ELECTROCARDIOGRAM, HEART RATE AND TEMPERATURE MONITORING SYSTEM

by

Phumzile Malindi

Dissertation submitted in compliance with the requirements for Masters Degree in Technology in the Department of Electrical Engineering (Light Current) at Technikon Natal

This Dissertation represents my own work

P Malindi

APPROVED FOR FINAL SUBMISSION

..... 23/10/2000.

Professor Vladimir B. Bajic, Pr. Eng., D.Eng. Sc.(EE) Date

Supervisor

Durban 14th of October 2000

Acknowledgments

I would like to express my appreciation to Professor Vladimir Bajic for always being there for me. His advice, encouragement, support and valuable suggestions have made this work possible.
Also, my thanks go to MR Stefan Mare, Head of Department of Electrical Engineering at Technikon Natal, for numerous comments and suggestions that contributed to the final results.
I also owe my thanks to Professors Monson H. Hayes and Willis J. Tompkins and Dr Brian Bramer, who though they never met me, have given valuable advice in my work.
I would like to thank the Cecilia Makiwane Hospital, especially senior medical superintendent Dr I Harris, for all the support they have given me.
I would also like to extend my appreciation to Leyworth Health Care (East London) for their donations, support and advice and to Japie Maré for helping me with Matlab software package.
This work would not have been possible without the support of my family, especially my wife for being so understanding and for being the source of inspiration for me at all times.

Finally, I thank God, our Celestial Father, for the life I am living, for the friends and advisors I have. I know that, without Him, nothing would have been possible.

ELECTROCARDIOGRAM, HEART RATE AND TEMPERATURE MONITORING SYSTEM

by

Phumzile Malindi

(ABSTRACT)

The purpose of this study is the development of an affordable computer-based electrocardiogram, heart rate and temperature monitoring system, that would complement those that are available on the market and contribute to the reduction of the shortage of these medical instruments in South African hospitals and clinics.

Electrocardiogram (ECG) refers to the graph that results from time versus voltage in a patient's chest. It reflects the rhythmic activity of the heart. For this reason the electrocardiogram has a diagnostic value that can be used by medical personnel to examine the biological (hence, clinical) behavior of the heart. The electrocardiogram can also be used to get the heart rate.

This thesis explained how to acquire ECG signals from the patient and also how to achieve a cheaper way of providing galvanic isolation, which is required for sensors that are attached to the human body. It also explains computer interfacing using the parallel port and computer-based processing of these ECG signals to determine the instantaneous value of the heart rate and also to reduce the interference that contaminates these signals.

In reducing interference, the performance of traditional IIR notch and adaptive filters, as noise cancelers, has been analyzed and compared. Least Mean Squares (LMS) and Normalized Least Mean Squares (NLMS) algorithms are the two algorithms that were considered in this study for adaptive noise canceling and their performance is evaluated and is compared based on their convergence rate, complexity and noise reduction.

The results obtained revealed that the use of filters can reduce the noise that contaminate the ECG signals and that the performance of adaptive filters produce better signal-to-noise ratio than the notch filter. The results also revealed that, though the NLMS algorithm is more complex than the LMS, the NLMS has a faster convergence rate and a better signal-to-noise ratio than the LMS algorithm.

Contents

List of Figures

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

In South Africa, as well as in the whole continent of Africa, there is a big shortage of medical equipment in hospitals, clinics and other health centers, especially equipment for taking an electrocardiogram. This is mainly because some of this equipment is too expensive and many health centers in Africa cannot afford them. This has resulted in most of the clinics in rural areas being without any facility for taking electrocardiograms of their patients.

Because of the absence of this equipment in clinics or hospitals, the medical staff in those health institutions have to resort to other methods as discussed by Mulder and Nowlan (1986). For example, in order to determine the pulse rate of a patient the medical examiner places the three middle fingers lightly over the selected pulse area and evaluate pulse rhythm by observing the time interval between two consecutive beats. If the rhythm is regular the pulse rate will be determined by counting the number of pulses for half a minute and multiply the result by two to get the actual rate. Alternatively, if the rhythm is irregular the counting is for a minute. This technique does not guarantee accuracy due to inevitable human errors and subjective judgements. Another inherent problem with similar techniques is their inability to ensure continuous monitoring of the patient's heart rate or temperature.

An inexpensive and reliable instrument for measurement and monitoring of the patient's heart rate, temperature and for recording the electrocardiograms can be a solution to some of the above mentioned problems. The instrument should also provide memorizing and printing functions for the recorded data. To achieve this, several trade-offs have to be dealt with. Normally, to make such a design the amount of hardware has to be minimal to keep the cost low. This implies a solution that is predominantly software-based. The hardware design requirements will have to

accommodate the use of isolation amplifiers, which are mandatory for medical electronics in which electrodes are applied to humans as discussed by Horowitz and Hill (1994) and Stulens (1996).

Since temperature and heart rate are closely related, as discussed by Solomon, Adragna and Schmidt (1990) and Shier, Butler and Lewis (1996), their recording has to be done virtually in parallel.

1.2. ELECTROCARDIOGRAPH

The electrocardiograph or ECG machine is an instrument that is used to record the pattern of electrical activity associated with the contraction and relaxation of cardiac muscle during heartbeat. Such a recording of the electrical activity of the heart is a completely noninvasive process that produces an electrocardiogram (ECG), which is a graphic recording of the voltage changes during the heartbeat. The electrocardiogram is useful in diagnosing the state of the heart. Therefore, it can be said that the electrocardiograph permits the clinician to determine electrical and mechanical defects of the heart. (Tompkins, 1995)

The electrical currents generated and transmitted through the heart also spread throughout the body, thus, resulting in electrical potential also distributed at the body surface.
These potentials, on the body surface, are measured using ECG (EKG) sensors that are attached to the body of the patient. These ECG sensors produce electrical signals with frequency range from 0.05 Hz to 100 Hz, which are analogs of the actual signals. These signals, which are about 50 μV in infants and 1 mV to 5 mV in adults, are conditioned and processed by the electrocardiograph, which records changes and shows the ECG waveform on a graphic display.

The Clinical electrocardiography uses three basic techniques, namely, the standard clinical ECG with twelve leads, the vectorcardiogram with three leads and the monitoring ECG with one or two leads.

In the case of standard clinical ECG, twelve different potentials, called ECG leads, are recorded from the body surface of a resting patient. In a vectorcardiograph three body surface potentials are used as inputs to a three-dimensional vector model of cardiac excitation. This produces a vectorcardiogram (VCG) which is a graphical view of the excitation of the heart. Finally, for long term monitoring, one or two leads are monitored or recorded to look for life-threatening disturbances in the rhythm of the heartbeat (Tompkins, 1995). This technique, also called arrhythmia, is the one that is used in this project.

In monitoring lead systems, electrodes or sensors are placed so that the primary ECG signal has a large peak to ensure a high signal-to-noise ratio for beat detection (Tompkins, 1995). The most commonly used position of the electrodes is the chest, where one electrode is placed closer to the heart and the other one at a greater distance from the heart.

1.3. THE ELECTROCARDIOGRAM

The electrocardiogram is the tracing of the rhythmic electrical activity of the heart that is made by the electrocardiograph and is abbreviated by ECG or EKG (derived historically from the German spelling "electrokardiogram").

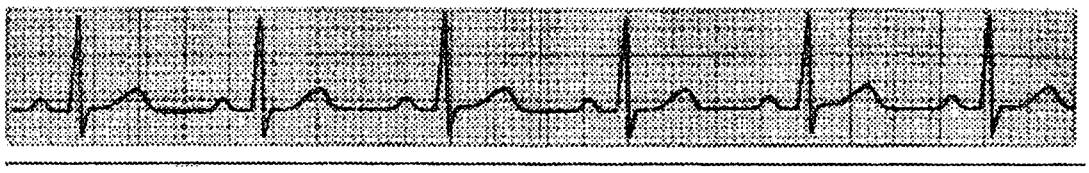

Figure 1.1 Electrocardiogram

The electrical activity of the heart is based on the ability of excitable tissue, such as heart muscle called myocardium, to change its membrane permeability to sodium (Na^+) and potassium ions

(K^+). This results in a changing electrical field (dipole) due to the movement of these ions across the cell membrane.

In the myocardium there are cells called pacemaker cells, which are contained within the sinoatrial (SA) node and in stimulus conductive tissues, like bundle branches, bundle of His, Purkinje fibers and internodal conductive bundles (Bañez, 1999). These cells are all electrically excitable, with each type of cell exhibiting its own characteristic action potential as shown in Figure 1.3 and they are responsible for establishing the normal heart rhythm (House, 1997). Cardiac muscle goes from relaxed (diastole) to the contractile (systole) state after the onset of "electrical depolarization" and returns from contraction to relaxation after "electrical repolarization".

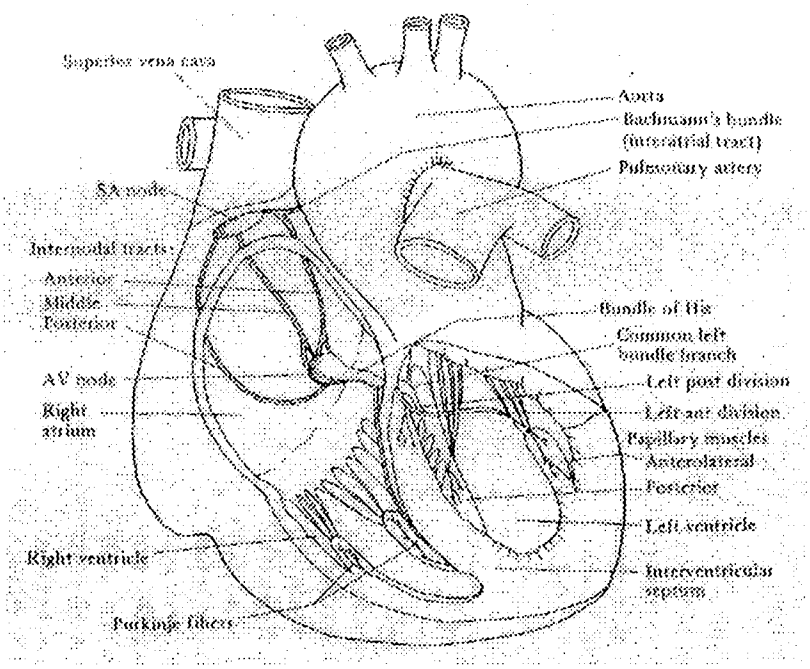

Figure 1.2. Distribution of specialized conductive tissues of the heart
(From J.G. Webster, Medical Instrumentation (Application and Design), 1978).

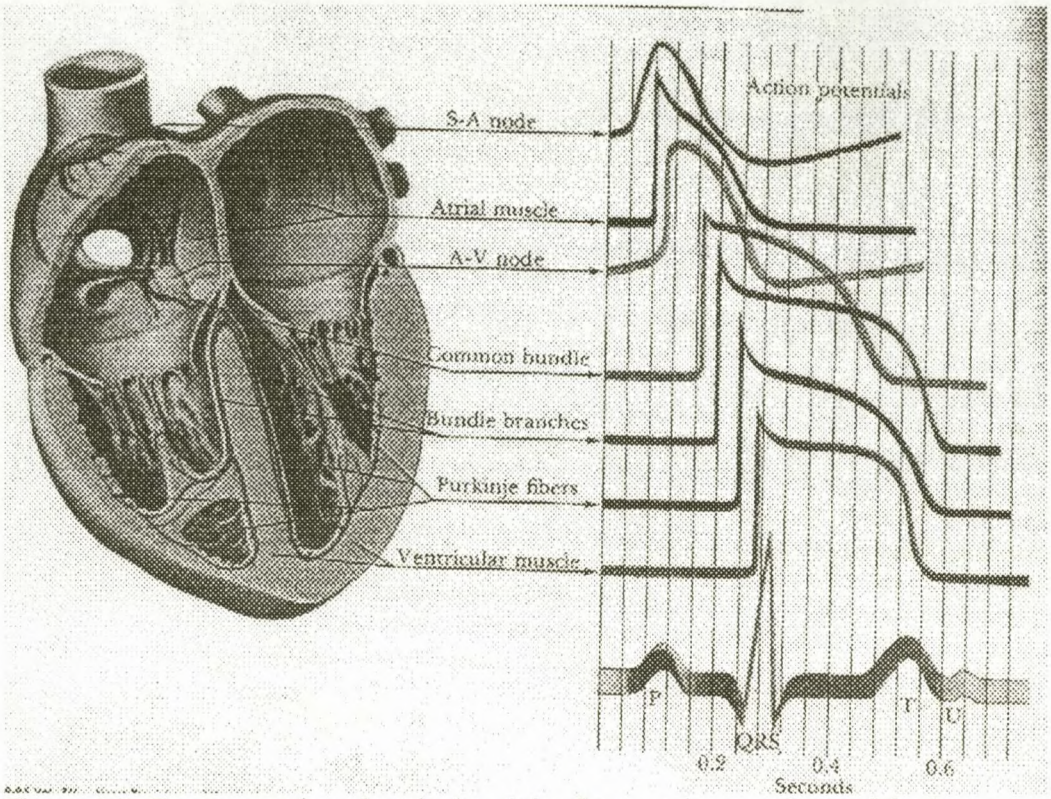

Figure 1.3. Representative electrical activity from various regions of the heart. Bottom trace is scalar ECG. (From J.G. Webster, Medical Instrumentation (Application and Design), 1978).

At the onset of the heart cycle, impulses from the SA node induce the right atrium to depolarize. This depolarization spreads across the atrial muscle causing atrial contraction, increasing atrial pressure and forcing blood into the ventricles. Once the blood has been pumped into the ventricles, the atrial muscle repolarizes and the depolarization is passed to the ventricles via the atrioventricular (AV) node, Bundle of His and the Purkinje fibers. During ventricle depolarization, the ventricular muscles contract, thus pumping blood out of the heart (Rhoades and Pflanzer, 1996, House, 1997).

1.3.1. Components of the normal Electrocardiogram

A normal electrocardiogram is characterized by five peaks and valleys labeled with successive letters of the alphabet P, Q, R, S and T (Greene, 1987).

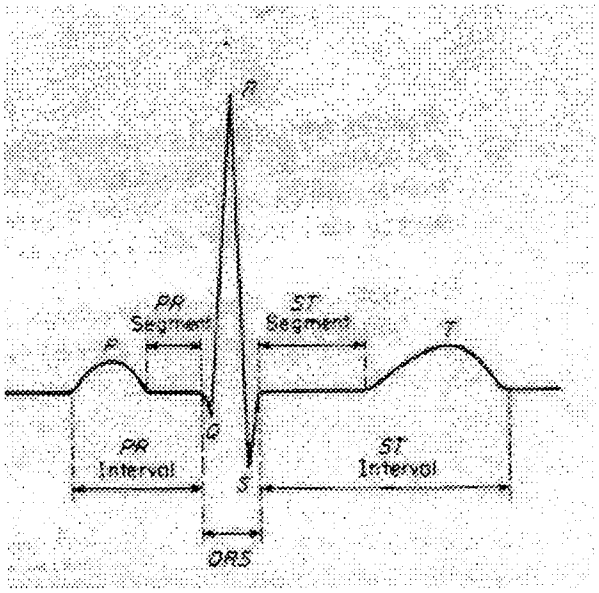

Figure 1.4. Components of the electrocardiogram (From R. Plonsey, Electrocardiography and Biopotentials, 1996)

Thus the ECG is said to consists of the P wave, QRS complex and T wave (Greene, 1987), all of which result from the passage of the cardiac impulses through the ventricles. In the normal electrocardiogram, the Q and S waves are often much less prominent than R wave and sometimes unidentifiable, but the wave is still referred to as the QRS complex or wave.

The P wave results from electrical currents generated as the atria depolarize prior to contraction, and it represents the spread of excitation and contraction of atrial tissue of both atria. It is normally upright, of amplitude about 0.2 mV (between 0.1 - 0.3 mV) and lasts about 0.1 sec. Currents generated when the ventricles depolarize prior to contraction cause the QRS complex. "Q" is an initial downward deflection; "R", a large upward deflection; "S", a downward deflection that follows, sometimes, to below the base line. The duration of the QRS complex is approximately 0.08 sec. Therefore, both P wave and the components of the QRS complex are DEPOLARIZATION WAVES. Currents generated as the ventricles recover from the state of depolarization cause the T wave. This process occurs in the ventricular muscle about 0.25 sec after depolarization, and this wave is known as a REPOLARIZATION WAVE. Thus, the electrocardiogram consists of depolarization and repolarization waves.

1.3.2. ECG and Temperature

As temperature increases, the degree of ionization also increases (Pham and Usher, 1994) and since the electrical activity of the heart is due to ionization, the body temperature will affect the heart rate and so as the electrocardiogram (Solomon, Schmidt, Adragna, 1990).

1.4. HEART RATE AS DETERMINED FROM THE ECG

The heart rate refers to the number of beats per minute. For each heart beat there is a corresponding peak called R, in the ECG. These R peaks or QRS complexes can be used to determine the instantaneous value of the heart rate.

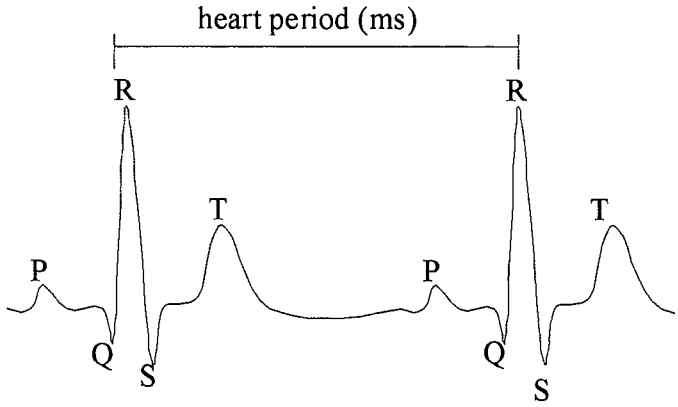

Figure 1.5. Electrocardiogram (ECG) showing two complete normal heart cycles

The time interval between two successive beats is the reciprocal (inverse) of the heart rate, for example, if one beat occurs every 0.8-sec, then the heart rate would be:

1 beat * (60/0.8)/min = 75 beats/min

Thus the general equation will be:

$$Heart\ Rate\ (beats/\min) = heart\ period \times 60\,000 \qquad (1.1)$$

where heart period is the time interval between the two successive R peaks in milliseconds.

CHAPTER 2

DATA ACQUISITION

2.1. Introduction

In order to sense and measure the ECG signals and temperature, sensors are used. The outputs of the sensors need to be conditioned and be further converted from analogue to digital format that is readable by the computer. This chapter explains how to sense, condition, isolate, multiplex and analog-to-digital convert these temperature and ECG signals so that they can be readable by the computer.

2.2. ISOLATED DATA ACQUISITION SYSTEM

This is the primary part of this project. It is where the physical variables, temperature and heart beats, are sensed, conditioned and converted into a digital format readable by the computer. It is also here where galvanic isolation is provided.

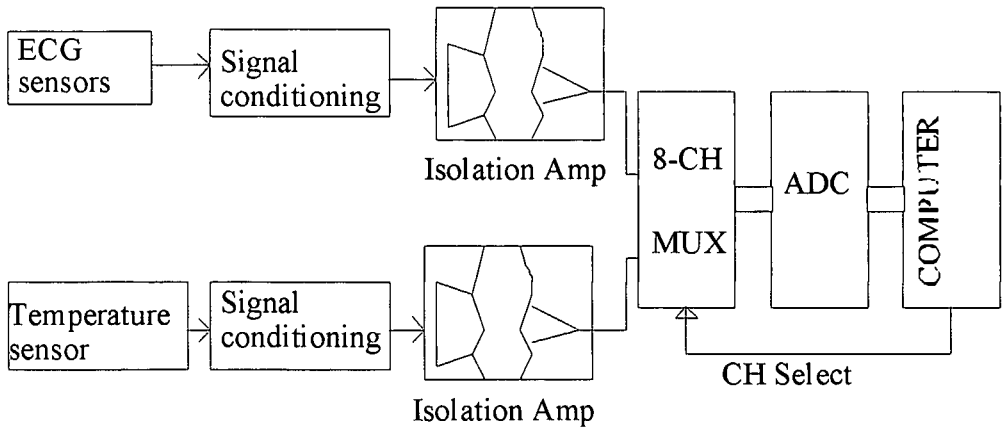

Figure 2.1 Isolated Data Aquisition System

Figure 2.1 depicts the block diagram of this data acquisition system. It consists of transducers (sensors) that convert temperature and heart beats to corresponding electrical signals. These

electrical signals are amplified and filtered by signal conditioning stages. After signal conditioning there are isolation amplifiers to provide isolation between sensing transducers and the rest of the circuit, including the computer that is powered from ac mains voltage. The isolated outputs are multiplexed and converted into a digital format before they are fed to the computer.

2.3. SENSING AND SIGNAL CONDITIONING

To sense electrical activity of the heart, metal electrodes (called ECG electrodes) are used. To sense temperature, a platinum resistive thermometer is used.

2.3.1. ECG sensing and conditioning

The ECG is obtained by measuring differentially between two ECG electrodes placed on the chest. A third standard skin electrode, placed on the right leg, is used as earth.
The signal conditioning circuit for ECG is shown in Figure 2.2, below.

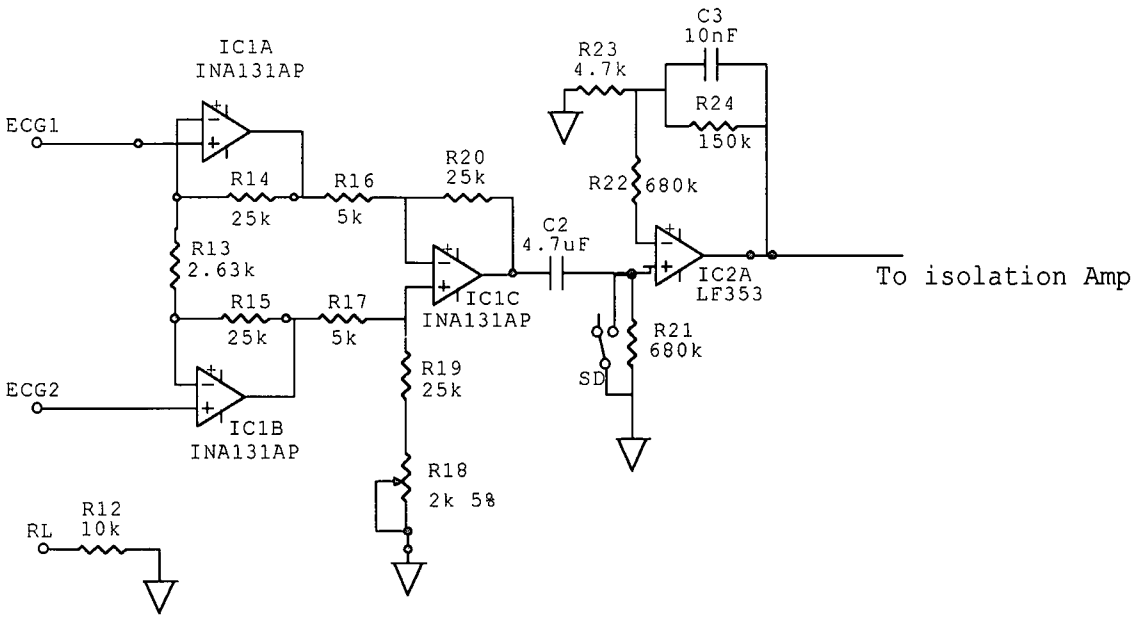

Figure 2.2 ECG Conditioning Circuit

The outputs of the ECG electrodes are fed to an instrumentation amplifier that provides high input impedance, gain and common mode rejection ratio. The instrumentation amplifier also

eliminates any interference that is common to both leads that are connecting the sensors to the monitoring instrument (Tompkins and Webster, 1988). This instrumentation amplifier consists of three operational amplifiers, IC1A, IC1B and IC1C, opamps IC1A and IC1B provide amplification to the ECG signals from the sensors.

They are designed to have equal gain given by

$$A_{1A} = A_{1B} = (1 + 2\frac{R_f}{R_{13}}) \qquad (2.1)$$

(where $R_f = R_{14} = R_{15}$)

$$\Longrightarrow A_{1A} = A_{1B} = (1 + 2*\frac{25k}{2.63k})$$

$$= 20.011$$

$$= 26.026 \text{ dB}$$

After amplification the two signals are fed to IC1C that is configured as a differential amplifier with a gain that is determined by R_{20} and R_{16}, that is

$$A_{1C} = -\frac{R_{20}}{R_{16}} \qquad (2.2)$$

$$= -\frac{25k}{5k}$$

$$= -5$$

the gain will be 5 or 13.98 dB and the minus sign indicates that the output signal will be inverted. This differential amplifier will produce an output that is equal to the difference of the two input signals amplified five times. It will also try to cancel the noise that is common on its input terminals.

The overall gain of the instrumentation amplifier will be the product of the two gains and is given by

$$A_{1T} = A_{1A} * A_{1C} = A_{1B} * A_{1C} \qquad (2.3)$$

$$= 20.011 * 5$$

$$= 100.055$$

$$= 40 \text{ dB}$$

The output of the instrumentation amplifier is passed through a high pass filter, formed by C_2 and R_{21} with a cutoff frequency of

$$f_{ch} = \frac{1}{2\pi R_{21} C_2} \qquad (2.4)$$

$$= \frac{1}{2\pi * 680k * 4.7\mu}$$

$$= 0.05 \text{ Hz}$$

Switch, SD, across R_{21}, may be momentarily closed when the output saturates. From the high pass filter the ECG signal goes to an active low pass filter formed by IC2A, R_{22}, R_{23}, R_{24} and C_3. R_{23} and R_{24} determine the gain whereas R_{24} and C_3 determine the cutoff frequency of this low pass filter. The gain is

$$A_{2A} = 1 + \frac{R_{24}}{R_{23}} \qquad (2.5)$$

$$= 1 + \frac{150k}{4.7k}$$

$$= 32.915$$

$$= 30.348 \text{ dB}$$

and the cutoff frequency for the low pass filter is

$$f_{ch} = \frac{1}{2\pi R_{24} C_3} \qquad (2.4)$$

$$= \frac{1}{2\pi * 150k * 10n}$$

$$= 106.10 \text{ Hz}$$

Figure 2.3 depicts the frequency responses of a high pass filter, a low pass filter and that of them cascaded. As it is reflected on Figure 2.3 (c), the overall response of the two filters is that of a band pass filter that passes all frequencies from 0.05 to 106.10 Hz.

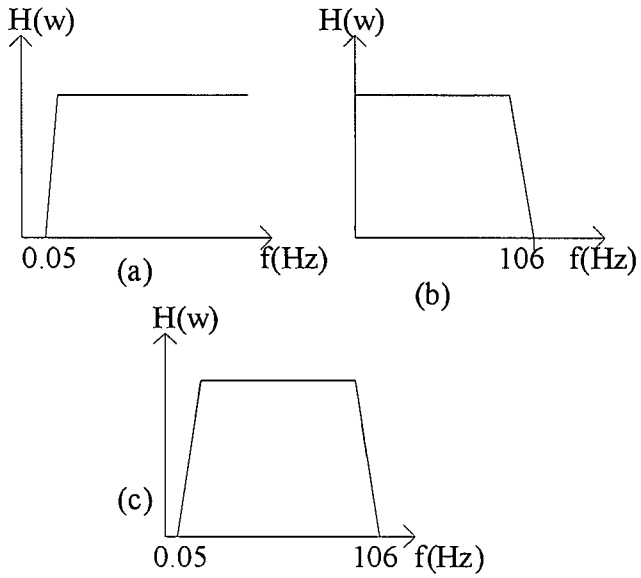

Figure 2.3 Frequency response curves for the (a) high pass filter, (b) low pass filter and (c) cascaded high pass and low pass filter

2.3.2. Temperature sensing and conditioning

To sense temperature, a platinum resistive thermometer (PT100) has been used because of its stability, sensitivity and linearity (Barney, 1988). This resistive temperature detector (RTD) has a positive temperature coefficient (i.e. its resistance increases with temperature and vice versa). To convert temperature to voltage the Wheatstone bridge has been used because it is ratiometric (that is, it compares the ratios from the same supply), so the null does not shift with variations in supply voltage (Horowitz and Hill, 1994). This was not very satisfactory since the resistance of the sensor will also include the resistance of the connecting wires in series with the resistance of the sensor (RTD) and this resistance will be a variable dependent on the temperature gradients along the connecting wires.

To cancel the effects of the resistance of the connecting wires a PT100/3 (that is a platinum resistive thermometer with three leads) is used. The third lead is introduced so that the two leads to the sensor are on either side of the bridge and thus effectively cancel, while the third wire functions as the extended supply to configure the bridge as shown in the temperature conditioning circuit, Figure 2.4.

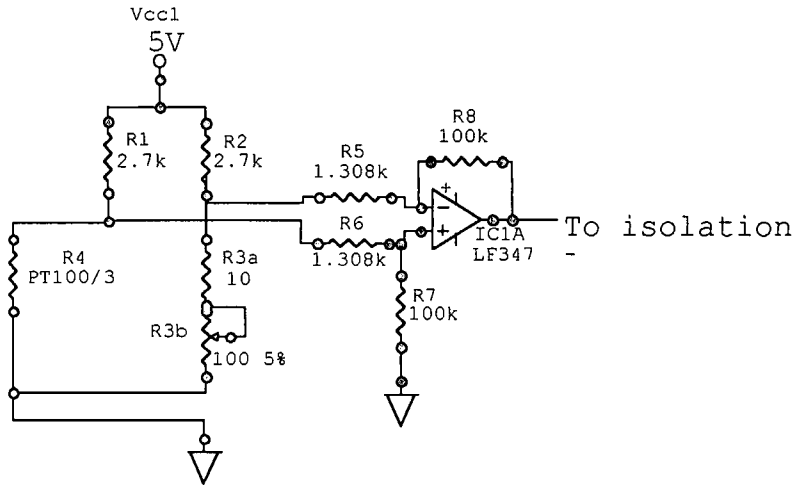

Figure 2.4. Temperature Conditioning Circuit

The bridge is balanced when R_3 is equal to R_4 and this is made to occur at 0 ^{0}C. As temperature rises above 0 ^{0}C towards 100 ^{0}C the resistance of the RTD (or R_4) increases from 100Ω to 138.5Ω. At 0^0 C, when the bridge is balanced, $R_4 = R_3 = 100\ \Omega$ and since $R_1 = R_2$,

$$V^- = V^+ = \frac{R_3 V_{CC}}{R_3 + R_2} \qquad (2.7a)$$

$$= \frac{5*100}{(100+2700)}\ \text{V}$$

$$= 178.57\ \text{mV}$$

At 100^0 C voltage V^- remain constant at 178.57 mV while voltage V^+ changes to

$$V^+ = \frac{R_4 V_{CC}}{R_4 + R_2} \qquad (2.7b)$$

$$= \frac{5*138.5}{(138.5+2700)} \text{ V}$$

$$= 243.97 \text{ mV}$$

As this value of R_4 increases, the bridge is no longer balanced and this results in an out-of-balance voltage that is used to determine the value of R_4. The values of input resistors R_5 and R_6 and the feedback resistor R_8 are chosen such that at 100^0 C the difference of the two voltages multiplied by the gain is equal to 5 volts. That is A_1*(243.97 - 178.571) mV = 5 volts, where the gain is

given by

$$A_1 = \frac{R_8}{R_5} = \frac{5\text{V}}{65.40\text{mV}}$$

$$= 76.45$$

$$= 37.67 \text{ dB}$$

2.4. ISOLATION

Galvanic isolation is required for many circuits that are found in medical systems since there are sensors that are attached to the patient. This has been traditionally accomplished by means of transformers and optocouplers, with transformers used to couple AC signals and optocouplers used primarily for DC signal coupling (Stulens, 1996).

The author has identified a LOC110 as a cheaper technology to replace the expensive and bulky cumbersome transformers and non-linear optocouplers. This device is a linear optocoupler that features an infra-red LED optically coupled with two phototransistors as shown in Figure2.5. One phototransistor is typically used in a servo feedback mechanism to control the LED drive current thus compensating for the LED's nonlinear time and temperature characteristics. The other (output) phototransistor provides output current that is linear with respect to the servo LED current.

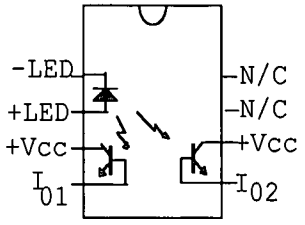

Figure 2.5 Linear optocoupler LOC110

This device provides up to 5 kV-peak isolation and it can couple both DC (temperature) and AC (ECG) signals. Figures 2.6 and 2.7 show the implementation of LOC110 in providing isolation for temperature and ECG, respectively.

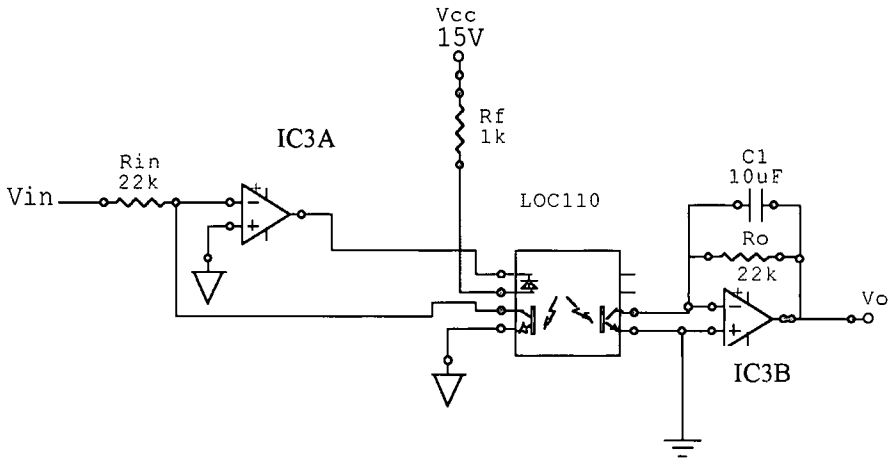

Figure 2.6 Isolation Amplifier (Photovoltaic Operation) for temperature

TheLOC110 devices are used in a photovoltaic mode to achieve best linearity, lowest noise drift performance. In this mode the phototransistors within the LOC110 act as current generators. These phototransistors are connected across the opamp inputs. As input voltage, v_{in}, increases, the current through the LED increases and so does the optical flux. The LED flux is incident on the servo phototransistor, which starts current I_{in} to flow from the opamp's (IC3A) inverting input through the phototransistor. This servo photocurrent is linearly proportional to input voltage, $I_{in} = v_{in}/R_{in}$ and keeps the voltage on the inverting input equal to zero.

The flux from the LED is also incident on the output phototransistor. The output phototransistor generates an output current I_o that is proportional to the LED flux and LED current. This current I_o is fed into the output opamp (IC3B), which is configured as a current-controlled voltage source, to convert the output current to voltage, v_o, which is a product of I_o and resistor R_o. Since the same LED flux is incident on both phototransistors, the currents generated by the phototransistors will be the same. Therefore, the output voltage can be expressed in terms of input current, I_{in}, or input voltage, v_{in}, as follows

$$\begin{aligned} v_o &= I_{in} \times R_o \\ &= I_{in} \times R_o \\ v_o &= \frac{R_o}{R_{in}} v_{in} \end{aligned} \qquad (2.8)$$

R_f is chosen such that the current I_f through the diode is less than or equal to 15 mA, thus

$$R_f = \frac{V_{cc}}{I_f} \leq 1k\Omega \qquad (2.9)$$

A capacitor is connected across the output resistor R_o, forming a low pass filter, to eliminate high frequency interference and also to ensure that the frequency of the ECG is less than or equal to the Nyquist frequency so as to avoid aliasing during sampling.

For ECG signals, the circuit in Figure 2.6 is modified as shown in Figure 2.7. Resistor R_{26} from 5 volts pre-biases the input amplifier such that a quiescent forward current in the LED is established. This is done to accommodate bipolar ECG signals. And at the output, a 10-kΩ variable resistor R_{28} is connected in series with R_{27} so that the gain can be varied from unity to about 1.5.

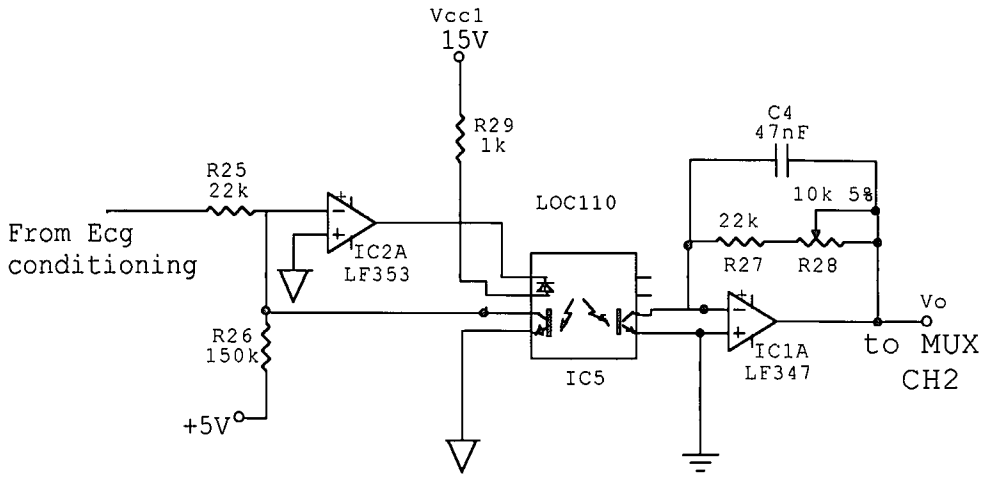

Figure 2.7. ECG Isolation Amplifier (Photovoltaic Operation with pre-biased input)

In this photovoltaic mode, it is possible to preserve accurately the linearity compatible with a 14-bit DAC (Stulens, 1996).

2.5. POWER SUPPLY

The author has designed the power supply for the whole system to include isolated power for the conditioning circuits and the input section of the isolation amplifier. For the isolated power, a NMA1215S, dc-to-dc converter has been used and it offers 1kV DC isolation. The circuit diagram for the power supply is shown in the appendix, Figure 7.1. It converts 220V ac from mains to ± 12V and positive 5 V dc.

2.6. PRE-PROCESSING AND OUTPUT

2.6.1. Multiplexing and Analog to Digital Conversion

The temperature and the ECG signals from the two isolation amplifiers are fed into a pre-processing circuit, shown in Figure 2.8, for multiplexing and analog-to-digital conversion.

The eight-channel analog multiplexor, CD4051 (IC7), which is controlled by the computer selects the required signal or channel. The selected signal is converted to digital by the ADC0801

(IC8). This ADC0801 is an eight bit analog to digital converter with an accuracy of plus or minus ¼ LSB. The digitized signal is fed to the computer for further processing through the parallel port, and 74541 (IC9), which is an octal buffer, is used to provide buffering between the computer's parallel port and the preprocessing stage.

2.6.2. Outputs

Figure 2.8 also contains the output stage, which includes circuits for ADC initialization, high and low level alarms and systole tone. Another octal buffer 74541 (IC10) is used to provide buffering between the computer parallel port and output circuits where:

The first pin Y1 operates relay RLY1 that is used to extend earth to pins 3 and 5 (WR and INTR) of the ADC thereby initializing it. This is done before the start of the monitoring process.

Pins Y2 and Y3 are used to select the channel of the analog multiplexor, CD4051 (IC7).

Y4 is used to operate a buzzer so that it produces a tone (i.e. systole tone) for every heart beat detected.

Y5 is set high whenever there is a high level detected (whether ECG or temperature), this operates a buzzer to signal an alarm condition and illuminates LED2 to indicate a high level condition.

Y6 is set high whenever a low level is detected (whether ECG or temperature). This operates a buzzer to signal an alarm condition and illuminates LED3 to indicate low level condition.

IC11 will produce a high output to switch on the transistor, Q_1, whenever one of its inputs goes high. When Q_1 is turned on, it drives the buzzer to produce a sound.

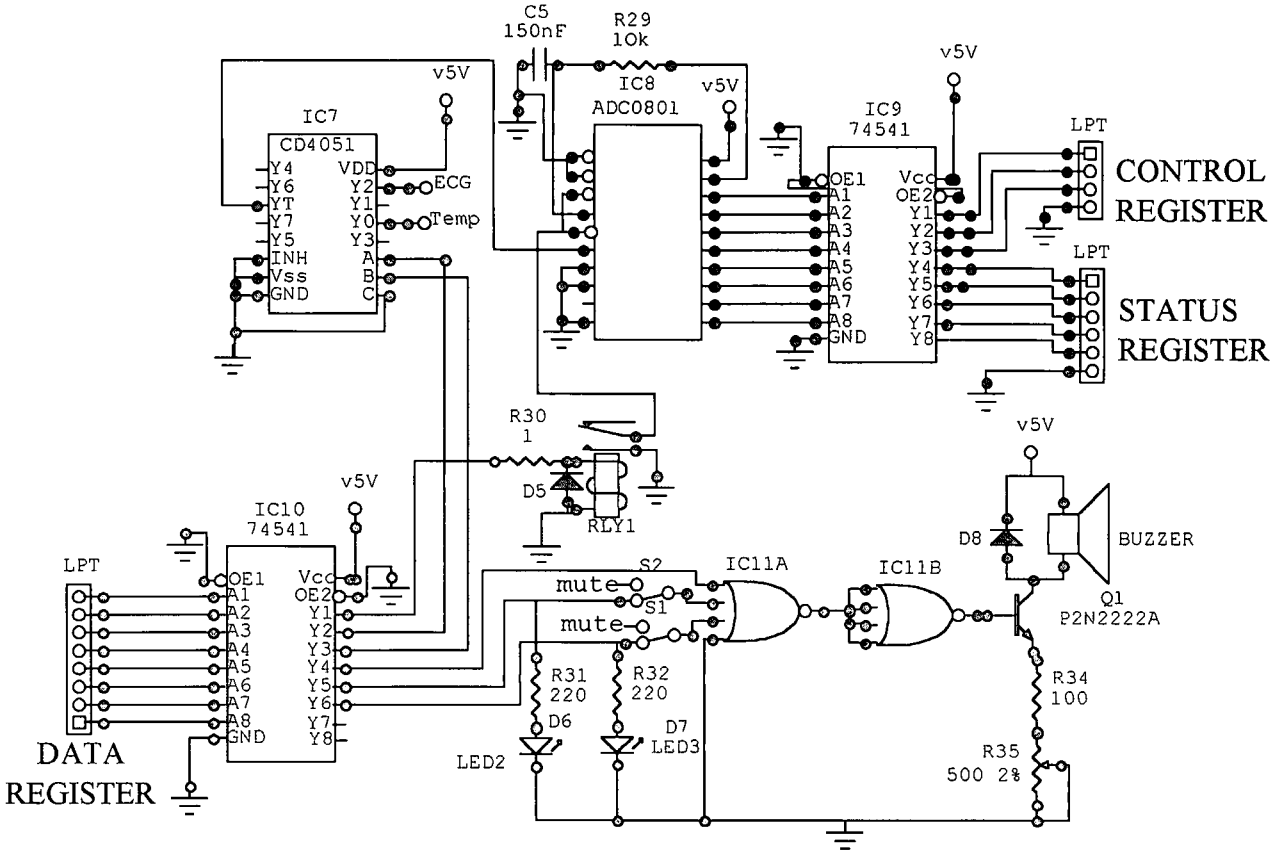

Figure 2.8 Pre-processing and output Circuits

CHAPTER 3

COMPUTER INTERFACING

3.1. Introduction

Once the signals are converted to a format that is readable by the computer, they are fed to the computer via the parallel port for further processing.
This chapter explains the parallel port and how it is used for monitoring and controlling purposes.

3.2. THE PARALLEL PORT

In the computer world the word port refers to a set of signal lines that the microprocessor, or CPU, uses to exchange data with other components. Therefore, it can be said that the port is used for communication and each personal computer has serial ports and one or more parallel ports. The serial port transfers one bit at a time while the parallel port transfers multiple bits at once. This means the transfer rate is faster when using a parallel port than when using the serial port.

The parallel port is also known as the printer port, because it was originally designed to connect the printer as reflected in the original names of the port's signals. However, these days the parallel port can be used to connect any peripheral. Over the years, several variations on the original parallel port's design have emerged, resulting in the introduction of improved versions of the parallel port, such as the enhanced parallel port (EPP), the extended capabilities port (ECP) and the PS/2-type (or simple bidirectional) port. The following is the summary of the available various types of the parallel port (Axelson, 1996):

3.2.1. Standard Parallel Port (SPP)

This is the original parallel port of an IBM PC and any port that emulates the original port's design. Other names used are AT-type or ISA-compatible. It is based on an existing Centronics printer interface.
SPP can transmit eight bits at once to a peripheral, using a protocol similar to that used by the original Centronics interface. The SPP does not have a byte (8 bits) wide input port, so it uses a Nibble mode that transfers each byte four bits at a time.

3.2.2. Simple Bidirectional (PS/2-type)

PS/2-type parallel port is a bidirectional data port that enables the computer to transmit or receive eight bits (byte) at once. It uses an eight-bit data-transfer protocol, called byte mode, to transfer data from the peripheral to the computer.

3.2.3. Enhanced Parallel Port (EPP)

EPP has bidirectional data lines and it can read or write a byte of data in one cycle of the ISA expansion bus, or about one microsecond, including handshaking, compared to four cycles for an SPP or PS/2-type port. It can also switch directions quickly so it is very efficient when used with devices that transfer data in both directions.
EPP can emulate SPP and sometimes can also emulate a PS/2-type.

3.2.4. Extended Capabilities Port (ECP)

ECP, like the EPP, is bidirectional and can transfer data at ISA-bus speeds. ECP has buffers and support for direct memory access (DMA) transfers and data compression. It can transfer large blocks of data.
ECP can emulate a SPP, PS/2-type and can also emulate an EPP.

3.2.5. Multi-mode Ports

Multi-mode ports are newer ports that can emulate some or all of the above port types. They often include configuration options that make all of the port types available, or allow certain modes while locking out the others.

3.3. PARALLEL PORT HARDWARE

The port's hardware includes, inter alia, a rear panel connector, circuits, interfacing cable and the system expansion bus.

The connector is a female 25-pin D-shell connector. The IEEE 1284 standard for the parallel port calls it the IEEE 1284-A connector (Axelson, 1996). This connector is for plugging in an interface cable to a printer or any parallel port device.

The parallel port circuits are inside the computer on the motherboard or on a card that plugs into the expansion bus. The port circuits connect to address, data, and control lines on the expansion bus, and in turn interface to the microprocessor and other system components.

The interface cable must have a male 25-pin D-shell connector to plug into the parallel port connector and since the parallel port uses TTL voltage levels, it is recommendable to limit the lengths of the interfacing cable to about 1 meter (Young, 1990).

3.4. ACCESSING THE PARALLEL PORT

MS-DOS refers to the parallel port as LPT1 (line printer) or LPT2 and LPT3 for additional ports. Each standard port has three registers; namely, data, status and control register. It also has three possible base addresses in memory: 3BCh, 378h, or 278h. These addresses can be changed by moving a jumper or running a setup utility. The data register has eight output lines and its address is the same as the base address, the status has five input lines and its address is base+1, and the control register has four bidirectional lines and its address is base+2. Therefore it can be said that the parallel port has (8 + 5 + 4) 17 lines or signals (Axelson, 1994).

When the computer boots up, a BIOS routine looks for a port at each of the three addresses in the order listed above. The BIOS do this by writing to the port and then reading back what was written. If the read is successful, the port exists. The first port found is called LPT1, the second, LPT2 and the third, LPT3. LPT1 may be at any of these addresses, LPT2 may be at 378h or 278h if LPT1 exists, and LPT3 can only be at 278h.

The parallel port can be accessed through the operating systems like dos, windows etc. or through BIOS. But for full access to all of the port's seventeen signals, you need to write and read directly to the port's data, status and control registers (Axelson, 1994).

3.4.1. The Data Register

In a standard parallel port, the Data Register (D0 to D7) holds a byte written to Data outputs and in bidirectional Data ports. When the port is configured as the input, the Data register holds the byte at the connector's Data pins. The Data register uses pins 2 through 9 on the parallel port connector.

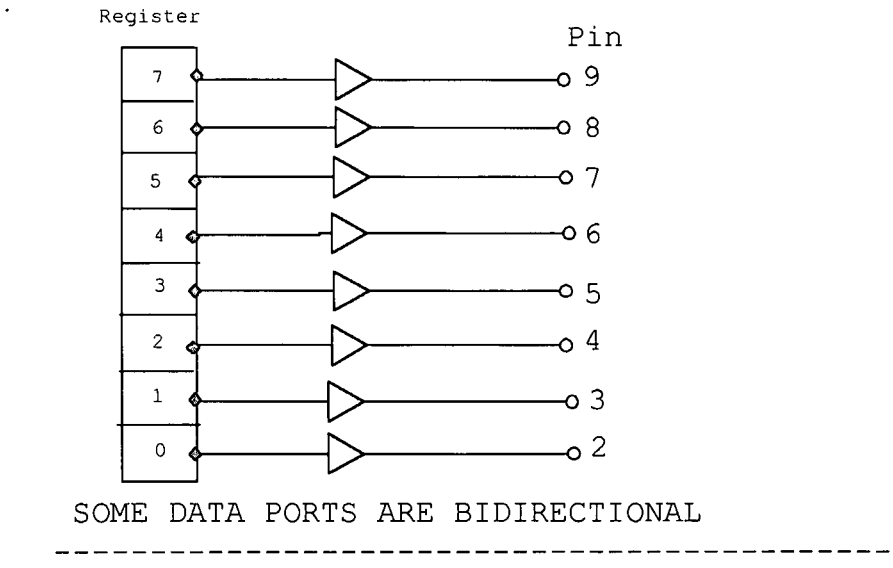

Figure 3.1 Data register (8 Outputs)

3.4.2. The Status Register

The Status register holds the logic states of five inputs, S3 through S7. Bit S7 is inverted at the connector. Bits S0 to S2 do not appear at the connector. This register is read-only except for S0, which is a timeout flag in the ports that support EPP transfers. S1 and S2 are not used.

The Status register uses pins 10 through 13 and pin 15 on the parallel port connector.

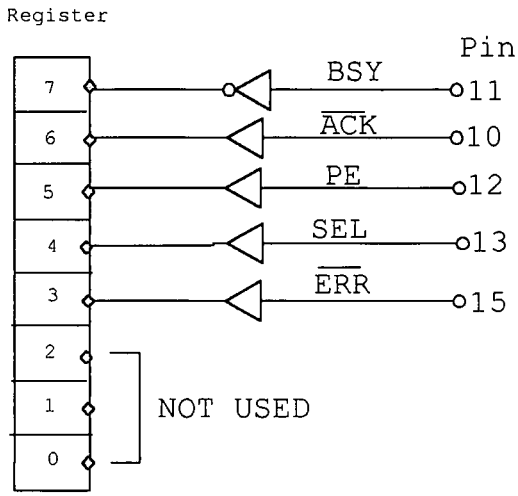

Figure 3.2 Status register (5 Input lines)

3.4.3. The Control Register

The Control register holds the logic state of bits, C0 through C3. Bits C4 through C7 do not appear at the connector. Conventionally, the control register is used as an output. However, on most SPPs, the control register can also be used as an input, by bringing high all four of the control port's outputs (Axelson, 1994).

Because bits C0, C1 and C3 are inverted at the connector, a 04 is written to the control register will bring bits C0 through C3 high.

The control register uses pins 1, 14, 16 and 17 on the parallel port connector.

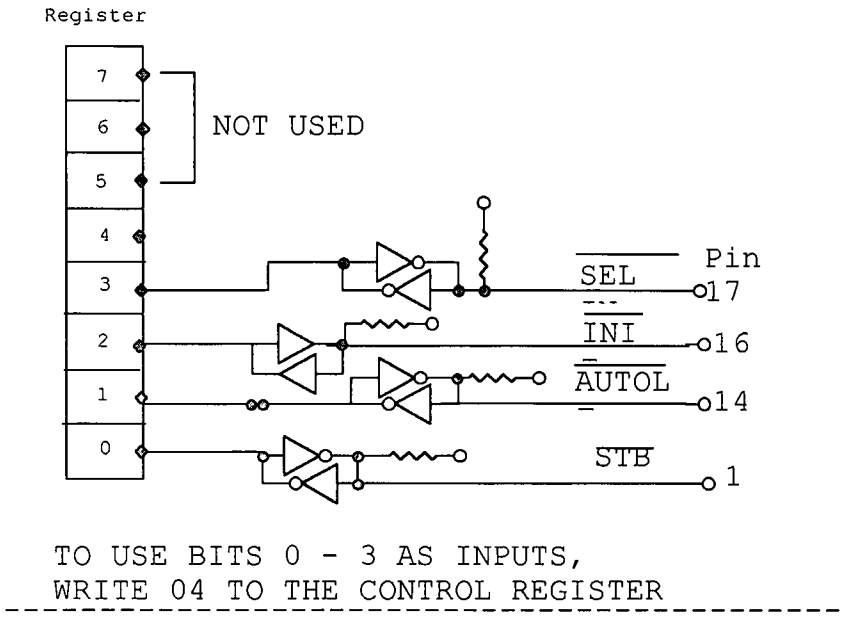

Figure 3.3 Control Register (4 Bidirectional lines)

Table 3.1 shows the possible register addresses of the parallel port and Table 3.2 summarize the signals at the parallel port connector. The signal names are those used by the parallel port in the original IBM PC.

Table 3.1 Possible Register Addresses of the Parallel port		
Data (Base address)	Status (Base + 1)	Control (Base + 2)
3 BCH	3BDH	3BEH
378H	379H	37AH
278H	279H	27AH

Table 3-1 Signals and their functions on a PC parallel port							
Pin: D-shell	Signal	Function	I/O at D-shell	Register		Inverted at connector	Pin Centronics
				Name	Bit		
1	Nstrobe	Strobe D0-D7	I/O	Control	0	Y	1
2	D0	Data Bit 0	O*	Data	0	N	2
3	D1	Data Bit 1	O*	Data	1	N	3
4	D2	Data Bit 2	O*	Data	2	N	4
5	D3	Data Bit 3	O*	Data	3	N	5
6	D4	Data Bit 4	O*	Data	4	N	6
7	D5	Data Bit 5	O*	Data	5	N	7
8	D6	Data Bit 6	O*	Data	6	N	8
9	D7	Data Bit 7	O*	Data	7	N	9
10	Nack	Acknowledge	I	Status	6	N	10
11	Busy	Printer busy	I	Status	7	Y	11
12	Paper End	Paper end or Out of paper	I	Status	5	N	12
13	Select	Printer selected	I	Status	4	N	13
14	NautoLF	Auto line feed	I/O	Control	1	Y	14
15	Nerror	Error (fault)	I	Status	3	N	32
16	Ninit	Initialize printer	I/O	Control	2	N	31
17	NselectIn	Select printer	I/O	Control	3	Y	36
18-25	GND	Ground	---	—	—	---	16, 19-30, 33
---	NC	No connection at PC	---	—	—	—	15,17,18, 34,35

*Some data ports are bidirectional (I/O)

3.4. MONITORING AND CONTROLLING USING A PARALLEL PORT

The standard parallel port offers a way to add user-accessible inputs and/or outputs to a personal computer to enable computer-based instrumentation or controls.

The standard PC's parallel port has eight outputs, five inputs and four bidirectional lines. Though many newer computers, the eight output (data) lines can also serve as inputs (that is, they are bidirectional). The discussion in this document will be based on a standard parallel port, where these eight data lines are for output only.

For alarms, ADC initialization, systole tone, controlling the multiplexor (channel selection) and other outputs, the data register has been used.

The control register has been configured to be used as input. This has been done by bringing high all four of the control port's output by writing 04 to it, its fourth line is ignored and the remaining three lines are combined with the five input lines from the status register to give the required eight input lines.

The value read from the control register has bits C0, C1 and C3 inverted, that is, they are compliments of their logic state at the connector. Software is used to invert these lines so that the actual value can be obtained from the control register. Exclusive-OR (XOR) with 0Bh (00001011) or 11 (decimal) does this (Axelson, 1994).

The last bit of the status register, S7, is the compliment of the logic state or inverse of that at pin11 at the connector. The software is again used to invert this line so that the actual value can be obtained from the control register. Exclusive-OR (XOR) with 80h (10000000) or 128 (decimal) does this (Axelson, 1994).

These first three bits, C0, C1 and C2, of the control register are combined with the last five bits, S3 through S7, of the status to make an eight-bit input that is used to read the values from the

eight-bit analog-to-digital converter (ADC). These values are read at a speed of 200 samples per second and are further processed by the computer. The computer that was used for testing this project is a 150 MHz Pentium personal computer with a 16 Mbytes random access memory (RAM) and to be able to print while monitoring, a second parallel port has been added.
Since for digitizing the analog signal, an ADC0801 has been used, the value read by the parallel port would not be the exact value. Thus, to get the exact value, the value read from the ADC must be multiplied by 0.01953 (which is the value of the LSB for the ADC080x series of ADCs).

CHAPTER 4

SIGNAL PROCESSING

4.1. Introduction

The signals from the isolated data acquisition system need to be further processed by the computer to get their diagnostic values. The ECG signals must be filtered to remove interference, which is mainly due to interference from mains.

This chapter presents various techniques that are used to eliminate the interference on the ECG signals. It also explains how the ECG signals can be further processed to get the instantaneous value of the heart rate.

4.2. ECG SIGNAL PROCESSING

The micro-volt ECG signals from the heart are very small and noise, such as residual electrode voltages and 50 Hz power line pickups can easily swamp out the signals being sensed. Though the use of an instrumentation amplifier with high common mode rejection ratio in the ECG conditioning circuit did reduce some of this noise, noise from the mains was still visible on the electrocardiograph as shown in the Figure 4.1. This requires that the acquired ECG signal must be further processed to remove the contaminating ac interference, thus improving the signal-to-noise ratio.

In this project, a notch and adaptive filters were implemented in real time to remove noise from the ECG signal and the performance of the filters was compared, based on noise reduction, for non-filtered and filtered ECG signals. The filters are discussed in the following sections.

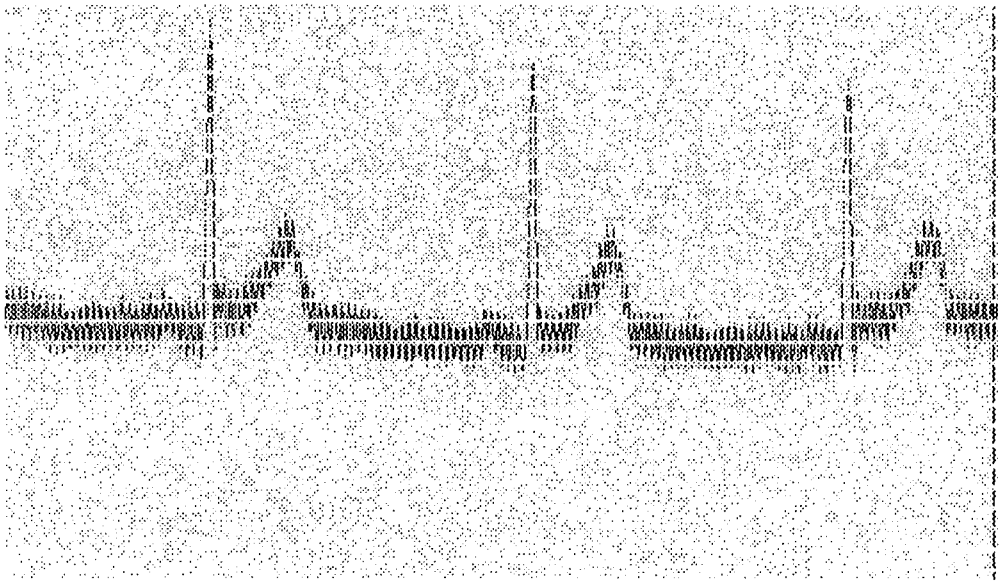

Figure 4.1 Contaminated ECG signal

4.2.1. NOTCH FILTER

A narrow band, notch filter was designed using Matlab's Signal Processing Toolbox (SPTool), to remove 50 Hz ac interference that contaminates the ECG signal.
The filter was designed to meet the following specifications:

Notch frequency	50 Hz
3 dB Width	± 2.5 Hz
Attenuation at Notch frequency	≥ 60 dB
Sampling frequency	200 Hz

Though most of the filters, within SPTool, could do the job, the Chebyshev Type 1 infinite impulse response (IIR) filter of the 10^{th} order was found to be the one that has the narrowest bandwidth, the highest attenuation at 50 Hz and fewer coefficients. Chebyshev Type 1 IIR filter is a recursive digital filter that is based on an analog filter prototype, with equal ripple in the passband and monotonic in the stopband (Ifeachor and Jervis, 1996).

The transfer function, H(z), for this notch filter is given by

$$\frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4} + a_5 z^{-5} + a_6 z^{-6} + a_7 z^{-7} + a_8 z^{-8} + a_9 z^{-9} + a_{10} z^{-10}}{1 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4} + b_5 z^{-5} + b_6 z^{-6} + b_7 z^{-7} + b_8 z^{-8} + b_9 z^{-9} + b_{10} z^{-10}}$$

where

$a_0 = 0.5835$; $b_1 = -0.0004$

$a_1 = -0.0003$ $b_2 = 4.0280$

$a_2 = 2.9175$ $b_3 = -0.0014$

$a_3 = -0.0011$ $b_4 = 6.4414$

$a_4 = 5.8351$ $b_5 = -0.0016$

$a_5 = -0.0016$ $b_6 = 5.0478$

$a_6 = 5.8351$ $b_7 = -0.0008$

$a_7 = -0.0011$ $b_8 = 1.8959$

$a_8 = 2.9175$ $b_9 = -0.0001$

$a_9 = -0.0003$ $b_{10} = 0.2592$

$a_{10} = 0.5835$

Converting from Z-domain to time domain gives a difference equation, which is:

$$\begin{aligned} y[0] = {} & 0.5835*x[0]-0.0003*x[1]+2.9175*x[2]-0.0011*x[3]+5.8351*x[4]- \\ & 0.0016*x[5]+5.8351*x[6]-0.0011*x[7]+2.9175*x[8]-0.0003*x[9] \\ & +0.5835*x[10]+0.0004*y[1]-4.0280*y[2]+0.0014*y[3]- \\ & 6.4414*y[4]+0.0016*y[5]-5.0478*y[6]+0.0008*y[7]-1.8959*y[8] \\ & +0.0001*y[9]-0.2592*y[10] \end{aligned} \qquad (4.2)$$

The filter realization for this filter is shown in Figure 4.2.

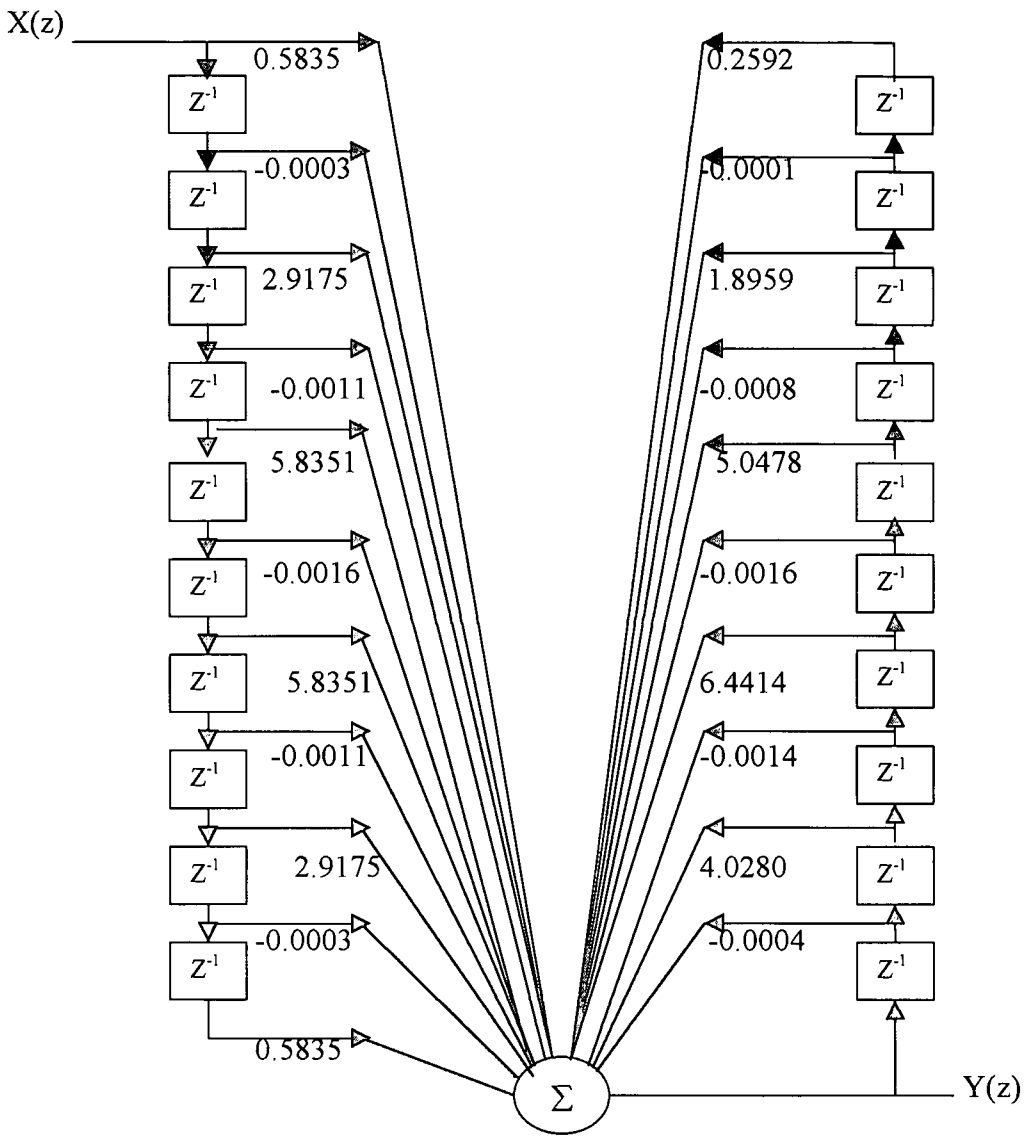

Figure 4.2. Tenth order 50 Hz Chebyshev type 1 notch IIR filter structure

The amplitude and phase responses for this filter are shown in Figure 4.3. It can be seen on the amplitude response curve that the attenuation at the notch frequency is about 150 dB.

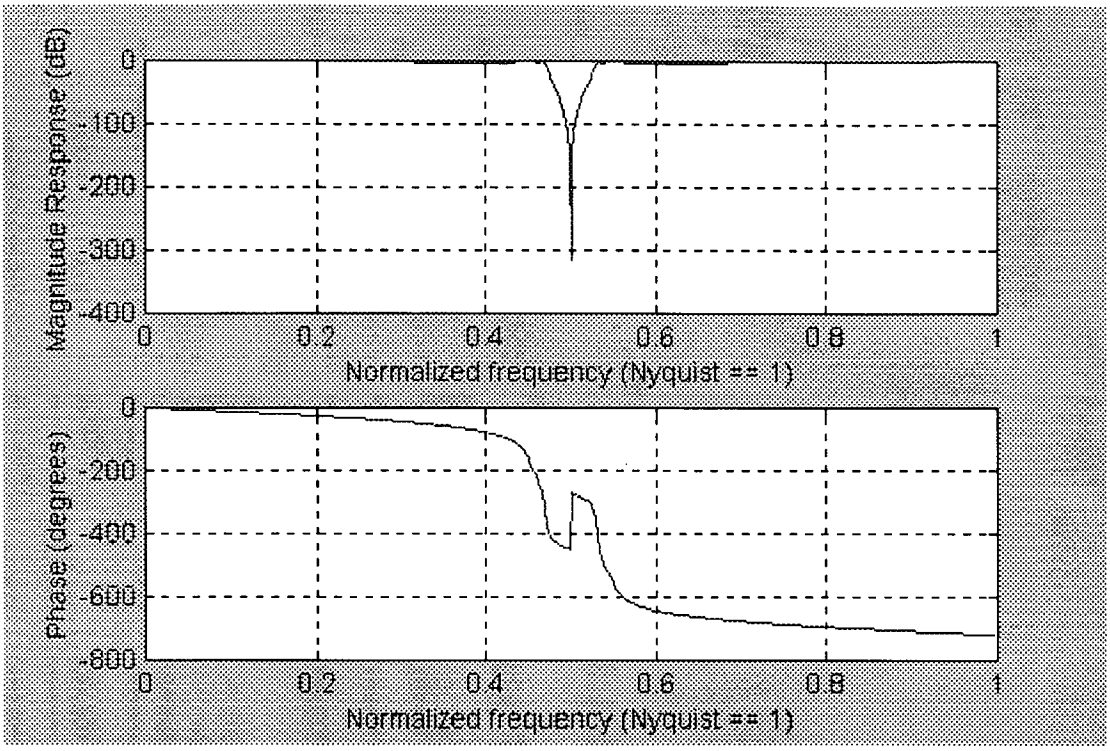

Figure 4.3. Amplitude and Phase Response of a 50 Hz Notch.

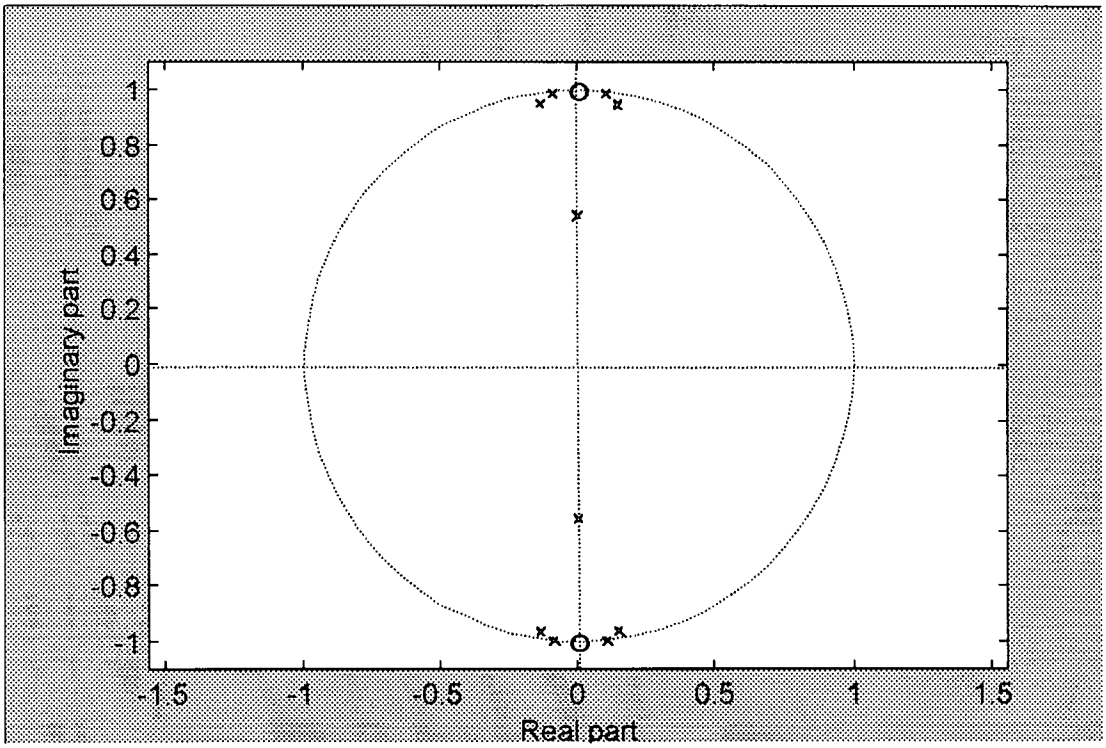

Figure 4.4. Pole – zero plot of the notch filter ('x' represents a pole and 'o' represents a zero)

4.2.1.1. Implementation of the Notch filter

The raw signal read from the isolated data acquisition system is fed to a narrowband notch filter to reduce the 50 Hz interference from the ac mains. The result is that the level of interference has been reduced as shown in Figure 4.5 below.

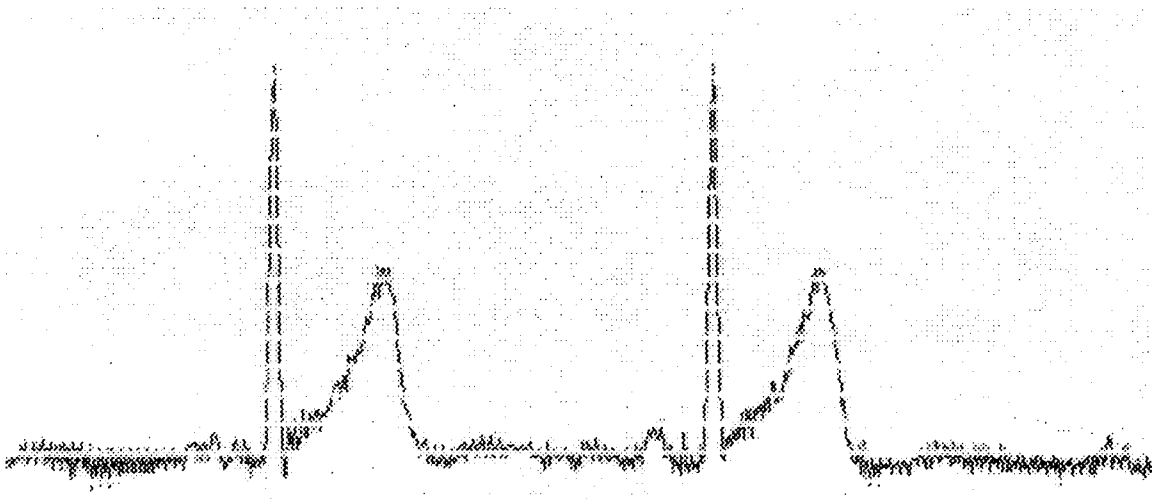

Figure 4.5 Filtered ECG using a notch

4.2.1.2. Practical Limitations of a Notch Filter

The Notch filter explained in section 4.2.1 above, removed the noise by attenuating the entire signal content at 50 Hz. This results in a loss of the frequency components of the ECG signal range around 50 Hz.

This violation of the frequency components of the ECG signal can be reduced by making the the bandwidth narrower. But narrowing the bandwidth of the notch filter requires a higher order of the filter resulting in many coefficients and more processing time. Another problem encountered

when using a narrow band notch filter is that it loses its effectiveness in removing the ac noise when the interference frequency shifts or deviates from 50 Hz.

4.2.2. ADAPTIVE FILTERING

Adaptive filtering offers another possibility of ac noise removal. Feeding the adaptive system with a reference signal, the interference can be highly reduced, if not eliminated.

The adaptive filter removes ac interference from the ECG signal using a reference input, which is subtracted from the corrupted signal by an adaptive estimation of amplitude and phase of the interference. This adaptive filter consists of a digital filter with adjustable coefficients and an adaptive algorithm, which is used to adjust the coefficients of the filter. Another advantage of using adaptive techniques is that they do not require a priori knowledge of signal or noise characteristics as do fixed notch filters. They can also be used not only to remove ac interference, but also for removing other artifacts from the ECG signal.

Figure 4.6 shows the model of an adaptive noise canceler, where y_k is the contaminated signal that contains both the desired ECG signal, represented by s_k, and the interfering noise, represented by i_k. There is no external reference signal from mains, but instead the reference signal, x_{k-M}, is derived from the corrupted ECG input, y_k, by delaying y_k so that x_{k-M} is not correlated with s_k (Hayes, 1996).

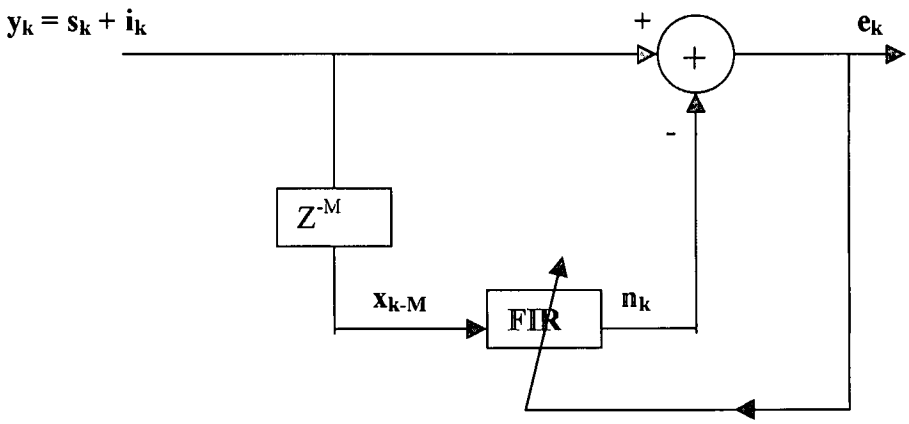

Figure 4.6 Adaptive Noise Canceler

4.2.2.1. Adaptive Filters as Noise Canceler

The adaptive filter is made up of two distinct parts: a digital filter with adjustable coefficients, and an adaptive algorithm that is used to adjust the coefficients of the filter (Ifeachor and Jervis, 1996). The input signal y_k and x_{k-M} (which is a delayed input) are applied simultaneously to the filter. By delaying y_k, x_{k-M} becomes less correlated with s_k whilst it remains correlated with the mains interference. The delayed input, x_{k-M}, is processed by a digital filter to produce an estimate of the noise, n_k. The estimate of the desired signal, e_k, is then obtained by subtracting the digital filter output, n_k, from the contaminated input signal, y_k:

$$e_k = y_k - n_k \qquad (4.3a)$$

$$= s_k + i_k - n_k \qquad (4.3b)$$

This estimate, e_k, serves two purposes: firstly, as an estimate of the desired signal and secondly as an error signal which is used in a feedback arrangement to modify the digital filter coefficients. This modification or adjustment of coefficients is done through an adaptive algorithm, to minimize interference and to produce the minimum error estimate of the desired signal, s_k.

This mean-squares error may be obtained by squaring equation (4.3b) and thereafter taking expectation (expected value, which is the mean value) as follows:

Squaring equation (4.3b) we have:

$$e_k^2 = s_k^2 + (i_k - n_k)^2 + 2s_k(i_k - n_k) \qquad (4.4)$$

Taking the expectations on both sides of equation (4.4) we get:

$$E[e_k^2] = E[s_k^2] + E[(i_k - n_k)^2] + 2E[s_k(i_k - n_k)] \qquad (4.5)$$

For an ECG signal that is not correlated with noise, s_k and i_k will be uncorrelated, then $E[i_k s_k] = 0$ (Ifeachor and Jervis, 1996) and the last term will become $-2E[s_k n_k]$ and equation (4.5) will become:

$$E[e_k^2] = E[s_k^2] + E[(i_k - n_k)^2] - 2E[s_k n_k] \qquad (4.6)$$

And s_k and n_k will also be uncorrelated, resulting in $E[n_k s_k] = 0$, the last term becoming zero and equation (4.6) becoming:

$$E[e_k^2] = E[s_k^2] + E[(i_k - n_k)^2] \qquad (4.7)$$

= Mean-squares error

It is evident in equation (4.7) that if the adaptive filter can be adjusted towards the optimum position, the remnant noise power and hence the total output power will be minimized (Ifeachor and Jervis, 1996). The desired signal power, $E[s_k^2]$, will not be affected by the adjustments (Tompkins, 1995), since s_k is uncorrelated with i_k. Thus

$$\min E[e_k^2] = E[s_k^2] + \min E[(i_k - n_k)^2] \qquad (4.8)$$

When this happens, the signal-to-noise ratio in the desired output is maximized and the filter is said to have adaptively learned to synthesize noise (i.e. $n_k \approx i_k$).

4.2.2.2. Digital Filter

The digital filter used in the adaptive noise canceler is realized using a finite impulse response (FIR), which is a nonrecursive digital filter structure. The FIR filter is chosen because of its simplicity and guaranteed stability (Ifeachor and Jervis, 1996). Figure 4.4 depicts this FIR filter. This filter is used to produce the estimate of i_k, n_k, and its output is given by

$$n_k = \sum_{c=0}^{N-1} w_k(c) x_{k-c} \qquad (4.8a)$$

$$= X_k^T W_k \qquad (4.8b)$$

where $w_k(c)$, c = 0, 1, 2, ..., are the adjustable coefficients or weights of the filter, and $x_k(c)$ and n_k are the input and output of the filter, respectively.

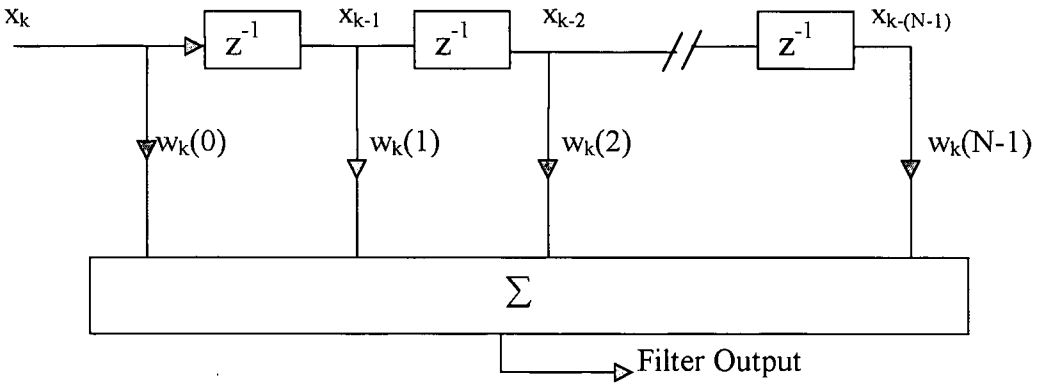

Figure 4.7. FIR filter structure

4.2.2.3. Adaptive Algorithms

In the adaptive noise canceler, shown in Figure 4.6, the coefficients of the digital filter are adjusted or modified using adaptive algorithms. This adjustment is done such that the estimate of the desired output (or error) signal, e_k, is minimized according to some criterion, e.g. in the least squares sense.

The commonly used algorithms include least mean squares (LMS), recursive least squares (RLS) and the Kalman filter algorithms. In terms of computing and storage requirements, least mean squares algorithm is the most efficient (Ifeachor, Jervis, 1996). Another advantage of using the least mean squares algorithm is that it does not suffer from the numerical instability problem inherent in the other two algorithms (Ifeachor, Jervis, 1996).

4.2.2.4. Least Mean Squares (LMS) Algorithm

The Least Mean Squares algorithm attempts to minimize the expected value of the square error, $E[e^2_k]$. It is based on the steepest decent algorithm where the weight vector is updated from sample to sample as follows:

$$W_{k+1} = W_k - \mu\nabla_k \qquad (4.9)$$

Where W_k and ∇_k are the weight and the gradient vectors of the error function, respectively, and μ is a step size that controls the stability and rate of convergence.

But according to the Wiener filter theory, the gradient, ∇, of the mean-squares error (MSE) can be obtained by differentiating the MSE equation, $J = \sigma^2 + 2P^TW + W^TRW$, with respect to the weight vector, W (Haykin, 1996). That is

$$\begin{aligned}\nabla &= \frac{dJ}{dW} + \frac{d(2P^TW)}{dW} + \frac{d(W^TRW)}{dW} \\ &= 0 - 2P + 2RW \\ \Rightarrow \nabla &= -2P + 2RW \qquad (4.10)\end{aligned}$$

where P is the cross-correlation between the primary input, x_k, and the secondary input y_k, and R is the autocorrelation of the primary input, x_k.

Since both P and R are not available, the instantaneous estimates are used for ∇.
Thus,

$$\nabla_k = -2X_k y_k + 2X_k X_k^T W_k \qquad (4.11a)$$

$$= -2X_k (y_k - X_k^T W_k) \qquad (4.11b)$$

Substituting equation (4.8b) for $X_k^T W_k$ in equation (4.11b) gives:

$$\nabla_k = -2X_k (y_k + n_k) \qquad (4.11c)$$

Substituting equation (4.3a) for $y_k + n_k$ in equation (4.11c) gives:

$$\begin{aligned} \nabla_k &= -2X_k(e_k) \\ &= -2e_k X_k \end{aligned} \qquad (4.11d)$$

Substituting equation (4.11d) for gradient in equation (4.9) gives:

$$W_{k+1} = W_k + 2\mu e_k X_k \qquad (4.12)$$

which is the Widrow-Hopf least mean-squares algorithm that is used for updating the weights of the filter (Ifeachor and Jervis, 1996).

The weights obtained using the LMS algorithm are only estimates that improve gradually with time, as the filter learns the characteristics of the input signals. Though for better performance it is required that these weights must converge to an optimum value, W_{opt}, which is a value that will occur when the gradient, ∇, in equation (4.10) is zero or when $W = R^{-1}P$. In practice, W_k never reaches the theoretical optimum (Wiener) solution, but fluctuates around it as shown in Figure 4.8.

When the ECG signal was fed to the LMS adaptive filter it took about 20 seconds for the filter to adapt or converge.

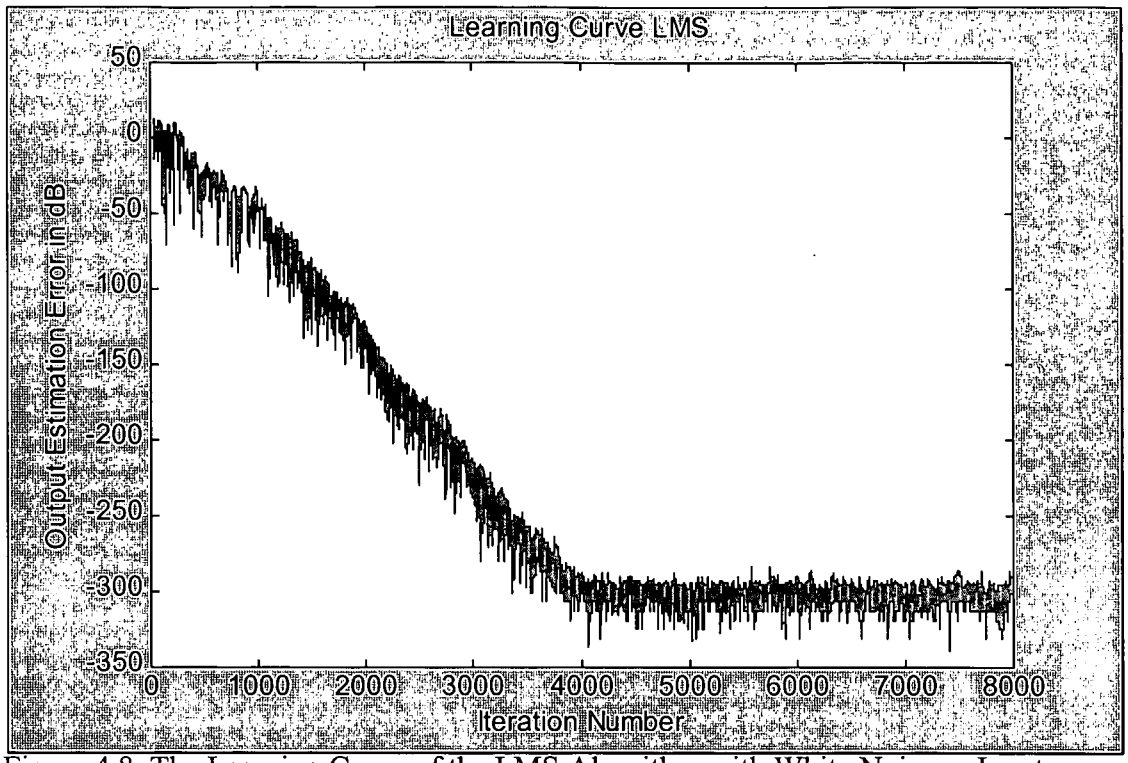

Figure 4.8. The Learning Curve of the LMS Algorithm with White Noise as Input

4.2.2.5. Normalized Least Mean Square (NLMS) Algorithm

The Normalized LMS algorithm is a modified version of the LMS algorithm. We have seen in the LMS algorithm, given by equation (4.12), that the correction factor, μ, which is applied to the weight vector, w_k, is directly proportional to the primary input vector, x_k. Therefore, when x_k is large, the LMS algorithm experiences a problem with gradient noise amplification.

With normalization of the step size, μ, however, this noise amplification can be eliminated.

This variant of the LMS algorithm, with normalization of the step size, is called Normalized LMS (NLMS) algorithm (Sankaran and Beex, 1996).

The normalization is achieved by putting a bound on the step size for mean-square convergence such that

and to incorporate this bound into the LMS adaptive filter, a time varying step size of the form

$$\mu_n = \frac{\beta}{x_k^H x_k} = \frac{\beta}{\| x_k \|^2} \tag{4.13}$$

is used, where β is the normalized step size (Hayes, 1996). According to the bound on the step size for the mean-square convergence, the value of the normalized step size is $0 < \beta < 2$. Replacing μ, in the LMS weight vector update equation (4.12), with μ_n in equation (4.13), leads to a Normalized version of equation (4.12), which is given by

$$w_{k+1} = w_k + \beta \frac{x_k^*}{\|x_k\|^2} e_k \tag{4.14}$$

(Hayes, 1996).

Although the NLMS eliminates the problem of noise amplification, a similar problem occurs when $\| x_k \|$ becomes too small. This problem can be surmounted by adding a small positive number, α, to $\| x_k \|^2$ such that the NLMS algorithm is

$$w_{k+1} = w_k + \beta \frac{x_k^*}{a + \|x_k\|^2} e_k \tag{4.15}$$

(Hayes, 1996) and (Sankaran and Beex, 1996).

The learning curve for the NLMS is shown in Figure 4.9 and the learning curves of both LMS and NLMS are shown in Figure 4.10.

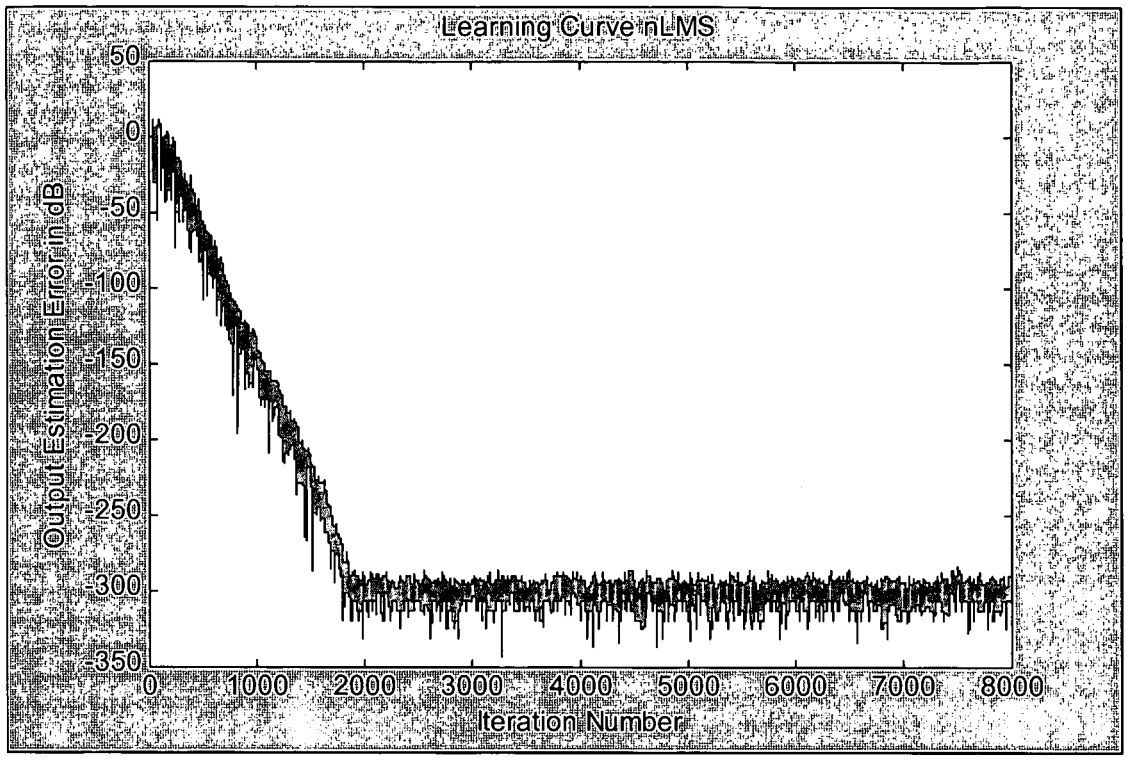

Figure 4.9. The Learning Curve of the NLMS Algorithm with White Noise as Input

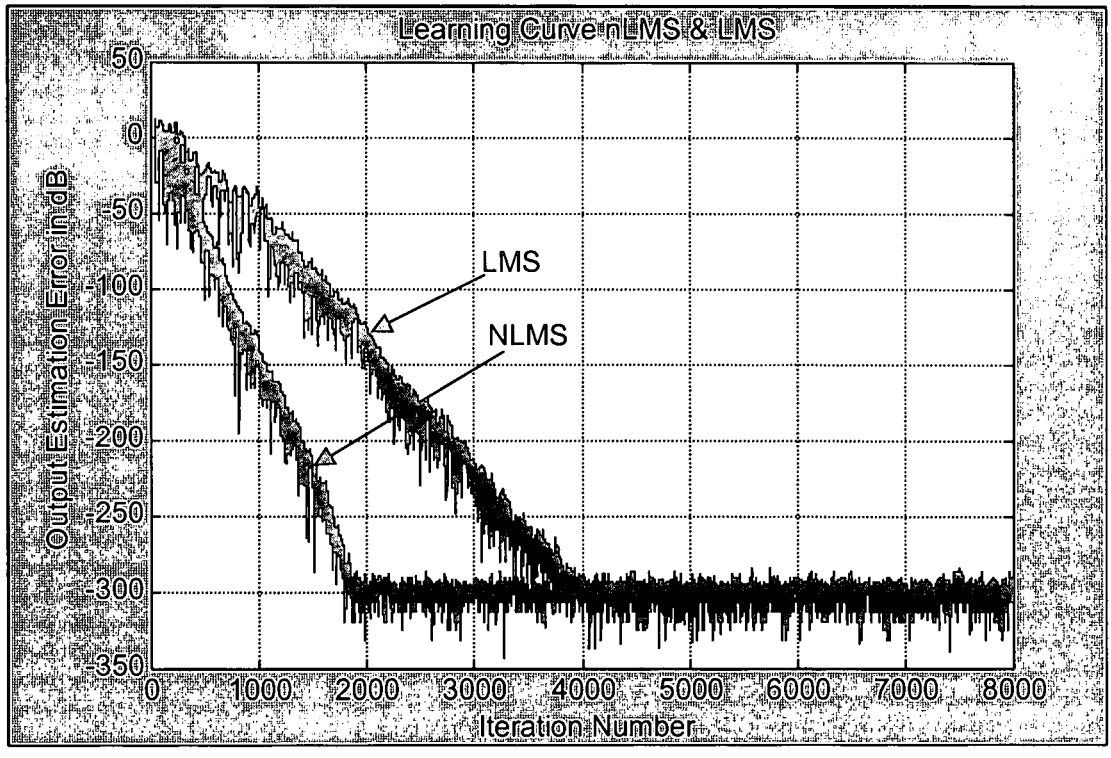

Figure 4.10. The Learning Curves of NLMS and LMS Algorithms with White

Figure 4.9. shows the learning curves of both NLMS and LMS algorithms, and comparing their learning curves it is evident that the NLMS yields a faster convergence rate than the LMS algorithm. This was also the case when the ECG signal was fed to a NLMS adaptive filter, it took less time to adapt than the LMS adaptive filter.

4.2.2.6 Implementation of the LMS and NLMS

The computational procedure of the LMS and NMLS algorithms is almost the same and is summarized below:

a) Initially, set each coefficient or weight, $w_k(c)$, c = 0, 1, 2, ..., N-1, to an arbitrary value that is fixed, such as zero.
b) Get the value of ECG signal, y_k and the reference signal, x_k

For each subsequent sampling instant, k = 1, 2, 3, ..., carry out the following steps:

c) Compute the filter output, n_k.
d) Compute the error estimate, e_k.
e) Compute the factor, g_k.
f) Update the weights, $w_{k+1}(c)$

The flowcharts for the LMS and NLMS adaptive filters are given in Figure 4.11 and Figure 4.12, respectively, and their outputs are depicted in Figure 4.13 and Figure 4.14, respectively.

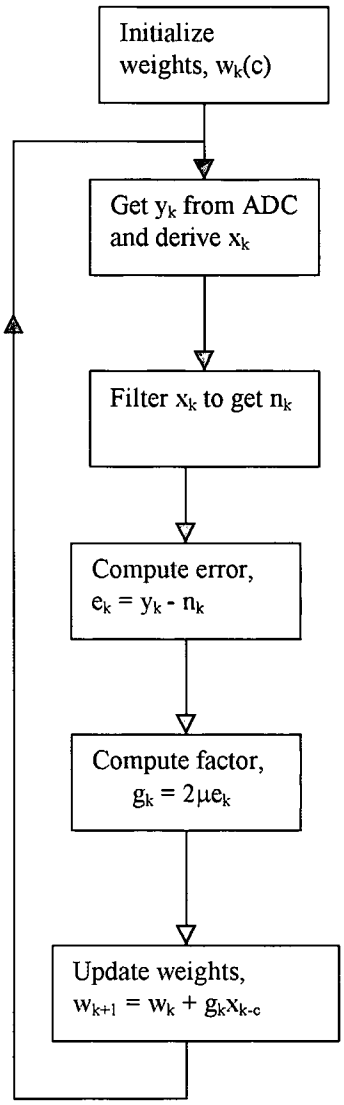

Figure 4.11 Flowchart for the LMS adaptive filter

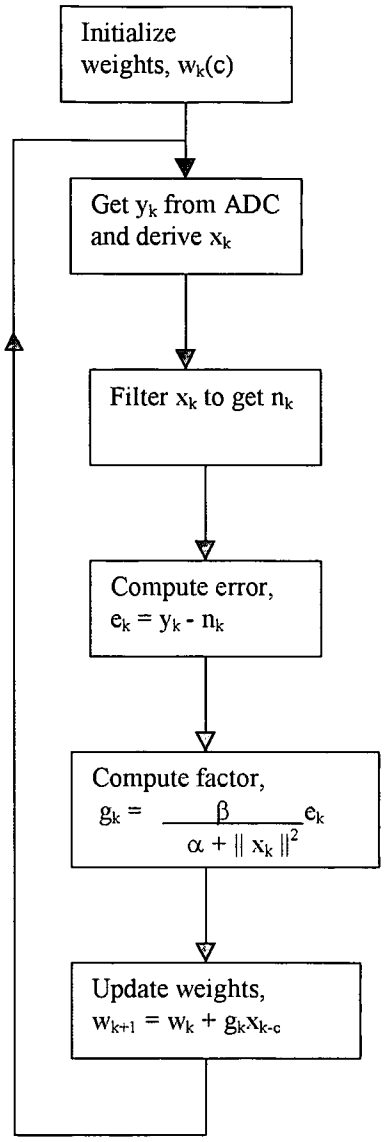

Figure 4.12 Flowchart for the NLMS adaptive filter

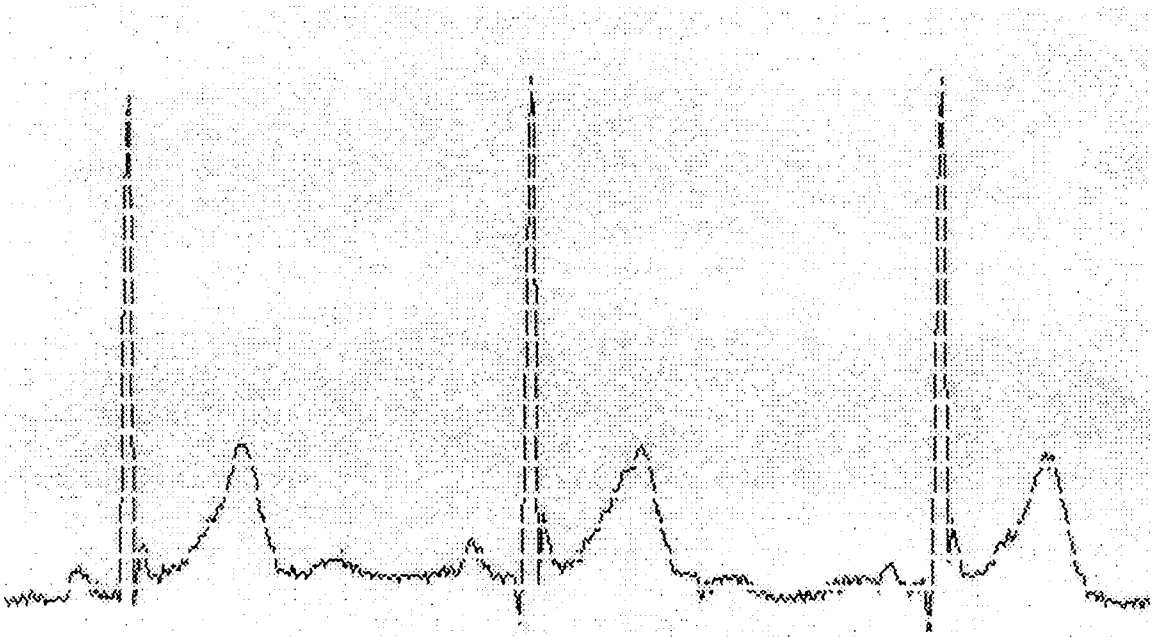

Figure 4.14 Filtered ECG signal using Adaptive LMS filter

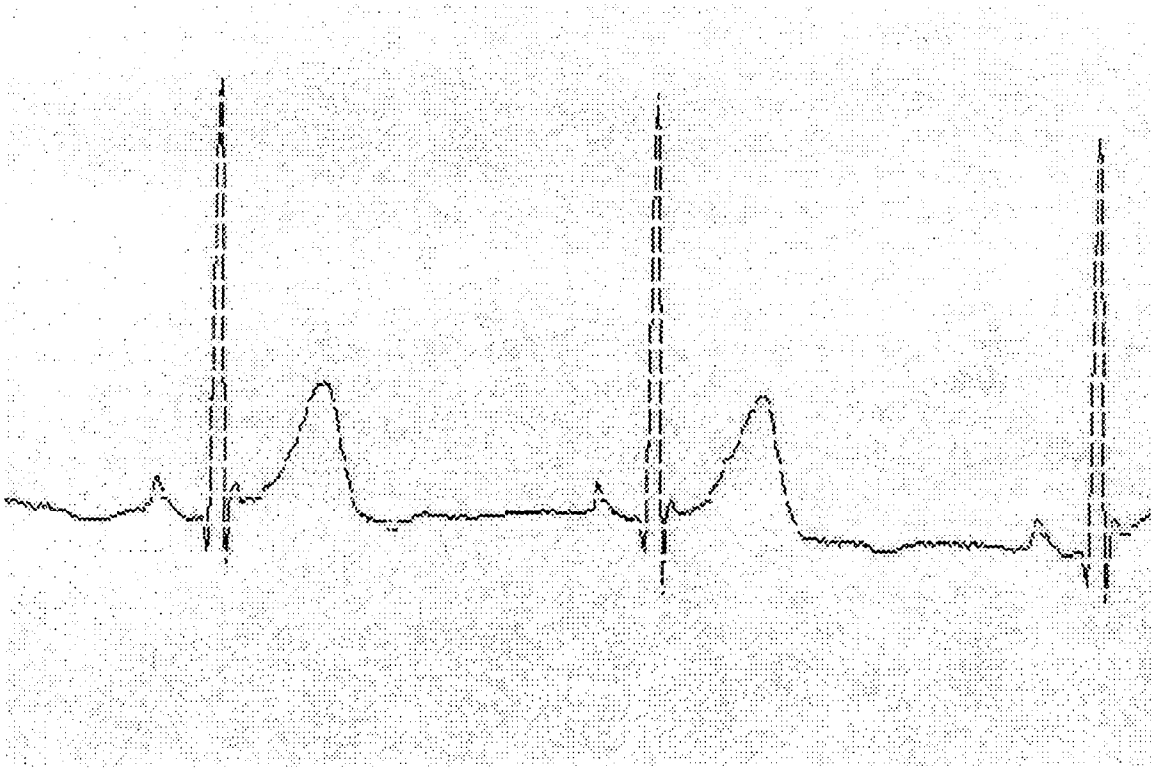

Figure 4.14 Filtered ECG signal using Adaptive NLMS filter

4.2.3. HEART RATE DETERMINATION

Now that the power line interference, high frequency electromyographic (EMG) noise, caused by muscle contractions, and the lower frequency noise caused by motion of electrodes (motion artifact) are reduced, further processing of the signal is done to get some of its diagnostic values, such as the heart rate.

In chapter one it was explained that the instantaneous heart rate can be obtained by taking the reciprocal of the time interval between the R-to-R peaks (in milliseconds) and multiply it by 60 000 (Ifeachor and Jervis, 1996). Therefore, in practice, to measure the heart rate a suitable algorithm must be employed to detect successive QRS complexes and from these calculate R-to-R intervals and the corresponding heart rate.

4.2.3.1. QRS Complex Detection

Most methods of QRS detectors reported in the literature consist of two cascaded blocks: a module that enhances the QRS complex and a module that performs the actual detection.

While emphasizing the signal of interest the first module usually reduces various kinds of noise and artifacts which are typically present in the ECG. This is done using a band pass filter. Though this band pass filtering technique maximizes the signal-to-noise ratio for detecting the QRS complex, it also attenuates non-QRS waves in the signal. The attenuation of P and T waves distorts the signal so much that the appearance of the filtered signal is not clinically acceptable (Tompkins 1995).

Since this study is not only concerned with heart rate but also with producing an electrocardiogram that is clinically acceptable, it is important that whatever technique used, there are no violations of the frequency components of the ECG.
Therefore, other technique, besides a band pass filter, has to be used to detect the QRS complexes.

The method that is implemented in this study recognizes QRS based on analysis of the slope and amplitude. Since the elimination of noise and artifact has been achieved by adaptive filtering in the previous stages, this method only covers the actual detection of QRS complex and it works as follows:

During the learning period, 640 samples of the ECG signals are taken and from these samples signal peaks are compared to get the one with the highest amplitude. Since the highest peak in any ECG signal always corresponds to the R peak of the QRS complex, then this highest amplitude will represent the R_{max}. To accommodate amplitude variability of the signal, this highest amplitude, R_{max}, is not used as a threshold point. Instead, a point that is 3 dB lower is used. Thus, the half power point of this R_{max} is determined and used as the threshold point and this threshold point and R_{max} are updated after every trace (or 640 samples).

The subsequent incoming real-time signal is compared with the R_{max} half power point, using the ascending slope, and if the level of the incoming ECG signal is greater than the threshold ($0.707R_{max}$), then a QRS complex is said to have occurred.

For every QRS detected, a systole tone is produced and the times of occurrence of the first two successive QRS complexes are recorded to get the beat period (or time interval between them). This interval, in milliseconds, between two successive R peaks is determined and its reciprocal is multiplied by 60 000 to get the instantaneous value of the heart rate.

4.2.3.2. Heart Rate Analysis

The value of the heart rate obtained is analyzed (by the computer) to determine whether it is within limits or not. If the heart rate is within limits, then it is regarded as normal, but if it is high, a high level alarm is set to indicate tachycardia. Similarly, if the heart rate is low a bradycardia alarm is set.

When this analysis is done, the age group of the patient monitored is taken into consideration, since the limits for a child younger than 4 years are not the same as that of an adult. The normal heart rate for a younger child is from 80 beats per minute to 200 beats per minute, whereas that for adults and older children is between 60 beats per minute and 100 beats per minute with less than 10 percent variations (Wartak, 1997). Based on these normal heart rates, the low limits are

set to be 75 beats per minute for younger children and 55 beats per minute for adults and the high limits are set to be 205 beats per minute for younger children and 105 beats per minute for adults.

4.3 TEMPERATURE PROCESSING

This monitoring system is designed such that, though it can measure temperature and ECG signals virtually in parallel, it gives the user the option of choosing whether the temperature measurements must be done or not. If the user chooses 'yes' to temperature measurement, the computer control the analog multiplexer, as explained in section 2.6, so that both temperature and ECG signals can be pre-processed and presented to the computer for further processing.

Since during temperature signal conditioning, 0^0C was changed to 0 V and 100^0C was made to correspond to 5 V, (to accommodate the analog-to-digital converter) each temperature value read is multiplied by 20 so as to convert it from 0 to 5 scale to a 0 to 100 scale.

Once this conversion is done the temperature is analyzed (by the computer) to determine whether it is within limits. If the temperature read is within limits (35 to 38 degrees Celsius), then it is regarded as normal, but if it is greater than 40 degrees Celsius a high level alarm is set to indicate hyperthermia. Similarly, if the temperature is below 35 degrees Celsius a hypothermia alarm is set.

CHAPTER 5

CONCLUSIONS

The purpose of this study was the development of an affordable personal computer-based temperature, heart rate and electrocardiogram monitoring system that uses three electrodes. As part of research, the investigation of cheaper technology that can be used to provide medical sensor isolation, and the development of signal processing techniques that can be used to eliminate the interference that contaminate the ECG signals, have been done.

Since the main reason for unavailability or shortage of these ECG equipment is their retail price that was ranging from R13 000 upward, the use of the personal computer has enabled the implementation of solution that is predominantly software-based. This has reduced the amount of hardware and cost of development to about 8 percent when the computer is excluded and to about 45 percent when personal computer is included.

The use of the computer has enabled the user to store and retrieve the ECG, heart rate and temperature recordings on a computer hard drive or diskette. It has also enabled the printing of current and stored results using ordinary printing paper instead of expensive thermal paper, that is used by other electrocardiograph machines. Another advantage of using computer is its e-mail capabilities that will enable the user to send the recordings to other physicians for consultation.

The developed monitoring system has all the basic features of the ECG machine, like systole tone, fully floating and isolated input, adaptive line frequency filter to suppress the 50 Hz mains interference and more. The low development cost and the low running cost has made this developed monitoring system to be more affordable than the traditional ECG machines and due to its affordability, every clinic, hospital and general practitioner can afford to have one thus improving their medical services to the broader community.

While the monitoring system developed here is working well, the eventual goal is to develop a personal computer-based system that would be able to monitor most of the electrophysiological signals, such as electroencephalography (EEG), electromyography (EMG), electroretinography (ERG) and electrooculugraphy (EOG).

CHAPTER 6

REFERENCES

Axelson, J. (1994). How to Use a PC's Parallel Port for Monitoring and Control Purposes, *Microcomputer Journal May/June* pp.14-25.

Axelson, J. (1996). *Parallel Port Complete*, Lakeview Research, Madison.

Bañez, J.G. (1999). Electrophysiology, Text from the internet address http://home.wish.net/~gallardo/ecgtutorial/chap2.html

Barney, G. C. (1988). *Intelligent Instrumentation*, Prentice Hall, London.

Greene, K. R. (1987). The ECG waveform. In *Balliere's Clinical Obstetrics and Gynaecology* (M. Whittle, ed.), Vol. 1, pp. 131-155.

Hayes, M. H. (1996). *Statistical Digital Signal Processing and Modelling*, John Wiley & Sons, New York.

Haykin, S. (1996). *Adaptive Filter Theory*, 3rd edition, Prentice Hall, Englewood Cliffs, New Jersey.

Horowitz, P. and W. Hill (1994). *Art of electronics*, 2nd edition, Cambridge University Press, Cambridge.

House, S.D. (1997). The Human Electrocardiogram, Text from the internet address: http://www.shu.edu/academics/art_sci/Unde.../physiology/frog/hearthuman/hearthum.htm

Ifeachor, E.C. and B. W. Jervis (1996). *Digital Signal Processing, a practical approach*, Addison-Wesley, Wokingham, England.

Mulder, M. and G. Nowlan (1986). *Practical guide for general nursing science - determining temperature and pulse rate of a patient,* Haum Educational Publishers, Pretoria.

Plonsey, R. (1996). *Electronic Engineer Handbook - Electrocardiography and Biopotentials* (4th edition), McGraw-Hill, New York.

Pham, H. and L. Usher (1994). Enzyme Immobilisation Methods, Text from the internet address: http://www.esb.ucp.pt/~bungah/immob/enzymeac.html

Rhoades, R. and R Pflanzer (1996). *Human Physiology*, 3rd edition, Saunders College Publishing, Fort Worth.

Sankaran, S. G. and A. A. (Louis) Beex (1996). Implementation and Evaluation of Echo Cancellation Algorithms, MSc. Eng. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Shier, D. N., J. L. Butler and R. Lewis (1996). *Hole's Human anatomy and Physiology - Electrocardiogram (clinical application)* (7th edition), WMC Brown, Dubuque, IA.

Solomon, E. P., R. R. Schmidt and P. J. Adragna (1990). *Human anatomy and physiology - Body temperature affects heart rate* (2nd edition), Saunders College Publishing, Fort Worth.

Stulens, P. (1996). Linear Optocoupler - application newsletter and notes. CP Clare Corporation.

Tompkins, W. J., editor (1995). *Biomedical Digital Signal Processing,* Prentice Hall, New Jersey.

Tompkins, W. J. and J.G. Webster, editors (1988). *Interfacing sensors to the IBM PC,* Prentice-Hall, Englewood Cliffs, New Jersey.

Wartak J. (1997). Electrocardiogram Rhythm Tutor, Text from the internet address: http://doyle.ibme.utoronto.ca/EKG/rhythm/EKGTUTORIAL.HTML

Webster J. G. (1978). *Medical instrumentation (Application and Design)*, Houghton Mifflin Company, Boston.

Young P. H. (1990). *Electronic Communication Techniques*, 2nd edition, Intercomputer Communications, Merill Publishing Company, Columbus.

CHAPTER 7

APPENDIXES

7.1. C PROGRAM FOR THE PROJECT

The program for the project is written in turbo C and is as follows:

```
/*************************************************************************/
/*          T E M P E R A T U R E  AND  E C G  MONITOR                    */
/*        This program reads Temperature and ECG signals via the         */
/*            parallel port and displays them on the screen              */
/*                                                                       */
/*        From the ECG signal, the beat (or heart) rate is determined    */
/*                              &                                        */
/*        There is a LMS, nLMS adaptive filter and a Notch to stop 50Hz  */
/*                                                                       */
/*                          by phumzile malindi                          */
/*                          NOVEMBER 1999                                */
/*                                                                       */
/*************************************************************************/


#include <time.h>
#include <stdio.h>
#include <dos.h>
#include<conio.h>
#include <stdlib.h>
#include <float.h>
#include <graphics.h>
#include <math.h>
```

```
#include <string.h>
#include <mem.h>
#include <ctype.h>
#include <io.h>
#include <alloc.h>
#include <bios.h>

#define     ESC     '\x1B'
#define     LPT1    0
#define     LPT2    1
#define     prn_putc(x)   biosprint(0,(x),LPT1)

#define SYSTOLE 8
#define ADC_START 1
#define CH_0 0
#define CH_1 2
#define CH_2 4
#define H_ALM 16
#define L_ALM 32
#define LEADS 64
#define u  0.004
#define e  0.0000015
#define beta  .025
#define order 4       /* filter length */
#define dly 3         /* delay */
#define Fs 200        /* sampling frequency */

 /*  Sets Epson printer to bit image mode.  */
 /* Nbytes is the number of bytes to print.  */

 static void bitImage(int Nbytes)
```

```
{
  register int    n1, n2;

   n2 = Nbytes >> 8;
   n1 = Nbytes - (n2 << 8);

   prn_putc(ESC);
   prn_putc('*');
   prn_putc(4);
   prn_putc(n1);
   prn_putc(n2);
}

/*  Get pixels from the screen and convert them to the printer's pin order.  */

static unsigned char getScrBits(int x, int y)
{
   unsigned char firePins;

   firePins  = (getpixel(x, y++)==0)? 0: 0x80;
   firePins |= (getpixel(x, y++)==0)? 0: 0x40;
   firePins |= (getpixel(x, y++)==0)? 0: 0x20;
   firePins |= (getpixel(x, y++)==0)? 0: 0x10;
   firePins |= (getpixel(x, y++)==0)? 0: 0x08;
   firePins |= (getpixel(x, y++)==0)? 0: 0x04;
   firePins |= (getpixel(x, y++)==0)? 0: 0x02;
   firePins |= (getpixel(x, y  )==0)? 0: 0x01;

   return    firePins;
}
```

```
void menu();                    /* menu for graphics scrn */
void initnlms();                /* Initializing wk(i) & xk-i */
double nlmsflt();               /* nLMS ANC */
double lmsflt();                /* LMS ANC */
double notch();                 /* 50 Hz NOTCH FILTER */
double QRS_BPF();               /* To Select QRS Complexes */
int printimage(int left, int top, int right, int bottom);      /* Graphics print Fn */
void graph_setup();              /* Detecting and initializing graphics hardware */
void Quit_monitor();            /* To stop monitoring process */
void Main_scrn();               /* The opening page */
void monitor_pgm();             /* To Monitor & Display results */
void port_set();                /* To change to another parallel port */
void temp_limit();              /* To set new temperature limits */
void hrate_limit();             /* To set new beat rate limits */
int amplify();                  /* To increase the amplitude of the ECG signal */
void rec_load(void);            /* Load records from the disk*/
void rec_save();                /* Save records to the disk */
void Keyboard();                /* check for a keystroke */
void prepare_l();               /* To prepare records to be loaded */
void prepare_s();               /* To prepare records to be saved */
void display_BW();              /* To display the records in Black & White */
void display_CLR();             /* To display the records in colour */
double get_input();             /* READ ADC */
double get_temp();              /* Read and process temperature */
double ecg_BPF();               /* To reduce base line shifts & hi_freq noise */
static void bitImage(int Nbytes);      /* Sets Epson printer to bit image mode.*/
static unsigned char getScrBits(int x, int y); /* Get pixels from the screen and
                                                  convert them to the printer's pin order. */

float Ecg = 0,  ecg=0, ecg_data[640], data[640], z = 0, zmax = 0, zave = 0, i_peak = 0;
float R_period = 0, t_period = 0, b_rate = 0, temperature = 0, x[order+1], w[order+1];
```

```
char temp[1000], status_t[80], status_hr[80], h_rate[1000], name[80], sex[8],
        t_type[12];
char pname[80], psex[8], t_status[1000], hr_status[1000], age[2],r_date[1000];
int  p_age= 0, hl_limit = 0, hu_limit = 0, date_r[80], CH = 0, ALM = 0, BASE = 0x278;
struct date t;
clock_t begin, end;
int n = 0, beat, maxx = 0, maxy = 0, d_out = 0, result_s = 0, zero = 0, result_c = 0;
float delay_t = 0, temp1, temp2, tempAve;
int c = 0, Ecg_gain = 30, ch, l_temp = 0, h_temp = 0, try;

struct p_struct
        {
        char name[40];
        char sex[4];
        char status_hr[20];
        char status_t[20];
        long  age;
        struct date rec_date;
        float temp;
        float rate;
        float ecg_rec[640];
        }p_datain[1], p_dataout[1];

FILE *fp, *infile;

int main(void)
{
 while(1){
     closegraph();
     clrscr();
     Main_scrn();
```

```
    }
}

/*********************************************************************
 Monitoring the patients ECG and temperature and displaying the results
 *********************************************************************/

void monitor_pgm()
 {
  clrscr();
  graph_setup();
  try = 0;
  /* Entering Patient's Particulars */
  printf("ENTER NAME:");
  gets(p_datain[0].name);
  printf("\nENTER SEX (F OR M):");
  scanf("%s",&p_datain[0].sex);
  printf("\nENTER AGE:");
  getdate(&p_datain[0].rec_date);
  scanf("%d",&p_datain[0].age);
  if(p_datain[0].age < 4){ hl_limit = 75;
                  hu_limit = 205;}
  if(p_datain[0].age >= 4){ hl_limit = 55;
                  hu_limit = 105;}
  fflush(stdin);                          // flush keyboard buffer
  printf("\nDo you also want to monitor the Patient's Temperature [Y/N]?");
  scanf("%c",&ch);
         switch(toupper(ch))
          {
           case 'Y':l_temp = 34;h_temp = 38;strcpy(t_type, "-(Patient)"); break;
```

```
case 'N':l_temp = 0;h_temp = 100;strcpy(t_type, "-(---)"); strcpy(status_t,
                                                    "NOT MEASURED");break;
        }
  printf("\n\n\n\n\n\n\n\n\n\n\n\t\t\t Starting, please wait....\n");
// clrscr();                          //clear scrn

/* Initialising ADC */
d_out = ADC_START;
outport(0x278,d_out);
delay(50);
outport(0x278,0);

/* Configuring Control register as an input */
outport(0x27a,04);

  initnlms();
  t_period = 1000/Fs;
  delay_t = t_period*25/1000;          /* (delay*sampling freq)/no. of bits */
/* READ CH 2 i.e. ECG to determine R peaks */

CH = CH_2;
d_out = CH + ALM;
outport(0x278,d_out);
 c = 0;

while( c <640)
{
Ecg = get_input()*Ecg_gain;
ecg_data[c] = nlmsflt();//QRS_BPF();
delay(t_period);
c++;
```

```
}

/* Start Monitoring */
while(!kbhit()){
   d_out = CH + ALM;

   if(ch == 'Y'|| ch == 'y'){get_temp();}
   else
fflush(stdin);                  // flush keyboard buffer

/* Getting the largest value of R peak */

data[0] = max(ecg_data[1], ecg_data[0]);

for(n = 1; n < 640; n++){
  data[n] = max(ecg_data[n+1], data[n-1]);
zmax = max(ecg_data[n+1], data[n]);
}

i_peak = (zmax*0.75 );        /* intermidiate value of R peak = Rmax-3dB */
  beat = 0;

/* determining the heart rate */

CH = CH_2;
d_out = CH + ALM;
outport(0x278,d_out);
c = 0;

R_period = (((end - begin))/ CLK_TCK) - delay_t;
b_rate = 60/R_period;
```

```
p_datain[0].rate = b_rate;
if(b_rate <= 1) {strcpy(status_hr, "NO SIGNAL"); try += 1;}
if(b_rate > hu_limit)
  { strcpy(status_hr, "TACHYCARDIA"); try += 1;}        /* High Beat Rate */
if(b_rate > 0 && b_rate < hl_limit)
  { strcpy(status_hr, "BRADYCARDIA"); try +=1;}       /* Low Beat Rate */
else if((b_rate <= hu_limit)&&(b_rate >= hl_limit))
        { strcpy(status_hr, "NORMAL"); }

if((b_rate > hu_limit)||(temperature > h_temp)){ALM = H_ALM;}
if((b_rate < hl_limit)||(temperature <= l_temp)){ALM = L_ALM;}
else if((b_rate <= hu_limit)&&(b_rate >= hl_limit)&&(temperature <=
h_temp)&&(temperature > l_temp)){ALM = 0;}

 /* Displaying the results*/

 clrscr();
setcolor(RED);
rectangle(0,0,maxx,maxy);
setbkcolor(7);
for(n=1; n<=15; n++)
{
setcolor(CYAN);
line(n*maxx/16,70,n*maxx/16,maxy);
line(0,(maxy*n/10)+70,maxx,(maxy*n/10)+70);
line(0,70,maxx,70);
line(0,72,maxx,72);
}

  setcolor(RED);
```

```
outtextxy(2,(maxy+15),"<P - PRINT> <S - SAVE> <X - QUIT>");

 sprintf(r_date,"Date       : %d-%d-%d",p_datain[0].rec_date.da_day,
  p_datain[0].rec_date.da_mon, p_datain[0].rec_date.da_year);
 outtextxy(6,40,r_date);
 sprintf(name,"Name       : %s",strupr(p_datain[0].name));
 outtextxy(6,10,name);

 sprintf(sex,"Sex        : %s",strupr(p_datain[0].sex));
 outtextxy(6,20,sex);
 sprintf(age,"Age        : %d years",p_datain[0].age);
 outtextxy(6,30,age);
sprintf(h_rate, "Heart Rate  : %.0f beats per minute",b_rate);
outtextxy(6,50,h_rate);
sprintf(hr_status, "%s", status_hr);
 outtextxy(350,50,hr_status);
sprintf(temp,"Temperature : %.2f Deg Celsius",temperature);
outtextxy(6,60,temp);
 sprintf(t_status, "%s", status_t);
 outtextxy(350,60,t_status);
moveto(0,zero);
c = 0;

/* READ CH_2 i.e. electrocardiography */

 CH = CH_2;
 d_out = CH + ALM;
 outport(0x278,d_out);
 c= 0;
while(c < 640)
 {
```

```
Ecg = get_input()*Ecg_gain;
ecg = nlmsflt();
ecg_data[c] = ecg;
p_datain[0].ecg_rec[c] = ecg;
if(ecg > i_peak){outport(0x278, d_out + SYSTOLE);}
if(ecg < i_peak){outport(0x278, d_out);}
if((ecg_data[c] > i_peak)&&(ecg_data[c-1] < i_peak)) /* determine the R peaks */
  { beat += 1;}
if(beat == 1){ begin = clock();
               }
if(beat == 2){ end = clock();
               }
 lineto(c, (zero-ecg));
 delay(t_period);
 c++;
}
if(try >= 10)
  { ALM = LEADS;
    closegraph();
    clrscr();
    printf("\n\n\n\n\n\n\n\n\n\n\n\t\t\t Auto Shut- off! Check the leads\a\a\a\n");
    printf("\n\t\t\t Press any key to continue.....\n");
    getch();
    Main_scrn();
  }

 c = 0;
}

menu();
}
```

```
/* menu for graphics display */
void menu()
   {
   ch = getch();

   switch(toupper(ch))
        {
         case 'S': rec_save();break;
         case 'P': prepare_s(); display_BW(); printimage(0, 0, maxx, maxy); Main_scrn();
         case 'X': Quit_monitor();
         }
        }
/************************ END MONITORING PROCESS ***************/

/*********************************************************************
   PREPARING THE DATA FOR PRINTING, SAVING AND LOADING
 *********************************************************************/

 /* copy to output structure for saving */

 void prepare_s()
    {
    closegraph();

    p_dataout[0] = p_datain[0];
    }

/* copy to input structure for loading */
 void prepare_l()
    {
    p_datain[0] = p_dataout[0];
```

```
    }

/* Change display to black and white for printing */
void display_BW()
   {
 closegraph();
 clrscr();
 graph_setup();
 setcolor(WHITE);
rectangle(0,0,(maxx-20),maxy);
setbkcolor(0);
for(n=1; n<=15; n++)
{
setcolor(CYAN);
//line(n*(maxx-20)/16,70,n*(maxx-20)/16,maxy);
//line(0,(maxy*n/10)+70,(maxx-20),(maxy*n/10)+70);
line(0,70,(maxx-20),70);
line(0,72,(maxx-20),72);
}

 setcolor(WHITE);

outtextxy(2,(maxy+15)," <P - PRINT> <S - SAVE> <X - QUIT>");

 sprintf(r_date,"Date       : %d-%d-%d",p_datain[0].rec_date.da_day,
 p_datain[0].rec_date.da_mon, p_datain[0].rec_date.da_year);
 outtextxy(8,32,r_date);
 sprintf(name,"Name       : %s",strupr(p_datain[0].name));
 outtextxy(8,8,name);

 sprintf(sex,"Sex        : %s",strupr(p_datain[0].sex));
```

```
 outtextxy(8,16,sex);
 sprintf(age,"Age        : %d years",p_datain[0].age);
 outtextxy(8,24,age);
sprintf(h_rate, "Heart Rate  : %.0f beats per minute",p_datain[0].rate);
outtextxy(8,40,h_rate);
sprintf(hr_status, "%s", status_hr);
 outtextxy(300,40,hr_status);
sprintf(temp,"Temperature : %.2f Deg Celsius",p_datain[0].temp);
outtextxy(8,48,temp);
 sprintf(t_status, "%s", status_t);
 outtextxy(300,48,t_status);
moveto(0,zero);

 for(c = 0; c <= 640; c++)
        {lineto(c, (zero-p_datain[0].ecg_rec[c]));}
}

/* Displaying in colour */
void display_CLR()
   {
 closegraph();
 clrscr();
graph_setup();
 setcolor(RED);
rectangle(0,0,maxx,maxy);
setbkcolor(7);
for(n=1; n<=15; n++)
{
setcolor(CYAN);
 line(n*maxx/16,70,n*maxx/16,maxy);
line(0,(maxy*n/10)+70,maxx,(maxy*n/10)+70);
```

```
line(0,70,maxx,70);
line(0,72,maxx,72);
}

 setcolor(RED);

outtextxy(2,(maxy+15)," <P - PRINT> <S - SAVE> <X - QUIT>");
 sprintf(r_date,"Date       : %d-%d-%d",p_datain[0].rec_date.da_day,
 p_datain[0].rec_date.da_mon, p_datain[0].rec_date.da_year);
 outtextxy(8,32,r_date);
 sprintf(name,"Name       : %s",strupr(p_datain[0].name));
 outtextxy(8,8,name);
 sprintf(sex,"Sex        : %s",strupr(p_datain[0].sex));
 outtextxy(8,16,sex);
 sprintf(age,"Age        : %d years",p_datain[0].age);
 outtextxy(8,24,age);
sprintf(h_rate, "Heart Rate  : %.0f beats per minute",p_datain[0].rate);
outtextxy(8,40,h_rate);
sprintf(hr_status, "%s", status_hr);
 outtextxy(300,40,hr_status);
sprintf(temp,"Temperature : %.2f Deg Celsius",p_datain[0].temp);
outtextxy(8,48,temp);
 sprintf(t_status, "%s", status_t);
 outtextxy(300,48,t_status);
moveto(0,zero);

 for(c = 0; c <= 640; c++)
        {lineto(c, (zero-p_datain[0].ecg_rec[c]));}
}
```

```
/*****************************************************************/
/*          FILTERING OF THE ECG SIGNAL                          */
/*****************************************************************/

/*50 Hz NOTCH FOR CANCELING MAINS INTERFERENCE (for 200 Hz SAMPLING
FREQ.  */

double notch(signal)
        float signal;                       /* x[n] */
    {
        int count;
        float x[11], y[11];

         signal = Ecg;

        for(count = 10;count > 0; count--)
          { x[count] = x[count - 1];}
          x[0] = signal;

 y[0] = (0.5835*x[0]-0.0003*x[1]+2.9175*x[2]-0.0011*x[3]+5.8351*x[4]-
        0.0016*x[5]+5.8351*x[6]-0.0011*x[7]+2.9175*x[8]-0.0003*x[9]+
        0.5835*x[10]+0.0004*y[1]-4.0280*y[2]+0.0014*y[3]-6.4414*y[4]
        +0.0016*y[5]-5.0478*y[6]+0.0008*y[7]-1.8959*y[8]+0.0001*y[9]
        -0.2592*y[10]);

        for(count = 10; count > 0; count--)
          {y[count] = y[count -1];}

        return(y[0]);
    }
```

```
/*      normalized LMS adaptive noise canceller    */
/*****************************************************/

/* initialise nLMS */
void initnlms()
      {
      int i;
      for(i = 0; i <= order; ++i){
         x[i] = 0;
         w[i] = 0;
         }
      }

/* nLMS filter */
double nlmsflt()
      {
      double nk = 0;
      float f, ek = 0;
      float signal;                              /* x[n] */
      int count;
      static float x[order+1];

       signal = Ecg;

      for(count = order;count > 0; count--)
       { x[count] = x[count - 1];}
        x[0] = signal;

      for (count = 0; count < order; ++count){     /* digital filtering */
         nk = nk + w[count]*x[0-dly];
         }
```

```
    ek = Ecg - nk;                          /* compute error */

    f = beta/(e+x[order]*x[0-dly]);          /* compute factor */
    for(count = 0; count < order; count++){  /*update coefficient */
      w[count] = w[count] + f*ek*x[0-dly];
      }
    return(ek);
  }

        /* LMS filter */
        /*************/

double lmsflt()
    {
    double nk = 0;
    float g, ek = 0;
    float signal;                    /* x[n] */
    int count;
    static float x[order+1];

     signal = Ecg;

    for(count = order;count > 0; count--)
     { x[count] = x[count - 1];}
      x[0] = signal;

    for (count = 0; count < order; ++count){   /* digital filtering */
      nk = nk + w[count]*x[0-dly];
      }
```

```
        ek = Ecg - nk;                          /* compute error */

        g = 2*u*ek;                             /* compute factor */
        for(count = 0; count < order; count++){ /*update coefficient */
          w[count] = w[count] + g*x[ 0-dly];
          }
        return(ek);
      }

/********************* END OF FILTERING PROCESS *****************/

/*****************************************************************
  PRINTING GRAPHICS RESULTS on an Epson compatible printer.
  PROTOTYPE: printimage(int left, int top, int right, int bottom);
 *****************************************************************/

 /*  Graphics print function.  */

 int printimage(int left, int top, int right, int bottom)
 {
   int      x, y, width, height;

   width  = right-left;
   height = bottom-top;

   /* Initialize line spacing to 7/72" */
   prn_putc(ESC);
   prn_putc('1');

   for (y=0; y<height; y+=8)
   {
```

```
            bitImage(width);

            for (x=0; x<width; x++)
                  prn_putc(getScrBits(x,y));

            prn_putc('\n');
      }
      return    0;
 }
 /*****************************END of PRINT fuctions*****************/
 /*************************************************************************/
/*                GRAPHICS DETECTION AND INITIALIZATION                  */
/*************************************************************************/

 void graph_setup()
      {
          /* request autodetection */

  int gdriver = DETECT, gmode, errorcode;

  /* initialize graphics mode */
  initgraph(&gdriver, &gmode, "a:\\");

  /* read result of initialization */
  errorcode = graphresult();

  if (errorcode != grOk)    /* an error occurred */
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();
```

```
    exit(1);            /* return with error code */
  }
        maxx = getmaxx();
        maxy = getmaxy() - 40;
        zero = (maxy/2) + 40;
 }

/************************ END of GRAPHICS functions ***************/

/*****************************************************************
        Quit monitoring and go back to the main (or opening ) page
*****************************************************************/

 void Quit_monitor()
        {
        cleardevice();
        closegraph();
        Main_scrn();
        return;
     }
/************************  END  ********************************/

/*****************************************************************
            THE MAIN PAGE (OR OPENING PAGE
*****************************************************************/

void Main_scrn(void)
{
closegraph();
clrscr();
 while(1)
```

```
{
clrscr();
printf("\n\tÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ\n");
printf("\tÛ                                                    Û\n");
printf("\tÛ     ELECTROCARDIOGRAPHY AND TEMPERATURE            Û\n");
printf("\tÛ                                                    Û\n");
printf("\tÛ                MONITORING SYSTEM                   Û\n");
printf("\tÛ                                                    Û\n");
printf("\tÛßßßßßßßßßßßßßßßßßßßßß    BY PHUMZILE MALINDI   ßßßßß Û\n");
printf("\tÛ      íííííííííííííííííííííííííííííííííííííííííííí     Û\n");
printf("\tÛ                                                    Û\n");
printf("\tÛ      ************** MENU **************            Û\n");
printf("\tÛ                                                    Û\n");
printf("\tÛ      M - MONITOR                                   Û\n");
printf("\tÛ      C - CHANGE PORT                               Û\n");
printf("\tÛ      T - CHANGE TEMPERATURE LIMITS                 Û\n");
printf("\tÛ      H - CHANGE HEART RATE LIMITS                  Û\n");
printf("\tÛ      A - INCREASE THE AMPLITUDE OF THE ECG         Û\n");
printf("\tÛ      L - READ FROM THE DISK                        Û\n");
printf("\tÛ      E - EXIT THE PROGRAM                          Û\n");
printf("\tÛ                                                    Û\n");
printf("\tÛ     Press one of the above keys to continue\a      Û\n");
printf("\tÛ                                                    Û\n");
printf("\tÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ\n");

switch(toupper(getch()))
 {
  case 'M': monitor_pgm(); break;
  case 'C': port_set(); break;
  case 'T': temp_limit(); break;
  case 'H': hrate_limit(); break;
```

```
 case 'A': amplify(); break;
 case 'L': rec_load();
 case 'E': exit(0);
 }
}
}
/************************ END of OPENNING PAGE ***************/


/*****************************************************************
        Save/load an entire screen image
*****************************************************************/

/* save the current screen */
void rec_save()
{
char fname[40];

 prepare_s();
 closegraph();
 clrscr();

printf("Enter the file name and directory [e.g. c:\\records\\name.dat] :\n");
gets(fname);

fp = fopen(fname, "wb");
if (fp == NULL)
 { printf("Error opening file.\n");}
 else
  {
     fwrite(&p_dataout,  sizeof(struct p_struct), 1, fp);
```

```
      printf("\n Saving...");
 }
 fclose(fp);
 }

void  rec_load(void)
{
  char f_name[40];

 clrscr();
 printf("Enter the file name and directory [e.g. c:\\records\\name.dat] :\n");
 gets(f_name);

 infile = fopen(f_name, "rb");
 if (infile == NULL)
   { printf("Error opening file.\n");
     getch();
     Main_scrn();
   }
 else
   {
     fread(&p_dataout,  sizeof(struct p_struct), 1, infile);
     printf("\n Loading...");
     delay(40);
     prepare_l(); display_CLR(); menu();
  }
 fclose(infile);
 }
/********************* END SAVE & LOADING of RECORDS*   ********/

/*****************************************************************
```

READING THE OUTPUT OF THE ADC & PROCESSING OF TEMPERATURE
**********/

```
double get_input()
 {
 float reading= 0;

  result_c = inportb(0x27a);    /* to read the control port */
  result_c = result_c^11;       /* to invert the bit 0, 1 & 3 */
  result_c = result_c&7;        /* to ignore bits 3 - 7 */
  result_s = inportb(0x279);    /* to read the status port */
  result_s = result_s^128;      /* to invert the bit 7 */
  result_s = result_s&248;      /* to ignore bits 0 - 2 */

  reading = (result_s + result_c)*0.01953;

  return(reading);
  }

 /* READ CH 0 i.e. temperature readings */

 double get_temp()
  {
 delay(1);
 CH = CH_0;                    /* Select CH 0 on the MUX */
 d_out = CH + ALM;
 outport(0x278,d_out);
 delay(5);
 temp1 = get_input();
 delay(1);
 temp2 = get_input();
```

```
tempAve = (temp1 + temp2)/2;
 temperature = tempAve*20 ;
  p_datain[0].temp = temperature;

 if(temperature > h_temp)
   { strcpy(status_t, "HYPERTHERMIA");}          /* High Temperature */
 if(temperature <= l_temp)
   { strcpy(status_t, "HYPOTHERMIA");}  /* Low Temperature */
   if((temperature > l_temp)&&(temperature <= h_temp))
   { strcpy(status_t, "NORMAL");}
   strcat(status_t, t_type);
   return 0;
   }
/***** END OF ADC READ & PROCESSING OF TEMPERATURE Fns ***/

/*****************************************************************
                    CHANGING THE DEFAULT SETTINGS
*****************************************************************/

  /* Changing Heart rate limits */

  void hrate_limit()
    {
    int low, hi;

    printf("\nEnter the low limit:>");
    scanf("%d",&low);
    printf("\nEnter the high limit:>");
    scanf("%d",&hi);
    hl_limit = low;
    hu_limit = hi;
```

```
    }

/* Changing temperature limits */

 void temp_limit()
    {
    int low, hi;

    printf("\nEnter the low limit:>");
    scanf("%d",&low);
    printf("\nEnter the high limit:>");
    scanf("%d",&hi);
    l_temp = low;
    h_temp = hi;
    }

 /* Changing the gain of the ECG signal */

int amplify()
  {
  int gain;
  printf("\n ENTER THE NEW GAIN:>");
  scanf("%d", & gain);
  Ecg_gain = gain;
  return 0;
  }
  /* Change to another parallel port */

  void port_set()
  {
  int p_port= 0x278;
```

```
printf("Enter the new parallel port:>");
scanf("%x",p_port);
 BASE = p_port;
 }
/******************** End changing default values ********************/
```

7.2. POWER SUPPLY

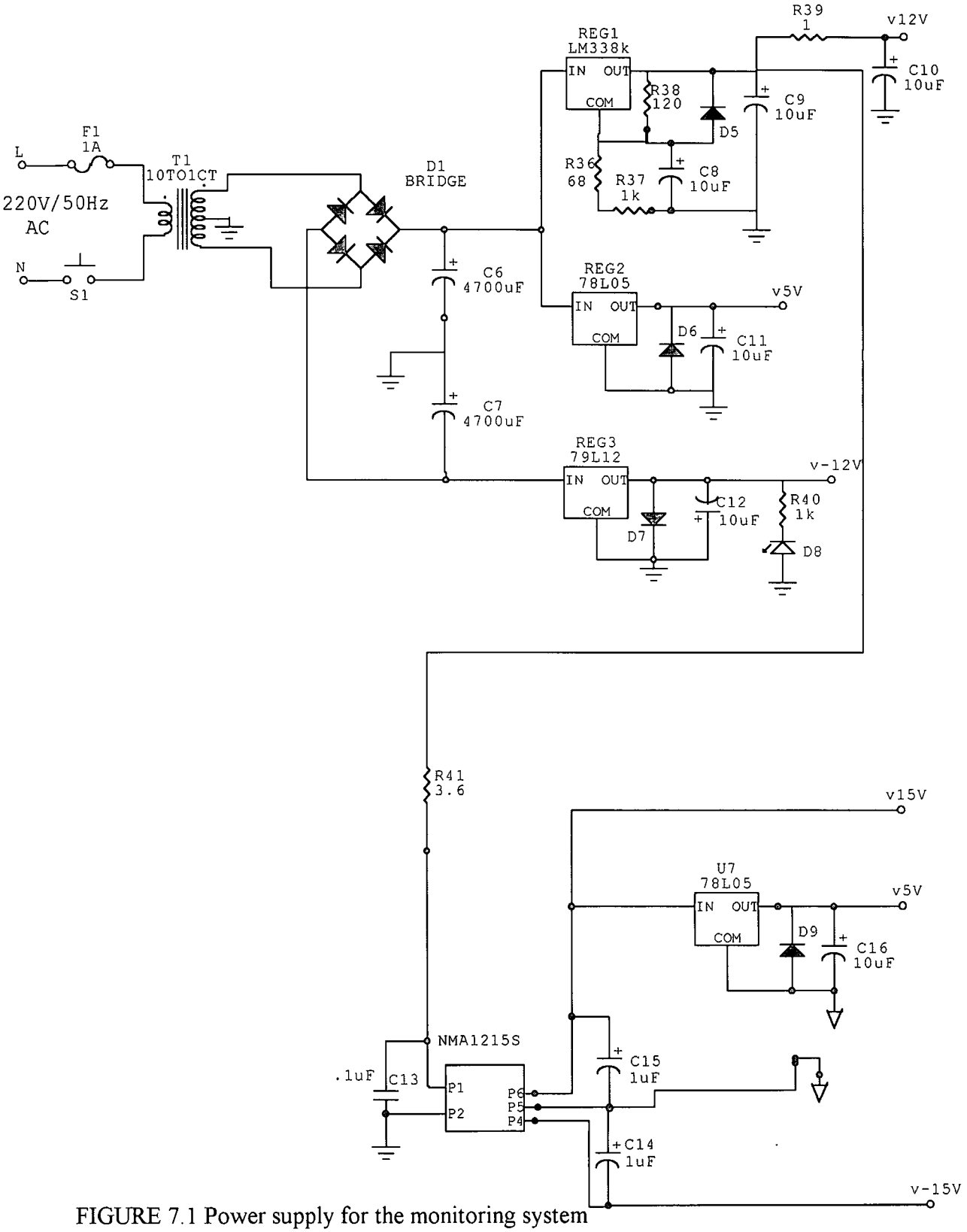

FIGURE 7.1 Power supply for the monitoring system

The power supply for this project is designed to have two stages, the primary and isolated secondary stage. The primary stage consists of a 10:1 center-tapped step down transformer, a bridge rectifier, smoothening capacitors C6 and C7, and three regulators for +12 V, +5 V and for –12 V.
The LED (D8) works with the ON/OFF switch to indicate when the power is on or off.

The secondary stage uses a NMA1215S DC-to-DC converter to produce a ±15 V from the regulated +12 V of the primary stage. This NMA1215s also provide about 1 kV isolated between secondary and primary voltages. From the +15 V another +5 V is derived, thus, making the secondary to have three output voltages, +15 V,

7.3 Acronyms

ADC	Analog-to-digital converter
DMA	Direct Memory Access
ECG	Electrocardiogram
ECP	Extended Capabilities Port
EKG	Electrokardiogram
EMG	Electromyogram
EPP	Enhanced Parallel Port
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
LMS	Least Mean Square
LOC	Linear Opto-coupler
MSE	Mean Square Error
NLMS	Normalized Least Mean Square
ODM	Open Development Method
OPAMP	Operational Amplifier
RLS	Recursive Mean Square
RTD	Resistive Temperature Detector
SPP	Standard Parallel Port
SPTool	Signal Processing Toolbox
VCG	Vectorcardiogram
XOR	Exclusive-OR