A SOFTWARE SYSTEM FOR
REAL-TIME CARDIAC
ACOUSTIC MAPPING

Dany Leong-Kon

Department of Electrical Engineering
McGill University
Montreal, Quebec, Canada

March 1997

A thesis submitted to the faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Engineering

0-612-29609-1

ABSTRACT

This thesis describes a real-time cardiac acoustic mapping system (RTCAMS) which displays the amplitude of the phonocardiogram (PCG) from the surface of the thorax through an acoustic probe composed of an array of 25 microphones. The essential components of the RTCAMS are the data acquisition, the graphic computation and display of the PCG envelopes and the acoustic maps. These processes are performed in parallel by distributing them in 2 digital signal processors (DSPs) on an add-on board in a personal computer (PC). Parts of the software for the DSPs are written in assembly language to improve the overall speed of the system. A real-time micro-kernel is implemented in the second DSP to improve the use of the DSP resources. As the RTCAMS uses the Microsoft™ Windows 3.1 operating system for its graphical user interface on a PC, a virtual device driver was developed to speed up the data transfer between the host PC and the DSPs and a double buffering technique was used to support real-time display of the graphics. Recording from 3 normal subjects was used to test the performance of the system. The preliminary results show that the system can display the spatial distribution of the PCG on the thorax at the speed of 33 frames per second, allowing the study of propagation patterns of the PCG in real time.

Résumé

Ce mémoire décrit un système de cartographie cardiaque sonore capable de calculer et d'afficher en temps réel la distribution spatiale de l'amplitude du phonocardiogramme (PCG) capté sur la surface du thorax à l'aide d'une sonde comprenant une matrice de 25 microphones. Les composantes principales du système sont l'acquisition des données, la production des cartographies cardiaques et des graphiques des enveloppes des PCGs ainsi que leur affichage. Ces composantes sont écrites en assembleur pour améliorer la performance du système et sont effectuées en parallèle par 2 processeurs spécialisés (DSPs) situés sur une carte insérée dans un ordinateur personnel. Un micro-noyau en temps réel est implanté sur le second DSP pour améliorer l'usage des ressources du DSP. Comme le système d'exploitation Windows 3.1 de Microsoft™ est utilisé pour son interface graphique, un algorithme a été implanté pour accélérer le transfert des données entre l'ordinateur hôte et les DSPs et une méthode de double tampon de mémoire a été utilisée pour supporter un affichage en temps réel (33 images par seconde). Des enregistrements ont été effectués sur 3 sujets normaux pour vérifier l'efficacité du système et sa capacité d'effectuer l'acquisition des PCGs, le calcul et l'affichage des graphiques des signaux et des cartographies en temps réel. Les résultats préliminaires obtenus sont très encourageants et démontrent l'utilité de la cartographie cardiaque sonore pour visualiser et étudier les patrons de radiation et de propagation du PCG sur la surface du thorax.

ACKNOWLEDGEMENTS

I would like to thank my supervisors, Dr. Louis-Gilles Durand and Dr. Howard C. Lee, for their patient guidance and advice throughout my Master's degree research and studies, and in writing the thesis.

I sincerely thank Michel Cozic for fruitful technical advice, Gelu Ionescu for writing part of the low-level driver software, and Jocelyn Durand for technical advice on the realization of the system hardware.

I am grateful to Xuan Zhang and Dr. Herkole Sava for proofreading the entire thesis and giving their suggestion on the system performance.

I would like to acknowledge the general support of the personnel of the Biomedical Engineering Laboratory of the IRCM. I also thank the volunteers who have participated as subjects for the evaluation of the system. I am also indebted to the Natural Sciences and Engineering Research Council of Canada for its financial support.

Finally, I would like to thank my parents and my sisters for their constant love and support. Without them, neither this thesis nor my education would have been achieved.

TABLE OF CONTENTS

Chapter 1

Introduction **1**

1.1 Motivation and Objectives 1

1.2 Thesis Overview 2

Chapter 2

Cardiac Acoustic Mapping System **4**

2.1 Introduction 4

2.2 The Phonocardiogram 5

2.2.1 The Heart Sounds 6

2.2.2 The Heart Murmurs 7

2.2.3 Sound Transmission 8

2.3 Surface Localization of Heart Sounds and Murmurs 8

2.4 Cardiac Acoustic Mapping 9

2.5 Examples of Similar Real-Time Systems 13

2.6 Applications of a Real-Time Cardiac Acoustic Mapping System 16

Chapter 3

The Real-Time System Hardware **18**

3.1 Introduction 18

3.2 Basic Hardware Architecture 18

3.2.1 The DSP Board 21

3.2.2 The Data Acquisition Board 22

3.2.3 The Acoustic Probe, the Analog Multiplexer, and the Analog Demultiplexer/Filter 24
3.2.4 The PC and the Monitor 24
3.3 The PCG Pre-Processing Board 24
3.3.1 The PCG Module 26
3.3.2 The Envelope Extractor, the Analog Switch, and the Sample/Hold Circuit 27
3.4 Integration of the PCG Pre-Processing Board in the System Hardware 27
3.5 Performance of the PCG Pre-processing Board 29

Chapter 4

The DSP Software 32
4.1 Introduction 32
4.2 The System Software Overview 33
4.3 The C40 Module A Software 34
4.3.1 The Main Function of the Software of DSP Module A 35
4.3.2 The Hardware Interrupts 38
4.3.2.1 The Host PC Interrupt 38
4.3.2.2 The C40 Module B Interrupt 40
4.3.2.3 The Data Acquisition Board Interrupt 41
4.4 The C40 Module B Software 43
4.4.1 The Main Function of the DSP B Software 44
4.4.2 The Hardware Interrupts 45
4.4.2.1 The Host PC Interrupt 45
4.4.2.2 The Timer Interrupt 46
4.4.2.3 The DMA Interrupt 47
4.4.3 The Design of the Real-Time Micro-Kernel 47
4.4.4 The Real-Time System Tasks 53
4.4.4.1 The DMA Channel 3 Task 53
4.4.4.2 The DMA Controller Process and the DMA Channel 1 Task 53
4.4.4.3 The PCG Graphic Process 54
4.4.4.4 The Acoustic Mapping Process 55
4.5 The C40 Module D Software 59

4.6 The C40 Module C Software 59
4.7 Conclusion 60

Chapter 5

The Host PC Software **61**

5.1 Introduction 61
5.2 The Windows 3.1 Architecture 62
5.3 The Virtual DSP Driver 64
5.3.1 The VDSPD Services for the RTCAMS Windows Application 66
5.3.2 The VDSPD Services for the DSP Board 67
5.4 The Dynamic Link Library 68
5.5 The RTCAMS Windows Application 69
5.6 Displaying graphics in Window 3.1 74
5.6.1 Fast Animation Technique 75
5.6.2 The Color Palette of the Cardiac Acoustic Map 76
5.6.3 Displaying the PCG Graphics 77
5.7 Conclusion 79

Chapter 6

Test Results **80**

6.1 Introduction 80
6.2 DSP system software 80
6.3 The Overall System 82
6.3.1 Methods 82
6.3.2 Results 83
6.4 Discussion 89

Chapter 7

Conclusion **91**

7.1 Summary 91
7.2 Suggested Future Developments 93

References **95**

Appendix A

The PCG Pre-Processing Board Schematic 99

Appendix B

The Acoustic Probe Schematics 104

Appendix C

The Analog Demultiplexer Schematics 110

Appendix D

The Sensitivity of the 25 Microphones of the Acoustic Probe 115

Appendix E

The PC I/O MAP of the DSP Board 117

LIST OF FIGURES

Fig. 2-1	Conventional auscultation areas, indicating the best transmission of the heart sound components on the surface of the chest. From Ref. 61	9
Fig. 2-2	The recording sites used by Chihara and their schematic representation. From Ref. 7.	10
Fig. 2-3	Mapping image of : (a) a healthy subject, and (b) a patient with myocardial infarction. From Ref. 7.	11
Fig. 2-4	Recording sites on the surface of the chest used by Çozic et al. From Ref. 8.	12
Fig. 2-5	Amplitude map of a normal subject during S1. From Ref. 8.	12
Fig. 2-6	Architecture of the first real-time imaging system developed for EIT. From Ref. 44.	14
Fig. 2-7	Architecture of the second real-time imaging system developed for EIT. From Ref. 15.	15
Fig. 3-1	A real-time cardiac acoustic mapping system	19
Fig. 3-2	Block diagram of the RTCAMS	20
Fig. 3-3	Block diagram of the QPC/C40	21
Fig. 3-4	Block diagram of the data acquisition board	23
Fig. 3-5	DSPLINK address space mapped on the first DSP module	23
Fig. 3-6	Block diagram of the PCG pre-processing board	25
Fig. 3-7	Block diagram of the PCG module	27
Fig. 3-8	Block diagram of the envelop extractor, the analog switch, and the sample/hold circuit	28
Fig. 3-9	Timing relationship between the channel number and the sample-and-hold signal	29
Fig. 3-10	Power spectrum of the data from channel 7 sampled at 5769 Hz, and driven with a full-scale (±3V) 200 Hz signal.	30
Fig. 3-11	Power spectrum of the first channel with grounded input, with a full scale (±3V) 200 Hz sine wave signal applied to the second channel.	31

Fig. 3-12	Power spectrum of the third channel with grounded input, with a full scale (±3V) 200 Hz sine wave signal applied to the second channel.	31
Fig. 4-1	The overall structure and data-flow diagram of the software developed on the DSP board	34
Fig. 4-2	Interactions between the C40 module A and the other devices.	34
Fig. 4-3	The flowchart of the Main Function of DSP A	36
Fig. 4-4	Flow of the cardiac data into the C40 module A	37
Fig. 4-5	Flowchart of the DAB ISR.	41
Fig. 4-6	Timing of the data acquisition and the DAB ISR.	42
Fig. 4-7	Flow of the PCGs and the CAMs in C40 module B.	43
Fig. 4-8	The flowchart of the Main Function of the DSP B	45
Fig. 4-9	The RTMK process descriptor	49
Fig. 4-10	Flowchart of the Scheduler function.	50
Fig. 4-11	Flowchart of the four most often used RTMK handling services.	51
Fig. 4-12	Merging example of one sine wave period. The original sine wave (a) is shrunk to half of its period (b), to a quarter of its period (c), and to a line for merging one full cycle (d).	55
Fig. 4-13	Relative position of each channel in the original 5x5 matrix before interpolation	58
Fig. 4-14	Computation of the two-dimensional Key's cubic convolution interpolation as a series of two one-dimensional interpolations.	58
Fig. 4-15	Flow of the cardiac data in the C40 modules C and D	60
Fig. 5-1	Enhanced Windows environment block diagram.	63
Fig. 5-2	The hierarchy of the different modules supported by the RTCAMS Windows Application	65
Fig. 5-3	Cardiac data transfer path requested by interrupt of the DSP board, and the I/O operation path requested by the RTCAMS Windows Application.	68
Fig. 5-4	RTCAMS Windows Application display.	70
Fig. 5-5	The class diagram of the RTCAMS Windows Application	71
Fig. 5-6	The Device Independent Bitmap structure.	76
Fig. 5-7	Scrolling the contents of the virtual window with the scroll bar of the visible window.	77
Fig. 5-8	The display process of the graphical PCG on the PCG window with t_0 marking the beginning of the acquisition and t_f the end. Two cases are	

	presented: a) the graphical PCG that is displayed when the buffer is not full; and b) when the buffer is full (overwriting the oldest data by the new data acquired and a gap).	78
Fig. 6-1	The three auscultation areas used for recording the CAMs.	83
Fig. 6-2	Arrangement of the microphones on the acoustic probe.	83
Fig. 6-3	The first sixteen PCG envelopes recorded at the apex area of subject #1. The envelopes shows S1 and S2 with the time reference 0 set at the onset of S1. The maximum amplitude of the envelopes is normalized to 1.0.	85
Fig. 6-4	CAMs of the first heart sound of subject #1 recorded at the apex area.	86
Fig. 6-5	CAMs of the second heart sound of subject #1 recorded at the apex area.	87
Fig. 6-6	The CAMs of the second heart sound of subject #2 recorded at the pulmonary area.	88
Fig. 6-7	The CAMs of the second heart sound of subject #3 recorded at the aortic area.	88

LIST OF TABLES

Table 4-1	Service identification number (ID NO.) sent by the host PC to request a service from the DSP A.	39
Table 4-2	Identification number (ID NO.) assigned to the commands sent by the DSP B.	40
Table 4-3	Service identification number (ID NO.) sent by the host PC to the C40 module B.	46
Table 5-1	The VDSPD services for the RTCAMS Windows application	66
Table 5-2	List of functions provided by the DLL	69
Table 5-3	Classes of the RTCAMS Windows Application	70
Table 6-1	Benchmark results for the PCG Graphic Process and the Acoustic Mapping Process.	81

List of Symbols

ADC	: Analog to Digital Converter
kHz	: 1000 Hertz
AMELIA	: Application Module Link Adapter
API	: Application Program Interface
BAR	: Base Address Register
BB	: Burr Brown
CAM	: Cardiac Acoustic Map
CPU	: Central Processing Unit
Control_DLL	: Class Control Dynamic Link Library
CDIBitmap	: Class Device Independent Bitmap
CmainFrame	: Class Main Frame
CmappingApp	: Class Mapping Application
CmappingWnd	: Class Mapping Window
CsignalWnd	: Class Signal Window
CLK	: Clock
DAB ISR	: Data Acquisition Board Interrupt Service Routine
DA ISR	: Data Acquisition Interrupt Service Routine
DEMUX	: Demultiplexer
DIB	: Device-Independent Bitmap
DM	: Diastolic Murmur
DSP	: Digital Signal Processor
DAC	: Digital to Analog Converter
DMA	: Direct Memory Access

DOS	: Disk Operating System
DSP B ISR	: DSP B Interrupt Service Routine
DSP ISR	: DSP Interrupt Service Routine
DLL	: Dynamic Link Library
EIT	: Electrical Impedance Tomography
EEPROM	: Electrically Erasable-Programmable Read-Only Memory
ECG	: Electrocardiogram
GIE	: Global Interrupt Enable
GUI	: Graphical User Interface
GDI	: Graphics Device interface
RGB	: Red Green Blue coded color
Hz	: Hertz
HRG	: High Resolution graphic
ID NO.	: Identification number
JTAG	: IEEE Standard 1149.1 for serial test bus
ISA	: Industrial Standard Architecture Bus
I/O	: Input/Output
IRQ	: Interrupt Request Level
ISR	: Interrupt Service Routine
LSW	: Least Significant Word (16-bit)
LSB	: Least Significant Byte (8-bit)
LIA	: Link Interface Adapter
LSI	: Loughborough Sound Images
MASD	: Maximum Amplitude Signal Detector
MFC	: Microsoft Foundation Classes
MS-DOS	: Microsoft-DOS
MOPS	: Million Operation per Second
MSW	: Most Significant Word (16-bit)
MSB	: Most Significant Word (8-bit)
MUX	: Multiplexer

OO	: Object Oriented Design
PC ISR	: PC Interrupt Service Routine
PC	: Personal Computer
PCG	: Phonocardiogram
PCS	: Process Control Structure
PIC	: Programmable Interrupt Controller
RAM	: Random Access Memory
RTCAMS	: Real-Time Cardiac Acoustic Mapping System
RTMK	: Real-Time Microkernel
S/H	: Sample and Hold
SRAM	: Static Random Access Memory
SM	: Systolic Murmur
TBC	: Test Bus Controller
TI	: Texas Instruments
TISR	: Timer Interrupt Service Routine
VxD	: Virtual Device
VDSPD	: Virtual DSP Driver
VM	: Virtual Machine
VMM	: Virtual Machine Manager
VPICD	: Virtual PIC Driver

Chapter 1

Introduction

1.1 Motivation and Objectives

Phonocardiography is a complementary diagnostic tool for cardiac auscultation and electrocardiography to detect abnormalities of the heart function. In a recent survey by Durand and Pibarot [14], several examples can be found on the application of spectral analysis of the phonocardiogram, and new areas of research and clinical applications of digital phonocardiography are described. This review and a further literature search show that only a few studies on cardiac acoustic mapping have been investigated.

Recently, Cozic et al. [9] designed a system to simultaneously record the phonocardiogram (PCG) envelopes from 22 sites on the surface of the thorax. The envelopes are extracted in real time by 22 envelope detector circuits, sampled at 250 Hz, and stored in a computer disk for off-line processing. Signal processing involves averaging and mapping of the envelopes. The results seem encouraging and promising because the recording made on 3 normal subjects provides information that could be used to localize the origin of the heart sounds and murmurs. The recent developments in low-cost fast data acquisition equipment and computers, and in digital signal processing methods, have made it possible to explore real-time cardiac acoustic mapping.

The basic objective of this project is to realize such a mapping system to display the acoustic maps at 33 frames per second, as the data is acquired. The system will be based on the one proposed by Cozic et al. The mapping rate will be increased above 5 kHz to support a larger frequency range for the heart sounds, the heart murmurs, and especially the clicks of mechanical heart valves. The processing of the PCG envelopes and the acoustic maps will be done on a personal computer (PC) under the Windows 3.1[1] environment to benefit from its graphical user interface. In order to achieve the required mapping speed of 33 frames per second, parallel processing on multiple high-speed digital signal processors (DSPs) in an add-on board for the PC is necessary. The major task of the project is the design and implementation of the software system for the basic objective.

1.2 Thesis Overview

Some essential background knowledge about phonocardiography is first introduced in the next chapter. This will be followed by a description of the surface localization of the heart sounds and murmurs on the thorax. Then, an overview of past research on cardiac acoustic mapping is described, including two real-time systems developed for a different cardiac application. Finally some potential applications of the real-time cardiac acoustic mapping system (RTCAMS) are suggested.

In Chapter 3, an overview of the hardware architecture of the RTCAMS is first presented. It is followed by a brief presentation of its various components, the data

[1] Windows 3.1 is a registered trademark for an operating system developed by the Microsoft Corporation.

acquisition process, and the design and implementation of a pre-processing circuit board as well as its integration in the system hardware.

Chapters 4 and 5 describe the design and implementation of the system software. The major part of the computation is carried out on the DSP board. This is necessary to avoid the host PC of doing any other tasks than displaying graphics and interacting with the user. Chapter 4 focuses on the software for the DSPs of the add-on board (the DSP board). It describes the partition of tasks between the four DSPs and the role of each task. The major tasks are controlling the pre-processing board, computing the PCG envelopes and the acoustic maps. Chapter 5 focuses on the program running on the host PC. It describes the technique used under Windows 3.1 to provide fast graphic display for the acoustic maps and the acoustic signals.

In Chapter 6, the test results of the RTCAMS are presented. It is followed by a description of the procedure used for the recording of the heart sounds from three normal subjects to test the RTCAMS. The limitations and the problems encountered during the development of the system are described, and some suggestions concerning the future improvements are proposed in Chapter 7. The five appendices contain the schematics of the electronic circuits of the system, the sensitivity of the microphones used in the acoustic probe, and the PC I/O map of the DSP Board. The program listings are given in Ref. 27.

Chapter 2

Cardiac Acoustic Mapping System

2.1 Introduction

Auscultation is a widely used noninvasive technique for diagnosis of heart disease. It helps to detect different cardiac malfunctions by listening to the heart sounds and murmurs from the chest wall. The recording of these sounds and murmurs is called the phonocardiogram or PCG, and its analysis provides complementary information about the chronology and frequency content of the heart sounds and murmurs. The digitized PCGs can be further processed, analyzed, and displayed. The PCG is most often analyzed in the time and frequency domains. The resulting information is related to the thoracic area from which the recording is made. By moving the microphone from one point to another on the chest wall, the location of different vibration maxima can be observed. However, phonocardiography has some serious limitations. For instance, the signal obtained is dependent on the force by which the microphone is pressed against the skin, the transmission characteristics of the heart-thorax acoustic system, and the location where the recording is performed.

Cardiac acoustic mapping is a complementary noninvasive method to provide visual information about the spatial distribution of the PCG on the surface of the chest wall and its evolution during the cardiac cycle. The spatial distribution of the PCG is obtained using an array of microphones attached to the body. The combination of the spatial and temporal information can provide a better understanding of the propagation of the PCG through the human chest. It can also provide basic information about the origin of the heart sounds and murmurs and about the propagation medium. The input-output operations of cardiac acoustic mapping are the acquisition of the PCGs and the display of the map, respectively. A RTCAMS provides an image of cardiac related changes (patterns). With a system capable of displaying at a suitable frame rate, it would be possible to observe the radiation and propagation patterns on the chest, and by suitable analysis of the images, to derive indices of cardiac performance on-line. The signal processing aspects of the mapping process are considerable due to the massive quantity of data that needs to be processed in a short time for dynamic display via a monitor.

2.2 The Phonocardiogram

Generally, the PCG is composed of two types of acoustic sounds: the heart sounds and the heart murmurs. In a typical cardiac cycle, up to four heart sounds can be found: The first (S1), second (S2), third (S3) and fourth (S4) heart sounds. S1 and S2 are normally observed in all subjects while S3 and S4 are normal in young persons, but pathological in older adults (>40 years). In addition to these four heart sounds, some other sounds such as opening clicks, snaps, and prolapsed sounds can sometimes be heard in abnormal hearts with valvular stenosis or defective mitral and tricuspid valves. Heart sounds are short duration transient signals while heart murmurs are relatively longer and noise like signals. They are generally caused by dysfunction of the cardiac valves, except for the innocent murmur which may occur during systole in young subjects with a strong normal heart. Heart murmurs are normally separated into two types according to the cardiac cycle chronology: the systolic murmurs (SMs) occurring between S1 and S2, and the diastolic murmurs (DMs) occurring between S2 and S1. The DMs can mask S3 and S4.

2.2.1 The Heart Sounds

Many controversial theories have been proposed so far for the mechanism responsible of the genesis of S1. The most popular and discussed theories can be classified into two basic theories: the valvular theory as supported by Craige and co-worker [10] and the cardiohemic theory proposed by Rushmer [40]. In the valvular theory, the heart sounds are believed to be caused only by the closure and tension of heart valves, while in the cardiohemic theory the cause of the heart sounds is due to the deceleration of the blood mass and the vibration of the heart muscles, and the heart valves do not contribute to the heart sounds. This theory states that the cardiac cavities, the valves, the blood mass, and the heart muscle form a whole complex system, the movement of each part being in unison with the others.

According to the valvular theory, S1 marks the onset of mitral and tricuspid valve closure, therefore there are two bursts of high-frequency vibrations (M1 and T1) in S1 corresponding to each valve closure and vibration. The mechanism responsible for the genesis of S2 is due to the closure and vibration of the aortic and the pulmonary valves which create respectively the aortic (A2) and the pulmonary (P2) components of S2. Similarly, S3 and S4 are created by the halting of the mitral valve leaflets at the end of opening.

On the other hand, the cardiohemic theory recognizes that up to four components may compose the S1 [38]. The first component is produced by the myocardial contraction at the onset of the contraction of the ventricle and is of low amplitude and low frequency. The second component is of higher frequency and is caused by the deceleration of the blood following the closure of the atrioventricular valves. The third component occurs at the time of opening of the aortic valve and is due to the oscillation of blood between the distending root of the aorta and the ventricular wall. The fourth component results from the turbulence of blood ejection in the aortic and pulmonary arteries during the early phase of systole. For S2, the cardiohemic theory supports two components A2 and P2, which however are caused by the deceleration and reversal of flow into the aorta and pulmonary artery, respectively. S3 is generated by the sudden deceleration of blood into the ventricles during the rapid filling phase of the early diastole [14]. S4 is also a ventricular filling sound as S3, but it occurs during the late diastolic filling phase following the contraction of the atria.

2.2.2 The Heart Murmurs

Systolic murmurs are generally classified as systolic ejection and pansystolic regurgitant murmurs [43]. Systolic ejection murmurs are due to the forward flow across an obstruction in the left or right ventricular outflow tract. These abnormalities can be caused by aortic or pulmonic valvular stenosis, for example. A systolic ejection murmur can also be heard in some persons without evidence of physiological or structural abnormalities of the heart. This kind of systolic murmurs is called innocent murmurs. It can be observed in less than half of all children and commonly in adolescent young adults. The pansystolic regurgitant murmurs are caused by retrograde flow from a high pressure chamber into a lower pressure chamber. Mitral and tricuspid regurgitation are typical examples.

Normally, the diastolic murmurs are caused by either a structural abnormality of the atrioventricular and semilunar valves or by increased flow across anatomically normal atrioventricular valves [42]. They can be classified as diastolic filling murmurs (or rumbles) and diastolic regurgitant murmurs. The typical pathological cases of the diastolic filling murmur are mitral stenosis and tricuspid stenosis where the blood is flowing through narrow mitral or tricuspid leaflets at greater than normal velocity. Aortic and pulmonary regurgitations are examples of diastolic regurgitant murmurs.

2.2.3 Sound Transmission

The transmitting media has a strong influence on the transmission of vibrations from the heart to the chest wall. Since the mass of the vibrating material (the heart, blood, and tissues) is important, low frequency sounds predominate in both the production and the transmission. This is unfortunate because the human auditory mechanism is particularly insensitive to low-pitched sounds. The most important loss of heart sound energy occurs in the compressible tissues (lungs) interposed between the heart and the chest wall. Thus the heart sounds and murmurs have maximum intensity in those surface areas to which the vibrations are transmitted directly through dense tissues or through a minimal thickness of the lung. Layers of fat also attenuate the heart sounds and murmurs.

2.3 Surface Localization of Heart Sounds and Murmurs

Clinical experience has well established the locations on the thorax for listening the heart sounds from particular origins. The sounds from the mitral, tricuspid, aortic, and pulmonary valves are better transmitted to their respective areas on the precordium as shown in Fig. 2-1. However, the specific localizations of these areas (in relationship to anatomy) are very difficult to explain on the basis of the transmission of the heart sounds through the thoracic tissues. The area where a specific heart sound is best heard does not correspond to the shortest distance from its source of origin, since it is not the point on the chest surface nearest to the valve. A sound arising in the thorax will reach some sites on the body surface earlier than at the other sites, and the differences between the times of arrival will depend on the specific origin of the sound, on the path which it takes to propagate to the surface, and on the velocity with which it travels.

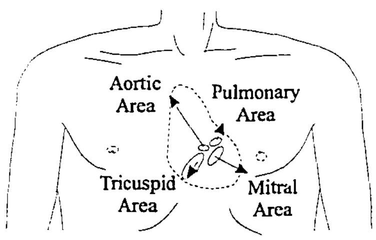


Fig. 2-1 Conventional auscultation areas, indicating the best transmission of the heart sound components on the surface of the chest. From Ref. 61

The systolic murmur originating from the aortic valve can be localized at various points on the precordium along a line paralleling the outflow tract of the left ventricle. The systolic murmur of pulmonary stenosis is heard more loudly in the pulmonary area and is also widely transmitted over the precordium. The mitral insufficiency produces murmurs which can be heard near the apex of the heart.

Diastolic regurgitant murmurs, produced by pulmonary and aortic insufficiency, can be heard in the pulmonary area and near the apex. Diastolic filling murmurs are localized to a very small area at or near the apical region on the precordium and are difficult to hear. The vibrations of these murmurs have a very low frequency and frequently escape detection by auscultation.

2.4 Cardiac Acoustic Mapping

Kozmann and Kenedi first developed a surface acceleration mapping for studying the origin of the first heart sound [25]. The aim of their study was to get the maximal possible information from the chest surface vibration by using a 3-accelerometer recording system, computerized data processing, and display of the results in reference to the chest surface. Ten 20-year old normal male subjects were selected for the study. Chest surface vibration measurements were carried out on the subjects in the supine position and 19 points were registered for the characterization of the vibration field. The accelerometer signals were recorded sequentially in 9 sessions, each session using 3 accelerometers simultaneously. One transducer was always at the same reference point while the other two were systematically changed. The electrocardiogram (ECG) signal was also recorded as the timing reference. The surface acceleration maps were computed and printed at intervals of 4 ms, or a sampling frequency of 250 Hz. Six most characteristic map patterns within a period of 256 ms from the starting of Q wave of the ECG were described in details for explaining the vibration of the heart for different phases of S1.

Later, Chihara et al. also developed a mapping representation of the heart sounds [7]. Twenty-five points on the chest surface were measured sequentially by using two microphones simultaneously. Fig. 2-2 shows the recording sites and their schematic representation. The recording was performed in a soundproof room on 20 healthy subjects and 19 patients. The purpose of this study was to find the difference between the normal and the abnormal hearts by imaging the degree of decrease in the peak frequency of S1. Fig. 2-3 is an example of the results obtained where (a) is the mapping image of a healthy subject and (b) is that of a patient with myocardial infarction.

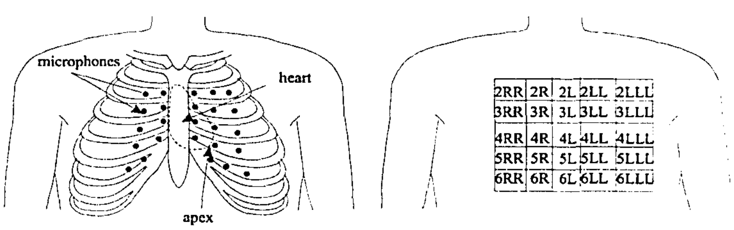


Fig. 2-2 The recording sites used by Chihara and their schematic representation. From Ref. 7.

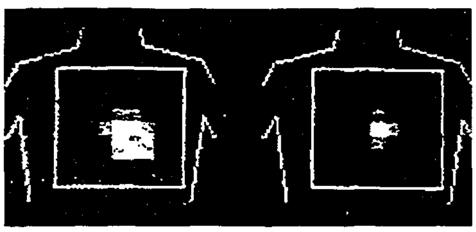


Fig. 2-3 Mapping image of : (a) a healthy subject, and (b) a patient with myocardial infarction. From Ref. 7.

Both studies of Kozmann and Chihara were performed on S1. In 1982, a complete chest wall mapping of the heart sounds and murmurs was developed by Okada [36]. The PCGs were taken from a matrix of 6x6 sites on the thoracic surface. The mapping images were generated from the envelopes of these PCGs by taking the absolute positive value of each signal sample. The recorded PCGs were synchronized by the time reference point of the R wave of the ECG. A sequential representation of the PCG map in each phase of the heart cycle could be displayed on the screen. These maps provided visualization of the changes in the sound distribution and the sites where the sounds and murmurs originate. In their paper, maps of S2 in different phases are illustrated for a healthy 22-year old male subject and for a 28-year old man with ductus arteriosus. It was found that for the normal subject, the two components (A2 and P2) of S2 could be clearly distinguished from each other, as A2 was stretching out to the aortic area and P2 extents to the pulmonary area. For the abnormal case, an intensified P2 was observed, and a murmur originating from a single location inside the ductus was also clearly demonstrated. Okada concluded that the cardiac acoustic maps could be useful not only for diagnosis but also for educational purpose for medical students.

As indicated in Okada's paper [36], the difficulty in acoustic mapping is the simultaneous recording of PCGs from different positions. All previous studies used a few microphones in a sequential recording procedure. The recent study of Cozic et al. [8,9] overcame this limitation by designing a computer-based system for simultaneously recording the PCGs from 22 points on the surface of the chest. A hardware PCG envelope extractor was applied to each PCG before digitization of the envelopes at a sampling interval of 4-ms. An ECG was also acquired as a timing reference signal. Each envelope was averaged over a series of cardiac cycles and interpolated in two dimensions to generate the iso-amplitude maps, which were displayed in pseudo-color. Three normal subjects were selected to study the acoustic maps. Fig. 2-4 shows the recording sites. Fig. 2-5 is an example of the maps obtained during the production of S1. The maps clearly show peak amplitude at the two expected positions (apex and tricuspid area) on the thorax, and they also show the radiation and propagation patterns of S1.

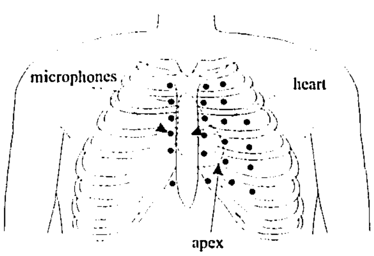


Fig. 2-4 Recording sites on the surface of the chest used by Cozic et al. From Ref. 8.

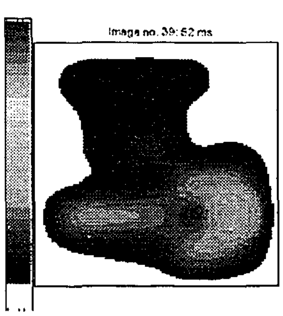


Fig. 2-5 Amplitude map of a normal subject during S1. From Ref. 8.

2.5 Examples of Similar Real-Time Systems

The signal processing aspects of a real-time mapping system are considerable. Due to the massive quantity of information that needs to be visualized as a function of space and time, integrated hardware envelope extraction [8,9], data compression [23], and multiresolution display are required [35,57,58]. However, recent introduction of fast and inexpensive computers, general-purpose data acquisition hardware, and high-speed digital-signal-processing computer add-on boards make it possible for researchers to develop a RTCAMS.

There has been no research on the development of a RTCAMS in the past. However, two real-time electrical impedance tomographic systems have been recently described [15,44]. Electrical Impedance Tomography (EIT) is a relatively new method of medical imaging that produces images based on electrical properties within the human body. Electrical currents are applied to a subject's body using electrodes attached to the skin, and the voltages developed on the electrodes are measured using the tomograph. Reconstruction algorithms use the applied current and the measured electrode voltages to compute the electrical conductivity and permitivity distribution in the body.

EIT resembles closely to the RTCAMS since both are using less than 30 sensors and are dealing with data acquisition and reconstruction images. However. EIT is not limited only to cardiac data, it can be used to measure the conductivity of any physiological volume or organ. Body surface potential mapping is an other type of cardiac mapping obtained by measuring on the torso or on the epicardial surface of the heart the potentials produced by the electrical generator within the heart. However, a large number of electrodes (more than 100) are generally required to produce maps with acceptable error. The computing requirements of these systems are very high [60]. Because the basic requirements of the real-time EIT systems are similar to our RTCAMS, they will be briefly reviewed below.

The common element in RTCAMS or EIT systems is the acquisition, image interpolation and visualization of the acquired cardiac data. Such systems require simultaneous recording from many sensors (microphones or electrodes), processing of the acquired signals, and a display for human visualization and evaluation. Since the volume of data is quite large, the cardiac information must be efficiently presented.

The first real-time EIT system uses an array of 16 electrodes on the surface of the body to reconstruct electrical impedance distribution images at a rate of 25 frames per second, allowing lung ventilation and lung perfusion to be observed in real time [44]. The system hardware consists of four floating point T800 transputers, as shown in Fig. 2-6. The transputer is a microprocessor which has the necessary hardware facilities for designing parallel processing systems. It has special communication links which allow several transputers to be connected together to form a network. As shown in Fig. 2-6, three transputers form a pipeline to perform the reconstruction of the images. The first transputer has two processes running in parallel. The buffer process receives the electrical measurements (voltages and currents) from the data acquisition system and collects them into frames of data. The preprocessor converts the measurements into

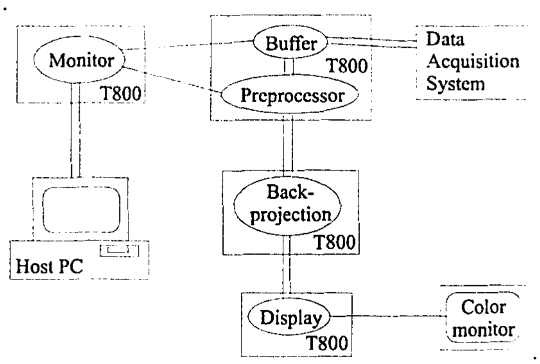


Fig. 2-6 Architecture of the first real-time imaging system developed for EIT. From Ref. 44.

transfer impedances and normalizes them. The results are passed to the second transputer, which reconstructs the image using the backprojection algorithm. It then scales the resulting pixel values in a form suitable for display and transfer them to the third transputer which contains a memory map into which the images are written. The third transputer interpolates between the reconstructed pixel values to enlarge the image which is then displayed on a color monitor. The fourth transputer serves as the interface between the host computer (an IBM PC) and the transputer pipeline. It has access to the host computer disk and receives its commands from the host computer through the user interface.

The second real-time EIT system is based on a commercially available coprocessor board that resides in a personal computer (PC), as shown in Fig. 2-7 [15]. It has 16 thoracic electrodes and reconstructs images at a rate of 18 frames per second. The acquired data are transferred from the data acquisition interface to the Alacron AL860 AT coprocessor board based on an Intel i860 DSP. The image reconstruction process is executed by the AL860 AT coprocessor board which maps the output of the reconstruction algorithm onto the video RAM of the high resolution graphics (HRG)

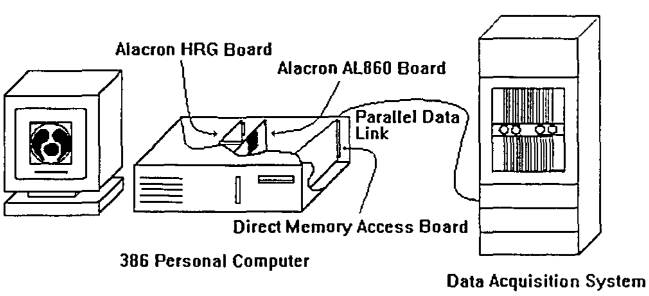


Fig. 2-7 Architecture of the second real-time imaging system developed for EIT. From Ref. 15.

board. The HRG board is a daughterboard to the AL860 AT coprocessor board.

Both EIT systems eliminate transmission of the image data across the slow internal data bus of the host PC by directly sending the reconstructed images to the monitor. The host PC is the interface with the user who controls the system. It also provides data storage facility (disk file) for the real-time system. The EIT software runs apart from the host PC, on a DSP board or network of transputers. The data acquisition hardware were, in both cases, custom designed according to the specific needs of the project. These two EIT systems have demonstrated the feasibility of a real-time imaging system built around a low-cost PC to provide real-time electrical impedance distribution maps.

2.6 Applications of a Real-Time Cardiac Acoustic Mapping System

The introduction of low-cost and high-performance integrated-circuit DSPs enables complex real-time applications of digital phonocardiography like the RTCAMS. One important objective of the RTCAMS is to provide an on-line visual representation of the heart sounds and murmurs on the surface of the thorax, or of transformed images or features, as a complementary tool to auscultation and phonocardiography, as well as for the teaching of auscultation. On-line processing is an important feature for clinical application which gives the ability for the physician to monitor at the same time the activity of the heart using other techniques like cardiac catheterization, intracardiac phonocardiography, and transesophageal Doppler echocardiography.

A RTCAMS could be used for clinical applications and for basic research. Examples of potential applications are:

1. The verification of basic hypotheses on the genesis of the first heart sound.

2. The study of the distribution of native and prosthetic valve sounds on the surface of the thorax.
3. The identification of optimal sites of radiation of stenotic and regurgitant murmurs.
4. The determination of the influence of positioning the microphone array on the reproducibility of the acoustic maps on the thorax.
5. The generation of reference isocontour and isochrone maps of the heart sounds and murmurs on the surface of the thorax for improving the teaching of auscultation.

Chapter 3

The Real-Time System Hardware

3.1 Introduction

This chapter first introduces the hardware architecture of the RTCAMS. Then, the essential information about each component is described in order to provide the basic background required to understand the design and implementation of the software for DSPs and the integration of the pre-processing board into the system hardware.

3.2 Basic Hardware Architecture

Fig. 3-1 shows an illustration of the real-time acoustic mapping system. Basically, it includes seven parts: the acoustic probe, the analog demultiplexer, the PCG pre-processing board, the data acquisition board, the DSP board, the host PC, and the monitor. The acoustic probe, shown at the right of Fig. 3-1, has twenty-five calibrated miniature microphones whose output signals are pre-amplified to provide good signal-to-noise ratios. The twenty-five PCG signals are transferred through a 6-foot flexible cable to the analog demultiplexer and the PCG pre-processing board by using time-division multiplexing. The analog demultiplexer reconstructs the twenty-five PCGs which are then high-pass filtered at 50 Hz by using a third-order Butterworth filters for each PCG. The time-division multiplexing is necessary to allow the use of a small and highly flexible cable between the acoustic probe and the analog demultiplexer to facilitate the handling and the positioning of the acoustic probe on the thorax of the patient by the physician.

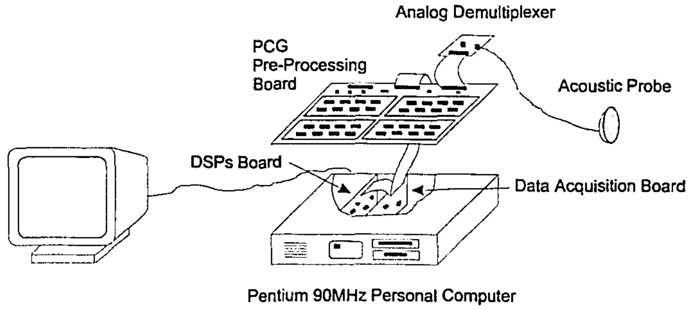


Fig. 3-1 A real-time cardiac acoustic mapping system

Fig. 3-2 shows a block diagram of the system. The PCG pre-processing board, whose basic function is to extract the PCG envelopes, is controlled by software to sample either the PCGs or their envelopes. The sampled PCG or PCG envelopes are then multiplexed, sequentially digitized by the data acquisition board, and stored on the DSP board. The analog multiplexer prior to the data acquisition board and the analog demultiplexer may appear to be redundant, but they were intentionally integrated in the system to provide flexibility for other applications not using the probe as the input device to the RTCAMS. The DSP board processes the PCGs for graphic representations, which can be the PCG signals, their envelopes, or the resulting acoustic maps. The graphic representations are sent to the PC to be displayed on a color monitor. The PC is the user interface of the system and it is used to send initialization parameters and receive processed data for display.

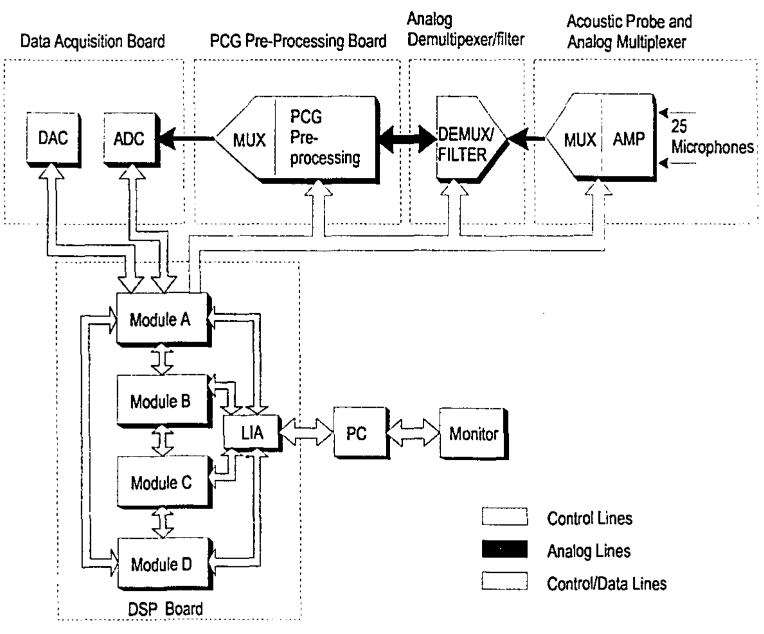


Fig. 3-2 Block diagram of the RTCAMS

The data acquisition board, the DSP board, the PC, and the color monitor were selected from commercially available computer equipment. The acoustic probe, the analog multiplexer, and the analog demultiplexer/filter were designed in the Laboratory of Biomedical Engineering of Institut de Recherche Clinique de Montreal (see Appendix B). The pre-processing board is an improvement and extension of the one developed by

Cozic et al. to support the real-time aspect of the RTCAMS. In this section, each component of the RTCAMS is briefly introduced to provide the basic information needed to understand the implementation of the new PCG pre-processing board and also the system software described in the next two chapters.

3.2.1 The DSP Board

The DSP board is a Spectrum QPC/C40 with four MDC40S modules, each of them based on a TMS320C40 integrated-circuit DSP running at 40 MHz. It uses the full 16-bit interface of the IBM PC/AT or compatible computer. The block diagram of the DSP board is shown in Fig. 3-3. Communication between the PC and the board takes place via the PC bus interface which provides access to the various facilities of the DSP board, such as resets, I/O ports, interrupts, and the Link Interface Adapter (LIA). The LIA enables the PC to communicate with each module via the parallel communication ports of the DSPs. As shown in Fig. 3-3, one parallel communication port from each module (A1, B1, C4, and D4) is routed to the PC bus through the LIA circuitry. Each

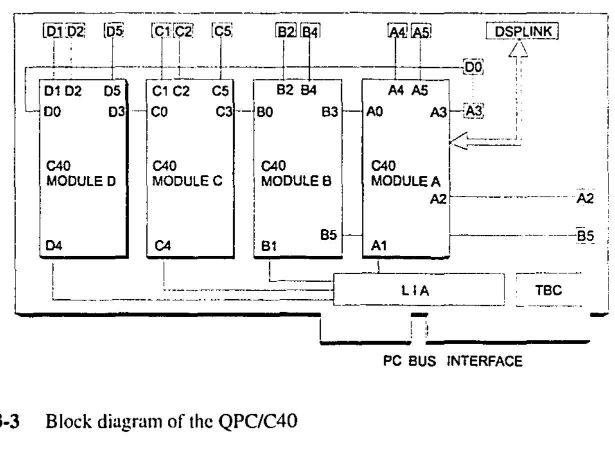


Fig. 3-3 Block diagram of the QPC/C40

module has its address reflected to the PC I/O interface. Data transfer can be requested independently by the four processors and the PC determines which one to service first.

The C40 modules are connected in a ring topology as shown in Fig. 3-3. The module A has access to the digital I/O peripherals of the DSP board via the DSPLINK interface. The DSPLINK is a Loughborough Sound Images (LSI) interface, which consists of a high speed bi-directional bus allowing direct Input/Output (to/from) the DSP without using the I/O bus on the PC. There are two boards that are connected to the DSP board through the DSPLINK interface: the data acquisition board and our customized PCG pre-processing board.

3.2.2 The Data Acquisition Board

The data acquisition board is a daughterboard to the QPC/C40 DSP board [49]. It can support two Burr-Brown (BB) analog modules, but only one is installed in the present board as shown in Fig. 3-4. A BB module carries one two-channel analog-to-digital converter (ADC) and one two-channel digital-to-analog converter (DAC). The ADC has a 16-bit resolution and a maximum sampling rate of 200 kHz. Only one channel of the ADC is used in the present real-time system. The other channel is left for future development. The DAC is not used for this project. The data acquisition board has its I/O mapped into the DSPLINK address space on the DSP module A [47,48], as shown in Fig. 3-5. The board is designed to be physically plugged into an expansion slot of the host PC. The connection, however, is not used to transfer data but for power and ground connections.

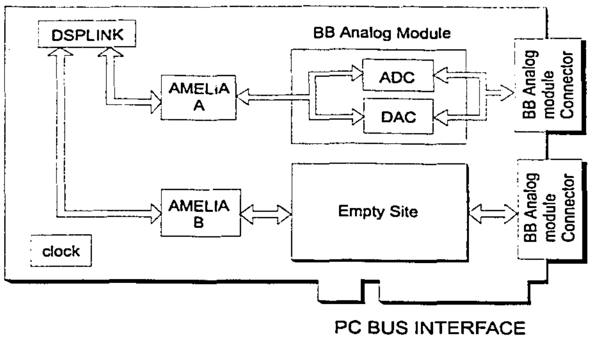


Fig. 3-4 Block diagram of the data acquisition board

Reserved for DSPLINK extension	B000 0000h
Data Acquisition Board	B000 0300h
Not Used	B000 0310h
PCG Pre-Processing Board	B000 0320h
Not Used	B000 0323h B000 03FFh

Fig. 3-5 DSPLINK address space mapped on the first DSP module

3.2.3 The Acoustic Probe, the Analog Multiplexer, and the Analog Demultiplexer/Filter

The acoustic probe is composed of twenty-five microphones disposed in an 5x5 array. The PCGs captured by the microphones are then pre-amplified with a gain of 20, time-division multiplexed, transmitted serially to the analog demultiplexer which reconstructs the twenty-five PCGs. The reconstructed signals are then high-pass filtered at 50 Hz (-3dB) before they are transferred to the PCG pre-processing board. The schematics of these three devices are shown in the Appendix B.

3.2.4 The PC and the Monitor

The PC and the monitor serve as the system hardware interface with the user. The acoustic maps and the acoustic signals are displayed using these two devices. They also allow the user to enter the parameters and the operating commands into the real-time acoustic mapping system. The PC is an IBM compatible with a Pentium processor operating at 90 MHz. The monitor is a 15" color video monitor.

3.3 The PCG Pre-Processing Board

Cozic et al. have developed a PCG pre-processing board that features twenty-two simultaneous sample and hold circuits, each of them coupled with an envelope extractor. However, this pre-processing board did not meet our technical specifications for the following reasons. First, the sample and hold circuit cannot operates at frequency higher than 250 Hz due to the technical limitations of the data acquisition board used. The frequency bandwidth must be extended above 5 kHz to include the high frequency components of the mechanical prosthetic heart valve sounds. Second and most important, the pre-processing board cannot be integrated in our system hardware because of incompatible I/O ports due to the different equipment used.

PCG Pre-processing Board

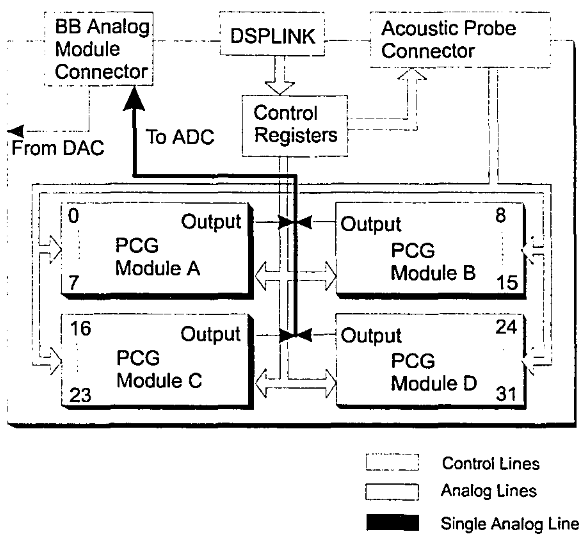


Fig. 3-6 Block diagram of the PCG pre-processing board

The main objectives of the PCG pre-processing board are to extract the envelopes of the twenty-five PCGs, to simultaneously sample and hold the envelopes, and to multiplex them for acquisition. The PCG pre-processing board shown in Fig. 3-6 supports four PCG modules, each of them comprising eight envelope extractors and sample/hold circuits. The pre-processing board circuit is designed to be modular to facilitate the expansion of input channels by increasing the number of sites for the PCG modules, according to the desired number of PCG sensors.

The PCG pre-processing board, like the data acquisition board, has its I/O mapped into the DSPLINK address space on the first DSP module as shown in Fig. 3-5. The PCG pre-processing board has its addresses located from B000 0320h to B000 0322h and they are used as follow :

- B000 0320h : Writing an 8-bit channel number at this location to choose the multiplexed channel for acquisition.
- B000 0321h : Writing an 8-bit command (FFFFh) at this location to reset the analog demultiplexer and the acoustic probe.
- B000 0322h : Writing a '0' at this location to select a PCG signal or a '1' to select a PCG envelope. This is a 1-bit command.

Writing a data into one of the three locations listed above results in storing the data into one of three registers called the control registers of the PCG pre-processing board as shown in Fig. 3-6. The contents of these control registers can affect the PCG modules, the analog multiplexer, the analog demultiplexer/filter, and the acoustic probe (see Fig. 3-2). Changes made to the control registers are directly reflected to the concerned devices. The main circuits of the PCG pre-processing board are the control registers and the DSPLINK interface. Their implementation as well as their utilization are defined in the schematic of the PCG pre-processing board presented in Appendix A.

3.3.1 The PCG Module

Each PCG module supports eight envelope extractors, eight sample/hold circuits, and one analog multiplexer, as shown in Fig. 3-7. Depending on which channel to be multiplexed, only one multiplexer of a PCG module is active to select the appropriate PCG for acquisition. There is an analog switch prior to the sample/hold circuit which selects either the PCG envelope or the PCG to be sampled and held.

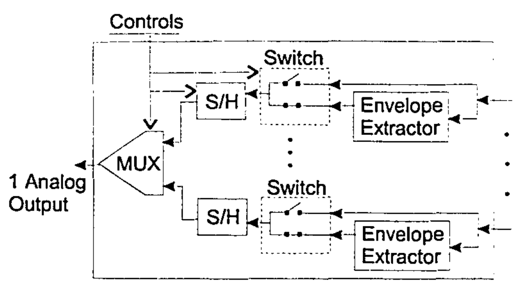


Fig. 3-7 Block diagram of the PCG module

3.3.2 The Envelope Extractor, the Analog Switch, and the Sample/Hold Circuit

This part of the hardware is very similar to the one designed by Cozic et al. [8]. However, there are two differences. First, the discrete components of the sample/hold circuit used as the analog switches were replaced by a specialized integrated circuit to allow a faster and more accurate bi-polar sample/hold function. Second, an analog switch was integrated prior to the sample/hold circuit for choosing between the PCG and its envelope. The block diagram of this part of hardware is illustrated in Fig. 3-8. More details about the implementation of this part of hardware are described in the thesis of Cozic and in Appendix A (for the schematic) of this thesis.

3.4 Integration of the PCG Pre-Processing Board in the System Hardware

The PCG pre-processing board must be synchronized with the data acquisition board to sample the twenty-five channels. However, there are no control lines between these two boards because the data acquisition board does not provide any additional I/O

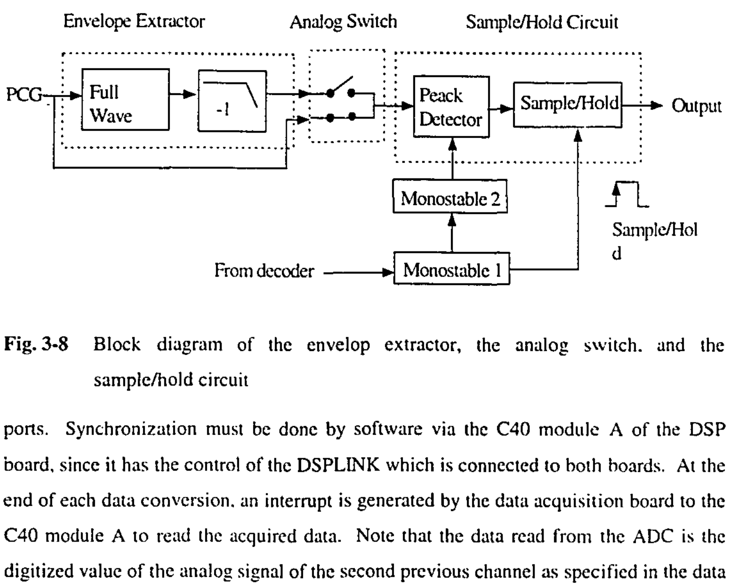


Fig. 3-8 Block diagram of the envelop extractor, the analog switch, and the sample/hold circuit

ports. Synchronization must be done by software via the C40 module A of the DSP board, since it has the control of the DSPLINK which is connected to both boards. At the end of each data conversion, an interrupt is generated by the data acquisition board to the C40 module A to read the acquired data. Note that the data read from the ADC is the digitized value of the analog signal of the second previous channel as specified in the data acquisition board manual [45]. At the same time, the number of the next channel is written to the appropriate control register to digitize the next PCG or PCG envelope.

At the transition from the last channel to the first channel, a sample/hold signal is generated and all sample/hold circuits simultaneously sample and hold the PCGs or the PCG envelopes. The timing relationship between the channel number and the sample/hold signal is shown in Fig. 3-9. An extra channel, channel 0, is inserted between channel 25 and channel 1, and it serves to sample all twenty-five analog signals. Note that data are continuously acquired while the system is started, even for the channel 0 which is ignored by the DSPs. The sample/hold signal is generated by the monostable 1 (see Fig. 3-8) after receiving a signal from a decoder when channel 0 is selected. The sample signal is first generated, and it is followed by the hold signal after 4.6μs.

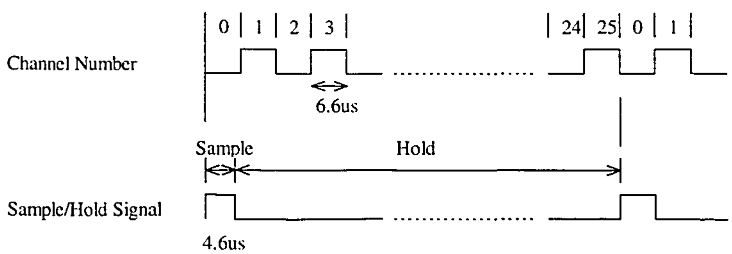


Fig. 3-9 Timing relationship between the channel number and the sample-and-hold signal

3.5 Performance of the PCG Pre-processing Board

Careful attention has been paid to the power supply, grounding arrangements, channels isolation, and acquisition timing to optimize the performance of the PCG pre-processing board. In particular, a sine wave with a frequency of 200 Hz was applied to several channels and an FFT analysis was performed on the digitized sine wave acquired by the data acquisition board. The frequency of 200 Hz was chosen for this test to provide a clear picture of the harmonics on a bandwidth of 3 kHz and to allow the DC component on the power spectrum to be visible. In this section, the result of the test made on channel 7 is presented but the results are similar for the other channels. The input signal, bypassing the envelope detector, was digitized at 5,769 Hz and a total of 62,914 samples was acquired. The signal was analyzed by using a 1024-point rectangular window, with a window shift of 512 samples. A mean spectrum was obtained by averaging the power spectra of all 120 segments. Fig. 3-10 shows the power spectrum of the sampled data of channel 7 driven with a full-scale (±3 V) 200 Hz signal. Note the low (-75 dB) noise floor and the harmonic frequencies at -40 dB or less.

A measure of crosstalk between three input channels of the PCG pre-processing board was also performed. A full scale 200 Hz sine wave was applied to one channel located between two adjacent channels with grounded input. Fig. 3-11 and Fig. 3-12 show the power spectrum of the first and the second grounded channels under test. In both channels, the residual amplitude of the fundamental and harmonic frequencies of the test sine wave has an amplitude below -60 dB.

Fig. 3-10 Power spectrum of the data from channel 7 sampled at 5769 Hz, and driven with a full-scale (±3V) 200 Hz signal.

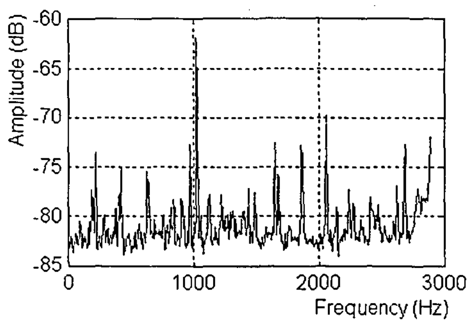


Fig. 3-11 Power spectrum of the first channel with grounded input, with a full scale (±3V) 200 Hz sine wave signal applied to the second channel.

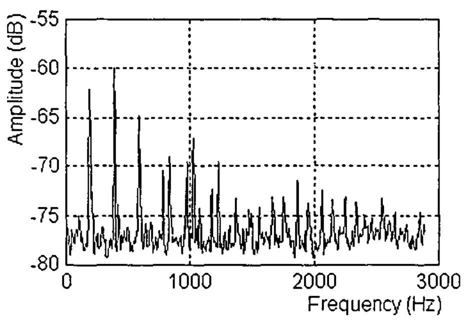


Fig. 3-12 Power spectrum of the third channel with grounded input, with a full scale (±3V) 200 Hz sine wave signal applied to the second channel.

Chapter 4

The DSP Software

4.1 Introduction

The software developed for the DSP board to support a RTCAMS is presented in this chapter. It includes the real-time micro-kernel, the parallel processing, and the multiprocessing routines for the DSPs. The design of the software for the data acquisition system must consider all aspects of the hardware that have been introduced in Chapter 3. The challenge of writing the software is to get the maximum performance of the available computing power, to give an informative presentation for human visual perception, and to maintain sufficient flexibility such that the system can be adapted to different experimental protocols. Like the hardware, the software structure is based on a modular approach that can utilize the system flexibility and allow expansion for future demands of research and clinical applications.

4.2 The System Software Overview

Massive data transfers requiring almost all the resources of the PC are involved in the RTCAMS to display graphics in real time (33 frames per second). Therefore, it is necessary to develop the main part of the software on the DSP board which has all the processing power and resources required. The host PC software thus serves as the interface between the user and the RTCAMS, and provides the graphic display of the cardiac acoustic maps (CAMs) and the PCGs or PCG envelopes. In the rest of this thesis, the term "PCGs or the PCG envelopes" will be referred to as the PCGs to simplify writing, unless it is specified as an envelope. In addition to the user interface function, the host PC spends a significant length of time transferring the cardiac data to and from the DSP board. These tasks require nearly all the computing power of the host PC when processing in real time. The details of this part of the software will be presented in the next chapter.

The overall structure and data-flow developed on the DSP board are shown in Fig. 4-1. The DSP software controls the pre-processing board and the data acquisition board to acquire the PCGs from 25 channels. It computes the graphic representation of the PCGs for visual monitoring, and the CAMs for studying the propagation of the PCGs. These tasks are distributed among two C40 modules, module A and module B. The graphical data are then transferred to the host PC for display. The C40 modules C and D are used only for their memories. The task assigned to these modules consists of saving the cardiac data into their on-module memory. Saving the cardiac data directly into a disk file or into the host PC memory while the system is running is not feasible due to the low bandwidth of the ISA bus where the DSP board is connected. The buffered cardiac data are transferred off-line to the host PC.

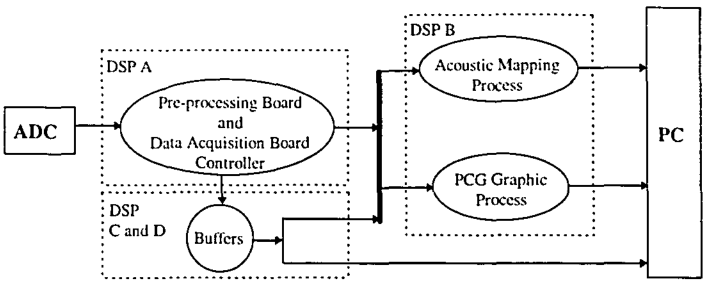


Fig. 4-1 The overall structure and data-flow diagram of the software developed on the DSP board

4.3 The C40 Module A Software

The C40 module A interacts with five different devices as shown in Fig. 4-2. The first device is the host PC from which it receives commands, and transfers the PCGs and the CAMs. The second device is the data acquisition board from which the cardiac data are read. The third device is the PCG pre-processing board, as explained in Section 3.3. The fourth device is the C40 module B to which the cardiac data are transferred to process the PCGs and the CAMs. The module A receives also commands from the C40

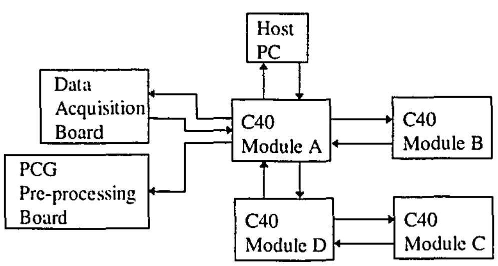


Fig. 4-2 Interactions between the C40 module A and the other devices.

module B which supervises the activity of the DSP board. The fifth device is the C40 module D to which the cardiac data are transferred and saved into the temporary memory for off-line saving into a disk file and for off-line processing. The C40 module D fills the memory of the C40 module C first with the transferred data, before storing data in its own memory. This procedure is necessary because there is no direct connection between module A and module C.

4.3.1 The Main Function of the Software of DSP Module A

The first function to be executed when the DSP A starts is the Main Function. As shown in Fig. 4-3, it contains thirteen steps. The first step allocates two buffers; a 10-kword (32-bit) circular buffer to store the cardiac data and a 5-kword buffer to store the packed cardiac data. The use of these two buffers is explained later in this section. In the second step, the program sets up and enables three hardware interrupts. The first hardware interrupt is generated by the data acquisition board when a new digitized data is ready to be read. The second and the third hardware interrupts are generated by the host PC and the DSP B, respectively, when they send commands to the DSP A. In the third step, two direct memory access (DMA) channels are initialized. The two channels, channels 0 and 3, are used to transfer the cardiac data to the C40 module B and the C40 module D. Using the DMA is important because it allows DSP A to perform another task in parallel. In step 4, the Main Function resets the analog demultiplexer and the acoustic probe to make sure that both devices are synchronized for the data transfer using a time-division multiplexing technique. This is achieved by writing the command *FFFFh* at address *B000 0321h*, as explained in Section 3.3. It then writes a *1* at address *B000 0322h* to select the PCG envelopes (step 5) and writes a *0* at address *B000 0322h* to sample all twenty-five channels beginning with channel 0 (step 6).

After performing the initialization steps 1 to 6, the Main Function signals to the host PC that the C40 module A is ready. It then waits for the host PC signal to indicate that the three other C40 modules are ready before continuing the program execution. The

Main Function then enters the Main Loop and it will never go out until the DSP board is reset. The DSP board is reset when the system is first started and before any programs are downloaded into the DSP board. However, the user can manually reset the DSP board through software running on the host PC. At step 9, the program is looping continuously, waiting for the start command from the user.

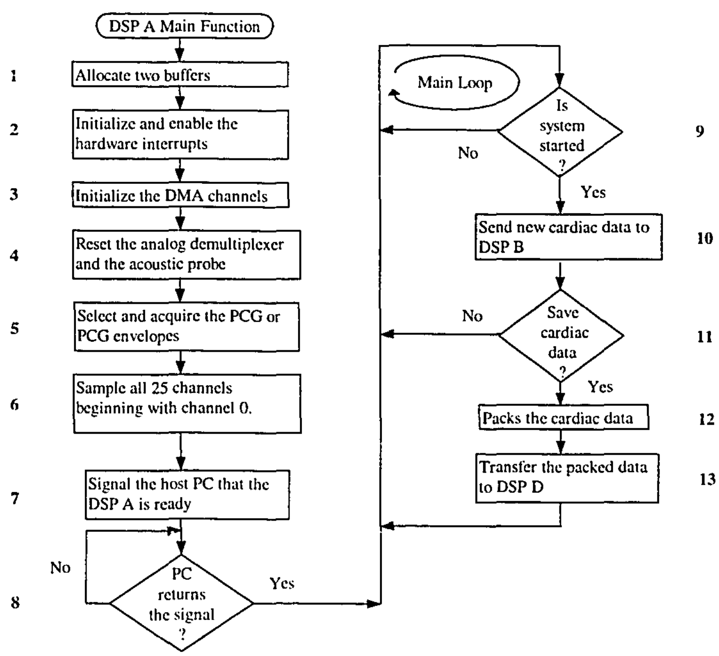


Fig. 4-3 The flowchart of the Main Function of DSP A

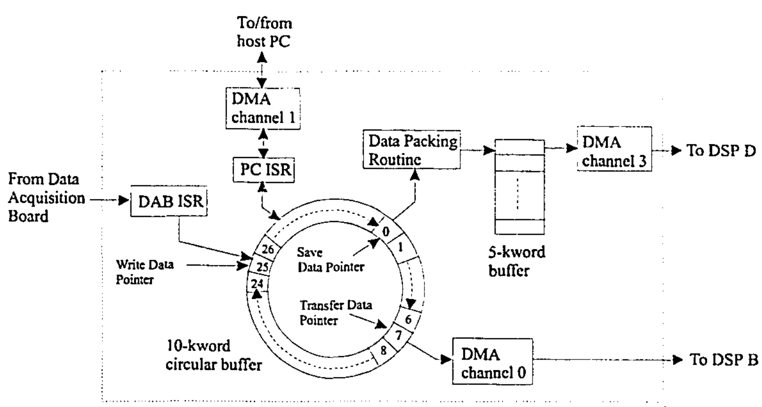


Fig. 4-4 Flow of the cardiac data into the C40 module A

Fig. 4-4 shows the cardiac data flow in the C40 module A when the system has started. First of all, when a cardiac acoustic signal is digitized, the data acquisition board interrupts the DSP A to get it to enter the Data Acquisition Board Interrupt Service Routine (DAB ISR). The DAB ISR retrieves the digitized value from the data acquisition board and stores it in the 10-kword circular buffer at the address pointed by the *Write Data Pointer*. This pointer is then incremented for the next value.

At step 10 of the Main Function, the cardiac data that are not yet transferred are sent to the C40 module B by using channel 0 of the DMA (see Fig. 4-4). The DMA transfers the cardiac data starting at the location pointed by the *Transfer Data Pointer* up to the newest stored cardiac data (See Fig. 4-4). At the end of the transfer, the *Transfer Data Pointer* will be at the address pointed by the *Write Data Pointer*. If the system is to save the cardiac data into the memory of modules C and D, the Main Function will enter step 12 of Fig. 4-3. A Data Packing Routine starts packing the cardiac data pointed by the *Save Data Pointer* up to the newest stored data. The Data Packing Routine is necessary to double the number of saved data by packing two 16-bit data, as determined by the 16-bit analog-to-digital converter, into one 32-bit data, since the C40 is a 32-bit processor. During the packing procedures, the packed cardiac data are stored temporarily in the 5-kword buffer allocated in step 1 of the Main Function. At the end of the packing procedure, the content of the 5-kword buffer is transferred to the C40 module D by the DMA using channel 3.

4.3.2 The Hardware Interrupts

As mentioned earlier, there are three hardware interrupts: the host PC interrupt, the C40 module B interrupt, and the data acquisition board interrupt. This section describes these hardware interrupts and how they are handled by the DSP A.

4.3.2.1 The Host PC Interrupt

There are five types of services that the host PC may request to the DSP A as listed in the Table 4-1. Each type is identified by a service identification number (ID NO.) which is sent by the host PC to the communication port 1 of the DSP A via the LIA (see Fig. 3-3). The communication port 1 then interrupts the execution of the DSP A to get it to enter into the PC ISR. The PC ISR retrieves the service ID NO. from the communication port 1, executes the requested service, and returns to the Main Function.

The first service is to save the cardiac data into the memory of modules C and D. Because there is a 385-kword of memory per C40 module and the cardiac data are packed into 32-bit, 1.54 million of samples can be saved, which represents approximately 10 seconds of recording. When the C40 module A receives the command ID NO. 1, it transfers it to the C40 module D, as explained in Section 4.5.

Table 4-1 Service identification number (ID NO.) sent by the host PC to request a service from the DSP A.

Service ID NO.	**Service Description**
1	Save the cardiac data into the memory of modules C and D.
2	Retrieve the cardiac data from the memory of modules C and D and transfer them to the host PC.
3	Retrieve the cardiac data sent by the host PC which are read from a disk file.
4	Set the PCG pre-processing board to acquire the PCG envelopes.
5	Set the PCG pre-processing board to acquire the PCGs.

The second service is to retrieve the packed cardiac data from the memory of modules C and D and transfer them to the host PC. Like for the first service, the C40 module A transfers the received command ID NO. 2 to the C40 module D. As mentioned previously, the RTCAMS must be stopped during this transfer. The retrieving mechanism consists of retrieving the oldest packed cardiac data first by blocks of 10k words. The retrieved packed cardiac data are stored temporarily into the 10-kword circular buffer and transferred to the host PC via the communication port 1 and the LIA (see Section 3.2.1). Retrieving the cardiac data is done by the DSP A and the transfer to the host PC is done by the DMA using the channel 1. The channel 1 of the DMA is used to transfer the packed cardiac data in both directions between the DSP A and the host PC. depending on the service (2 or 3) requested by the host PC. The process of retrieving a block of 10-kword packed cardiac data followed by the transfer to the host PC is repeated until all the packed cardiac data saved into modules C and D are transferred to the host PC. At the end of the transfer, the PC ISR exits and returns to the Main Function.

The third service allows the DSP A to retrieve the packed cardiac data read from a disk file by the host PC. The retrieving mechanism is similar to the second service described above except that the data transfer is in the opposite direction. The fourth service simply writes a *1* or a *0* at address *B000 0322h* to select the PCG envelopes or the PCGs (see Section 3.3 and 3.4).

4.3.2.2 The C40 Module B Interrupt

The C40 module B controls the activity of the whole DSP board, therefore sending commands to the DSP A to start or to stop the system. Table 4-2 summarized the ID NO. assigned to each command. Each time the DSP B sends a command, it interrupts the DSP A to get it to enter the DSP B ISR and executes the associated program.

Table 4-2 Identification number (ID NO.) assigned to the commands sent by the DSP B.

Command ID NO.	**Command Description**
1	Start the system and start the data acquisition board .
2	Start the system, retrieve and unpack the packed cardiac data saved into DSP B and C.
3	Stop the system.

The first two commands initialize the three pointers of the 10-kword circular buffer to point to the same element (see Fig. 4-4). This procedure is equivalent to emptying the circular buffer. If the command ID NO. is *1*, the DSP B ISR performs three steps; it first writes a *0* at address *B000 0322h* for sampling all twenty-five channels (see Section 3.3), sets the data sampling rate to 150 kHz, and starts the data acquisition board. If the command ID NO. is *2*, the DSP B ISR prepares the DSP A to retrieve and unpack the packed cardiac data saved into DSP B and C and transfer them to the DSP B for generating the graphical display of the PCGs and the CAMs. The command ID NO. *3* stops the data acquisition board and aborts any data transfer from the PC to the C40 module A.

4.3.2.3 The Data Acquisition Board Interrupt

As described earlier in this chapter, the data acquisition board interrupts the DSP A every time it finishes digitizing a signal value. The DSP A enters the DAB ISR to switch to the next channel for the next acquisition, to read the new cardiac data from the data acquisition board, and to store it into the 10-kword circular buffer. The flowchart of the DAB ISR is shown in Fig. 4-5. Switching to the next channel is done by writing the

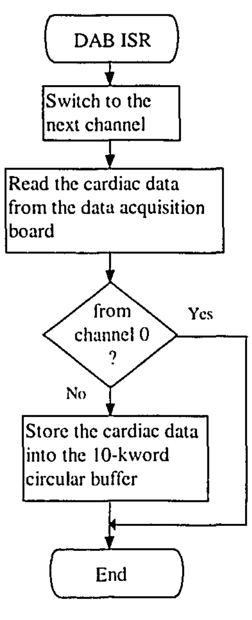


Fig. 4-5 Flowchart of the DAB ISR.

next channel number at address *B000 0322h*, as explained in Section 3.3. The data acquired from channel 0 is not valid (see Section 3.4) and consequently not saved into the 10-kword circular buffer.

Because the data acquisition operates at 150 kHz, the DAB ISR is executed 150,000 time per second. To allow sufficient DSP resources to process other tasks, the DAB ISR must be very short and optimized for speed. In our application, the execution of the DAB ISR takes 42 clock cycles plus 6 clock cycles to switch from the Main Function to the DAB ISR, and return. Because there are 150,000 interruptions per second, the DSP A executes 7.2 10^6 clock cycles per second for the DAB ISR which is equivalent to 36% of the DSP A resources, for a 40 MHz TMS320C40 DSP processor operated at 20 MOPS.

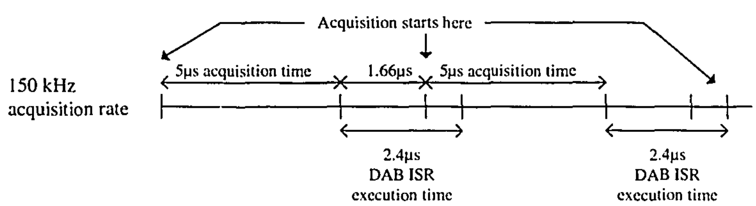


Fig. 4-6 Timing of the data acquisition and the DAB ISR.

Fig. 4-6 shows the timing of the data acquisition and the DAB ISR execution time at 150 kHz. The time required to acquire an analog signal is 5µs and this is independent of the data acquisition rate. The DAB ISR requires approximately 2.4µs to execute but the switching to the next channel occurs within the first 1µs. It is essential that the next conversion begin only after that the next channel is enabled. Therefore the minimum sampling interval is 6µs between two data conversions, which is equivalent to a maximal sampling frequency of 166 kHz. However, the data acquisition board is set to operate at

150 kHz which is the highest configurable frequency rate below 166 kHz [45]. This leaves 1.66μs for channel switching.

4.4 The C40 Module B Software

Computing the CAM and the PCG graphic display are the two main tasks of the C40 module B. Two other tasks are also included in the module B: to receive the cardiac data from the C40 module A and to transfer the resulting PCGs and CAMs to the host PC. These four tasks must operate in parallel to achieve the real-time display of the PCGs and the CAMs. To achieve this goal, a real-time micro-kernel had to be developed.

The flow of the PCGs and the CAMs is shown in Fig. 4-7. The cardiac data sent by the C40 module A arrives at the communication port 3 of the DSP B. The presence of a cardiac data in that port activates the DMA which starts retrieving the cardiac data using its channel 3 and stores it into an 10-kword circular buffer using its own destination address register. This destination address register can be accessed by the PCG Graphic

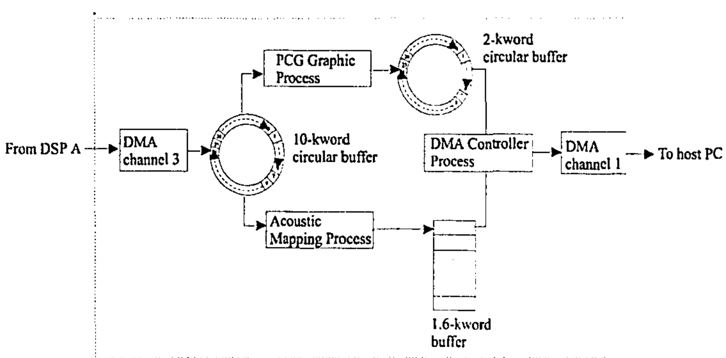


Fig. 4-7 Flow of the PCGs and the CAMs in C40 module B.

Process and the Acoustic Mapping Process to identify the newest cardiac data being stored into the 10-kword circular buffer. The Graphic Process uses the *PCG Pointer* to retrieve the oldest cardiac data up to the newest stored cardiac data. The retrieved cardiac data are then processed and transformed into its graphical representation based on an array of 8-bit values which represent the position of several pixels on the display window. At the end of the PCG Graphic Process, the resulting PCG graphics are packed and stored into a 2-kword circular buffer. The Acoustic Mapping Process retrieves the twenty-five newest cardiac data from the 10-kword circular buffer and computes the CAM. The 8-bit values of the resulting CAM are then packed and stored into a 1.6-kword buffer. The PCGs and the CAMs are later transferred to the host PC by the DMA using channel 1.

The four tasks of the C40 module B are distributed among the DSP B and its DMA. The first task, which retrieves and stores the cardiac data from the communication port 3 to the 10-kword circular buffer, is completely carried out by the DMA and it does not require any intervention from the DSP B. The second task, the PCG Graphic Process, and the third task, the Acoustic Mapping Process, are both time-consuming tasks and are handled by the DSP B. The last task is operated by both the DSP B and the DMA. The DSP B programs the DMA to transfer the PCG or the CAM graphic to the host PC. The transfer is, of course, handled by the DMA which then frees the DSP B for other tasks.

4.4.1 The Main Function of the DSP B Software

As shown in Fig. 4-8, the Main Function of the DSP B is similar to the Main Function of the DSP A, therefore it is described only briefly here. The first step allocates three buffers: one 10-kword circular buffer, one 2-kword circular buffer, and one 1.6-kword buffer. The second step initializes three hardware interrupts: one generated by the host PC to send commands to the DSP B, one generated by the timer of the DSP B to activate the Acoustic Mapping Process 33 times per second, and one generated by the DMA to transfer the next graphic to the host PC when the on-going transfer is finished. The third step initializes the DMA channel 1 and 3 for data transfer, as explained in the

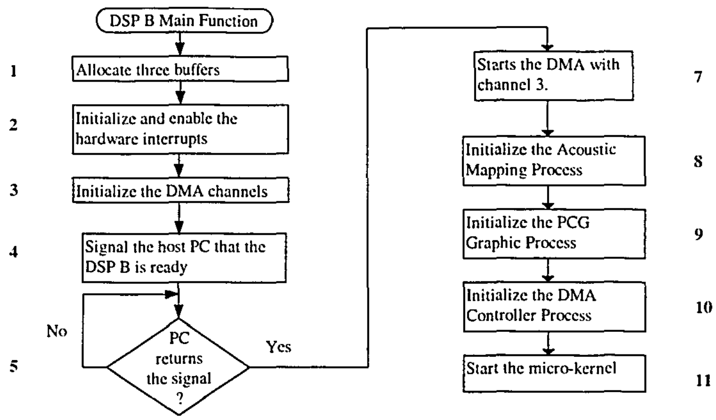


Fig. 4-8 The flowchart of the Main Function of the DSP B

previous section. Steps 4 and 5 signal the host PC that the C40 module B is ready and waiting for other C40 modules before continuing its execution. Step 7 starts the DMA with channel 3 to store the incoming cardiac data into the 10-kword circular buffer. Steps 8, 9 and 10 create and initialize the three tasks performed by the DSP B. These steps are necessary to register the three tasks into the micro-kernel. In the last step, the Main Function starts the micro-kernel which distributes the DSP processing time among the three tasks such that they can perform in parallel. Section 4.4.3 describes in details the micro-kernel and its relation with the three tasks.

4.4.2 The Hardware Interrupts

There are three hardware interrupts in the DSP B. The first is generated by the host PC, the second by the timer of the C40 module A, and the third by the DMA.

4.4.2.1 The Host PC Interrupt

The host PC interrupt works similarly to the one in the C40 module A. Table 4-3 shows five types of services that the C40 module B handles. The first service starts the C40 modules A and B in real-time mode. This is done by starting the timer to activate the Acoustic Mapping Process 33 times per second and by sending the command ID NO. 1 to the C40 modules A (see Table 4-2). The second service is to start the C40 modules A and B to process the cardiac data saved into the memory of DSP B and C. This service also starts the timer for 33 interrupts per second and it sends the command ID NO. 2 to the C40 module A. The third service stops the C40 module A and B by stopping the timer. The fourth service reports the currently opened windows by the RTCAMS on the monitor to the C40 module B. Normally there are two windows opened; one for the CAM and one for the PCGs. Only the graphic with the opened window is computed by the DSP B and transferred to the host PC. The fifth service specifies the upper and lower limits of the PCGs on the display window area, and specifies the number of data points to merge for the construction of the PCGs.

4.4.2.2 The Timer Interrupt

Table 4-3 Service identification number (ID NO.) sent by the host PC to the C40 module B.

Service ID NO.	Service Description
1	Start C40 modules A and B in real-time mode.
2	Start the C40 modules A and B to process the cardiac data saved into the memory of DSP B and C.
3	Stop the C40 modules A and B.
4	Report the currently opened windows by the RTCAMS on the monitor to the C40 module B.
5	Specify the upper and lower limits of the PCGs on the display window area, and specifies the number of data points to merge for the construction of the PCGs (see Section 4.4.4.3)

When the timer interrupt arrives, the DSP B enters the Timer Interrupt Service Routine (TISR). This routine then sends a message to the micro-kernel to activate the Acoustic Mapping Process. This is necessary to make sure that 33 CAMs can be computed every second. Once the CAM is created, the Acoustic Mapping Process releases the DSP B for another task and put itself into a *wait state* for the next timer interrupt. The details about the micro-kernel and the tasks are explained in Section 4.4.3.

4.4.2.3 The DMA Interrupt

At the end of a transfer to the host PC, the DMA interrupts the DSP B to execute the DMA Controller Process which reprograms the DMA for the next transfer. This is necessary because, the DSP B has to determine which acoustic graphic needs to be sent; the PCGs or the CAM. Note that the DMA Controller Process is not an interrupt service routine but a task, as described later in Section 4.4.4.2.

4.4.3 The Design of the Real-Time Micro-Kernel

Although there are many well-known real-time commercial operating systems, these are often too expensive and too all-purpose to be optimal in real-time applications that have rigorous response time requirements. To deliver total control over the interrupt state of the DSP board, a real-time micro-kernel (RTMK) was entirely written for the C40 module B and its design was closely tied to the specific application and based on technical facets described in [3,4,26]. The micro-kernel provides three specific functions: task scheduling, task dispatching, and inter-task communication. The scheduler determines which process will run next in a multi-tasking system, the dispatcher performs the necessary work to start that task, and the inter-task communication is used to synchronize the activities between running tasks.

The micro-kernel designed for the C40 module B is based on the RTMK written for embedded systems [4]. This RTMK provides a fast context switching to spend as little time as possible in the kernel, a simple interface to a standard programming language, and interruption of the kernel to allow high-priority immediate processes hooked to an interrupt. The RTMK also supports a subset of the original *Sceptre* standard [3]. The Sceptre, defined by several European software houses, proposes a simple but realistic set of commands for a real-time kernel. The entities of our RTMK include :

- **Tasks.** In the C40 module B, the real-time system has three tasks: the PCG Graphic Process, the Acoustic Mapping Process, and the DMA Controller Process. Switching from one task to another is like switching from one DSP to another. Therefore, each task must have its own copy of all registers of the DSP, which is called the process context. During process changes or context switches, the environment of the executing process (the content of the DSP registers) is saved into the process context while re-installing the next scheduled process context. Consequently, the context must contain all the necessary data to maintain the integrity of the DSP operation. The area where all this data is stored is a process descriptor as shown Fig. 4-9. The process context is stored into the variable *context*. The task priority is handled by the structure *priority* and *pmask* fields. The *pmask* field is simply a mask that is pre-calculated during task definition in order to accelerate the context switches by avoiding unnecessary left shifts at run time. The *expected_signals* field is a mask. If bit n is set in this mask, signal n can activate the process. In other words the process is waiting for signal n to arrive before it can be scheduled. The *received_signals* is another 32-bit field in which each bit set to 1 means that the corresponding signal has been received by the process; but if the process is not waiting for this signal, then the process will not be scheduled until the corresponding *received_signals* is set by the process.

```
typedef unsigned long word32;        /* Double Word for a max of 32 processes kernel */

  typedef word32 SIGNALS;                /* Signal type                               */

  struct PCS {
    my_jmp_buf context;                        /* CPU Context (registers)
*/
    SIGNALS expected_signals;
    SIGNALS received_signals;
    word32 pmask;
    word32 priority;
  };

 typedef struct PCS *PROCESS;    /* pointer of PROCESS CONTEXT STRUCTURE */
```

Fig. 4-9 The RTMK process descriptor

- **Immediate Tasks.** In the DSP B, the real-time system has two immediate tasks: the host PC interrupt and the timer interrupt. In the RTMK, an immediate task is an interrupt service routine which does not have a process context. However, the immediate task saves at the beginning of the task execution the DSP registers that it will use. Therefore, the immediate task can restore the content of these registers before exiting itself without disrupting the execution of the interrupted process. Usually immediate tasks are not interruptible and they must be kept as small as possible especially in a real-time system where timing is highly critical. On the other hand, a task is usually a big process and is always interruptible by any immediate tasks.

- **Scheduler.** A scheduler is a software module in charge of handling the DSP resources and dispatching it to a set of processes. In the RTMK, the scheduler of the immediate processes is the interrupt mechanism of the processor. while the scheduler of standard processes is part of the kernel. The flowchart of the *Scheduler* function is shown in Fig. 4-10. This function is called when a waiting process requests for the DSP B, or when a process releases the DSP B because it is waiting for a signal. In the first case, the waiting process is passed as the parameter *next_process* to the *Scheduler* function. The scheduler then saves the *context* of the current process and restores the *context* of the *next_process* to allocate the DSP B to the waiting process. In the second case, the *Scheduler* function is called with the *next_process* equal to 0. In that case, the scheduler finds among the ready processes the one that has the highest priority and allocates the DSP B to it.

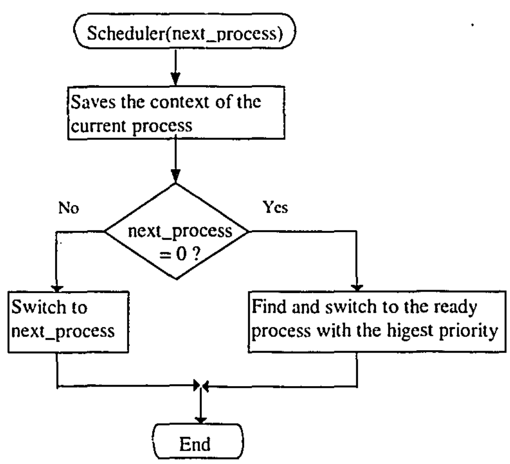


Fig. 4-10 Flowchart of the Scheduler function.

In our implementation, each process is associated with a set of signals. Each signal can be *arrived* or *not_arrived*. The process specifies its own activation condition through these signals. A process can be in one of the following states:

- **Waiting**. The process is suspended until one of its signals has arrived.
- **Ready**. The process has received a signal and will continue execution when the kernel allocates it DSP time.
- **Progress**. The process is currently running.

Sending or receiving signals is done through kernel services by the process. In these primitives, the signals that have arrived are checked against those expected. If a match occurs, the DSP can be reallocated to another process, depending on its priority. The RTMK provides four important processes handling services (see Fig. 4-11) :

- Send (*process, signals*). Sets the bits contained in *signals* with the corresponding bits in the *received_signals* field of the *process* descriptor and calls the *Scheduler*

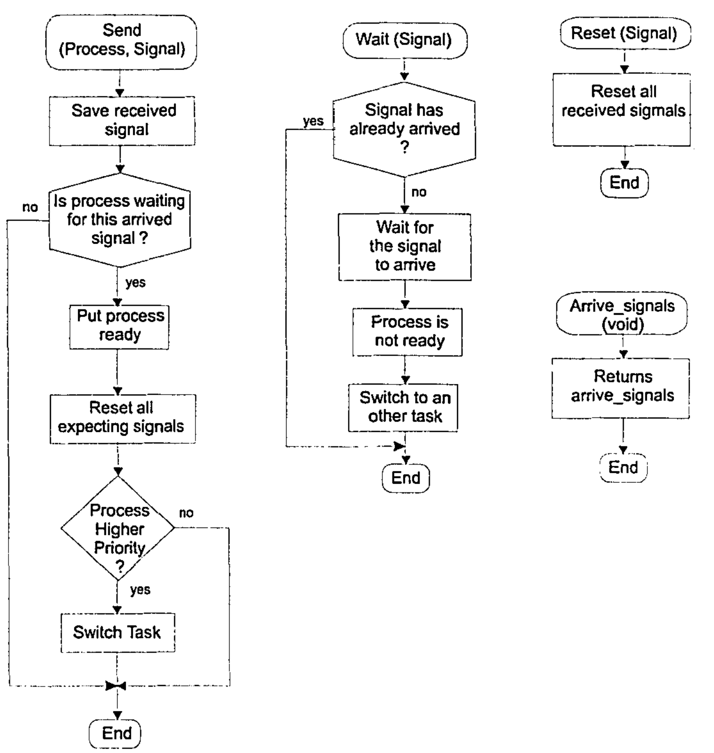


Fig. 4-11 Flowchart of the four most often used RTMK handling services.

with *process* as the parameter.

- Wait (*signals*). Sets the process descriptor *expected_signals* field and calls the scheduler, which suspends the process. This service returns the list of received signals. The return value distinguishes between the signals that triggered the process, since two signals can be activated at the same time.
- Reset (*signals*). Clears the bits corresponding to *signals* in the process descriptor *arrived_signals* in order to permit another cycle. By clearing these bits just before the next call to the wait service, all the events that occurred during the processing are ignored.
- Arrive_signals (*void*). Returns the field *arrived_signals* to the caller.

The send(*process, signals*) and wait(*signals*) services serve many useful purposes. For example, they may be used to guard against concurrent access of shared variables among a number of concurrent tasks, or to synchronize the execution of two tasks. In our project, both the Acoustic Mapping Process and the PCG Graphic Process need to transfer their resulting graphics to the host PC via the LIA which is mapped into a single address. To avoid the conflict of simultaneous access, the two tasks need to synchronize their access to this single resource by sending a signal to a third task which is the DMA Controller Process. When the DMA Controller Process receives a signal from a task, it then programs the DMA and starts the data transfer if the DMA is not used. If the DMA is used then the DMA Controller Process blocks itself (with wait(*DMA ready*)) and the data transfer is delayed until the DMA is free. Every time the DMA ends transferring data to the host PC, it interrupts the DSP B to send the *DMA ready* signal (with send(*PC server, DMA ready*)) to the DMA Controller Process to reschedule another data transfer.

4.4.4 The Real-Time System Tasks

As stated in the beginning of this section (4.4), the DSP B has four tasks. In the following, a description as well as the integration of the four tasks in the real time and multitasking environment of the DSP B are described .

4.4.4.1 The DMA Channel 3 Task

The DMA channel 3 is set to continuously receiving any incoming cardiac data from the C40 module A. Once it is initialized, the DMA will run independently from the DSP B during the RTCAMS execution. As soon as the communication port 3 receives a cardiac data from the C40 module A, it sends an interrupt signal to the DMA which starts retrieving the cardiac data and places it into the 10-kword circular buffer. Allocating this task to the DMA will allow the DSP B to compute the PCGs and the CAMs in real time. In fact, if this task is not handled by the DMA, the DSP B must allocate approximately 30% of its computing resources to do this task, thus slowing down considerably the processing of the other tasks.

4.4.4.2 The DMA Controller Process and the DMA Channel 1 Task

The DMA Controller Process serves for two purposes in the C40 module B software. First it controls the DMA channel 1 to transfer the PCGs or the CAMs to the host PC. Second it arbitrates the request from the PCG Graphic Process and the Acoustic Mapping Process to transfer their cardiac graphics to the host PC, as explained at the end of Section 4.4.3. However, this Process is relatively small because it assigns the main purpose of its task (transferring the cardiac graphics) to the DMA channel 1. The DMA is capable of handling two transfers from two different channels without penalties [59], thus the DMA channels 1 and 3 can operate in parallel. Note that the Acoustic Mapping Process must wait for the CAM to be totally transferred to the host PC before storing the next CAM into the 1.6-kword circular buffer. This approach is implemented because there is no advantage of computing more CAMs if the DMA cannot transfer all of them to the host PC. This limitation does not exist for the PCG Graphic Process because the 2-kword circular buffer is large enough to insure that it will not be filled out before the next PCGs transfer occurs.

4.4.4.3 The PCG Graphic Process

The goal is to enable the user to view the PCGs and evaluate the quality of the digitized cardiac data. Twenty-five PCGs are computed from the cardiac data retrieved from the 10-kword circular buffer of the C40 module B and drawn on the PCG Window with a minimum processing from the host PC. The PCG Graphic Process can compile all incoming cardiac data in the 10-kword circular buffer and generate the graphics in real time. Each PCG is a set of data which are plotted versus its indices onto the PCG Window and each PCG is assigned to its own plotting area which is 35 pixels high. More details about the display of the PCGs are given in Chapter 5.

The RTCAMS can plot 150,000 data points in one second (for the ~150 kHz sampling rate excluding the channel 0), but using this approach to display the PCGs would not be practical for two reasons. First, this will require more processing time for the host PC for displaying the PCGs and thus slow down the displaying rate of the CAMs. Second, the PCGs are displayed on the PCG Window in a very short period of time, due to the limiting display length of the PCG Window, so that the user cannot control efficiently the quality of the PCGs. Instead, displaying the PCGs in a reasonable speed was accomplished by allowing some data points to be represented by a vertical line corresponding to their maximum and minimum values, enabling the user to enlarge or shrink the time period displayed. The number of data points to merge is specified by the user and sent by the host PC with the service ID NO. 5, as explained in Section 4.4.2.1. The procedure of merging the data point is common in real-time monitoring systems that has multiple channels at acquisition rate higher than 100 kHz [63]. An example of merging one period of a sine wave is illustrated in Fig. 4-12 to show the effectiveness of this procedure. The original sine wave is shrunk to half of its period, to a quarter of its period, and to a line for merging one full cycle.

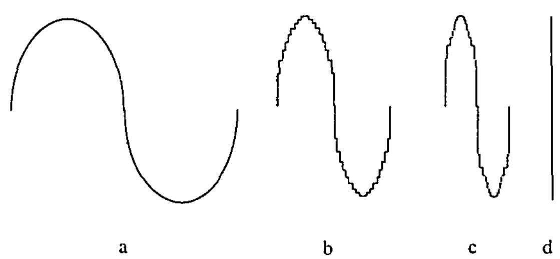


Fig. 4-12 Merging example of one sine wave period. The original sine wave (a) is shrunk to half of its period (b), to a quarter of its period (c), and to a line for merging one full cycle (d).

Because the result of a merged segment of data points is a vertical line, only the positions of the two extremities are needed to be passed to the host PC in order to redraw this vertical line on the PCG Window. Therefore, the 2-kword circular buffer contains a series of twenty-five interleaved PCGs which are composed of a pair of data points, i.e. the positions of the minimum and maximum extremity of the merged segment.

4.4.4.4 The Acoustic Mapping Process

The Acoustic Mapping Process generates a sequence of images from the cardiac circular buffer of the C40 module B. These images represent the temporal evolution of the amplitude of the PCGs. The main role of the process is interpolating the digitized cardiac data from 25 microphones, and rendering an image with a size large enough to be visualized on the host PC color monitor. The spatial interpolation is required to enlarge the dimension of the CAM and to reduce the intensity discontinuity between two adjacent pixels. There are many well known method of interpolation for digitized images

[18,21,24,28,39,41], and a comparison of some of these methods was investigated by Cozic et al. [9] who found that the cubic convolution interpolation of Keys [24] was one of the best method for cardiac acoustic mapping. Their choice was based on many factors including the computing time, the quality of the image, and the human visual perception. The short computing time required for interpolating an image is definitely an important asset for our real-time system. For each interpolated point, the cubic convolution requires only 3 additions and 4 multiplication and this is relatively small compared to other methods. Two-dimensional interpolation with the cubic convolution is developed as a cascade of two one-dimensional interpolations, one in the horizontal axis and the other in vertical axis, in order to reduce the number of operations and the required memory to handle intermediate results [28]. The human visual perception is sensitive to the rapid changes of intensity, i.e. the high frequency components in an image [37]. The cubic convolution interpolation reduces this by filtering the original image (5x5 matrix) with a low pass-filter so as to attenuate the high frequency components of the resulting interpolated image (100x100 matrix).

The cubic convolution interpolation proposed by Keys is briefly described in this section. Consider the signal $f(t_n)$ defined by the sequence of points $\{t_n: n = 0, 1, ..., N\text{-}1\}$ acquired from a continuous signal $\bar{f}(t)$. The interpolation process is to approximate the original signal $\bar{f}(t)$ by a continuous function $\bar{g}(t)$. Cozic et al. [9] show that the cubic convolution interpolation of a segment $[t_n, t_{n+1}]$ is expressed by :

$$\begin{aligned} \bar{g}(t) = {} & \frac{C_{n-1}}{2}(-d^3+2d^2-d)+\frac{C_n}{2}(3d^3-5d^2+2)+ \\ & \frac{C_{n+1}}{2}(-3d^3+4d^2+d)+\frac{C_{n+2}}{2}(d^3-d^2) \end{aligned} \tag{4.1}$$

where $d = (t-t_n)/\Delta \qquad 0 \leq d < 1$

$C_n = f(t_n) \qquad for\ n = 0, 1, ..., N\text{-}1$

and Δ the sampling interval.

At the boundary $n=0$ *and* $n=N-1$, C_{-1} and C_N are given by:

$$C_{-1} = f(t_2) - 3\,f(t_1) + 3\,f(t_0) \tag{4.2}$$

$$C_N = 3\,f(t_{N-1}) - 3\,f(t_{N-2}) + f(t_{N-3}) \tag{4.3}$$

It is more convenient to write Equation (4.1) in a matrix form by defining :

$$G_n = \begin{bmatrix} f(t_{n-1}) & f(t_n) & f(t_{n+1}) & f(t_{n+2}) \end{bmatrix} \; P \tag{4.4}$$

where :

$$P = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1A} \\ P_{21} & P_{22} & \cdots & P_{2A} \\ P_{31} & P_{32} & \cdots & P_{3A} \\ P_{41} & P_{42} & \cdots & P_{4A} \end{bmatrix}$$

with : G_n : the interpolated value in the segment $[t_n, t_{n+1}]$

A : the number of points to interpolate inside the segment $[t_n, t_{n+1}]$

$$\left.\begin{aligned} P_{1i} &= (-d_i^3 + 2d_i^2 - d_i)/2 \\ P_{2i} &= (+3d_i^3 - 5d_i^2 + 2)/2 \\ P_{3i} &= (-3d_i^3 + 4d_i^2 + d_i)/2 \\ P_{4i} &= (+d_i^3 - d_i^2)/2 \end{aligned}\right\} \quad d_i = \frac{i}{A+1} \quad i = 1,2,\ldots,A$$

The matrix P needs to be computed only once at the beginning of the program execution because every segment has the same fixed number of points A to interpolate. In our application, there are 24 points to interpolate inside each segment $[t_n, t_{n+1}]$ to construct an 100x100 matrix from the original 5x5 matrix. In this case, A=24 and the matrix P is:

$$P = \begin{bmatrix} -0.0225625 & -0.0405 & \ldots & -0.0011875 \\ 0.9939375 & 0.9765 & \ldots & 0.0298125 \\ 0.0298125 & 0.0685 & \ldots & 0.9939375 \\ -0.0011875 & -0.0045 & \ldots & -0.0225625 \end{bmatrix}$$

The position of each channel inside the original 5x5 matrix is illustrated in Fig. 4-13 and it reflects the relative position of each microphone inside the acoustic probe, which are disposed in a rectangular area and are equally spaced.

Ch 1	Ch 2	Ch 3	Ch 4	Ch 5
Ch 6	Ch 7	Ch 8	Ch 9	Ch 10
Ch 11	Ch 12	Ch 13	Ch 14	Ch 15
Ch 16	Ch 17	Ch 18	Ch 19	Ch 20
Ch 21	Ch 22	Ch 23	Ch 24	Ch 25

Fig. 4-13 Relative position of each channel in the original 5x5 matrix before interpolation

Interpolating the original 5x5 matrix is performed in two steps by successive applications of the one-dimensional Key's cubic convolution method, as shown in Fig. 4-14. A 5x100 matrix is obtaining by a cubic convolution interpolation along each row. The final result is then obtained by interpolating along each column of the intermediate 5x100 matrix. It should be noted that the same results would be obtained by first interpolating along the columns and then along the rows of the matrix.

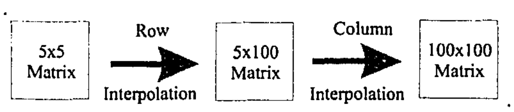


Fig. 4-14 Computation of the two-dimensional Key's cubic convolution interpolation as a series of two one-dimensional interpolations.

Each value of the resulting 100x100 matrix is converted from floating point (32-bits) to an unsigned byte (8-bits) value representing a color map index of 256 colors.

This color map is created by the host PC, as will be described in Chapter 5. Each four consecutive bytes of data is then packed into one 32-bit value and saved in the circular buffer (Fig. 4-3) to speed up the data transfer to the host PC.

4.5 The C40 Module D Software

The Main Function of the C40 module D has only three steps. In the first step, the DSP D initializes the DMA channels 0 and 3 to transfer the packed cardiac data between the DSP A and DSP D (channel 0) and between the DSP D and DSP C (channel 3), in both directions. In the second step, the DSP D signals the host PC that it is waiting for the other DSPs to be ready before continuing its program execution. The third step is a loop that waits for commands sent by the DSP A which is not done by interruptions like the C40 module A and B. If the command has a service ID NO. equal to 1, the DSP D programs the DMA to retrieve the first 385-kword of packed cardiac data from its channel 0 (sent by the DSP A) and to transfer them to the DSP C through channel 3 (see Fig. 4-15). At the end of this transfer, the DMA initializes itself and transfers the second 385-kword to the C40 D on-module memory. If the command sent by the DSP A has an ID NO. equal to 2, the DSP D programs the DMA channels 0 and 3 to transfer the cardiac data in the opposite direction. The cardiac data of the DSP C are sent before that of the DSP D.

4.6 The C40 Module C Software

The C40 module C software is the same as that of the C40 module D, except that it does not initialize the DMA channel 3 and does not transfer any data to that channel, as shown in Fig. 4-15. All incoming packed cardiac data are stored in its on-module memory if the command ID NO. sent by the DSP C is 1; otherwise it transfers its packed cardiac data back to the DSP C through the DMA channel 0.

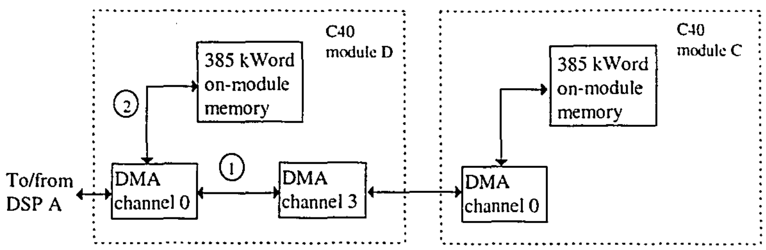


Fig. 4-15 Flow of the cardiac data in the C40 modules C and D

4.7 Conclusion

Three major time-consuming tasks are identified in this projects to be the DAB ISR which has a short execution time but is called frequently, the PCG Graphic Process which computes the graphical display of the PCGs, and the Acoustic Mapping Process which interpolates an array of 5x5 PCGs envelopes to a CAM of 100x100. The three tasks are distributed in two DSPs with the DAB ISR implemented in DSP A and the two other processes in DSP B. All three algorithms are written in assembler to improve the speed of the system. A real-time micro-kernel is implemented in the second DSP to optimize the use of the DSP resources among the following three parallel processes: the Acoustic Mapping Process, the PCG Graphic Process, and the transfer of both graphical data to the host PC. The DSP C and D are used to save the digitized cardiac data into their memories for off-line saving into a disk-file and for off-line processing.

Chapter 5

The Host PC Software

5.1 Introduction

The Windows 3.1[1] was chosen as the operating system for the RTCAMS because of two main reasons. First, it was one of the most widely used operating system in the world when this project began. Second, it provides a graphical user interface (GUI) and a color map image device which can be easily adapted to our specific application. Indeed, the Windows GUI provides ergonomic use of the program, allowing the user to directly manipulate graphic objects on the video screen, like dragging windows, pushing buttons, and using scroll bars. Also, it allows the program to display the graphics of the PCGs and the CAM on two separate windows. In addition, Windows 3.1 offers benefits for a RTCAMS requiring large memory.

In the development of the RTCAMS on Windows 3.1, we have met two limiting factors: the slow graphic display under Windows 3.1 and the slow data transfer over the internal ISA bus of the host PC by which graphic data and the cardiac data are transferred from the DSP board to the host PC disk. In order to display the CAMs in real time (i.e.

[1] Windows 3.1 is a registered trademark for an operating system developed by the Microsoft Corporation.

33 CAMs per second), the software that deals with the data transfer and the graphic display must be optimized. The display of the PCGs is not a time consuming process because it involves only drawing lines (see Section 4.4.4.3) and the PCG graphic data are relatively small compared to the CAM data.

To partially overcome the poor performance of Windows 3.1 in displaying sequential images in real time, a technique called double buffering used in multimedia applications and graphic intensive applications is applied. A Microsoft library called the WinG application program interface (API) was also used to speed up the graphic display to 33 CAMs per second.

A software called the Virtual DSP Driver (VDSPD) is developed to manage the cardiac data transfer from the DSP board to the host PC. The software resides in the kernel-level of Windows 3.1 and is hand-optimized in assembler to provide the highest possible performance for the data transfer between the host PC and the DSP board. The VDSPD has the direct access privilege to the I/O ports of the DSP board and therefore provides fast services.

A brief description of the Windows 3.1 architecture is presented in the next section. It is followed by the implementation of the VDSPD, the techniques used to display images in real time, and the overall application structure that runs under Windows 3.1.

5.2 The Windows 3.1 Architecture

When Windows 3.1 is invoked on an appropriately configured system, it runs in an "enhanced" mode to use the Intel 80386 and later microprocessors which support multiple, independent memory regions. Enhanced Windows uses this to build multiple, independent virtual machines (VMs), each capable of running an application program, as shown in Fig. 5-1. Each virtual machine has its own address space, I/O port space, and interrupt vector table. Multiple virtual machines can be running simultaneously, with each under the illusion that it has complete control of the processor. The first VM created is called the System VM. This is the virtual machine in which the Windows graphical user interface runs. Non-Windows applications (DOS applications) run in VMs of their own and they multitask preemptively. However, all Windows applications run inside the unique system VM and do not support the preemptive multitasking.

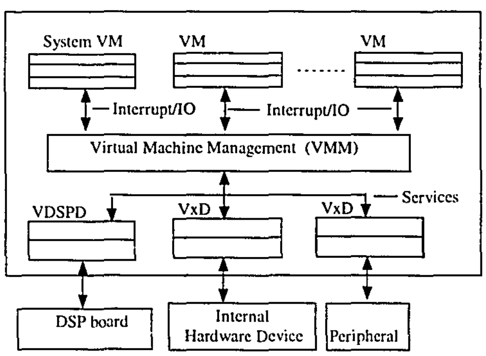


Fig. 5-1 Enhanced Windows environment block diagram.

Enhanced Windows supports this multitasking virtual machine environment with a sophisticated set of services provided by the virtual drivers (VxDs), which give access to all the system resources, including memory management, scheduling, and the hardware devices including the DSP board. By writing a VxD for a particular hardware device, the Windows Enhanced mode enables multiple VMs (and the System VM) to share that single hardware or to allow only one VM (or the System VM) to use that single hardware. The Virtual Machine Manager (VMM) supporting a multitasking operating environment provides services that control the main memory, the CPU execution time, and the peripheral. Like the VxDs, the VMM is part of the Windows 3.1 kernel but is transparent to the programmer. The Microsoft Windows Device Development Kit [55] provides more information about the Windows 3.1 architecture in the 386 enhanced mode.

5.3 The Virtual DSP Driver

The VDSPD can be considered as an application that contains its own variables. The VDSPD is part of the Windows 3.1 kernel and it will remain active during the whole session of Windows 3.1, even if the RTCAMS Windows Application is not executed. The initialization hooks the DSP ISR to the specified PC IRQ line used by the DSP board. After being retrieved from the *system.ini* file by the VDSPD, the three base addresses and the IRQ line number are checked for their consistency. The base addresses of the Block 0 must be between *300h* and *370h*. The base address of the Block 1 must be greater or equal to the base address of Block 0 plus *20h*, because each Block consists of 32 bytes, and the base address of the LIA Block must be greater or equal to the base address of Block 1 plus *20h*. The IRQ line must be either 4, 5, 6, 7, or 9. The initialization procedure is aborted if one of the information retrieved from the *system.ini* file does not agree with the restriction mentioned above; however, Windows 3.1 will continue loading other VxDs.

Hooking the DSP ISR to the specified PC IRQ line is accomplished by the *VPICD_Virtualize_IRQ* service provided by the VPICD [55]: a hardware interrupt managed in the host PC by a programmable interrupt controller, or PIC. In Windows 3.1, the virtual PIC driver (VPICD) intercepts all interrupts and dispatches them to the VxD that handles the corresponding interrupt service routine. Hooking the DSP ISR to the specified PC IRQ means that the VPICD will schedule the execution of the DSP ISR whenever the DSP interrupt occurs. The VPICD handles all IRQ lines, each of which can be individually enabled (unmasked) or disabled (masked) by writing to the PIC I/O ports. At this point of initialization, the VDSPD disables the PC IRQ line used by the DSP board.

The VDSPD serves as the interface between the System VM and the DSP board with its four C40 modules. It provides services to the RTCAMS Windows Application as well as to the DSP board (see Fig. 5-2). The VDSPD is programmed such that only the System VM can request its services. All Windows applications running inside the System VM can request any service from the VDSPD, but only the RTCAMS Windows Application has been programmed to request services from the VDSPD.

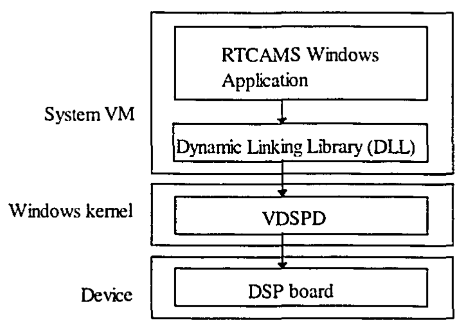


Fig. 5-2 The hierarchy of the different modules supported by the RTCAMS Windows Application

The services for the RTCAMS Windows Application are low-level functions that only read or write in the DSP board I/O map interface. Each low-level function is written in assembly language and is called directly by a dynamic link library (DLL) which "wraps" them into high-level C-language functions that can be used by the RTCAMS Windows Application, as explained later in Section 5.4. The services for the DSP board perform the cardiac data transfer from the DSP board to the host PC and they can be requested by all four C40 modules. To request a service, a C40 module must write the service ID (described in Section 5.3.3) to the communication port connected to the host PC. The presence of a data in that communication port will generate an interrupt to the host PC to get it to enter the DSP ISR to perform the requested cardiac data transfer.

5.3.1 The VDSPD Services for the RTCAMS Windows Application

Because the RTCAMS Windows Application can not access the PC I/O map of the DSP board, it must call the services provided by the VDSPD to do so. However, not all the VDSPD services provide I/O operation, some services return the values of the variables of the VDSPD to the RTCAMS Windows Application, and some services allow both the VDSPD and the RTCAMS Windows Application to share buffers. Fourteen registers of the PC I/O map of the DSP board are used in this project. The description of these PC I/O map registers are in Appendix E.

Table 5-1 The VDSPD services for the RTCAMS Windows application

Service Name	Service Description
VDSPD_API_Read_LIA_Reg	Read LIA Register (16-bit)
VDSPD_API_Write_LIA_Reg	Write LIA Register (16-bit)
VDSPD_API_Read_LIA32	Read two LINK DATA Register (LSW & MSW)
VDSPD_API_Write_LIA32	Write two LINK DATA Register (LSW & MSW)
VDSPD_API_Get_Int_Number	Get IRQ Line number used by the DSP board
VDSPD_API_Write_Control_Reg	Write value to the Control Register
VDSPD_API_Write_BAR	Write value to the BAR
VDSPD_API_Reg_Share_Data	Register shared data between the VDSPD and the application
VDSPD_API_Empty_Buffers	Empty PCG and MAP Buffers
VDSPD_API_Unlock_ISA_Bus	Unlock the ISA bus

Table 5-1 shows a list of services provided by the VDSPD. The first two services read/write any 16-bit register within the LIA Block. The next two services read/write a 32-bit data from/to the *LINK DATA* registers. The *VDSPD_API_Get_Int_Number* service returns the IRQ line number stored in one of the variables of the VDSPD. The *VDSPD_API_Write_Control_Reg* service writes in the *Control* register, and the *VDSPD_API_Write_BAR* service writes the two base addresses of Block 1 and the LIA Block (from the variables of the VDSPD) to the *BAR* register.

The *VDSPD_API_Reg_Share_Data* function assigns a buffer that can be accessed by both the VDSPD and the RTCAMS Windows Application. The buffer must be allocated by the RTCAMS Windows Application before assigning it to the VDSPD. Two buffers can be assigned to store the received PCG graphics and the CAM. The *VDSPD_API_Empty_Buffers* function clears all assigned buffers. The *VDSPD_API_Unlock_ISA_Bus* function enables the next CAM transfer from the DSP board to the host PC and its use is explained in the next section.

5.3.2 The VDSPD Services for the DSP Board

There are two services that the C40 modules can request to the VDSPD: transfer of the PCGs or of the CAM to the host PC. The data must be placed into the communication port in following order to request a service from the VDSPD. The service ID is first (1 for a PCG transfer, and 2 for a CAM), followed by the number of cardiac data to be transferred, and finally the cardiac data. If the communication port interrupt is enabled, a data present in that communication port will generate an interrupt to the VPICD which transfers the program execution to the DSP ISR.

The DSP ISR performs the following tasks:

1. Disables the interrupt from the DSP board.
2. Finds the source of interrupt (which C40 module communication port ?).
3. Retrieves the requested service ID from that communication port.
4. Retrieves the number of data to transfer.
5. Starts the data transfer (reads from that communication port and writes to the corresponding registered buffer),
6. If it is a CAM transfer, blocks any subsequent request for a CAM transfer.
7. Schedules to send a signal to the RTCAMS Windows Application, indicating that a cardiac data has been received after the DSP ISR has exited.
8. Enables the interrupt from the DSP board.
9. Exits the DSP ISR.

Fig. 5-3 shows the cardiac data transfer path requested by the interrupt, and the I/O operation path requested by the RTCAMS Windows Application. The DSP ISR treats the PCG Graphic Buffer as a circular buffer, and the CAM Buffer as a linear buffer that can only handle one entire CAM. Therefore after receiving a CAM, the DSP ISR blocks any subsequent request for a CAM transfer until it is enabled again. This is necessary to prevent a newly received CAM being altered by a subsequent CAM transfer. After the CAM has been displayed from its buffer, the RTCAMS Windows Application will enable the next CAM transfer by calling the *VDSPD_API_Unlock_ISA_Bus* function.

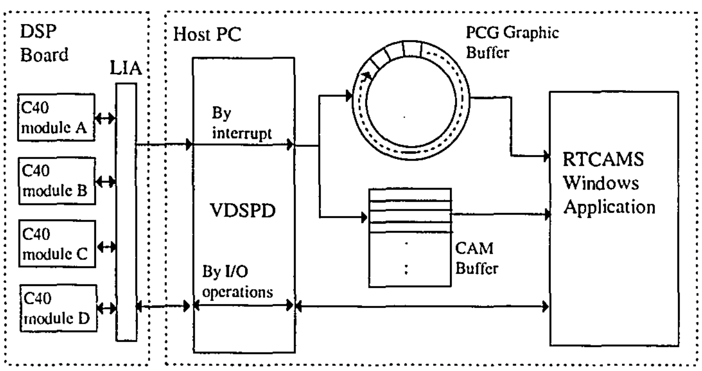


Fig. 5-3 Cardiac data transfer path requested by interrupt of the DSP board, and the I/O operation path requested by the RTCAMS Windows Application.

5.4 The Dynamic Link Library

The dynamic link library (DLL) serves as an interface between the RTCAMS Windows Application and the VDSPD and it resides in the System VM, as shown in Fig. 5-2. The DLL provides high-level functions that can be directly used by the RTCAMS Windows Application to request one or more services from the VDSPD.

Table 5-2 shows a list of nine functions provided by the DLL. The *Reboot* function resets the DSP board and the four DSPs by writing '0' following by '1' to the five least significant bits of the *Control* register [50]. The *Send DSP executable code* function reads a C40 executable file and downloads it to a specified C40 module via its communication port. After being reset, a C40 module scans all its communication ports and boots from the communication port that receives the first data [22]. Therefore, the *Send DSP executable code* function must be called after the *Reboot* function. The other functions of Table 5-2 are self explanatory and are not described in this section.

Table 5-2 List of functions provided by the DLL

Functions	**Description**
Reboot	Reboot the DSP board,
Send DSP executable code	Download a program to a C40 module,
Register shared buffer	Register an allocated buffer to the VDSPD,
Empty buffers	Empty buffers used by the VDSPD,
Read LIA status	Read the *Status* register,
Interrupt select	Select an IRQ line used by the DSP board,
Interrupt enable	Enable interruptions from the DSP board,
Interrupt disable	Disable interruptions from the DSP board,
Send command	Send a command (start, stop, save, and replay) to a specified C40 module.

5.5 The RTCAMS Windows Application

The real-time CAM application is a program interface that let the user interactively view the acoustic map and the signals on two separate windows, as shown in Fig. 5-4. The implementation of the application is based on the object oriented design and is written with the Microsoft Foundation Classes (MFC) in Visual C++ 1.51.

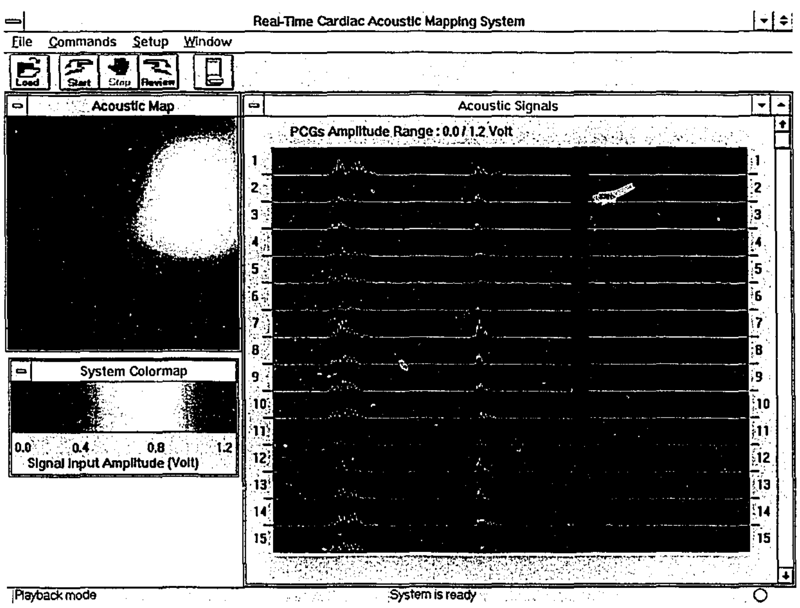


Fig. 5-4 RTCAMS Windows Application display.

Table 5-3 Classes of the RTCAMS Windows Application

Classes	Descriptions	Functions
CmappingApp	Class Mapping Application	Main function of the RTCAMS Windows Application
CMainFrame	Class Main Frame	Control the Main Frame Window of the RTCAMS Windows Application
CsignalWnd	Class Signal Window	Control the PCG Windows and the display of the PCGs
CmappingWnd	Class Mapping Window	Control the Mapping Windows and the display of the CAM
Control_DLL	Class Control DLL	Provide the access to the function of the DLL
CDIBitmap	Class Device Independent Bitmap	Provide functions to manipulate the Device Independent Bitmap (DIB)
CWinG	Class Win Game	Provide the access to the WinG library

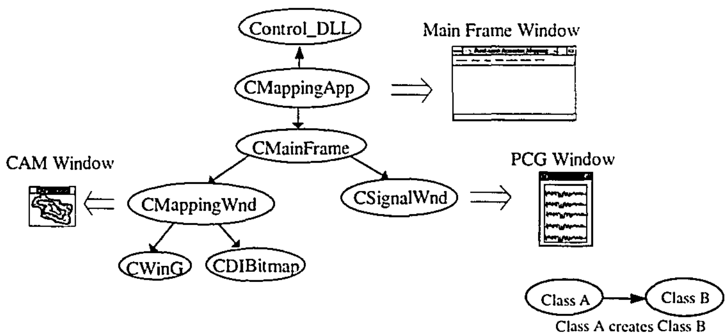


Fig. 5-5 The class diagram of the RTCAMS Windows Application

There are seven major objects in the application which are defined from their respective classes, as shown in Table 5-3 and Fig. 5-5. The CMappingApp is the main class of the RTCAMS Windows Application and it performs different level of initialization because it is the first object created when starting the application. The initialization consists of the following procedures:

1. Create the object Control_DLL for access to the functions provided by the DLL.
2. Reboot the DSP board and the four C40 modules.
3. Load the base addresses of Block 1 and LIA Block to the BAR register in order to access the registers of these two blocks.
4. Disable the hardware interrupt from the DSP board.
5. Select the communication port 1 of the C40 module A and B for interrupting the host PC.
6. Retrieve from the *mapping.ini* file the upper and lower limits for the display of the PCGs on its window area and the number of data points to merge for the construction of the PCGs, as explained in Section 4.4.4.3. The *mapping.ini* file is an initialization file for the RTCAMS Windows Application.

7. Create a CMainFrame object and display the Main Frame Window of the RTCAMS Windows Application. The Main Frame Window is the application interface without the PCG Window and the CAM Window.
8. The Main Frame Window then creates the CSignalWnd and the CMappingWnd which display the PCG Window and the CAM Window, respectively.
9. The CSignalWnd allocates a buffer and identifies it to the VDSPD as the PCG buffer.
10. The CMappingWnd creates two objects; the CDIBitmap and the CWinG. The CDIBitmap allocates a buffer and identifies it to the VDSPD as the CAM buffer. The CWinG allows the use of the WinG library to speed up the image display.
11. The CMainFrame now waits for any action from the user.

The CMainFrame has the control of the menu bar, and the status bar at the bottom of the Main Frame Window (see Fig. 5-4). Therefore, it performs an action when the user clicks on the menus. There are four menus in the RTCAMS Windows Application as shown in Fig. 5-4 and two of them are common in all Windows based applications. The two common menus are the *File* menu which allows the user to save (or to retrieve) an acoustic data to (from) a file, and the *Window* menu to select a window to be placed in front of the others. The other menus are the *Command* menu to send commands (start, stop, save, and replay) to a C40 module, and the *Setup* menu to enter parameters used by the application (the lower and upper limits for the PCG graphics and the number of point to be merged for the construction of the PCGs).

After the RTCAMS Windows Application has been started and the initialization of the CMappingApp has been performed, the user can set up the upper and lower limits for the PCGs on its display window area and the number of data points to merge for the construction of the PCGs from the *Setup* menu. Before invoking the *Setup* menu, the system must stop running. The new setup information is then saved into the *system.ini* file. This file is used to store the setup parameters that can be recalled by the RTCAMS Windows Application the next time it starts.

At this point, the application is started with the *Start* command through the following procedures:

1. Download the executable code to the C40 modules.
2. Verify that all four C40 modules are booted correctly and are ready to execute.
3. Signal to the four C40 modules to begin their program execution.
4. Send the currently opened window(s) to the C40 module B by sending the service ID NO. 4 first, as explained in Section 4.4.2.1.
5. Send the upper and lower limits for the display of the PCGs on its window area and the number of data points to merge for the construction of the PCGs. This is done by first sending the service ID NO. 5 to the C40 module B following by the upper limit, the lower limit, and the number of data points to merge.
6. Send the *Start* command ID NO. 1 to the C40 module B to start the system and the data acquisition board.
7. Enable the hardware interrupt from the DSP board to the host PC.

At this point, the user should see the PCGs and the CAM on their respective windows. To save the digitized cardiac data, the user needs to click on the *Save* option from the *Command* menu which sends the command ID NO. 1 to the C40 module A, as explained in Section 4.3.2.1. As the PCG graphic keeps scrolling on its window, its color will change from its original yellow color to the gray color when recording. The gray colored PCGs are the PCGs that are saved into the memory of the C40 module C and D. After approximately 10 sec of recording, the color of the PCGs returns to yellow. The user can repeat this saving procedure as many times as necessary to record the adequate PCG; the previous recording being overwritten and lost.

To save the recorded PCGs onto a disk file, the user must first stop the system with the *Stop* option in the *Command* menu. This option sends the stop command ID NO. 3 to the C40 module B. The user then selects the *Save* option from the *File* menu and enters the file name to which the cardiac data are saved. This option sends the command ID NO. 2 to the C40 module A and empties the registered PCG Graphic Buffer.

This buffer is used to store temporarily the digitized cardiac data sends by the C40 module A before storing into a specified disk file.

To retrieve, compute, and display the PCGs and the CAMs of the cardiac data saved into a disk file, the user opens a source file with the *Open* option in the *File* menu. Then, he starts the system with the *Replay* option in the *Command* menu. This option sends the command ID NO. 3 and 2 respectively to the C40 module A and B, as explained in Section 4.4.2.1. To stop the system, the user must use the *Stop* option in the *Command* menu as explained earlier.

The implementation of the classes of the RTCAMS Windows Application is not described in this thesis because they are part of Windows programming and they are discussed in many Visual C++ programming books. Instead, the methods of displaying the PCG and the CAM graphics, which are specific for this application, are emphasized and explained in the next section.

5.6 Displaying graphics in Window 3.1

While enhanced graphic display is the basis of the Windows user interface, it remains one of the Windows worst performance functions. There are several reasons for the slow graphic performance of Windows. The most basic one is that Windows must satisfy the needs of every running applications. Some applications must interact with the user; some may be time-critical, and others calculation intensive, or resource (memory, devices) demanding. Windows must allocate its limited resources so as to keep all these applications running. One result of this multitasking is that Windows cannot afford to give an application direct access to the video hardware, thus slowing down the graphic display. To partially overcome this problem, a fast animation technique of multimedia and game applications was applied in this project.

5.6.1 Fast Animation Technique

To implement a fast animation with acceptable results (smooth and flicker-free animation), the double-buffering technique was used [17]. This technique requires the RTCAMS Windows Application to manage a buffer to construct an entire CAM Window image. The CAMs are drawn directly to that buffer instead of being displayed to the monitor. When finished, the image is copied from the buffer to the video adapter memory which directly defines the CAM Window image, updating the display accordingly. This fast block-copy operation is done by a Microsoft library (WinG) which provides fast access to the video adapter memory.

This technique allows a smooth animation because the entire CAM is transferred at once to the graphic adapter memory. This smooth animation can not be achieved if the CAM received from the DSP board is drawn directly into the video adapter memory due to the slow ISA bus and Windows which does not allow direct accesses to the memory of the video card.

The buffer allocated for the entire CAM Window image is called the memory device context which has a "display surface" existing only in memory. The display surface is in fact a device-independent bitmap (DIB) that contains the CAM. The DIB was introduced in Windows 3.0 and it was designed to address bitmap portability. A DIB defines a bitmap's dimension, its color map, and its pixels in a single structure (see Fig. 5-6). The CDIBitmap (see Section 5.5) allocates a contiguous memory space for a DIB that can handle an 80x80 image size, coded to 8-bit per pixel, and a palette of 256 colors. The memory space used to store the image pixels is registered to the VDSPD as the buffer to store CAM transferred by the C40 module B. Any changes made to that bitmap are directly reflected in the display surface of the memory device context.

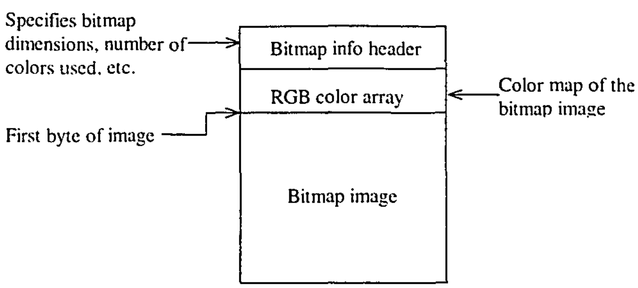


Fig. 5-6 The Device Independent Bitmap structure.

To get the bitmap on the display once the CAM is received from the DSP board, the function *BitBlt* (stand for bit-block transfer) of CWinG is called. This Microsoft library function stretches the DIB image to fit the size of the displayed window and maps the result into the video memory card. The size of the display area of the CAM Window is set by default to 300x300 pixels. The display area can be changed from 80x80 pixels to a maximum of 300x300 pixels.

5.6.2 The Color Palette of the Cardiac Acoustic Map

The displayed CAM is pseudo-color encoded by the color map of the DIB. The colors are a variant from the Hue-saturation-value color map that has its color varying from dark blue to dark red, passing through light blue, green, yellow and red. The color map is an array of 256 RGB colors that can be indexed by the image pixels received from the C40 module B which are encoded into 8 bits per pixel. The dark blue color indicates a low intensity acoustic signal and the dark red is for a maximum intensity. A corresponding palette is then created for the application. This color map is also referred as the logical palette of the RTCAMS Windows Application.

Windows GUI offers a palette manager to map the logical palette to the hardware system palette. This operation requires each pixel to index the logical palette which is then translated by the palette manager to the hardware system palette before display. In a real-time application, the palette manager must be bypassed because it gives a poor performance. To allow this, the hardware system palette must have exactly the same colors as the logical palette. This procedure, however affects other applications which can have their color changed, but this technique greatly enhances the video performance. which is the first priority in this application.

5.6.3 Displaying the PCG Graphics

The twenty-five acoustic signals received by the host PC can be displayed on the PCG Window with little computation. These signals are processed and conditioned by the C40 module B such that the host PC can draw the received acoustic signals onto a

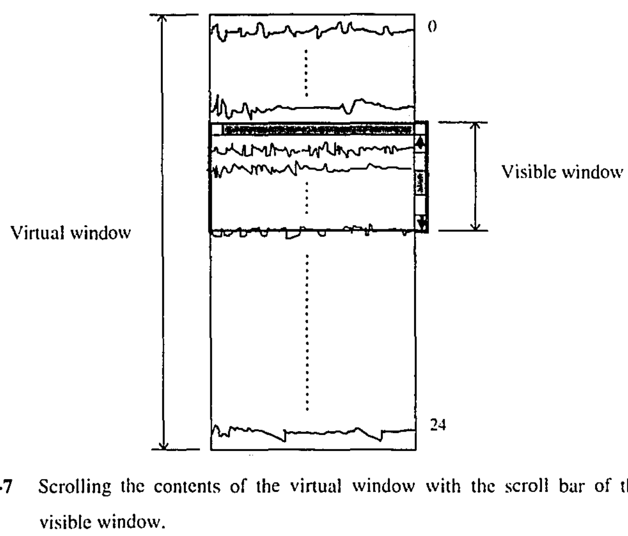


Fig. 5-7 Scrolling the contents of the virtual window with the scroll bar of the visible window.

virtual signal window. Unfortunately, the virtual window cannot be directly mapped onto the visible signal window which has a vertical length smaller than the virtual window, as shown in Fig. 5-7. Even on a 1024 x 768 pixel screen, the visible window cannot be stretched enough to display all the twenty-five signals. This problem is solved by adding a scroll bar on the visible window such that the contents of the virtual window can be scrolled up and down to display different groups of PCG signals.

The display software of the PCGs can display up to 14 channels at one time, each in its own graph area (see Fig. 5-4). The speed of plotting is of paramount importance in this project because the display should appear as a smoothly flowing strip chart. The PCGs are plotted using the CRT mode, simulating the sweep of an oscilloscope. The PCG Graphic Buffer contains a series of 25 interleaved PCGs. The size of this buffer is equal to the length of the PCG Window multiplied by 25x2 (pair of data per vertical line for each PCG). The CRT mode plots these vertical lines versus the indices of the corresponding PCG in the PCG Graphic Buffer. As the PCG Graphic Buffer is a circular

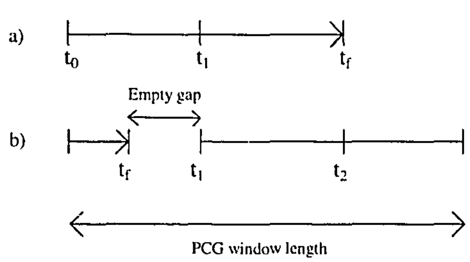


Fig. 5-8 The display process of the graphical PCG on the PCG window with t_0 marking the beginning of the acquisition and t_f the end. Two cases are presented: a) the graphical PCG that is displayed when the buffer is not full; and b) when the buffer is full (overwriting the oldest data by the new data acquired and a gap).

buffer, the new data is stored continuously at the end of the buffer overwriting the oldest data of the buffer. This process is reflected on the PCG Window: the PCGs are plotted from the left to the right of the displayed window, as illustrated in Fig. 5-8. In order to distinguish the new from the last PCG displayed when the buffer is full, a gap overwriting the oldest acquired data is placed between the last sample acquired (t_f) and the first sample (t_l) of the buffer, as shown in Fig. 5-4 and Fig. 5-8.

5.7 Conclusion

The VDSPD is developed to allow direct access to the DSP board within Windows 3.1. This virtual device driver is highly optimized and is in charge of the data transfer between the DSP board and the host PC. The display software of the CAMs is the main factor that limits the overall performance of the RTCAMS and it relies heavily on the WinG library from Microsoft to provide a fast copy of a memory DIB onto the screen. In the present implementation, the RTCAMS is capable of rendering 33 frames per second for a CAM size of 300x300 pixels. In practice, no other Windows application and DOS application should be running at the same time with the RTCAMS application to avoid sharing the limiting resources (memory, devices, etc.) of Windows 3.1 and slowing down the performance of the RTCAMS.

Chapter 6

Test Results

6.1 Introduction

The performance of the DSP system software is presented first. It is followed by a description of the tests on the overall RTCAMS by recording the heart sounds from three normal subjects. The chapter concludes with a discussion on the operating system.

6.2 DSP system software

A performance benchmark of the DSP B was first obtained to estimate the performance of the DSP system software, since the PCGs and CAMs are computed by the module B. The benchmark consists of measuring the percentage of time spent by the PCG Graphic Process and the Acoustic Mapping Process. These two tasks are the only time-consuming processes. Because the program running on the host PC is limited to read the graphics sent by the C40 modules, the host PC software does not influence the DSP software performance. However, the data transfer from the DSP B to the host PC is taken into account in the benchmark. As specified in Section 4.4.4.2, the host PC may decrease the benchmark of the Acoustic Mapping Process if the CAMs are not transferred fast enough due to the slow display of the CAMs on the monitor. Table 6-1 gives an average result of the benchmark taken from ten program executions having a duration of four minutes each.

Table 6-1 Benchmark results for the PCG Graphic Process and the Acoustic Mapping Process.

Task	Time spent to perform the task (msec)	Process activation frequency	Task priority	Time spent after 10 sec of execution (sec)	DSP B resources used (%)
PCG Graphic Process merging 20 data points to produce 25 pair of data for 25 PCGs	2.34	DSP B idling	Background (lowest priority)	3.38	34
Cardiac Acoustic Mapping Process to generate one CAM	17.6	33	Front end (highest priority)	5.28	53

In this study, The PCG Graphic Process merges 20 data points to form one displayed line. In general, merging 20 to 40 data points is considered to be necessary to display the PCGs at a convenient speed. Merging 20 data points takes more time than merging 40 data points because it produces more displayed lines for the same time period. To produce twenty-five PCG displayed lines, the PCG Graphic Process takes about 2.34 msec. To compute one CAM, the Acoustic Mapping Process needs 17.6 msec. Because the PCG Graphic Process is assigned the lowest priority, it is executed in the background when no other task requires the DSP B. However, assigning different priorities to these two tasks does not seem to change the benchmark results. After 10 second of program execution, the PCG Graphic Process spent 3.38 sec which is equivalent to 34 % of the DSP B resources, and the Acoustic Mapping Process spent 5.28 sec or 53 % of the DSP B resources. The remaining 13 % is spent by the DMA Controller Process and the kernel.

6.3 The Overall System

Tests were carried out to determine the performance of the overall RTCAMS, which includes the system hardware described in Chapter 3, the DSP board system software described in Chapter 4, and the host PC system software described in Chapter 5. This benchmark consists of measuring the number of CAMs that can be displayed in one second on its maximum window size of 300x300 pixels while displaying the PCGs at the same time. The results, obtained by averaging the values of ten measurements performed during approximately twenty seconds of recording, show that the RTCAMS always displays 33 frames per second. The RTCAMS was tested by recording heart sounds from 3 normal subjects. The details are presented in the following.

6.3.1 Methods

The recording sessions were carried out in a quiet room. The subjects lay in the supine position and were told to relax. The acoustic probe was placed on the surface of the subject's thorax. The number and the positions of recording sites for producing the CAMs are important points to be considered, as emphasized in previous studies [2,9,16,36]. They are determined according to the hypothesis to be verified, the information desired, and the method of interpolation. In the tests, the acoustic probe was placed on three specific areas: namely the mitral, pulmonary, and aortic areas, as shown in Fig. 6-1. The acoustic probe captures the 25 PCGs over a square area of 100 cm^2 with the configuration of the microphone placement shown in Fig. 6-2. When the RTCAMS was displaying the CAMs and the graphic of the 25 PCG envelopes in real time, the PCG envelopes were carefully monitored to ensure that they have no artifact noise. Such noise is usually caused by the manipulation of the sensitive acoustic probe. When the graphs of the PCG envelopes are consistent, they were saved in the computer memory and the corresponding CAMs were analyzed in detail. All the results to be presented take into consideration the different sensitivity of the microphones by applying the procedure proposed by Cozic [8], on the recorded data off-line.

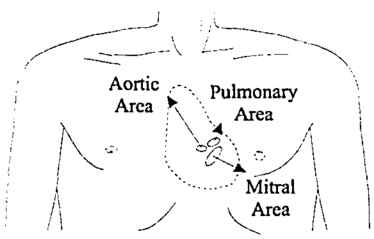


Fig. 6-1 The three auscultation areas used for recording the CAMs.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Fig. 6-2 Arrangement of the microphones on the acoustic probe.

6.3.2 Results

Fig. 6-3 shows the first 16 PCG envelopes recorded at the apex area of subject #1. The chronological positions of S1 and S2 on the PCG envelopes are clearly distinguishable. They are separated by a 220-ms interval, with S1 occurring first. The shape of their amplitude envelopes, their duration, and their number of components vary from one microphone to another. In general, S1 lasts longer than S2 (90 ms vs. 60 ms) and has a lower relative amplitude (0.7 vs. 1.0). In most cases, two components are detected in S1 with the first beginning between 0 and 44 ms followed by the second at 44 ms. S2 also reveals two close components in some envelopes (channels 1, 10, 11, and 13 for example) and a single component for the other envelopes (channels 3, 7, 8, and 9, for example).

Fig. 6-4 and Fig. 6-5 show the CAMs of S1 and S2 computed from the PCG envelopes of Fig. 6-3. Both figures clearly show the evolution of the acoustic pattern. Two components can be observed in Fig. 6-4: the first one between 8 and 44 ms, and the second one between 48 and 72 ms. In the real-time CAMs, the patterns are changing rapidly. The patterns obviously depend on the configuration of the positions of the microphones of the probe. This is a subject matter which needs further investigation, but it is outside the scope of the present project. Two acoustic patterns are also observed in Fig. 6-5. The first one occurs between 324 ms and 340 ms and the second one between 344 ms and 370 ms.

Fig. 6-6 and Fig. 6-7 show the CAMs of S2 recorded at the pulmonary area of subject #2 and at the aortic area of subject #3, respectively. The two figures show two components as in Fig. 6-5 but the propagation patterns are different. In Fig. 6-6, the first component is stronger than the second one. The first component appears at the lower right of the CAM and splits into 3 components (325 ms), which rapidly attenuate at 329 ms. The second one is composed of two peaks that fade gradually (333 - 361 ms). In Fig. 6-7, the first component appears at 364 ms at the bottom of the CAM which is followed by a stronger peak at the lower right of the image (368 ms) and a wider component at the upper left of the image (373 to 381 ms). The second component is composed of two peaks at the bottom of the CAM (386 - 407 ms).

The two main components of the first heart sound of Fig. 6-4 were found to begin at 8 and 44 ms, respectively. These two components correspond to the mitral (M1) and tricuspid (T1) components of S1, as confirmed by the sequence of valvular events demonstrated by Braunwald et al. [5,6] and by Moscovitz et al. [34]. According to many workers in this field [5,6,30,31,34], the second component following the first by 30 to 40 ms can be attributed to the closure of the tricuspid valve. Similarly, the two components observed in S2 are associated to the closure of the aortic (A2) and pulmonary (P2) valves whose time intervals are found to be 20 ms in Fig. 6-5 (324 to 344 ms), 12 ms in Fig. 6-6 (321 to 333 ms), and 17 ms in Fig. 6-7 (364 to 381 ms). The onset of these two components is known to be greatly influenced by the respiration of the subject as well as by other physiological factors.

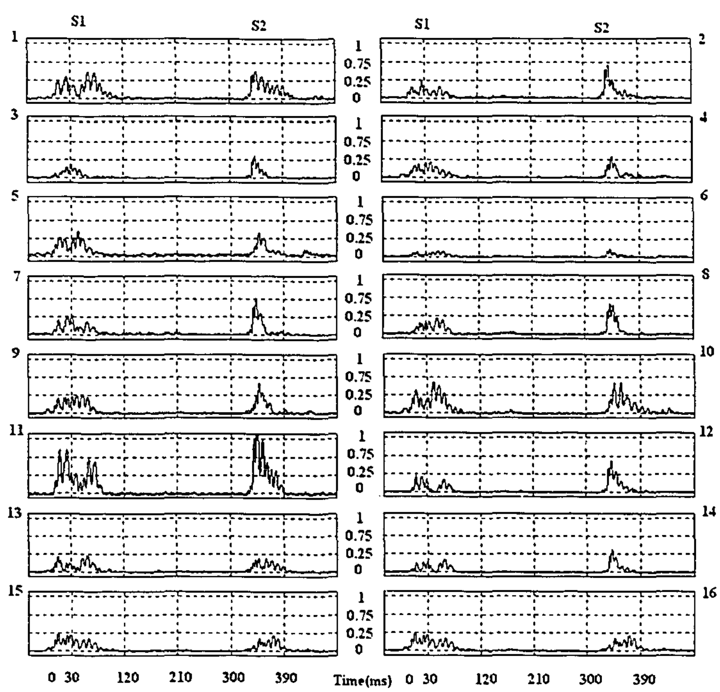


Fig. 6-3 The first sixteen PCG envelopes recorded at the apex area of subject #1. The envelopes shows S1 and S2 with the time reference 0 set at the onset of S1. The maximum amplitude of the envelopes is normalized to 1.0.

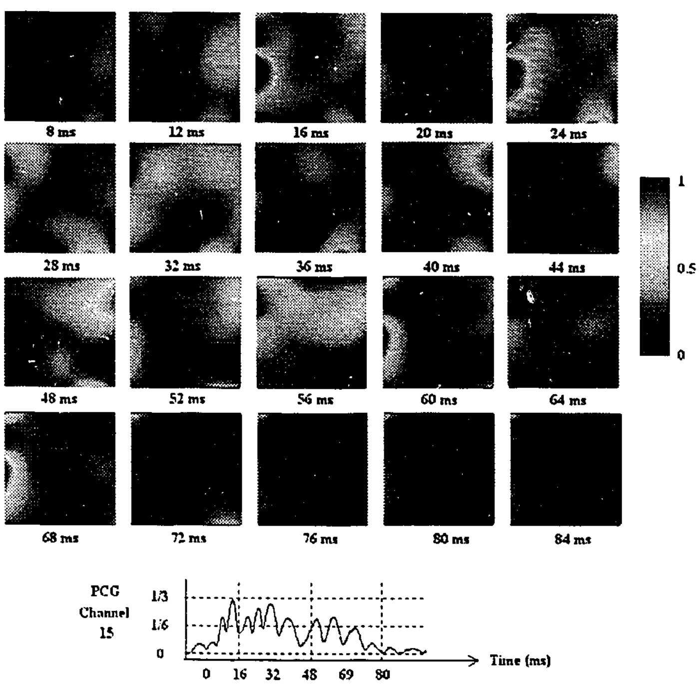


Fig. 6-4 CAMs of the first heart sound of subject #1 recorded at the apex area.

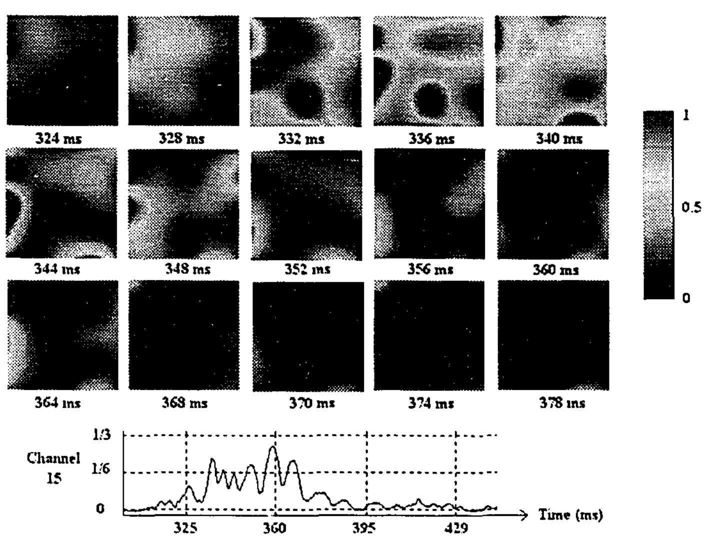


Fig. 6-5 CAMs of the second heart sound of subject #1 recorded at the apex area.

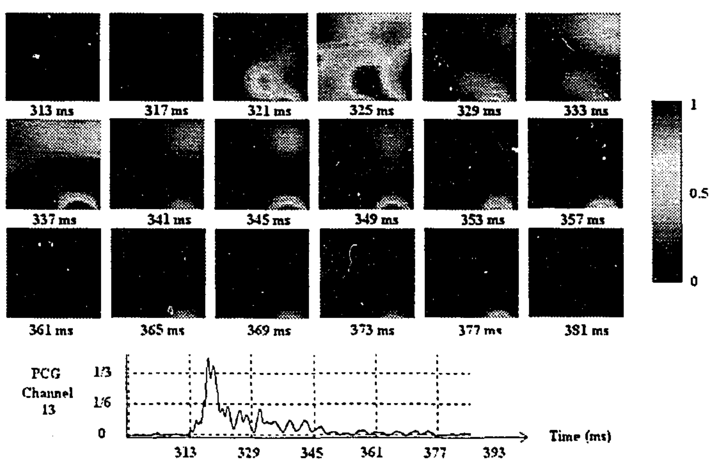


Fig. 6-6 The CAMs of the second heart sound of subject #2 recorded at the pulmonary area.

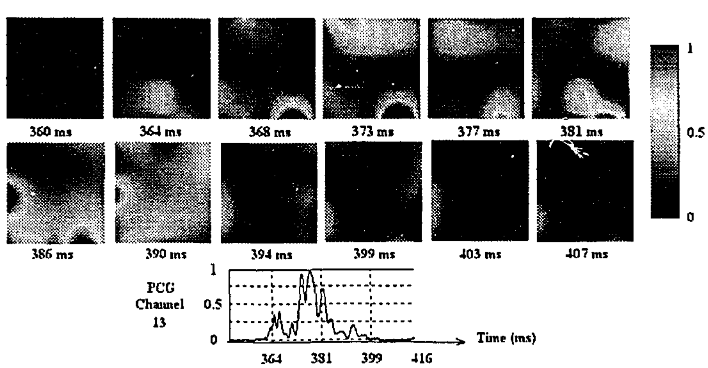


Fig. 6-7 The CAMs of the second heart sound of subject #3 recorded at the aortic area.

6.4 Discussion

The user interface of the RTCAMS does not allow the change of the system parameters while it is processing the PCGs or their envelopes. The system parameters include the amplitude range of the PCGs, the compression factor for the graphical display of the PCGs, and the playback speed of the saved PCGs, as explained in Chapter 4 and 5. In order to change one parameter, the system must be stopped, and thus makes it inconvenient for clinical applications. Removing this limitation is not feasible in Windows 3.1[1] which we considered the most suitable available operating system when we started on the project. When the user is changing one of the system parameters, Windows 3.1 allocates all the CPU time to the user's action, which freezes the background tasks of the RTCAMS such as displaying the received graphical PCG envelopes from the DSP board and refreshing the CAMs. This problem can be solved by programming the RTCAMS under a multitasking operating system like Windows 95[1].

Windows 95 would be a better operating system for the RTCAMS for many reasons. First, as it is a 32-bit operating system, data transfers are done in 32-bit instead of 16-bit. A fast data transfer is an important feature to speed up the RTCAMS in which massive graphical data transfers from memory to memory are involved, such as scaling the CAMs to fit the window. However, data transfer from the DSP board to the host PC will be done still in 16-bit due to the ISA bus architecture of the host PC in which the DSP board is connected. Second, there is an improvement in Windows 95 over Windows 3.1 in displaying images in term of speed, a feature that is primordial for the RTCAMS. Third, because Windows 95 is a multitasking operating system, features such as changing the system parameters while the RTCAMS is processing the PCGs would be possible.

[1] Windows 3.1, and Windows 95 are a registered trademark for the operating systems developed by the Microsoft Corporation.

Porting the current 16-bit Windows 3.1 application to Windows 95 can be done without much difficulty. The virtual device driver (VDSPD) of the DSP board can be recompiled with the new assembler, MASM 6.11[2], with a few modifications in the source code in order to allow Windows 95 to recognize it. The recompiled VDSPD with MASM 6.11 could be used either in Windows 3.1 or Windows 95. Note that the current VDSPD is already a 32-bit device driver. In the dynamic link library (DLL), only the part of the software that communicate with the VDSPD requires some modifications, since it is coded in assembler and manipulating only the first 16-bit of the CPU registers. After these modifications, the DLL must be recompiled with Visual C++ 2.0. The RTCAMS application can be recompiled as is, or with just a few modifications performed with Visual C++ 2.0. The software for the DSPs requires no change.

[2] MASM is a registered trademark of Microsoft Corporation.

Chapter 7

Conclusion

7.1 Summary

Previous studies in cardiac acoustic mapping involve the sequential acquisition of the PCGs, envelope detection and synchronization, interpolation, and display. The mapping process required long time consuming tedious work on the data acquired on the surface of the thorax. In this thesis, the design, implementation, and testing of a system for real-time cardiac acoustic mapping are presented. This system displays in real-time a visual distribution of the PCG amplitude on the surface of a human chest.

The real-time cardiac acoustic mapping system (RTCAMS) was implemented by using dedicated hardware and software. An acoustic probe with 25 integrated highly sensitive microphones is used to record the PCG from the surface of a subject's thorax. The 25 recorded PCGs are then transferred to an analog demultiplexer/filter via a small cable by using time-division multiplexing. The analog demultiplexer/filter reconstructs and filters the 25 PCGs. A specially designed PCG pre-processing board extracts the envelopes from the 25 PCGs and multiplexes them prior to their analog-to-digital conversion.

In this system, data acquisition and upload to a high-speed DSP board are processed in parallel with the computation of the CAMs, the computation of the 25 graphical PCGs or their envelopes, and the transfer of both graphical data to the host PC. The two parallel processes are distributed in two TMS320C40s on a Spectrum DSP board. The three essential components of the RTCAMS are the data acquisition performed by the first DSP, the computation algorithms of the CAMs and the graphical display of the CAMs and the PCG envelopes performed by the second DSP. All three algorithms are written in assembler to improve the overall speed of the system. A real-time micro-kernel is implemented in the second DSP to improve the use of the DSP resources among the following three parallel tasks: the computation of the CAMs, the computation of the graphical PCGs, and the transfer of both graphical data to the host PC.

The RTCAMS uses Windows 3.1 for its graphical user interface. To support a real-time display of the CAMs and the graphical PCGs, three special techniques were incorporated in Windows programming. First, because of the enormous data transfers, a virtual device driver was developed to speed up the data transfer between the host PC and the DSP board. Second, since Windows 3.1 has a poor performance in displaying images in real time, a double buffering technique was used. Third, the Windows 3.1 system palette was modified to map exactly the color palette of the RTCAMS to avoid Windows 3.1 doing the color translation prior to the display. The RTCAMS was tested by recording heart sounds from 3 normal subjects. The results show that the system can process the CAMs in real time (33 CAMs per seconds).

7.2 Suggested Future Developments

The main task of the present project was on the development of the software for the RTCAMS and thus certain limitations of the system have not been dealt with. First, the acoustic probe is sensitive to ambient noise and therefore recording must be performed in a quiet room. The acoustic probe has a thin layer of silicon which covers the surface of the open cavities of the air-coupled microphones (see Appendix B). This material is used to electrically isolate the subject from the microphones and to facilitate the cleaning of the acoustic probe. However, it has the drawback of attenuating the sensitivity of the microphones by approximately 10 dB. In addition, the sensitivity of the microphones varies from one channel to another with differences in amplitude of +6/-12 dB, as shown in Appendix D. This variation should be compensated on-line by the procedure proposed by Cozic [8], for example.

The same gain factor is integrated in each PCG channel of the present system. However, the intensity of the PCGs reaching the surface of the thorax is variable depending on the subject and the recording location on the thorax for a given subject. Therefore, it would be better to have a software controllable gain amplifier for each PCG channel in order to scale the PCGs to the input amplitude range of the data acquisition board and thus maximize the signal-to-noise ratio. A user interface allowing the software to control the gain of each channel at any time should be implemented.

One important feature that needs to be included is the acquisition of the ECG as a reference for determining the cardiac events (systole and diastole). Integrating the ECG in the system software can be done easily because it can be viewed as an additional PCG signal. For the system hardware, one PCG channel with a permanent bypass of the PCG envelope detector could be used. Indeed, the pre-processing board was originally designed to accept up to 32 input channels. The ECG would allow to delimit the cardiac cycles and perform averaging of the PCGs on multiple cardiac cycles. Ensemble averaging would improve the quality of the CAMs, and it should be a part of the future development.

References

[1] **J. Abram,** "Current concepts of the genesis of heart sounds I. First and second sounds", *JAMA*, Vol. 239, pp. 2787-2789, 1978.

[2] **D.T. Barry, and J. Wood,** "Time frequency transforms of the human first heart sound", *Annual Int. Conf. of the IEEE Eng. In Med. and Biol. Society*, Vol. 13, pp. 2097-2098, 1991.

[3] **Y. BNY,** *Sceptre*, the French computing-standards bureau, 1982.

[4] **J.F. Bortolotti, P. Bernard, and E. Bouchet,** "RTMK : A real-time microkernel", *Dr. Dobb's Journal*, pp.70-77, May 1994.

[5] **E. Braunwald, A.P. Fishman, and A.F. Cournand,** "Time relation of dynamic events in the cardiac chambers, pulmonary artery and aorta", *Circulation Research*, Vol. 3, p. 100, 1956.

[6] **E. Braunwald, and A.G. Morrow,** "Origin of heart sounds as elucidated by analysis of the sequences of cardiodynamic events", *Circulation*, Vol. 18, p. 971, 1958.

[7] **K. Chihara, A. Kitajima, and Y. Sakurai,** "Considerations on imaging method of diagnostic information of heart sounds", *Letters, Trans. of the IECE of Japan*, Vol. E63, no. 9, pp. 670-671, Sept. 1980.

[8] **M. Cozic,** *Réalisation d'un système de cartographie cardiaque sonore destiné au dépistage des pathologies valvulaires*, École Polytechnique, Université de Montréal, 1994, thesis/dissertation.

[9] **M. Cozic, L.G. Durand, and R. Guardo,** "Development of a cardiac acoustic mapping system", *Proc. 17th Annual Int. Conf. IEEE Eng. Med. & Biol. Soc.*, pp. 1029-1030, 1995.

[10] **E. Craige,** "On the genesis of heart sounds. Contributions made by echocardiographic studies", *Circulation*, Vol. 53, pp. 207-209, 1876.

[11] **L.G. Durand,** "Digital signal processing of the phonocardiogram". *IEEE Eng. & Biol. Soc.*, 9th Ann. International Conf., pp. 1-2, 1987.

[12] **L.G. Durand, and R. Guardo,** "A model of the heart-thorax acoustic system, applications of computers in medicine", *IEEE Eng. In Med. & Biol. Society*, pp. 29-41, 1982.

[13] **L.G. Durand, Y.E. Langlois, T. Lanthier, R. Chariella, P. Coppens, S. Carioto, S. Bertrand-Bradley,** "Spectral analysis and acoustic transmission of mitral and aortic valve closure sounds in dogs. Part 1 : modeling the heart/thorax system", *Med. & Biol. Eng. & Compt.*, Vol. 28, pp. 169-277, 1990.

[14] **L.G. Durand, and P. Pibarot,** "Digital signal processing of the phonocardiogram — review of the most recent advancements", *CRC Critical Reviews Biomed. Eng.*, Vol. 23, pp. 163-219, 1995.

[15] **P.M. Edic, G.J. Saulnier, and J.C. Newell,** "A real-time electrical impedance tomograph", *IEEE Trans. on Biomed. Eng.*, Vol. 42, no. 9, pp. 849-859, September 1995.

[16] **J.J. Faber, and A.C. Burton,** "Spread of heart sounds over chest wall", *Circulation Research,* Vol. XI, p. 96, July 1962.

[17] **J. Finnegan,** "Implementing games for Windows using the WinG API and the WaveMix DLL", *Microsoft System Journal*, pp. 61-70, Jan 1995.

[18] **D. FRASER,** "Interpolation by the FFT Revised — an experimental investigation", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 37, no. 5, pp.665-675, May 1989.

[19] **K. Hazzah,** "Port I/O under Windows", *Windows Developer's Journal*. pp. 33-39, June 1996.

[20] **P. Heintzen,** "The genesis of the normally split first heart sound", *Am. Heart J.*, Vol. 62, p. 332, 1961.

[21] **H.S. HOU, and H.C. ANDREWS,** "Cubic splines for image interpolation and digital filtering", *IEEE Trans. Acoust., Speech, Signal Processing*. ASSP-26, pp.508-517, 1978.

[22] **B. Hök, K. Nilson, and H. Bjelkhagen,** "Imaging of chest motion due to heart action by means of holographic interferometry", *Med. Biol. Eng. Comput.*, Vol. 16, pp. 363-368, 1978.

[23] **S.M.S. Jalaleddine, C.G. Hutchens, R.D. Strattan, W.A. Coberly,** "ECG data compression techniques — a unified approach", *IEEE Trans. on Biomed. Eng.*, Vol. 37, no. 4, pp. 329-335, April 1990.

[24] **R.G. Keys,** "Cubic convolution interpolation for digital image processing". *IEEE Trans. on Acous., speech, and Signal Processing*, Vol. ASSP-29, no. 6, pp. 1153-1160, August 1981.

[25] **G.Y. Kozmann, And P. Kenedi,** "Surface acceleration mapping a new method for heart sound investigation", 7^{th} *Eur. Congr. Cardiol.*, KFKI 76-35, 1, 1976.

[26] **P.A. Laplante,** *Real-time systems design and analysis an engineer's handbook*, ed. IEEE Press, IEEE Computer Society Press, 1993.

[27] **D. Leong-Kon,** *RTCAM source code version 1.0*. IRCM, Biomed. Department, 1996.

[28] **J.S. LIM,** *Two dimensional signal and image processing*. Englewood Cliffs, NJ: Prentice Hall, 1990.

[29] **A.A. Luisada,** *The sounds of the normal heart*, Warren H. Green, St. Louis, 1972.

[30] **A.A. Luisada, C.K. Liu, C. Aravanis, M.R. Testelli, and J. Morris,** "On the mechanism of production of the heart sounds", *Am. Heart J.*, Vol. 55, p. 383, 1958.

[31] **A.A. Luisada, and G. Bartolo,** "High frequency phonocardiography", *Am. J. Cardiol.*, Vol. 7, p. 51, 1961.

[32] **V.G. McKusick,** *Cardiovascular sound in health and disease*, The William & Wilkins Company, Baltimore, 1958.

[33] **K. Minhas, and B. Gasul,** "Systolic clicks: A clinical, phonocardiographic, and hemodynamic evaluation", *Am. Heart J.*, Vol. 57, p. 49, 1959.

[34] **H.L. Moscovitz, A.J. Gordon, E. Braunwald, S.S. Amram, S.D. Sapir, R.P. Lasser, A. Himmelstein, and M.M. Ravitch,** "The hemodynamics of the left side of the heart as studied by simultaneous left atrial, left ventricular and aortic pressures; particularly in reference to mitral stenosis", *Circulation*, Vol. 12, p. 59, 1955.

[35] **S. Muraki,** "Volume data and wavelet transforms", *IEEE Computer Graphics & Applications*, Vol. 13, no. 4, pp. 50-56, July 1993

[36] **M. Okada,** "Chest wall maps of heart sounds and murmurs", *Comput. Biomed. Res.*, Vol. 15, pp. 281-294, 1982.

[37] **W.K. Pratt,** *Digital image processing*, John Wiley & Sons, New York, 1991.

[38] **R.M. Rangayyan, and R.J. Lechner,** "Phonocardiogram signal analysis : a review", *CRC Critical Reviews in Biomed. Eng.*, Vol. 15, no. 3, pp. 211-236, 1988.

[39] **A.C. Robert,** "A new approach to the interpolation of sampled data", *IEEE Trans. on Medical Imaging*, Vol. 15, no 3, pp. 369-376, June 1996.

[40] **R.F. Rushmer,** *Cardiovascular dynamics*, Fourth edition, W.B. Saunders company, Philadelphia, 1976.

[41] **R.W. SCHAFER, and L.R. RABINER,** "A digital signal processing approach to interpolation", *Proc. IEEE*, Vol. 61, pp. 692-702, June 1973.

[42] **J.A. Shaver,** "Diastolic murmurs, heart disease and stroke", Vol. 2, pp. 98-103, 1993.

[43] **J.A. Shaver,** "Systolic murmurs, heart disease and stroke", Vol. 2, pp. 9-17, 1993.

[44] **R.W.M. Smith, I.L. Freeston, and B.H. Brown,** "A real-time electrical impedance tomography system for clinical use — design and preliminary results", *IEEE Trans. on Biomedical Eng.*, Vol. 42, no. 2, pp. 133-139, Feb. 1995.

[45] *Burr-Brown analog daughter module user's manual*, version 1.03, Spectrum Signal Processing Inc., Nov. 1993.

[46] *C40 network API support for PC systems user guide*, version 1.10, Spectrum Signal Processing Inc., March 1995.

[47] *Daughter module developers' kit designer documentation*, version 1.0, Spectrum Signal Processing Inc., March 1993.

[48] *DSPLINK technical application note*, issue 1.32, Spectrum Signal Processing Inc., Nov. 1991.

[49] *PC daughter module carrier board technical reference manual*, version 1.00, Spectrum Signal Processing Inc., Sept. 1992.

[50] *Quad C40 processor board technical reference manual*, version 1.03, Spectrum Signal Processing Inc., Sept. 1994.

[51] *TIM-40 module and carrier board PC support manual*, version 1.10, Spectrum Signal Processing Inc., March 1995.

[52] *TMS320C40 development system board user's manual*, Special short version manual, Spectrum Signal Processing Inc., March 1995.

[53] *Quad C40 processor board user's guide*, version 1.00, Spectrum Signal Processing Inc., June 1993.

[54] *TMS320C4x User's Guide*, revision c, Texas Instruments, August 1993.

[55] *Microsoft Windows Device Development Kit – Virtual Device Adaptation Guide*, version 3.0, Microsoft Corporation, 1990.

[56] **P.O. Stein, H.N. Sabbah, I.B. Lakier, and S. Goldstein,** "Frequency spectrum of the aortic component of the second heart sound in patients with normal valves, aortic stenosis, and aortic porcine xenografts", *Amer. J. Cardiol.*, Vol. 45, pp. 49-52, 1980.

[57] **H Tao, and R. Moorhead,** "Progressive transmission of scientific data using biorthogonal wavelet transform", *IEEE visualization '94 Proceedings*, Washington, D.C., pp. 93-99, Oct. 1994

[58] **H. Tao, and R. Moorhead,** "Lossless progressive transmission of scientific data using biorthogonal wavelet transform", *IEEE International Conference on Imaging*, Austin, TX, Vol. 3, pp. 373-377, Nov. 1994.

[59] "*TMS320C40 Floating-Point DSP - User's Guide*", Revision A, Texas Instruments, Oct. 1991.

[60] **C.W. Thomas, K. Laurita, G. Sun, J. Liebman, and A.L. Waldo,** "a cardiac potential mapping system", *Journal of Electrocardiology*, Vol. 22, pp. 64-71, 1989.

[61] **A.G. Tilkian, and C.M. Boudreau,** *Understanding heart sounds and murmurs with an introduction to lung sounds*, Philadelphia, W.B. Saunders Company, 1984.

[62] **J. Wiley,** *Communication Systems*, John Wiley & Sons, New York, 1978.

[63] **J. Wiliams,** "Examining the LabOBJX Real-Time Chart Control - Developing trending software for real-time applications", *Dr. Dobb's Journal*, pp.88-92, Oct. 1997.

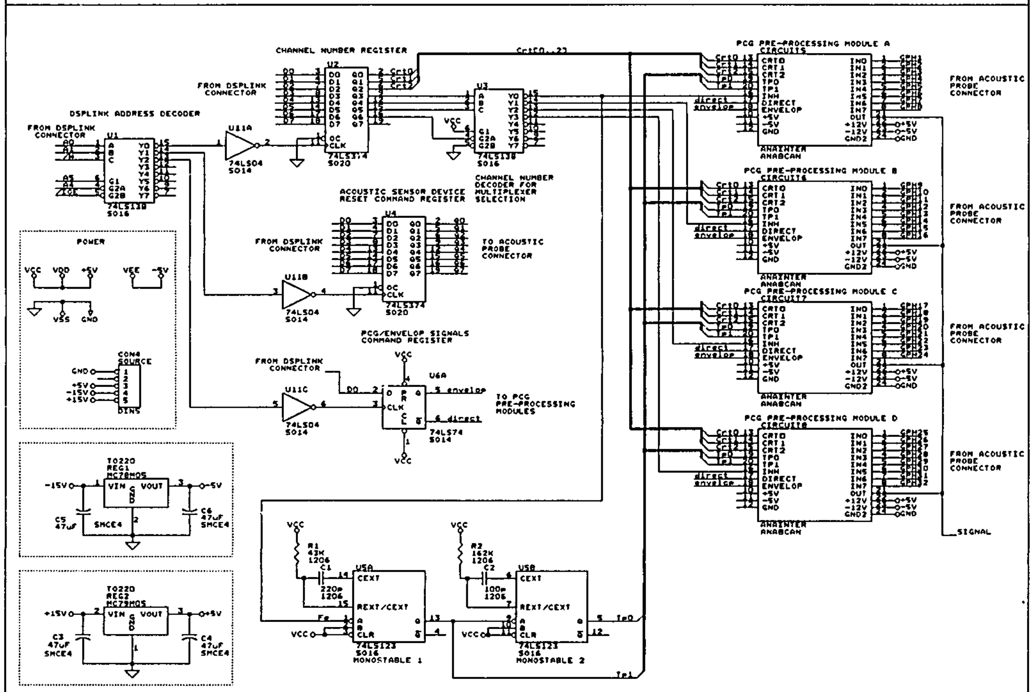
[64] **W. Wulf, and M. Shaw,** *Global variables considered harmful*, Sigplan Notices, pp. 28-34, 1973.

Appendix A

The PCG Pre-Processing Board Schematics

PROJET SONDE 32 CANAUX: PLAQUETTE MAITRESSE

JD. 1997. 02



LABORATOIRE DE GENIE BIOMEDICAL - INSTITUT DE RECHERCHES CLINIQUES DE MONTREAL

TEL: (514) 987-5564

FAX: (514) 987-5705

ADRESSE ELECTRONIQUE: DURANDJ@IRCM.UMONTREAL.CA

CONCEPTION: DANY LEONG ET JOCELYN DURAND,ING.

INSTRUMENT: SONDE 32 CANAUX

DOCUMENT : OrCAD\SOND_97MB\SOND_MBR.SCH

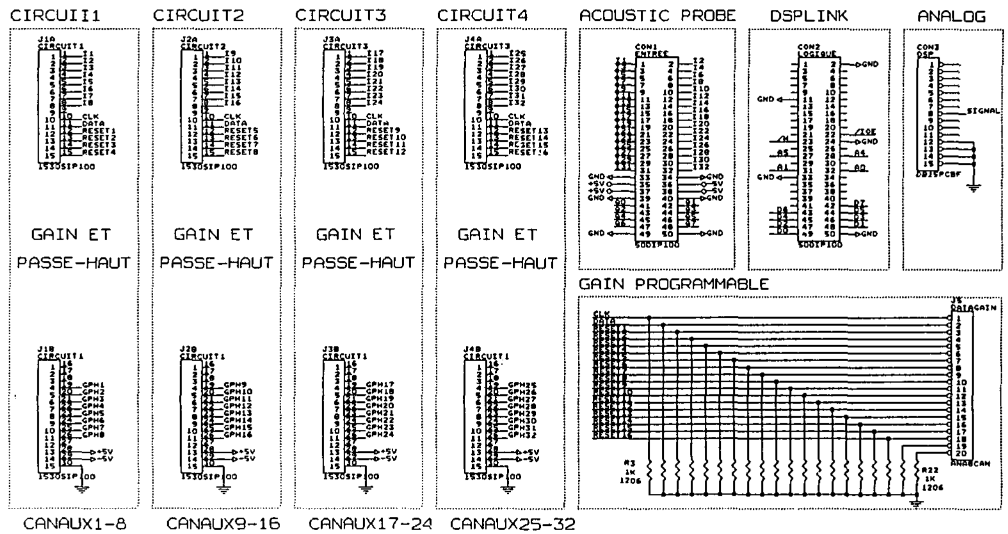
DATE : FEVRIER 1997

FORMAT B

PAGE 1 DE 2

VERSION: 1997.02

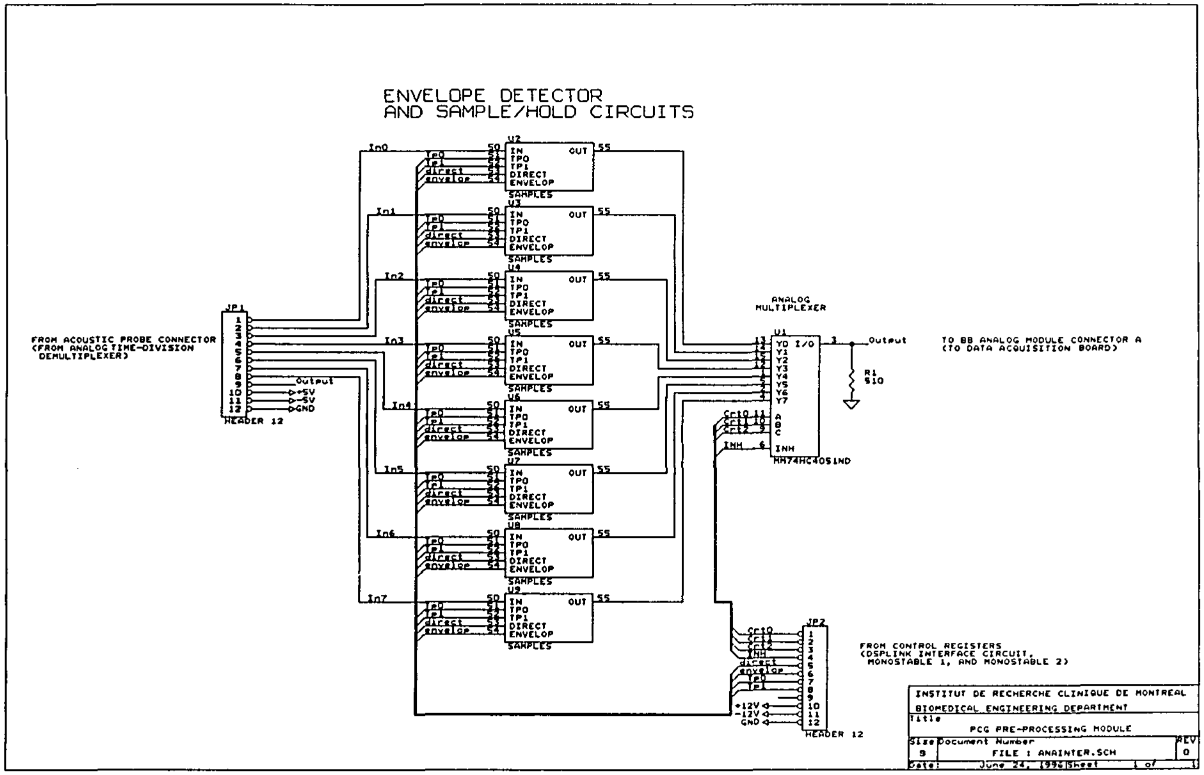
PROJET SONDE 32 CANAUX: RACCORDEMENTS — JD. 1997. 02

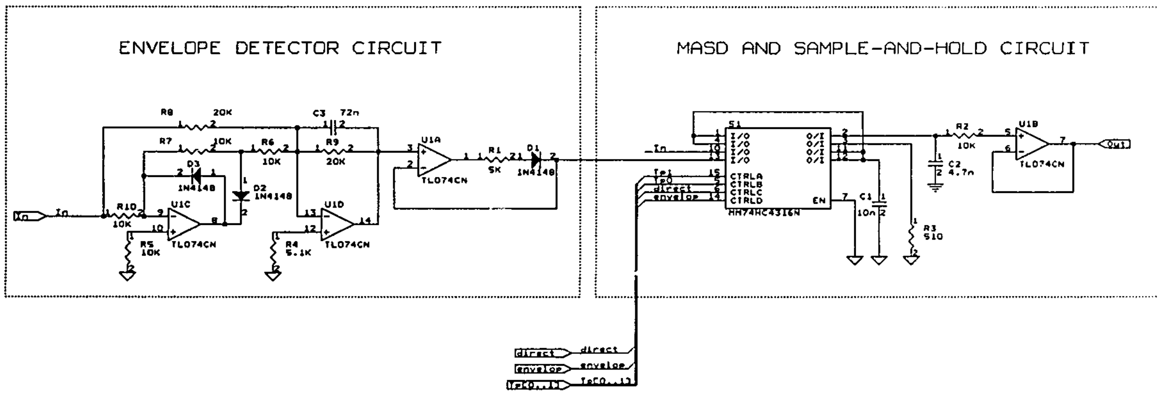


GAIN PROGRAMMABLE ET FILTRE PASSE-HAUT 3eORDRE

LABORATOIRE DE GENIE BIOMEDICAL – INSTITUT DE RECHERCHES CLINIQUES DE MONTREAL
TEL: (514) 987-5564
FAX: (514) 987-5705
ADRESSE ELECTRONIQUE: DURANDJ@IRCM.UMONTREAL.CA

CONCEPTION: JOCELYN DURAND, ING.
INSTRUMENT: SONDE 32 CANAUX — FORMAT B
DOCUMENT : OrCAD\SON_97HB\SOND_HRD.SCH — PAGE 2 DE 2
DATE : FEVRIER 1997 — VERSION: 1997.02

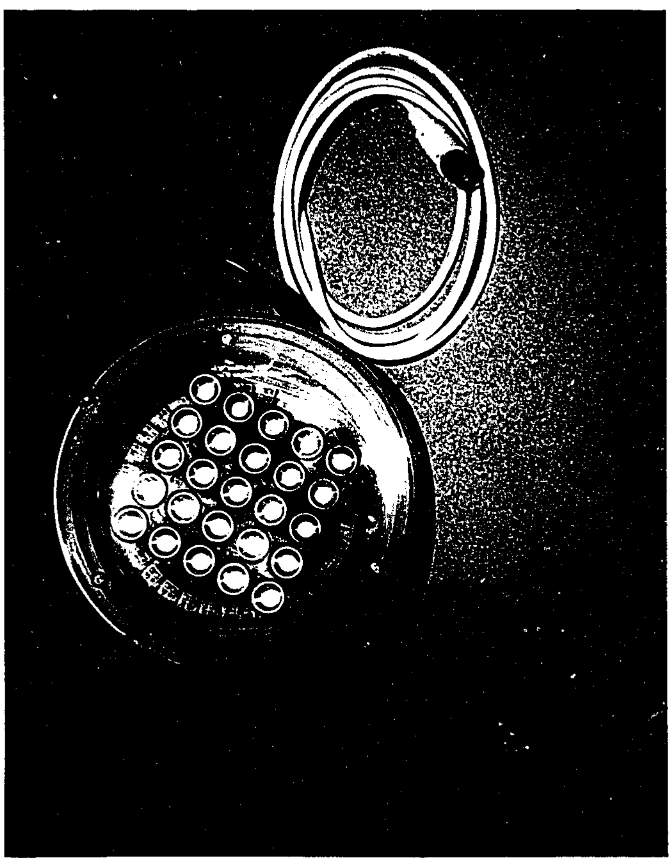

ENVELOPE DETECTOR
AND SAMPLE/HOLD CIRCUITS
FROM ACOUSTIC PROBE CONNECTOR
(FROM ANALOG TIME-DIVISION
DEMULTIPLEXER)
JP1
HEADER 12
Output
+5V
-5V
GND
In0
In1
In2
In3
In4
In5
In6
In7
IN
TP0
TP1
DIRECT
ENVELOP
SAMPLES
OUT
ANALOG
MULTIPLEXER
U1
MM74HC4051ND
INH
R1
510
Output
TO BB ANALOG MODULE CONNECTOR A
(TO DATA ACQUISITION BOARD)
FROM CONTROL REGISTERS
(DSPLINK INTERFACE CIRCUIT,
MONOSTABLE 1, AND MONOSTABLE 2)
JP2
HEADER 12
+12V
-12V
GND
INSTITUT DE RECHERCHE CLINIQUE DE MONTREAL
BIOMEDICAL ENGINEERING DEPARTMENT
Title
PCG PRE-PROCESSING MODULE
Size
Document Number
REV
B
FILE : ANAINTER.SCH
0
Date:
June 24, 1996
Sheet
1 of 1

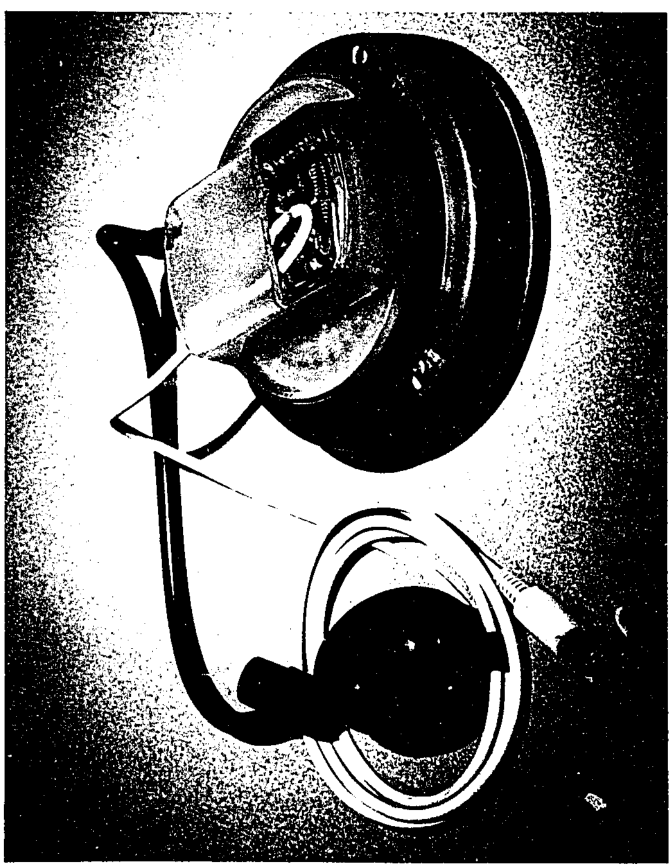


INSTITUT DE RECHERCHE CLINIQUE DE MONTREAL		
BIOMEDICAL ENGINEERING DEPARTMENT		
Title: ENVELOPE DETECTOR AND SAMPLE/HOLD CIRCUIT		
Size: B	Document Number: FILE : SAMPLE2.SCH	REV: 0
Date: June 24, 1996	Sheet 1 of 1	

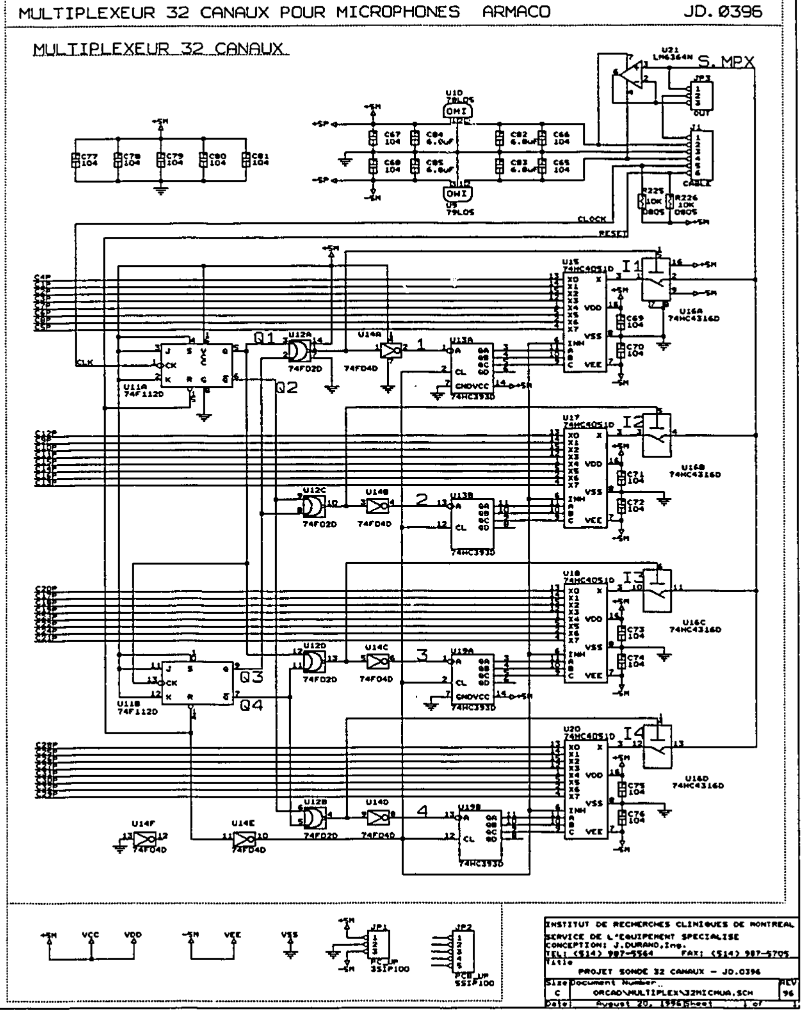
Appendix B

Schematics and Illustrations of the Acoustic Probe

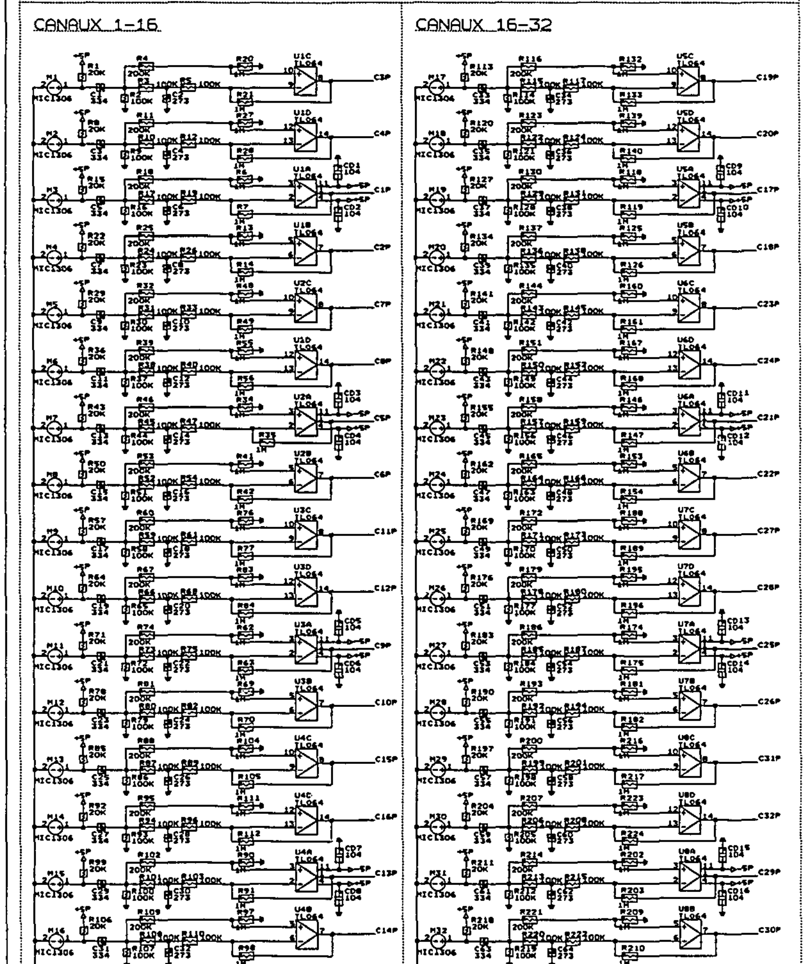




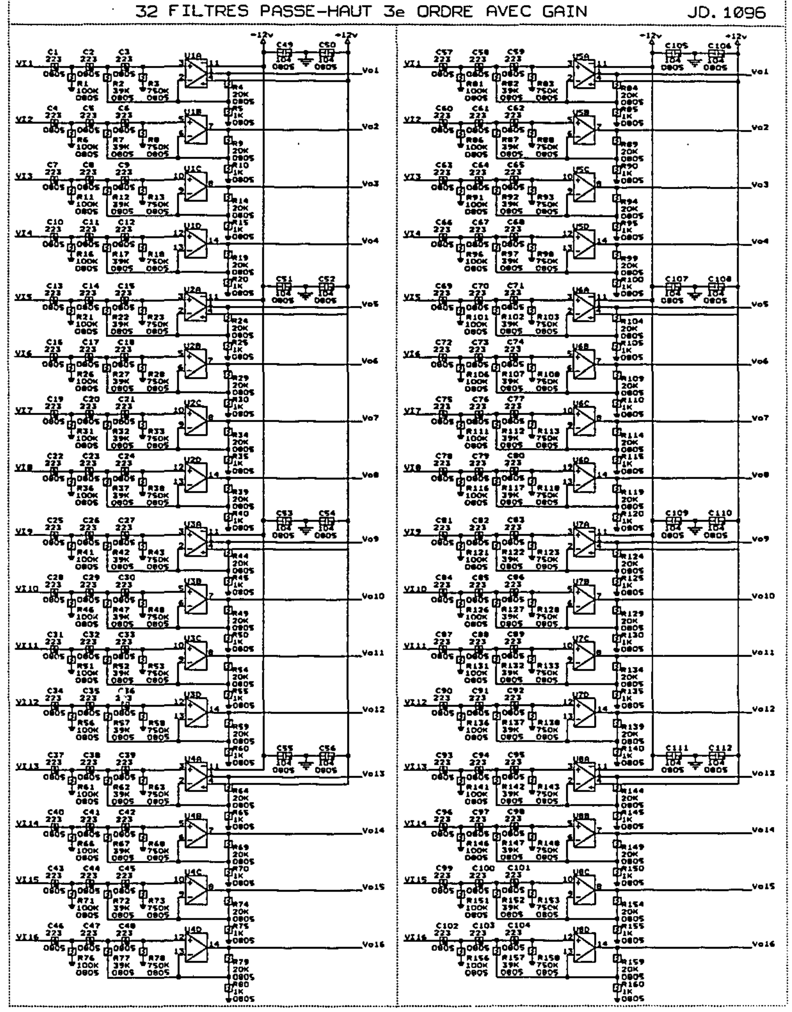
MULTIPLEXEUR 32 CANAUX POUR MICROPHONES ARMACO JD. Ø396



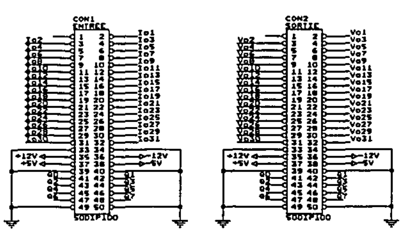
PREAMPLIFICATEUR 32 CANAUX POUR MICROPHONES ARMACO JD. Ø396



32 FILTRES PASSE-HAUT 3e ORDRE AVEC GAIN JD. 1096



CIRCUIT DE RACCORDEMENT JD.Ø496

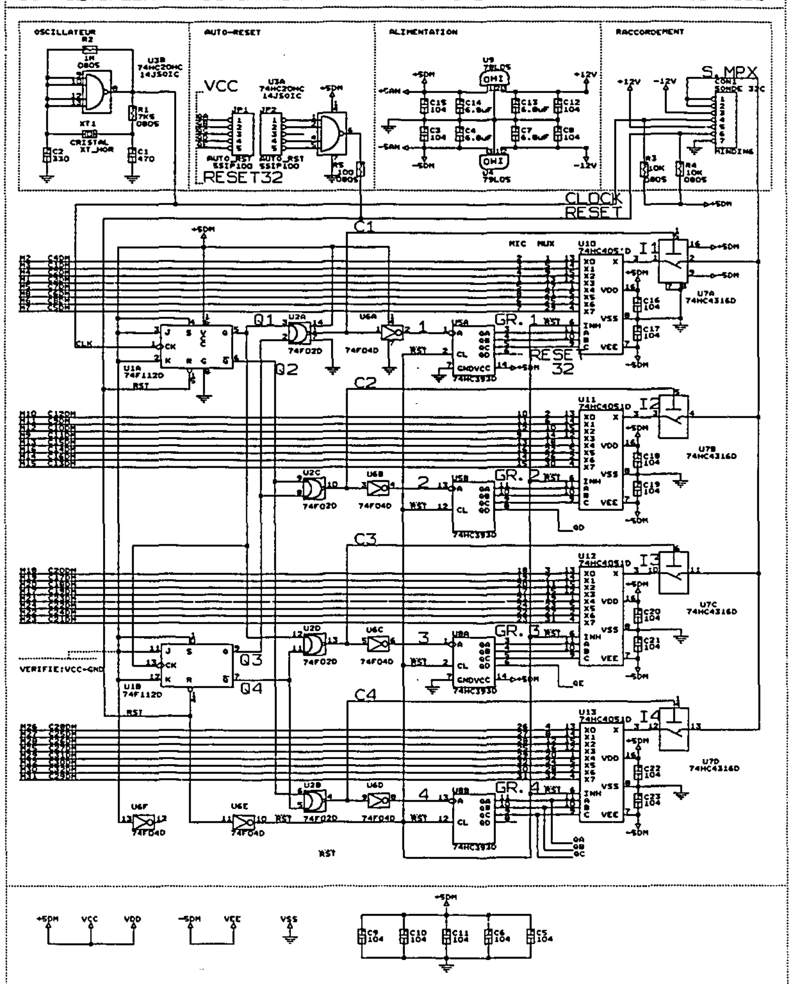


INSTITUT DE RECHERCHES CLINIQUES DE MONTREAL		
INSTRUMENT: SONDE 32 CANAUX CONCEPTION: JOCELYN DURAND, Ing. TEL: (514) 987-5564 TEL: (514) 987-5705		
Title PROJET: FILTRE PASSE-HAUT 32 CANAUX AMPLIFIE		
Size C	Document Number ORCAD\MULTIPLX\32MICFPH.SCH	REV 96
Date: February 6, 1997	Sheet 1 of 1	

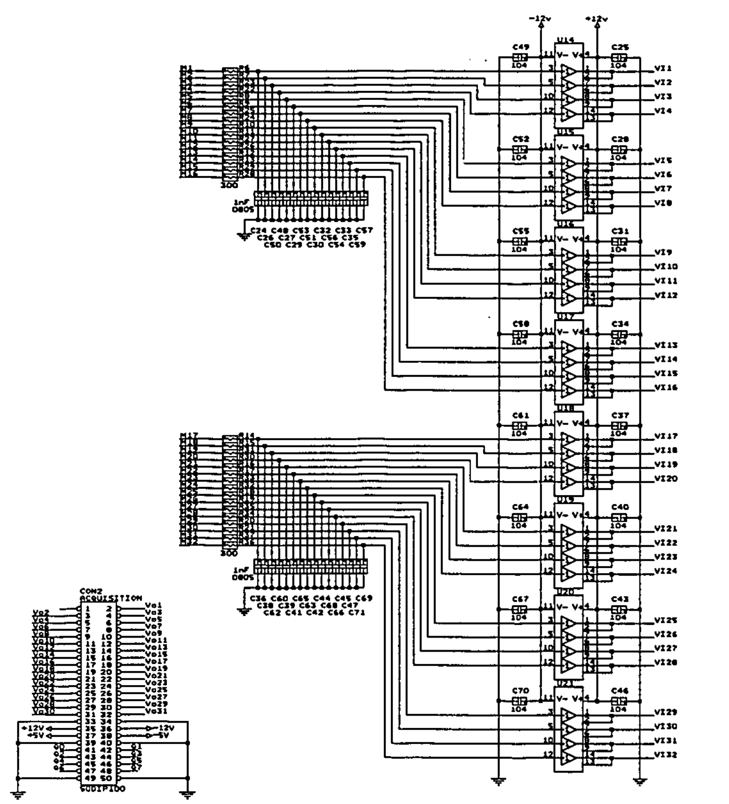
Appendix C

The Analog Demultiplexer Schematics

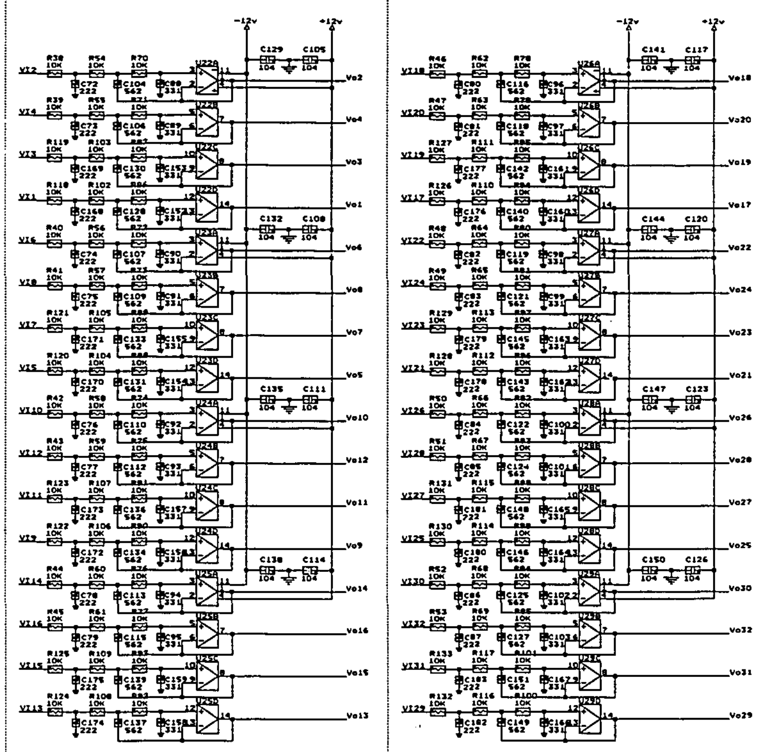
DE-MULTIPLEXEUR 32 CANAUX POUR MICROPHONES ARMACO JD.0396



CIRCUIT DE MAINTIENT 32 CANAUX JD.Ø496



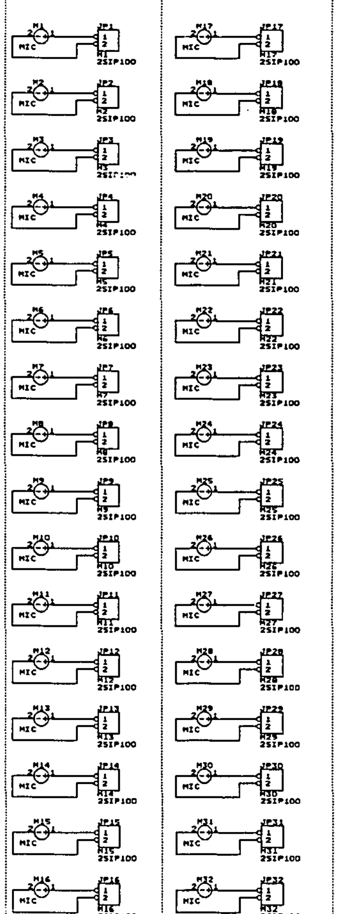
32 FILTRES PASSE-BAS 3e ORDRE @ 10 KHZ. JD.0496-CORRIGE 25.10.96



INSTITUT DE RECHERCHES CLINIQUES DE MONTREAL		
INSTRUMENT: CONCEPTION: JOCELYN DURAND, Ing. TEL: (514) 987-5564 FAX: (514) 987-5705		
Title PROJET: DE-MULTIPLEXEUR 32 CANAUX		
Size D	Document Number ORCAD\MULTIPLX\DMUX32CA.SCH	REV 962
Date: February 6, 1997	Sheet 1 of 1	

PREAMPLIFICATEUR 32 CANAUX POUR MICROPHONES ARMACO JD.Ø896

CANAUX 1-16 CANAUX 16-32



INSTITUT DE RECHERCHES CLINIQUES DE MONTREAL SERVICE DE L'EQUIPEMENT SPECIALISE CONCEPTION: J.DURAND,Ing. TEL: (514) 987-5564 FAX: (514) 987-5705		
Title PROJET SONDE 32 CANAUX - JD.0896		
Size A	Document Number ORCAD\MULTIPLEX\32MICMUB.SCH	REV 96
Date:	August 15, 1996 Sheet 1 of 1	

Appendix D

The Sensitivity of the 25 Microphones of the Acoustic Probe

25 CHANNEL PROBE

Sensitivity of the 25 microphones:

Microphone	DC offset (mvolts)	Signal 100 Hz (mvolts)	Signal 100 Hz 0dB=200mv	Signal 200 Hz (mvolts)	Signal 200 Hz 0dB=200mv	Notes
1	-20	300	+3.50	400		
2	-20	200	0	300		
3	-20	300	+3.50	400		
4	-25	300	+3.50	400		
5	-25	300	+3.50	350		
6	-10	400	+6.00	600	+6.00	strong
7	-10	350	+4.86	400		
8	0	300	+3.50	300		
9	-30	300	+3.50	300		
10	-40	250	+1.94	400		
11	-20	100	-6.00	100		
12	-30	350	+4.86	375		
13	-20	400	+6.00	400		
14	-10	400	+6.00	500		
15	+10	400	+6.00	600		
16	-10	300	+3.50	400		
17	-20	400	+6.00	500		
18	-50	50	-12.00	100	-12.00	weak
19	-30	350	+4.86	450		
20	-40	350	+4.86	400		
21	-20	150	-2.50	200		
22	0	250	+1.94	300		
23	-20	350	+4.86	500		
24	-40	100	-6.00	300		
25	-10	350	+4.86	400		

Appendix E

The PC I/O MAP of the DSP Board

Table1 shows the fourteen registers of the PC I/O map of the DSP board which are used in this project. The *Control* register resets the DSP board and its four C40 modules, and selects the PC IRQ line [41]. The *BAR* register is used to set the base addresses of Block 1 and the LIA Block by software. The next four pairs of *Link Data* registers are the links to the communication ports of the four C40 modules. Each pair of registers can read or write 32-bit data from/to a communication port of a C40 module in two operations, 16-bit at a time, due to the 16-bit ISA bus. The *Status* register has four bits composed of a flag from the DSP of each C40 module, each flag indicating that the corresponding C40 module is ready (1) or not (0) for the execution of the loaded program. The purpose of these flags is to synchronize the execution of all C40 modules by starting them at the same time. In response, when all the modules are ready, the PC sends the "start" signal to the communication ports of all DSPs.

The *Interrupt Status* register contains four bits, each of them, if set, indicates the presence of a data in the corresponding communication port of a C40 module, and it can be read from the corresponding *Link Data* register. The *Interrupt Status* and the *Interrupt Select* registers share the same address in the LIA Block. Thus, reading from that address will give the *Interrupt Status*, while writing to that address will set the *Interrupt Select* register to select one or multiple communication ports of the C40 modules that can cause interrupts to the PC by using the *Interrupt Enable* register. Additional information, can be obtained from the Quad C40 Processor Board Technical Reference Manual [41].

Table 1 PC I/O Map of the DSP board

Address (hex)	Register	Read/Write
Base Address of Block 0 + 10	Control	W
Base Address of Block 0 + 18	BAR	W
Base Address of LIA Block + 00	C_LINK DATA - C40 Module C - (LSW)	R/W
Base Address of LIA Block + 02	C_LINK DATA - C40 Module C - (MSW)	R/W
Base Address of LIA Block + 04	A_LINK DATA - C40 Module A - (LSW)	R/W
Base Address of LIA Block + 06	A_LINK DATA - C40 Module A - (MSW)	R/W
Base Address of LIA Block + 08	D_LINK DATA - C40 Module D - (LSW)	R/W
Base Address of LIA Block + 0A	D_LINK DATA - C40 Module D - (MSW)	R/W
Base Address of LIA Block + 0C	B_LINK DATA - C40 Module B - (LSW)	R/W
Base Address of LIA Block + 0E	B_LINK DATA - C40 Module B - (MSW)	R/W
Base Address of LIA Block + 10	Status	R
Base Address of LIA Block + 12	Interrupt Status/ Select (read/write)	R/W
Base Address of LIA Block + 14	Test	R/W
Base Address of LIA Block + 16	Interrupt Enable	W

LSW = Least Significant Word (16-bit)
MSW = Most Significant Word (16-bit)