Timer1 and Timer2. All three timers can be clocked with the Phi-Clock/4, a Prescaler Clock or external signals from Timer I/O pins, (TIO01 and TIO2). Timer1 and Timer2 can also be clocked respectively by overflow events of Timer0 and Timer1. However, the overflow of Timer2 cannot be cascaded to another register. However, Timer2 can trigger an interrupt. The input stage of each timer has an exclusive-OR gate followed by a 4-to-1 multiplexer that permits the selection of the clock signal. The timer module contains eight read/write registers, which are located in the X memory. Each register has two sets of 16-bit registers. One set of bits is for the preload register, (TPR[x]), and the other set is for the count register, (TCT[x]). Timer0 and Timer1 share one 16-bit register, TCR01, with 8 bits assigned to each timer. Timer2 uses only 8 bits of the 16 bit register TCR2. Figure 4.15 presents the block diagram of the Timer Module.
Every time the output signal from a multiplexer transitions from low to high, the corresponding count register decrements the value loaded by the preload register. When the count registers reach zero an overflow signal is generated, which can trigger an interrupt or enable another timer. Afterwards, the count register reloads the value stored by the preload register and resumes its countdown. The reload and countdown sequence events are repeated until either the preload value is clear or the timer is disabled through bit 15 and bit 7 in TCR01 and bit 7 in the TCR2, which are called Timer Enable bits, (TE). TE bits must be set to “1” to enable the corresponding timer. Bits 9 and 8 of TCR01 determine the MUX input, which is to be applied to Timer1. Similarly, bits 1 and 0 for both TCR01 and TCR2 control the clock signal for Timer0 and Timer2. These two
bits are called Event Select bits, (ES). Table 4.10 catalogs the clock source/Event Select bits association.

**Table 4.10: Clock Source/Event Select Bits Association**

<table>
<thead>
<tr>
<th>Event Select Bits</th>
<th>Clock Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Internal Phi Clock/4</td>
</tr>
<tr>
<td>01</td>
<td>Internal Prescaler Clock</td>
</tr>
<tr>
<td>10</td>
<td>Previous Timer Overflow from Timer1 or 2</td>
</tr>
<tr>
<td>11</td>
<td>External Event from TIO pin</td>
</tr>
</tbody>
</table>

The remaining bits were not of concern since they were not used during this research. Table 4.11 and Table 4.12 present the TCR01 and TCR2 register formats and their control bits.

**Table 4.11: TCR01 Register Format**

<table>
<thead>
<tr>
<th>TCR01 Control Bit</th>
<th>Timer1</th>
<th>Timer0</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>TE</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>ES</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>ES</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>TE</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ES</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>ES</td>
<td>0</td>
</tr>
</tbody>
</table>

*Reserved Bits for future compatibility.

**Table 4.12: TCR2 Register Format**

<table>
<thead>
<tr>
<th>TCR2Control Bit</th>
<th>Reserved</th>
<th>Timer2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>ES</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>ES</td>
</tr>
</tbody>
</table>

*Reserved Bits for future compatibility.

This arrangement enables the programming of each register to cycle through a fixed interval, which is determined by the size of the preload value. The frequency of the clock signal determines the count register decrement rate.

The timers can run independently or they can be cascaded to increase time resolution that can range from nanoseconds to seconds. During this research Timer1 and Timer0 were used in a cascaded configuration. Timer2 was not utilized during this
research. Since Timer0 was clocked by the Phi Clock/4, the count register started
decrementing when bit 7 of TCR01 was set to “1”. Every time the count register reached
0 an overflow signal was generated and the count register, TCT0, was reloaded with the
value set in the preload register, TPR0. Since Timer1’s Event Select bits were set to
“10”, Timer1’s count register started decrementing every time there was a low to high
transition from Timer0’s overflow signal and bit 15 on TCR01 was set to “1”. Therefore,
the “Timer1&Timer0 Timing module” could be activated if bit vector
“1000010100000000” was assigned to TCR01. Table 4.13 illustrates how to activate the
Timer1&Timer0 Timing module.

Table 4.13: TCR01 Register for Activation of the Timer1 & Timer0 Timing Modules

| TCR01 Control Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|
| Value             | 1  | 0  | 0  | 0  | 0  | 0  | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Reserved Bits for future compatibility.

The deactivation of the Timing module was accomplished by simply assigning bit
vector “0000000000000000” to the TCR01 register, which stopped the decrementing
process in both TCT1 and TCT0.

4.2.4 Generation of Interrupts

To Motorola DSP5824 uses a programming technique called Interrupt Generation
in order to handle events asynchronously. This method allows an event to interrupt the
normal operation of the DSP processor and to force it to execute a different set of
instructions called an Interrupt Service Routine, (ISR). Once the task, called by the
interrupt, has been completed the processor resumes operation at the point where it was
interrupted. Interrupts can be either internally generated, such as the overflow signal from a timer, or externally triggered. The IRQA and IRQB push-buttons are examples of ways to produce externally generated interrupts. Figure 4.16 presents a diagram of the activity that ensues when an interrupt occurs.

![Figure 4.16: Interrupt Service Routine Diagram](image)

The processor initially executes the main program. At any time in the program an interrupt can be triggered by internal or external events. When the processor recognizes an interrupt, it completes the current instruction in the main program while it performs a check of the priority of the interrupt. The priority level of an interrupt allows the DSP processor to arbitrate pending interrupt requests and select the one to handle first in case two or more interrupts are generated at the same time. The processor recognizes two interrupt priority levels, which are designated as level 1, (IPL1), and level 0, (IPL0). Level 1 interrupts have the highest priority and are always executed before any level 0
interrupts. Both IPL1 and IPL0 are enabled or disabled via a register called the Interrupt Priority Register, (IPR). The Interrupt Priority Structure is presented in Tables 4.14 and 4.15.

Table 4.14: Interrupt Priority Structure; Level 1
Priority Level 1, (Non-maskable)

<table>
<thead>
<tr>
<th>Interrupt Source</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware Reset</td>
<td>Highest</td>
</tr>
<tr>
<td>COP watchdog Timer Reset</td>
<td>↓</td>
</tr>
<tr>
<td>Illegal Installation Trap</td>
<td>↓</td>
</tr>
<tr>
<td>Hardware Stack Overflow</td>
<td>↓</td>
</tr>
<tr>
<td>OnCE Module Instruction Trap</td>
<td>↓</td>
</tr>
<tr>
<td>Software Interrupt (SWI)</td>
<td>Lowest</td>
</tr>
</tbody>
</table>

Table 4.15: Interrupt Priority Structure; Level 0
Priority Level 0, (Maskable)

<table>
<thead>
<tr>
<th>Interrupt Source</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRQA Push-Button</td>
<td>Highest</td>
</tr>
<tr>
<td>IRQB Push-Button</td>
<td>↓</td>
</tr>
<tr>
<td>Synchronous Serial Interface</td>
<td>↓</td>
</tr>
<tr>
<td>Reserved</td>
<td>↓</td>
</tr>
<tr>
<td>Timer Module</td>
<td>↓</td>
</tr>
<tr>
<td>Serial Peripheral Interface 1</td>
<td>↓</td>
</tr>
<tr>
<td>Serial Peripheral Interface 0</td>
<td>↓</td>
</tr>
<tr>
<td>Real Time Timer</td>
<td>↓</td>
</tr>
<tr>
<td>Port B GPIO</td>
<td>Lowest</td>
</tr>
</tbody>
</table>

When an interrupt is to be processed the processor freezes the Program Counter, (PC), which determines the address of the next instruction to be executed. The PC and Status register, (SR), which describe the status of the processor, are pushed onto the stack in order to provide a return address and the processor condition at the moment of interruption. When the processor finishes executing the ISR the return address will be
used to return to the point in the main program where it left off. In order to process the interrupt the processor maps the appropriate interrupt vector, which is located in the lower area of the program memory, $P$, of the DSP56824. The interrupt vector points to a set of memory locations. Each pair of the set of memory locations corresponds to a particular interrupt. The first word of the interrupt vector corresponds to a Jump to Subroutine, (JSR), instruction. The second word contains the memory address of the interrupt handler. The PC only fetches the handler’s address while the interrupt controller supplies the JSR instruction. Table 4.16 presents the association of the main interrupts and their corresponding interrupt vectors.

Table 4.16: Main Interrupt Sources and their Interrupt Vectors

<table>
<thead>
<tr>
<th>Interrupt Source</th>
<th>Interrupt Starting Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware RESET</td>
<td>$0000</td>
</tr>
<tr>
<td>IRQA</td>
<td>$0010</td>
</tr>
<tr>
<td>IRQB</td>
<td>$0012</td>
</tr>
<tr>
<td>Port B GPIO interrupt</td>
<td>$0014</td>
</tr>
<tr>
<td>Real-time interrupt</td>
<td>$0016</td>
</tr>
<tr>
<td>Timer0 overflow</td>
<td>$0018</td>
</tr>
<tr>
<td>Timer1 overflow</td>
<td>$001A</td>
</tr>
<tr>
<td>Timer2 overflow</td>
<td>$001C</td>
</tr>
</tbody>
</table>

After resolving the interrupt priority arbitration, the processor places the JSR into the instruction stream so that it is fetched next and releases the PC. Arbitration among any other pending interrupts is permitted at this stage. This continuing arbitration allows an executing interrupt routine to be interrupted by a higher priority interrupt. When the
processor completes all the ISR instructions it uses the PC and SR to return to the main program.

The Interrupt Priority Register, (IPR), is responsible for the determination of which DSP peripherals can generate interrupts. The SR register enables these interrupts. The IPR is a 16-bit register whose format is presented in Table 4.17.

Table 4.17: Bit Configuration of the ISR Register

| Bit name | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|----------|--------|--------|--------|--------|--------|--------|------|------|------|------|------|------|------|------|------|
| GPIO     | Timer  | SPI0   | SPI1   | Module | Reserved| SSI   | *    | *    | B    | IBL1 | IBL0 | IBINV| IAL1 | IAL0 | IAINV|

*Reserved Bits for future compatibility.

Two interrupt sources were used during this research. The first was Port B GPIO and the second was the IRQB push-button. The GPIO was used to trigger interrupts on every low to high or high to low transition of the Air Detected signal from the air detector module. The IRQB push-button was used to reset Timer1 and Timer0 registers and to clear all the outputs generated by the GPIO after an air detection occurred. This reset process stopped the audio alarm, unclamped the blood line clamp and allowed the visual alarm to turn green. Therefore, only bit 15 and bits 5 through bit 3 of the ISR register were of concern. If bit 15 was asserted, the Port B GPIO lower inputs were allowed to generated interrupts. Bits 5 through bit 3 were reserved for the IRQB push-button. In order to trigger an interrupt when the IRQB button was pushed required bit 5 through bit 3 to receive bit vector “110”. The IBL0 bit enabled the IRQB push-button while IBL1 and the IB-INV bits made the interrupt falling edge sensitive. Tables 4.18 and 4.19
present respectively the IPR register format required to enable Port B GPIO interrupt
generation and IRQB push-button interrupt generation.

**Table 4.18: IPR Register for Port B GPIO Interrupt Generation**

<table>
<thead>
<tr>
<th>IPR Control Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Reserved Bits for future compatibility.

**Table 4.19: IPR Register for IRQB Push-Button Interrupt Generation**

<table>
<thead>
<tr>
<th>IPR Control Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Reserved Bits for future compatibility.

The status register enabled both non-maskable, (IPL1), and maskable, (IPL0),
interrupts. Bit 9 and bit 8, which are termed Interrupt Mask bits, enabled or disabled
respectively IPL1 and IPL0 interrupts. Since both Port B GPIO and IRQB were IPL0
interrupts, the only bit to be asserted was bit 8. Table 4.20 presents the format for the SR
register and its control bits.

**Table 4.20: SR Register to Enable IPLO Interrupts**

<table>
<thead>
<tr>
<th>SR Control Bit</th>
<th>Mode Register</th>
<th>Condition Code Register</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>#</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>Value</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

*Reserved Bits for future compatibility.

#Bits of not concern for the purpose of Interrupt Generation.
5.1 Metrowerks CodeWarrior Development Environments

The Metrowerks CodeWarrior IDE is software used to program the DSP56824EVM board. This software is used in combination with the Motorola Embedded Software Development Kit, (SDK), to develop, test and debug DSP applications. The SDK library provides a multitude of useful DSP functions to simplify the development process. The CodeWarrior software and the DSP56824 hardware communicate via a parallel cable that connects the evaluation board to the host PC. The program is written in a mixed language that comprises C and Assembly language. The CodeWarrior IDE has an extensible architecture that uses plug-in compilers and linkers to specifically target the DSP56824 board. After the code is compiled the linker links together all the files required by a project. The Linking process insures that each sub program communicates properly with the other elements in the project. After successful compilation and linkage of the code the program can be downloaded to the DSP program memory via the parallel port for code debugging. CodeWarrior offers many features that facilitate debugging. One of the most important features is the breakpoint, which allows the programmer to halt code execution and manually step in or out of the code. This
capability results in an easier and more effective debugging process. Figure 5.1 presents
a picture of the CodeWarrior IDE Debugging Window.

![CodeWarrior IDE Debugging Window](image)

Figure 5.1: CodeWarrior IDE Debugging Window

### 5.2 Air Detection Algorithm

The objective of the Air Detection algorithm, developed during this research, was
to measure the pulsewidth of the *Air Detected* signal, which was generated by the Air
Detector Module. The value of the pulsewidth was used to control activation of the
blood line clamp and trigger the alarms. Section 5.3 presents a general overview of the
algorithm and the processes performed by the DSP processor.

Initially the program sets Port B GPIO to support interrupt generation on input pin
1 and activates the green LED of the visual alarm. Next, a 70 MHz clock signal is
synthesized and applied to DSP’s internal timers. After the timers are activated all
preload and count registers of Timer1 and Timer0 are initialized and enabled for falling
edge transition interrupts on Port B GPIO input pin1. When these initialization activities
are completed, the program places the processor in a while loop to wait until an external interrupt occurs. Figure 5.2 flowcharts the main program processes.
When a falling edge transition causes an interrupt to be generated the program jumps to the Interrupt Service Routine, (ISR), associated with the GPIO. The ISR causes the processor to check whether the global variable FLAG is cleared or asserted. Since FLAG is initially loaded with “0”, the program starts the Timer Module. The PBINT register is modified to allow the detection of a rising edge transition on GPIO pin1 while the status of FLAG is changed from “0” to “1”. The program exits the ISR and waits. In the meanwhile, the timer module keeps counting until a rising edge transition interrupt on pin 1 occurs. When the interrupt is generated the program jumps again to the ISR. This time, the timer module is stopped since variable FLAG is asserted. The program uses the values stored in Timer1 and Timer0’s count registers and the clock frequency to calculate the time resolution and the time duration of each timer. This information is utilized to calculate the Total Elapsed Time, which is equivalent to the pulsewidth of the Air Detected signal. All preload and count registers are reset, PBINT is modified to recognize falling edge transition interrupts on GPIO pin 1 and variable FLAG is cleared. The processor compares the Total Elapsed Time with a fixed value of 0.003 seconds, which represents the pulse width threshold. If the Total Elapsed Time is greater than the fixed value, the DSP processor activates three logic-high outputs of the GPIO. The three logic-high outputs trigger the red LED of the visual alarm, the audio alarm and the blood line clamp. Simultaneously, GPIO interrupt generation is deactivated, which leaves only the IRQB push button interrupt enabled. If Total Elapsed Time is less than the fixed value the program exits the ISR and waits for a new falling edge transition interrupt. Figure 5.3 flowcharts the Port B Interrupt Service Routine operations.
Figure 5.3: Port B Interrupt Service Routine Flow Chart

The IRQB interrupt was used to release the Air Detector system from the alarm status. When the IRQB push button is pressed, the three GPIO outputs, which are applied to the visual alarm, audio alarm, and the blood line clamp are deactivated. The green LED of the visual alarm is activated. All preload and count registers are reset, PBINT is modified to recognize falling edge transition interrupts on GPIO pin 1, and variable \textit{FLAG}
is cleared. The processor exits the Interrupt Service Routine and waits for a new falling edge transition interrupt to be generated. Figure 5.4 flow charts the IRQB Push-button Interrupt Service Routine operations.

Figure 5.4: IRQB Push-button Interrupt Service Routine Flow Chart
5.2.1 Registers Settings

The DSP56824 has 128 words of on-chip program ram that are reserved for on-chip peripheral registers. The hexadecimal addresses for this ram span $FFC0 to $FFFF. Table 5.1 catalogs the registers used during this research and the corresponding hexadecimal memory addresses.

Table 5.1: On-Chip Peripheral Registers and their Memory Addresses

<table>
<thead>
<tr>
<th>Register</th>
<th>Memory Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt priority register, (IPR)</td>
<td>$FFFB</td>
</tr>
<tr>
<td>Bus control register, (BCR)</td>
<td>$FFF9</td>
</tr>
<tr>
<td>PLL control register 1, (PCR1)</td>
<td>$FFF3</td>
</tr>
<tr>
<td>PLL control register 0, (PCR0)</td>
<td>$FFF2</td>
</tr>
<tr>
<td>Port B data register, (PBDR)</td>
<td>$FFEC</td>
</tr>
<tr>
<td>Port B data direction register, (PBDDR)</td>
<td>$FFEB</td>
</tr>
<tr>
<td>Port B Interrupt register, (PBINT)</td>
<td>$FFEA</td>
</tr>
<tr>
<td>Timer control reg. 1 &amp; register 0, (TCR01)</td>
<td>$FFDF</td>
</tr>
<tr>
<td>Preload register for timer 1</td>
<td>$FFDC</td>
</tr>
<tr>
<td>Preload register for timer 0</td>
<td>$FFDE</td>
</tr>
<tr>
<td>Count register for timer 1</td>
<td>$FFDB</td>
</tr>
<tr>
<td>Count register for timer 0</td>
<td>$FFDD</td>
</tr>
</tbody>
</table>

The program used a `#include` instruction to call a subroutine that defined each register and its corresponding memory address. This allowed the processor to match the register name, which can be defined by the programmer, to the physical register location and to create predefined pointers to the registers for read/write purposes. To set a value in a register, the pointer to that particular register must be called and set equal to the desired value. Since all the registers were 16-bit wide, hexadecimal notation was used. For example, if pin 1 of the GPIO was to be set for output, the Port B Data Direction Register must be written with bit vector “000000000000000001”, which is equivalent to
$0001. Since the \#include subroutine defines the Port B Data Direction Register to be at location $FFEB as “PBDDR”, a pointer to the register called “*PBDDR” was created. In order to write $0001 into the Port B Data Direction Register the instruction

*PBDDR=0X0001

was used.

As mentioned in Chapter 4, several registers had to be configured initially. The main program performed the initialization function. A simplified version of the main program is presented as Code 5.1.

```
int main(void)
{
    int i;
    *PBDDR = 0x0F00;
    *PBDR = 0x0400;
    *PBINT = 0x0101;
    *IPR = 0x8000;
    *PCR0 = 0x280;
    *PCR1 = 0x4208;
    //Enable all levels of interrupts
    asm(bfset #$0100,sr);
    asm(bfclr #$0200,sr);
    *TCR0 = 0;
    *TPR0 = 0xFFFF;
    *TPR1 = 0xFFFF;
    *TCTR0 = 0xFFFF;
    *TCTR1 = 0xFFFF;
    while(i)
    {
        asm(nop);
        asm(nop);
    }
    return 0;
}
```

Code 5.1: Simplified Version of the Main Program
The PBDDR register is set to 0x0F00 to enable four outputs on the Port B GPIO.
The PBDR register is loaded with $0400 in order to activate the green LED of the visual alarm while the PBINT and IPR registers are configured so that GPIO pin 1 can trigger on a falling edge transition interrupt. PCR0 and PCR1 are set in such a way that a 70 MHz clock signal for the timers’ registers is synthesized. The preload and count registers for both Timer1 and Timer0 are loaded with an initial value of $FFFF. The Timing module is initially turned off via TCR01. Using Assembly Language bit manipulation, the Status Register is accessed and modified to enable all level interrupts. After execution of these preliminary register setup procedures, the processor enters a “while loop” and waits for an interrupt event to occur. The complete version of the main program is presented in Appendix A.

5.2.2 GPIO Interrupt Service Routine

In order to initialize the GPIO Interrupt Service Routine, the program is required to write the ISR interrupt vector into the interrupt vector table. To accomplish this step an assembly language routine labeled *pmemwrite*() was written. The prototype for *pmemwrite*() is:

```c
void pmemwrite(WORD value_V, WORD address_ADD);
```

The *pmemwrite*() function writes the 16 bit “value_V” at memory location “address_ADD”. This function is called four times. The first two times the function is called to write the JSR instruction, (0xE9C8), at 0x0012 and the name of the ISR at 0x0013 for the IRQB Interrupt. The last two times the function is called to write the JSR instruction, (0xE9C8), at 0x0014 and the name of the ISR at 0x0015 for the Port B GPIO
interrupt. The need of this function was dictated by the fact that Program memory could not be accessed via pointers. The code presented as Code 5.2 the operation of the Setup ISR using the `pmenwrite()` function.

**Code 5.2: Setup Interrupt Service Routine**

```c
//setup Interrupt Service Routines (ISRs)

pmemwrite((WORD)0xE9C8,(WORD)0x0012);  //IRQB Interrupt
pmemwrite((WORD)Irqb_ISR,(WORD)0x0013);

pmemwrite((WORD)0xE9C8,(WORD)0x0014);  //Port B GPIO Interrupt
pmemwrite((WORD)ISR,(WORD)0x0015)
```
5.2.3 Air Detector Input Signal Detection

The Air Detector Input signal Detection method was developed in accordance with the state diagram presented in Figure 5.5.

![State Diagram for the Air Detector Input Signal Detection Method](image)

Figure 5.5: State Diagram for the Air Detector Input Signal Detection Method

The state diagram of Figure 5.5 shows that there are five distinct states, which the processor might enter. In state 1, the processor enters an infinite loop and does not
perform any operation. While in state 1 the processor waits for an interrupt event to occur. This state is governed by the detection of an interrupt event from the Port B GPIO input and the status of variable \textit{FLAG}. Since there are two inputs four possible events can occur. Table 5.2 details the input combinations and the corresponding \textit{Next States}.

<table>
<thead>
<tr>
<th>Interrupt Event Detection</th>
<th>Flag Status</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>0</td>
<td>State 1</td>
</tr>
<tr>
<td>FALSE</td>
<td>1</td>
<td>State 1</td>
</tr>
<tr>
<td>TRUE</td>
<td>0</td>
<td>State 2</td>
</tr>
<tr>
<td>TRUE</td>
<td>1</td>
<td>State 3</td>
</tr>
</tbody>
</table>

Regardless the status of \textit{FLAG}, if an interrupt event on the GPIO is not detected the processor stays in the same state and waits for an external event. If an interrupt is generated and \textit{FLAG} is set to “0” the processor goes to State 2. However, the next state will be State 3 if \textit{FLAG} is set to “1”. All four input combinations simply affect the next state and no outputs are produced.

In State 2 the processor expects a Falling Edge Transition on GPIO pin 1. If this condition is true the next state will be State 3 and output \( Z_1 \) is generated. \( Z_1 \) is a set of instructions for the purpose of:

- Resetting preload and count timers,
- Starting the timer module,
- Turning off the green LED of the visual alarm,
- Modifying PBINT register to allow rising edge transition interrupt generation on GPIO pin 1,
- Asserting variable \textit{FLAG}. 

69
The code presented as Code 5.3 reflects the output of State 2.

Code 5.3: Output of State 2

```c
    case 0:
    for(j = 0; j < 150;j++)                 //Short Delay in nanoseconds range
        {asm(nop);}
    *TPR0 = 0xFFFF;                        //Timer0 Preload Register Reset
    *TPR1 = 0xFFFF;                        //Timer1 Preload Register Reset
    *TCTR0 = 0xFFFF;                       //Timer0 Count Register Reset
    *TCTR1 = 0xFFFF;                       //Timer1 Count Register Reset
    *TCR01 = 0x8280;                       //Start Timer Module
    *PBINT = 0x0100;                       //Rising Edge Transition Enable
    *PBDR = 0;                                  //Green LED Disable
    FLAG = 1;
    break;
```

If a falling edge transition interrupt is not generated the processor returns to State 1 and no output is generated.

In State 3 the DSP processor expects the reception of a rising edge transition interrupt. If a rising edge transition interrupt is generated State 4 becomes the next state and Z2 is produced. The following operations are executed in response to Z2:

- Timer Module is stopped,
- *Total Time Elapsed* is calculated and printed on the console,
- Preload and Count registers are reset,
- Falling Edge Transition interrupt generation is enabled via PBINT.

The code presented as Code 5.4 reflects the output of State 3.
As described in Chapter 4, the Phi Clock/4 is synthesized in order to provide the timer module with a time base. Since the Phi Clock runs at 70 MHz, the Phi Clock/4’s frequency is 17.5 MHz. Since Timer0 is clocked by the Phi Clock/4 signal, the timer decrements every 57.143 ns, \((T = 1/f = 1/17.5 \text{ MHz})\). This value is stored in \(\text{Timer0\_resolution}\). If the Timer0 Count Timer is loaded with \$FFFF\, it requires 65535 clock transitions before resetting and generating an overflow signal. In terms of time, the time elapsed for a complete Timer0’s cycle is 3.7448 ms, \((65535 \times 57.143 \text{ ns})\), which corresponds to \(\text{Timer0\_duration}\). Therefore, Timer1 decrements by one every 3.7448 ms since Timer0 is cascaded to Timer 1.

Multiplying TCTR0’s counts, \((\text{low\_count})\), by \(\text{Timer0\_Resolution}\) and TCTR1’s value, \((\text{high\_count})\), by \(\text{Timer0\_Duration}\) yields \(\text{Time\_Elapsed}\), which is given by:

\[
\text{Time\_Elapsed} = (65535 - \text{high\_count}) \ast \text{timer0\_duration} + (65535 - \text{low\_count}) \ast \text{timer0\_resolution}.
\] (5.1)
If a falling edge transition occurs when the current state is State 3 the processor returns to State 1. However, due to the nature of the Air Detect Signal, it is not possible to obtain two consecutive falling edge transition interrupts. If such a situation occurs the processor can be taken to an Error State or simply forced to return to State 1, which restarts the entire sequence.

State 4 performs a comparison between the Total Time Elapsed variable and a fixed value equal to 0.003. If Total Time Elapsed is greater than the fixed value, the processor enters State 5 and outputs Z₃, which performs the actions:

- Activate Blood Line Clamp,
- Activate Audio and RED LED Visual alarms,
- Modify IPR register to disable Port B GPIO interrupt and enable IRQB interrupt,
- Set variable FLAG to “0”.

If Total Time Elapsed is less than 0.003 the processor returns to State 1 and only the green LED of the visual alarm is activated. The code presented as Code 5.5 describes the behavior of the processor in State 4.

```
Code 5.5: Behavior of State 4

if (time_elapsed > 0.003s) {
    printf("Alarm\n");
    *PBDR = 0x0B00;  //Activate alarms outputs
    *IPR=0x0012;     //Disable GPIO interrupt and enable IRQB
}
else
{
    *PBDR = 0x0400;  //Activate Green LED
    FLAG=0;
}
break;
```
In State 5, the DSP processor stops and waits for the operator to stop the alarms by pressing the IRQB pushbutton. Activation of the IRQB pushbutton initiates execution of the operations:

- Preload and Count Timers are reset,
- The IPR register is modified to enable Port B GPIO interrupts and disable the IRQB interrupt,
- Falling Edge Transition interrupt generation is enabled via PBINT,
- The Blood Line Clamp is deactivated,
- The Audio and red LED Visual Alarm are deactivated,
- The green LED of the Visual Alarm is activated.

If IRQB is not pressed no further operations are executed.

The code presented as Code 5.6 describes the operations executed in State 5.

```
Code 5.6: Behavior of State 5

for(I = 0; I < 0x40000; I++)          //Add delay to debounce the switch
{
    asm(nop);
}
*TPR0 = 0xFFFF;         //Timer0 Preload Register Reset
*TPR1 = 0xFFFF;         //Timer1 Preload Register Reset
*TCTR0 = 0xFFFF;        //Timer0 Count Register Reset
*TCTR1 = 0xFFFF;        //Timer1 Count Register Reset
*PBINT = 0x0101;         //Falling Edge Transition Enable
*IPR = 0x8000;              //Enable GPIO interrupt and disable IRQB
*PBDR = 0x0400;          //Activate Green LED
```

Notice that a short delay is added to debounce the IRQB switch. This concludes the discussion of the Software. The complete program code is presented in Appendix A.
CHAPTER 6

EXPERIMENTAL RESULTS

A system validation test and a system performance test were conducted to verify the functionality of the system developed during this research. Bovine blood was used as a replacement for human blood, during testing. The choice of bovine blood was dictated by the fact that the viscosity and density of bovine blood are similar to those in human blood. Additionally, bovine blood and human blood have comparable hematocrit levels, which are defined as the measure of the proportion of blood volume occupied by red blood cells. Bovine blood was slowly raised to body temperature, \((37 \pm 1) ^\circ C\), using a warm water bath that was heated by a common laboratory heater. A peristaltic pump was employed to produce a blood flow of 300mL/min. Figure 6.1 provides a picture of the bovine blood test setup.
6.1 System Validation Test

The purpose of the system validation test was to ensure that the DSP based Air Detector System consistently, reliably and accurately detected air bubbles in the blood. The system was tested using 40, 60, 80 and 100µL air bubbles. For each air bubble size, three air bubbles were injected. The objective of the test was to compare the percent error between the pulsewidth of the analog signal generated by the Air Detector Module and the pulsewidth calculated by the DSP board. The success criteria for this test stated that the percent error, for each air detection, had to be less than 1%. An unsuccessful comparison for any of the 12 test runs would automatically deem the entire system validation test to be a failure. For these tests an oscilloscope, set in Trigger Mode, was connected to measure the Air Detector Module’s output signal. The DSP algorithm was programmed to output, on the console, the value of the `time_elapsed` variable. Therefore, each air bubble injection event yielded two comparable values. The values compared were the oscilloscope measurement and the system pulsewidth.
The system validation test, using bovine blood, was performed without anomalies. Data were recorded and percent errors were calculated. Table 6.1 presents the data and results of the system validation test. The percent errors for each of the 12 test trials were within 1%. Therefore, the DSP based Air Detector System successfully passed the validation test.

<table>
<thead>
<tr>
<th>Oscilloscope Pulse (ms)</th>
<th>System Pulse (ms)</th>
<th>Percent error %</th>
<th>Pass/Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>40 µL air bubble</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUN 1</td>
<td>2.44</td>
<td>2.454</td>
<td>0.57377049</td>
</tr>
<tr>
<td>RUN 2</td>
<td>3.1</td>
<td>3.123</td>
<td>0.74193548</td>
</tr>
<tr>
<td>RUN 3</td>
<td>2.54</td>
<td>2.563</td>
<td>0.90551181</td>
</tr>
<tr>
<td>60 µL air bubble</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUN 1</td>
<td>4.3</td>
<td>4.324</td>
<td>0.55813953</td>
</tr>
<tr>
<td>RUN 2</td>
<td>4.28</td>
<td>4.298</td>
<td>0.42056075</td>
</tr>
<tr>
<td>RUN 3</td>
<td>4.86</td>
<td>4.893</td>
<td>0.67901235</td>
</tr>
<tr>
<td>80 µL air bubble</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUN 1</td>
<td>5.9</td>
<td>5.934</td>
<td>0.57627119</td>
</tr>
<tr>
<td>RUN 2</td>
<td>6.26</td>
<td>6.292</td>
<td>0.51118211</td>
</tr>
<tr>
<td>RUN 3</td>
<td>6.4</td>
<td>6.383</td>
<td>0.265625</td>
</tr>
<tr>
<td>100 µL air bubble</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUN 1</td>
<td>7.16</td>
<td>7.152</td>
<td>0.11173184</td>
</tr>
<tr>
<td>RUN 2</td>
<td>7.22</td>
<td>7.235</td>
<td>0.20775623</td>
</tr>
<tr>
<td>RUN 3</td>
<td>7.34</td>
<td>7.325</td>
<td>0.20435967</td>
</tr>
</tbody>
</table>

6.2 System Performance Test

The system performance test was executed in order to collect data to study the performance of the system with different sizes of air bubbles. The analysis was required in order to verify that the system would successfully stop, via the blood line clamp, air bubbles larger than 60µL. The test was performed with bovine blood at body temperature and a blood flow rate of 300mL/min. A series of air bubbles, ranging from 10 to 100µLs,
were injected inside the blood tubing and the pulsewidths measured by the DSP algorithm were recorded. For each air bubble size, ten air bubble injections were performed and the corresponding pulsewidth measurements were recorded. The complete data are presented in Appendix C.

The data collected allow the generation of several plots, which were used to analyze system performance. The first plot was an XY-Scatter plot. In this plot, each measurement is represented by a data point whose X-value represents the air bubble size and the Y-value is the measured pulsewidth. Since ten injections were performed for each air bubble size, the plot displayed a series of sequences of 10 points, which were plotted vertically. This type of graph indicates the consistency and reliability of the measurements. For example, if ten measurements are collected by injecting ten 50µL air bubbles and the plot shows a sequence of points scattered vertically, the measurements will be inconsistent. Such a plot could arise due to a system failure.

Analysis of the scatter plot of system performance revealed that each air bubble size yielded sequences of data points, which were uniformly distributed around a center point, with no scattered points. Therefore, the system developed during this research was considered to be consistent and reliable. Figure 6.2 presents the XY-Scatter plot of system performance.
In order to display the data distribution as a function of air bubble size, pulsewidth and test run, a three dimensional column plot was generated. The column plot provides an intuitive view of the data distribution. The column plot generated from the system performance test data demonstrated that the system was not capable of detecting 10 and 20µL air bubbles. However, the system detected 30µL air bubbles six out of ten times. This behavior was considered acceptable since the air detector module’s piezo ceramic was calibrated to reliably detect air bubbles larger than 40µL. In addition, the plot shows that the pulsewidth becomes gradually larger as the air bubble size increases. These phenomena are illustrated by the column plot of system performance presented in Figure 6.3.
Figure 6.3: 3D Column Plot of System Performance

A bar graph was generated, from the system performance data, which displayed the average of the ten test runs for each air bubble size. The bar at the top of each column, in Figure 6.4, indicates the standard deviation of the group. The bar graph display of system performance clearly indicates a linear relationship between air bubble size and pulsewidth. Since 30µL air bubbles were detected six out of ten times, the standard deviation for this data set was visibly higher. Figure 6.4 presents the Average and Standard Deviation Plots of system performance.
The strong correlation between air bubble size and pulsewidth was also observed in the Line plots of the average of the ten test runs for each air bubble size, which were generated from the system performance data. By intersecting the data curve with a horizontal straight line, which represents the 3ms threshold, it is possible to predict the theoretical air bubble size associated with the 3ms threshold. During these tests the threshold air bubble size was 45µL. This means that 45µL air bubbles are likely to produce pulsewidths of 3ms. The Line plots of system performance are presented in Figure 6.5.
Data associated with the frequency of activation of the blood line clamp and the triggered alarms was gathered for each air bubble size. A percentage value was calculated, as the ratio of the number of times the measured pulsewidth was larger than 3ms to the sample size of 10 test runs, for each air bubble size. The calculations indicated that the system could not halt air bubbles smaller than 30µL. The chances of stopping 40 and 50µL air bubbles were 40% and 70% respectively. The uncertainty was due to the fact that 40 and 50µL air bubbles produce pulsewidths that are very close to the 3ms threshold. However, the system successfully stopped all air bubbles greater than or equal to 60µL. The data demonstrates that the system performed in accordance with the specifications. Figure 6.6 presents the plots of system performance with respect to detection percentages related to air bubble sizes.
Figure 6.6: Detection Percentage Plot of System Performance
CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

This research addressed the design of a DSP based air detector system to prevent air embolism during hemodialysis therapy. The research successfully demonstrated that the air detector system developed was capable of reliably detecting 60µL or larger air bubbles in blood. The experimental results demonstrated that the system was reliable, fast and accurate for detecting air bubbles. With respect to the software perspective, implementation of the DSP algorithm to detect air bubbles was demonstrated to be highly effective as shown by the system validation test data. The average percent error was found to be less than 1%. With respect to the hardware viewpoint, the DSP56824 was found to be very suitable for accurately measuring the pulsewidth of the signal generated by the Ultrasound Air Detector. Even though the system was feasible from the design point of view, the DSP based Air Detector System developed in this research was not cost-effective due to the high cost of a DSP board and the CodeWarrior IDE software. Therefore, the manufacturability of the system in large scale on hemodialysis machines is not suggested. However, the system can be used as a test fixture to test air detector modules before their installation in a hemodialysis machine.
Future investigations should incorporate the development and implementation of a test fixture to simultaneously test multiple air detector modules. The fixture designed during this research could be used to test the reliability and the performance of air detector modules before they are installed in hemodialysis machines. Future investigations should also consider the idea of developing a mechanical sub-system for breaking down and/or dissolving a large bubble into smaller ones when air is detected in blood and the blood line clamp is activated. Mechanical vibration could be an appropriate approach for breaking up large air bubbles. The vibration should be strong enough to break the air bubble but, at the same time, not strong enough to damage biological structures in the blood such as red and white blood cells.
REFERENCES


APPENDICES
# Appendix A

## Air Detector DSP Code

```c
#include <stdio.h>
#include "56824Registers.h"

/*Global variables*/
int flag = 0;

/*Prototypes for our interrupt handlers*/
void ISR(void);     //IRQA interrupt handler
void Irqb_ISR(void);

int main(void)
{
    int i;

    //setup Interrupt Service Routines (ISRs)
    pmemwrite((WORD)0xE9C8,(WORD)0x0014); //Write JSR instruction
    pmemwrite((WORD)ISR,(WORD)0x0015);  //ISR address
    pmemwrite((WORD)0xE9C8,(WORD)0x0012); //Write JSR
    pmemwrite((WORD)Irqb_ISR,(WORD)0x0013); //IRQB's ISR address

    *PBDDR = 0x0F00;  //Set PBDDR register. pin1: input; pin9 thru 12: outputs
    *PBDR = 0x0400;   //Enable Green LED of visual alarm
    *IPR = 0x8000;  //Enable GPIO interrupt by setting the INTERRUPT PRIORITY REGISTER:
                    //binary 1000000000000000 = hex 8000*/

    *PBINT = 0x0101;    //Falling edge interrupt enabled
    *BCR = 0;               //Initialize BCR for zero wait states
    *PCR0 = 0x0280;    //3.6864 MHz * 19 = 70 MHz
    //Enable PLL using the oscillator clock
    *PCR1 = 0x4208;    //Enable PLL
    printf("Clock running at %3f MHz\n", 19* 3.6864);  //Print Phi clock's frequency
    for(i1 = 0;i < 0x1fff;i++)   //Delay to meet lock spec
```

88
Appendix A: (Continued)

{  
    asm(nop);  
    asm(nop);  
    asm(nop);  
}

for(i = 0;i < 0x1fff;i++) //Delay to meet lock spec
{
    asm(nop);  
    asm(nop);  
    asm(nop);  
    asm(nop);  
}

//Enable all levels of interrupts
asm(bfset #$0100,sr);  
asm(bfclr #$0200,sr);  

//Preload and Count Timers initialization
printf("Initializing timers\n");

*TCR01 = 0;   //Turn off Timing module
*TPR0 = 0xFFFF;  //Timer0 Preload Register Reset
*TPR1 = 0xFFFF;  //Timer1 Preload Register Reset
*TCTR0 = 0xFFFF;  //Timer0 Count Register Reset
*TCTR1 = 0xFFFF;  //Timer1 Count Register Reset

//Processor sits in an infinite loop waiting for an Interrupt event to occur
while(i)
{
    asm(nop);  
    asm(nop);  
}

return 0;

}  //end main()

void ISR(void)
#pragma interrupt

{  
    //Variable Declarations

    WORD low_count;
    WORD high_count;
    float time_elapsed;
    float clock;
    float timer0_resolution;
    float timer0_duration;
    int j;
    
    return 0;

}  //end ISR()
switch(flag)    //check variable flag status
{
    case 0:
        for(j = 0; j < 150; j++)   //nanoseconds delay
        {
            asm(nop);
        }
        *TPR0 = 0xFFFF;  //Timer0 Preload Register Reset
        *TPR1 = 0xFFFF;  //Timer1 Preload Register Reset
        *TCTR0 = 0xFFFF;  //Timer0 Count Register Reset
        *TCTR1 = 0xFFFF;  //Timer1 Count Register Reset
        *TCR01 = 0x8280;  //Start Timing module
        *PBINT = 0x0100;  //Rising Edge Transition Enable
        *PBDR = 0;   //Green LED Disable
        printf("Timer on\n");
        break;
    case 1:
        *TCR01 = 0x0000;  //Stop Timing module
        //Read Timer0 and Timer1 count value.
        low_count = (WORD)*TCTR0;
        high_count = (WORD)*TCTR1;
        clock = (3.6864 *19)/4;  //Calculate Phi Clock/4 frequency
        timer0_resolution = 1/clock * 0.000001;  //Calculate Timer0 Resolution
        timer0_duration = 0xFFFF * timer0_resolution;  //Calculate Timer Duration
        time_elapsed = (float)(65535-high_count)* timer0_duration +
        (float)(65535-low_count)* timer0_resolution;  //calculate time_elapsed
        printf("Timer off\n");
        printf("Time = %e%s
",time_elapsed, "sec");
        //Reset Preload and Count Timers
        *TPR0 = 0xFFFF;
        *TPR1 = 0xFFFF;
        *TCTR0 = 0xFFFF;
        *TCTR1 = 0xFFFF;
        *PBINT = 0x0101;
        *PBDR = 0;
Appendix A: (Continued)

if (time_elapsed>0.003)   //Compare to Air Detected Threshold
{
    printf("Alarm\n");
    *PBDR = 0x0B00;  //Activate alarms outputs
    *IPR = 0x0012;   //Disable GPIO interrupt and enable IRQB
}
else
    *PBDR = 0x0400; //Activate Green LED
break;
} //end switch structure

flag = (flag == 0) ? 1: 0;   //Toggle Flag
for (j = 0;j < 1000;j++)   //Short delay
{
    asm(nop);
}

} //end ISR()

void Irqb_ISR(void)
{
    #pragma interrupt
    long i;

    //Add delay to debounce the switch
    for(i = 0;i < 0x40000;i++)
    {
        asm(nop);
    }

    //Reset Preload and Count Timers
    *TPR0 = 0xFFFF;
    *TPR1 = 0xFFFF;
    *TCTR0 = 0xFFFF;
    *TCTR1 = 0xFFFF;
    *PBINT = 0x0101;    //Falling Edge Transition Enable
    *IPR = 0x8000;    //Enable GPIO interrupt and disable IRQB
    *PBDR = 0x0400;   //Activate Green LED
}

} //end Irqb_ISR()
Appendix B

“56824Register.h” Code

#define IPR  (WORD*)0xFFFB  //Interrupt priority register
#define BCR  (WORD*)0xFFF9  //Bus control register
#define PCR1 (WORD*)0xFFF3   //PLL control register 1
#define PCR0 (WORD*)0xFFF2   //PLL control register 0
#define PCD  (WORD*)0xFFEF  //Port C data register
#define PCDDR (WORD*)0xFFEE  //Port C data direction register
#define PCC  (WORD*)0xFFED  //Port C control register
#define PBDR  (WORD*)0xFFEC //Port B data register
#define PBDDR (WORD*)0xFFEB //Port B data direction register
#define PBINT (WORD*)0xFFEA //Port B Interrupt register

//Timer defines
#define TCR1  (WORD*)0xFFDF  //Timer control register 1
#define TCR2  (WORD*)0xFFDA  //Timer control register 2
#define TPR0  (WORD*)0xFFDE  //Preload register for timer 0
#define TPR1  (WORD*)0xFFDC  //Preload register for timer 1
#define TPR2  (WORD*)0xFFD9  //Preload register for timer 2
#define TCTR0 (WORD*)0xFFDD  //Count register for timer 0
#define TCTR1 (WORD*)0xFFDB  //Count register for timer 1
#define TCTR2 (WORD*)0xFFD8  //Count register for timer 2
Appendix C

System Performance Test Data

Table C.1: 10 to 50 µL Air Bubble Injections

<table>
<thead>
<tr>
<th>Temp</th>
<th>Flow Rate</th>
<th>Temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>37° C</td>
<td>300 mL/min</td>
<td>37.1 C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.454</td>
<td>3.942</td>
</tr>
<tr>
<td>RUN 2</td>
<td>0</td>
<td>0</td>
<td>2.112</td>
<td>3.123</td>
<td>3.421</td>
</tr>
<tr>
<td>RUN 3</td>
<td>0</td>
<td>0</td>
<td>1.823</td>
<td>2.563</td>
<td>3.84</td>
</tr>
<tr>
<td>RUN 4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.343</td>
<td>2.927</td>
</tr>
<tr>
<td>RUN 5</td>
<td>0</td>
<td>0</td>
<td>1.995</td>
<td>2.866</td>
<td>3.529</td>
</tr>
<tr>
<td>RUN 6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3.555</td>
<td>3.234</td>
</tr>
<tr>
<td>RUN 7</td>
<td>0</td>
<td>0</td>
<td>2.002</td>
<td>2.123</td>
<td>3.123</td>
</tr>
<tr>
<td>RUN 8</td>
<td>0</td>
<td>0</td>
<td>2.393</td>
<td>2.982</td>
<td>2.993</td>
</tr>
<tr>
<td>RUN 9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3.453</td>
<td>3.523</td>
</tr>
<tr>
<td>RUN 10</td>
<td>0</td>
<td>0</td>
<td>1.732</td>
<td>3.342</td>
<td>2.534</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>0</td>
<td>0</td>
<td>1.2057</td>
<td>2.8804</td>
<td>3.3066</td>
</tr>
<tr>
<td>STAND. DEV.</td>
<td>0.0000</td>
<td>0.0000</td>
<td>1.0520</td>
<td>0.4956</td>
<td>0.4315</td>
</tr>
<tr>
<td>% Detection</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>40%</td>
<td>70%</td>
</tr>
</tbody>
</table>

Table C.2: 60 to 100 µL Air Bubble Injections

<table>
<thead>
<tr>
<th></th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN 1</td>
<td>4.324</td>
<td>5.234</td>
<td>5.934</td>
<td>6.793</td>
<td>7.152</td>
</tr>
<tr>
<td>RUN 3</td>
<td>4.893</td>
<td>6.324</td>
<td>6.383</td>
<td>6.902</td>
<td>7.325</td>
</tr>
<tr>
<td>RUN 4</td>
<td>3.793</td>
<td>6.001</td>
<td>6.021</td>
<td>6.829</td>
<td>6.834</td>
</tr>
<tr>
<td>RUN 5</td>
<td>3.932</td>
<td>5.766</td>
<td>6.091</td>
<td>6.552</td>
<td>6.992</td>
</tr>
<tr>
<td>RUN 6</td>
<td>4.29</td>
<td>5.231</td>
<td>5.713</td>
<td>5.991</td>
<td>7.532</td>
</tr>
<tr>
<td>RUN 7</td>
<td>4.217</td>
<td>5.372</td>
<td>5.923</td>
<td>6.199</td>
<td>7.239</td>
</tr>
<tr>
<td>RUN 9</td>
<td>3.937</td>
<td>4.992</td>
<td>6.012</td>
<td>6.625</td>
<td>7.592</td>
</tr>
<tr>
<td>RUN 10</td>
<td>4.124</td>
<td>5.783</td>
<td>6.241</td>
<td>6.462</td>
<td>7.623</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>4.1631</td>
<td>5.51014</td>
<td>6.1103</td>
<td>6.5547</td>
<td>7.2526</td>
</tr>
<tr>
<td>STAND. DEV.</td>
<td>0.3257</td>
<td>0.4482</td>
<td>0.2387</td>
<td>0.2936</td>
<td>0.2692</td>
</tr>
<tr>
<td>% Detection</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

93