

# **BIOMEDICAL SIGNAL PROCESSING USING TMS320C6713 DSK**

Project submitted in partial fulfillment of requirements

For the Degree of

## **BACHELOR OF ENGINEERING**

BY

**ELIZABETH ALANKARA**

**POOJA BANDEKAR**

**KAVITA DATE**

**APARNA BHUKTAR**

Under the guidance of

Internal Guide

**Prof. K. T. TALELE**



Department of Electronics Engineering

Sardar Patel Institute of Technology

University of Mumbai

2008-2009

BHARTIYA VIDYA BHAVAN'S  
**SARDAR PATEL INSTITUTE OF TECHNOLOGY**

MUNSHI NAGAR, ANDHERI (W),

MUMBAI - 400 058.

2008-09

## CERTIFICATE OF APPROVAL

This is to certify that the following students

**ELIZABETH ALANKARA**

**POOJA BANDEKAR**

**KAVITA DATE**

**APARNA BHUKTAR**

have successfully completed and submitted the project entitled

**“BIOMEDICAL SIGNAL PROCESSING USING TMS320C6713  
DSK”**

towards the fulfillment of Bachelor of Engineering course in Electronics of the

Mumbai University

---

**Internal Examiner**

---

**External Examiner**

---

**Internal Guide**

---

**Head of Department**

---

**Principal**

## **ACKNOWLEDGEMENTS**

We would like to thank our internal guide and head of department, Prof K. T. Talele, for his overwhelming support during the entire phase of our project. His able guidance was instrumental in us achieving our goal.

We would also like to express our sincere gratitude to Lect. R. G. Sutar and all those who have contributed to the development of this project. Their encouragement and guidance has contributed immensely in the successful completion of this project. We would like to thank our college Sardar Patel Institute of Technology for providing us with all the resources that we required to go through the various phases of the project. The project would not have shaped up the way it has without the support and cheerful encouragement of our professors. We are thankful to the entire staff of the Electronics Department.

Finally, we would like to thank all the people who have taken painstaking efforts and provided us with information needed for the successful completion of this project.

# TABLE OF CONTENTS

1. INTRODUCTION.....	1
1.1 PROBLEM DEFINITION.....	2
1.2 SUBJECT BACKGROUND.....	3
1.2.1 TRANSDUCERS.....	3
1.2.2 SIGNAL CONDITIONING.....	3
1.2.3 FILTERS.....	4
1.2.4 PROCESSING UNIT.....	4
1.3 CONCEPTUAL BLOCK DIAGRAM.....	5
1.4 SCOPE OF THE PROJECT.....	6
2. THE ELECTROCARDIOGRAM.....	7
2.1 INTRODUCTION TO ECG.....	8
2.2 ECG WAVES.....	9
2.3 ECG LEADS.....	10
2.3.1 STANDARD LIMB LEADS.....	10
2.3.2 3-AUGMENTED LIMB LEADS.....	11
2.3.3 6-PRECORDIAL LEADS.....	11
2.4 USES OF ECG.....	12
3. LITERATURE SURVEY.....	13
3.1 GENERAL DESCRIPTION.....	14
3.2 FUNCTIONAL BLOCK DIAGRAM.....	15
3.3 COMPONENT DESCRIPTION.....	16
3.3.1 BUFFER.....	17
3.3.2 SENSORS.....	17
3.3.3 MULTIPLEXERS.....	18

3.3.4 FILTERS.....	20
3.3.4.1 LOW PASS FILTER.....	21
3.3.4.2 HIGH PASS FILTER.....	22
3.3.4.3 NOTCH FILTER.....	24
3.3.5 TL082 OPERATIONAL AMPLIFIER.....	26
3.3.6 INVERTING AND NON-INVERTING AMPLIFIER.....	29
3.3.7 VOLTAGE REGULATOR.....	31
3.3.7.1 IC7905 REGULATOR.....	31
3.3.8 TMS320C6713.....	32
3.3.9 PATTERN CORRELATION.....	33
3.3.10 HEARTRATE DETECTION.....	37
4. HARDWARE IMPLEMENTATION.....	40
4.1 CIRCUIT DIAGRAM.....	41
4.2 WORKING.....	42
4.3 COMPONENT LIST.....	43
5. SOFTWARE IMPLEMENTATION.....	44
5.1 OVERVIEW OF CODE COMPOSER 3.1.....	45
5.2 ALGORITHMS.....	46
6. RESULT ANALYSIS AND CONCLUSION.....	47
6.1 RESULT ANALYSIS.....	48
6.2 CONCLUSION.....	49
7. FUTURE SCOPE.....	50
APPENDIX.....	52
REFERENCES.....	81

## LIST OF FIGURES

Figure 1 ECG Processing conceptual block diagram.....	5
Figure 2 ECG waves.....	8
Figure 3 ECG wave components.....	9
Figure 4 Standard limb leads.....	10
Figure 5 Axis positions of standard leads.....	11
Figure 6 Functional block diagram.....	15
Figure 7 Non-inverting buffer.....	17
Figure 8 ECG sensors.....	18
Figure 9 ICT4051B Pin Configuration.....	19
Figure 10 8:1 multiplexer truth table.....	19
Figure 11 Low pass filter.....	21
Figure 12 Frequency response of low pass filter.....	22
Figure 13 A passive analog first order high pass filter realized by RC circuit.....	23
Figure 14 Frequency response of high pass filter.....	24
Figure 15 Notch filter.....	25
Figure 16 Frequency response of notch filter.....	25
Figure 17 OP-AMP connection diagram.....	26
Figure 18 Internal schematic of IC TL082B.....	27
Figure 19 OP-AMP IC TL082B specifications.....	29
Figure 20 Inverting Amplifier.....	29
Figure 21 Non-inverting Amplifier.....	30
Figure 22 7905 Fixed voltage regulator configuration.....	31
Figure 23 TMS320C6713 Internal block diagram.....	33
Figure 24 Correlation patterns.....	35
Figure 25 Graphical interpretation of the multi-channel correlation results.....	36
Figure 26 An ECG wave.....	38
Figure 27 ECG wave 10 sec rule.....	38
Figure 28 ECG signal conditioning circuit.....	41
Figure 29 ECG displayed using CCS.v3.1.....	48

## **ABSTRACT**

Medical practitioners analyze the electrical activity of the heart in order to detect or predict various disorders by interpreting the information provided by the Electrocardiogram signal of the patient. This project aims at processing and matching of an ECG signal with a database for the detection of abnormal heart conditions based on pattern comparison. The basis for the pattern matching algorithm without feature extraction is an ECG wave from database with ECG signals and patient diagnosis information. The recognition takes place by comparing the wave forms of 3 leads of the examined ECG with the same leads of different ECGs from the database. The similarity of the compared ECG beats is calculated with the cross correlation function. By means of the algorithm, we select ECGs of the database which are the most similar to the examined ECG considering all of the individual leads. Using the available database information, the diagnosis which corresponds to the most similar ECGs of the database is then identified as the diagnosis for the examined ECG. In our project we have used a Texas Instruments TMS320C6713 DSK to generate and process the electrocardiogram signal in digital domain.

# **1. INTRODUCTION**



## **1.1 PROBLEM DEFINITION**

The Electrocardiogram (ECG) is one of the most important signals used by medical practitioners to analyze the electrical events in the human cardiac cycle. The ECG signal finds wide applications in evaluation of rhythm disorders, in screening tests for coronary heart diseases, etc. The ECG signal can be generated using a single lead, 3 leads or 12 leads.

The ECG signal generated using standard leads contains noise in the form of power line noise, baseline wander, and muscle noise etc. This noise needs to be filtered out using suitable filters.

Also, modern developments in the field of medical science have laid the emphasis on real-time monitoring, processing and analysis of electrocardiogram signal. The Electrocardiogram signal needs to be compared with a database of ECG signals corresponding to different heart conditions. Such a technique can be used for the detection of certain conditions which might otherwise go unnoticed when observed by the naked eye.

Here, there arises a need for design of a system which captures Electrocardiogram signal from human body, filters it to remove noise components present in the signal, samples and digitize it in real-time and correlates it with an already stored database. For real-time processing of Electrocardiogram data fast processors are required. The DSP processors can be the best choice for design of such real-time ECG signal.

## **1.2 SUBJECT BACKGROUND**

The technique is based on the hypothesis that those ECGs whose signal waveforms are in best agreement in their leads can in all probability be assigned to the same diagnosis. As this assumption is at the basis of medical experience, it should be possible to obtain the diagnosis for an unknown ECG from a comparison between the signal waveforms of this ECG with unknown diagnosis and an ECG of known diagnosis stored in a database. The technique thus starts from the assumption that the waveforms of the quasi-periodical signal sections of the leads of the 12-channel ECG represent a "finger print" of the ECG in question.

### **1.2.1 TRANSDUCERS**

A transducer is a device that converts a physical quantity into an electrical quantity such as voltage or current. The electrical activity of heart can be converted into a measurable electrical signal by using a suitable transducer. The performance of a system depends upon the quality of ECG signal obtained from transducer.

### **1.2.2 SIGNAL CONDITIONING**

A signal conditioning circuit helps to enhance the strength of a signal obtained at the output of transducer to a range that can be directly applied to a measuring or processing system. The ECG signal obtained at transducer output is in the range of mV and its current capacity is low. Hence, the signal needs to be passed through a set of buffers and voltage amplifiers. The signal conditioning circuit should have very high noise immunity so that it does not add noise to the originally weak ECG signal.

### **1.2.3 FILTERS**

A filter is a device which removes the unwanted frequency signals from the input signal.

The ECG signal from transducers needs to be filtered using following filters:

- High-Pass Filter
- Low-Pass Filter
- Band-reject Filter

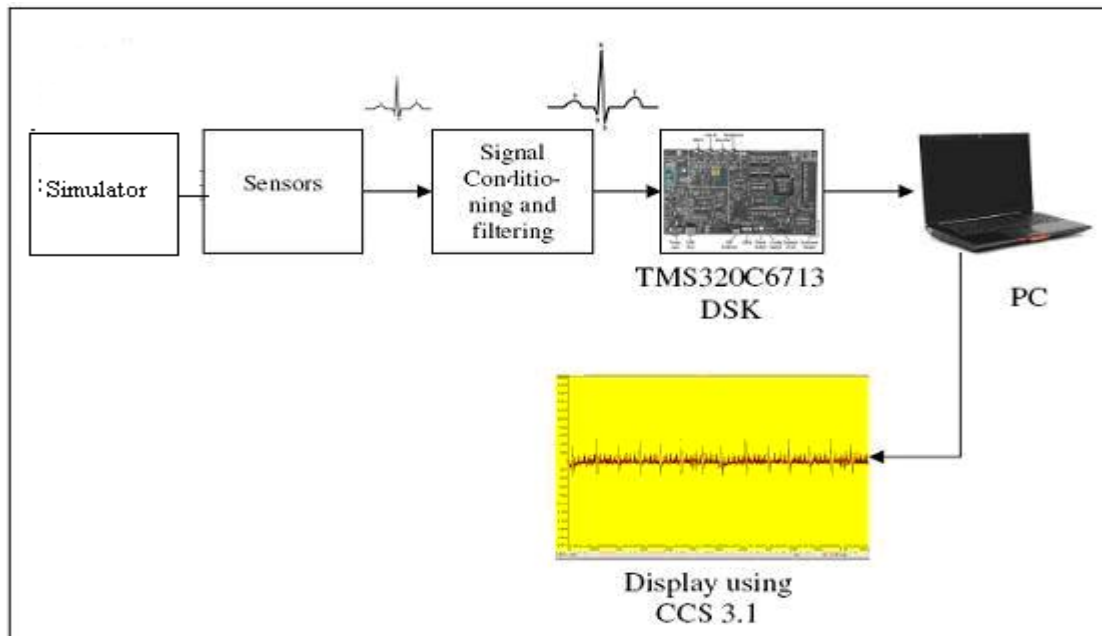
A high pass filter should be used to remove the DC component from the Electrocardiogram signal, noise due to baseline wandering, i.e., due to perspiration, respiration and body movements of patient and the noise from body electrodes. A low pass filter should be used to remove the high frequency components such as RF interference, Electromagnetic interference and muscle noise without affecting the high frequency components of desired ECG wave. A precise and accurate Band-reject (notch) filter should be used to remove power line noise of 50Hz frequency.

### **1.2.4 PROCESSING UNIT**

A processing unit can be a microcontroller, a microprocessor or any other hardware. For real time processing, it has been observed that ECG processing using Digital Signal Processor is most efficient. The TMS320C6713 is a fast special-purpose digital signal processor with a specialized type of architecture and instruction set appropriate for signal processing. The processing speed of DSP processor is large enough to digitize and process the incoming ECG signal in real – time.

### **1.3 CONCEPTUAL BLOCK DIAGRAM**

The following figure shows the conceptual block diagram of an ECG processing system



**Figure 1. ECG Processing Conceptual Block Diagram**

An ECG processing system consists of following:

- Input ECG signal
- Signal Conditioning Circuit
- TMS320C6713 DSK Board
- Personal Computer or a monitoring device
- Application Software (Code Composer Studio)

A real-time ECG signal is applied to the signal conditioning circuit. The signal level is enhanced by the conditioning circuit and is filtered by series of filters. The filtered signal is applied at the input of Digital Signal Processor where it is digitized and compressed. The ECG waveforms and the significant parameters of ECG are viewed on Personal Computer using the application software Code Composer Studio 3.1.

## **1.4 SCOPE OF THE PROJECT**

The term ECG processing can include digitization, noise removal, storage of the signal, correlation with database and displaying the results. Since the development of this project required a lot of literature survey, there was a limit on the implementation part of project.

It is not possible to incorporate all the required processing features since time and resources have been the major constraint. Hence taking into consideration above mentioned points, the scope of this project was limited to a smaller manageable size. The scope of this project includes:

- capturing an ECG signal from patients body, its amplification and conditioning
- digitization of captured signal
- filtering to remove noise components
- creation of a database
- correlation with the database
- displaying the obtained results

The filtering of ECG signal is done in analog domain using RC filter networks with proper cut-off frequencies whereas the compression/decompression and determination of heart rate is implemented in digital domain.

We have considered fair amount of modules for purpose of implementation of this project like study of Electrocardiogram waves, their frequency components, sources of noise along with the study of TMS320C6713 DSK and associated application software Code Composer Studio 3.1. The project has been designed taking into considerations the basic requirements of end user.

## **2. THE ELECTROCARDIOGRAM**

## 2.1 INTRODUCTION TO ECG

An electrocardiogram (ECG) is a time-varying signal that represents the activity of the heart. Each event has a distinctive waveform, the study of which can lead to greater insight into a patient's cardiac pathophysiology. An ECG can also be defined as a time-varying signal reflecting the ionic current flow which causes the cardiac fibers to contract and subsequently relax.

An Electrocardiogram signal can be used for detection of coronary artery disease, cardiomyopathies and left ventricular hypertrophy. It can also provide information for evaluation rhythm disorders.

The ECG can be obtained by recording the potential difference between various electrodes placed on the surface of the skin, at specific locations. A single normal cycle of the ECG occurs with every heart beat. The various components of ECG signal are as shown in the below:

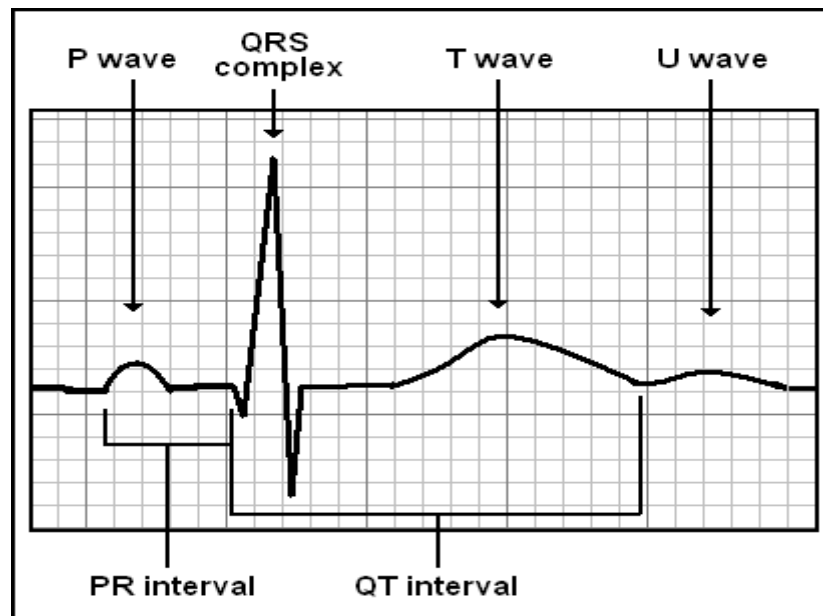


Figure 2. ECG waves

## 2.2 ECG WAVES

The ECG signal comprises of various waves and these waves are explained below:

P wave: The sequential depolarization of the right and left atria

QRS complexes: Right and left ventricular depolarization

T wave: Ventricular repolarization

U wave: After depolarization in the ventricles

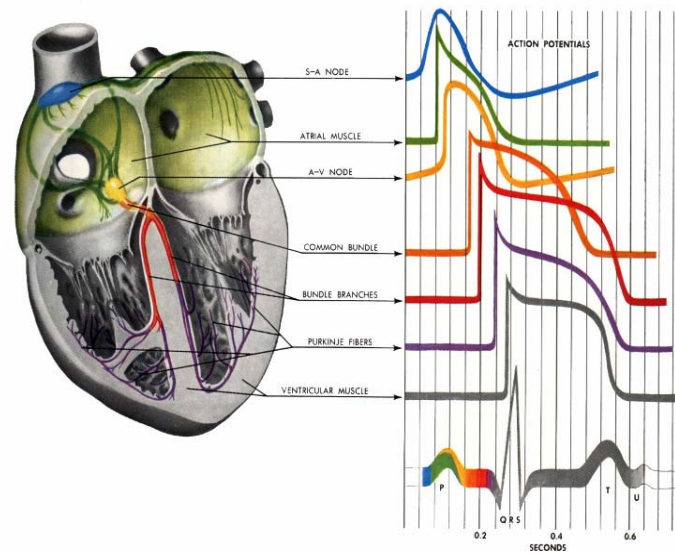
The following figure shows how the ECG signal is formed due to the combination of such waves.

The typical values of ECG are as given below:

P wave	0.25mV
R wave	1.60mV
Q wave	0.4mV
T wave	0.1mV- 0.05mV

Duration of Interval:

P-R Interval	0.12 - 0.20sec
Q-T Interval	0.35 - 0.44sec
S-T Segment	0.05 - 0.15sec
QRS Interval	0.09 sec
P Interval	0.11 sec



**Figure 3. ECG wave components**



## 2.3 ECG LEADS

ECG leads are electrodes which measure potential difference between two points on the body or between one point on the body and a virtual reference point with zero electrical potential located at the centre of heart. The leads which measure the potential difference between two points on the body are called as bipolar leads and the leads which measure potential on one point on the body w.r.t virtual reference point are called as unipolar leads. A standard ECG measuring apparatus consists of 12 leads:

- Standard Limb Leads
- 3 Augmented Limb Leads
- 6 Precordial Leads

### 2.3.1 STANDARD LIMB LEADS:

Bipolar recordings utilize standard limb lead configurations. Lead I has positive electrode on the left arm, and the negative electrode on the right arm. In this and the other two limb leads, an electrode on the right leg serves as a reference electrode for recording purposes. In the lead II configuration, the positive electrode is on the left leg and the negative electrode is on the right arm. Lead III has the positive electrode on the left leg and the negative electrode on the left arm. These three bipolar limb leads roughly form an equilateral triangle that is called Einthoven's triangle as shown in the figure.

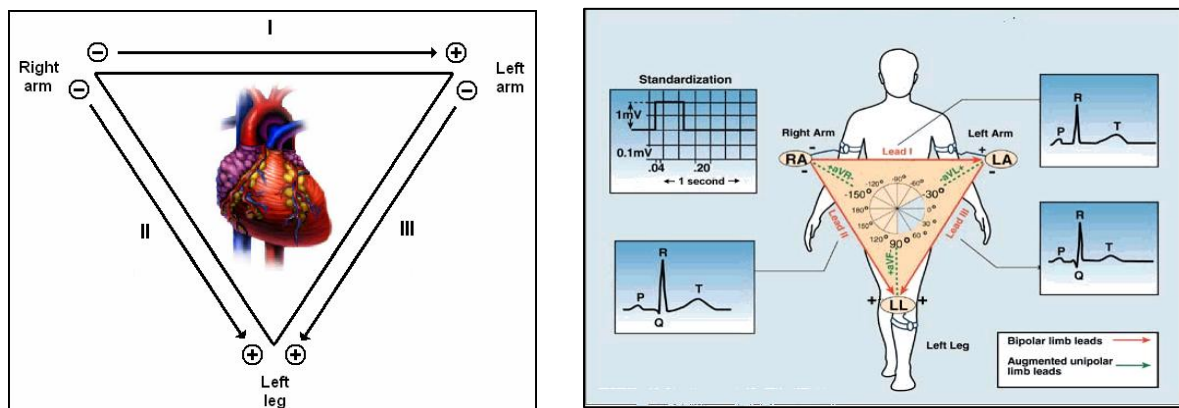


Figure 4. Standard Limb Leads

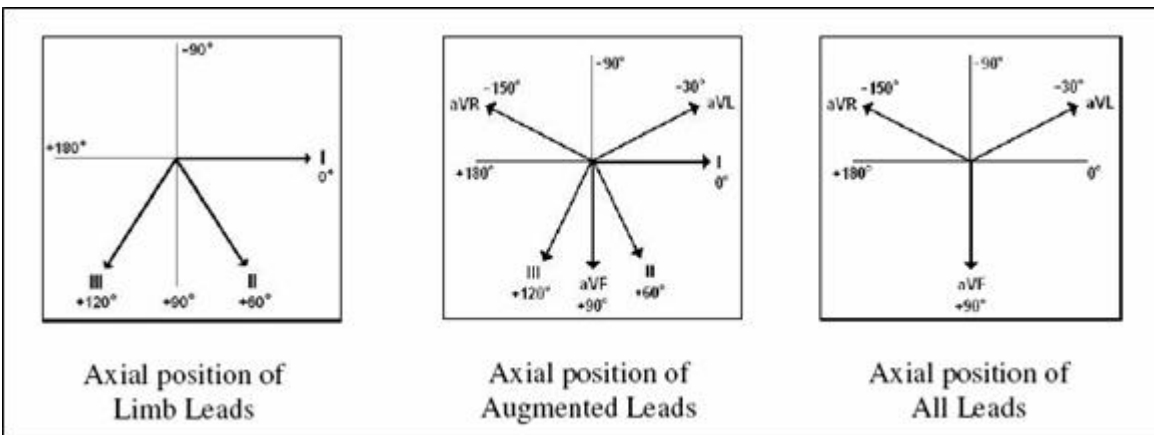
### 2.3.2 3-AUGMENTED LIMB LEADS:

There are three augmented unipolar limb leads. These are termed unipolar leads because there is a single positive electrode that is referenced against a combination of the other limb electrodes. The positive electrodes for these augmented leads are located on the left arm (aV<sub>L</sub>), the right arm (aV<sub>R</sub>), and the left leg (aV<sub>F</sub>). The aV<sub>L</sub> lead is at -30° relative to the lead I axis; aV<sub>R</sub> is at -150° and aV<sub>F</sub> is at +90°.

### 2.3.3 6-PRECORDIAL LEADS:

There are six Precordial, unipolar chest leads. This configuration places six positive electrodes on the surface of the chest over different regions of the heart in order to record electrical activity in a plane perpendicular to the frontal plane.

The axis of a particular lead represents the viewpoint from which it looks at the heart.



**Figure 5. Axis Positions of Standard Leads**

## **2.4 USES OF ECG**

The various applications of ECG are as follows:

1. In Cardiac stress test, wherein the patient walks on a treadmill while connected to an electrocardiogram machine
2. In Screening test, which identifies coronary artery disease prior to the onset of symptoms
3. Preoperatively to rule out coronary artery disease
4. Provide information in the presence of metabolic alterations such as hypercalcemia, hypocalcemia, hyperkalemia and hypokalemia
5. With known heart disease, monitor progression of the disease
6. It is used in evaluation of rhythmic disorders by observing electrocardiogram
7. It is the basic cardiologic test and is widely used to diagnose patients with suspected or known heart disease

### **3. LITERATURE SURVEY**

### **3.1 GENERAL DESCRIPTION**

The Literature survey included the study of various components of an electrocardiogram signal, their typical voltage values and time intervals. It was also necessary to study the various components required for generation and processing of electrocardiogram signal. The different algorithms for correlation and heart rate detection were studied and evaluated.

Also the technicalities involved in use of Digital signal Processor, its interfacing with PC and ECG generation circuit were studied.

Thus, the literature survey included:

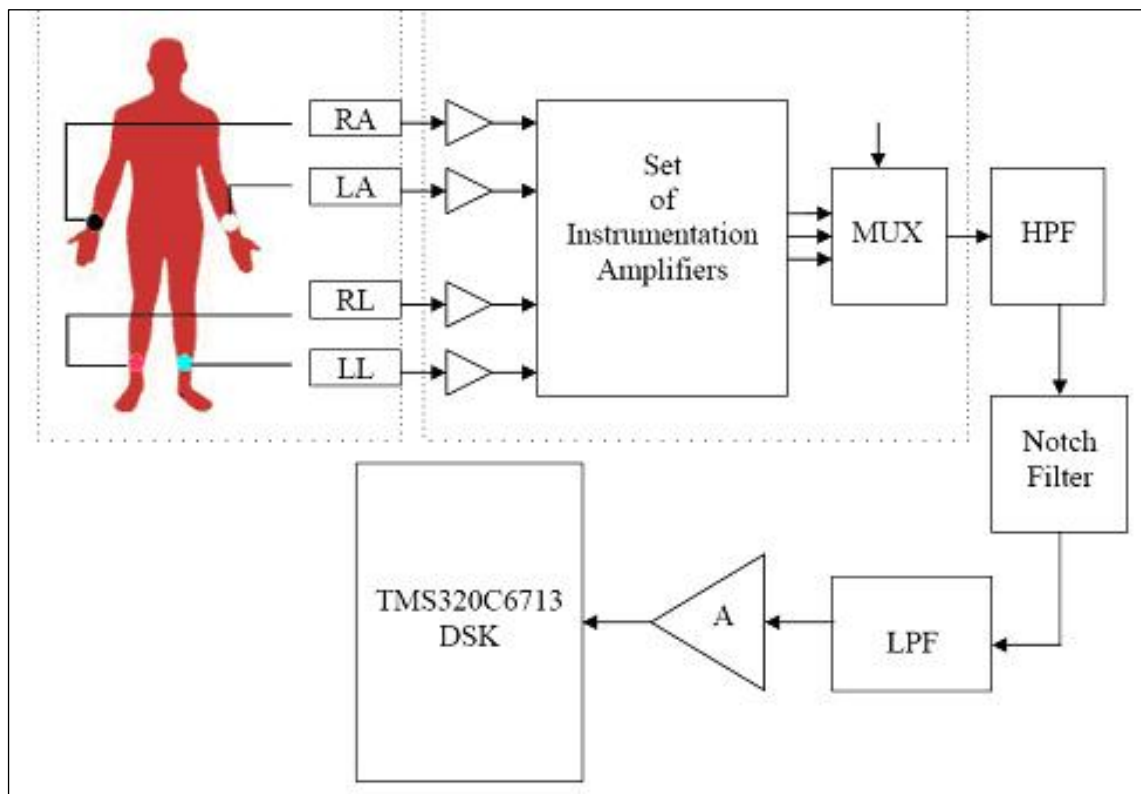
- Study of ECG waves and its parameters
- Detailed study of components such as OP-AMP TL082, Analog MUX, LM7805/7905
- Detail study of the TMS320C6713 DSK
- Since we have developed our source codes using application software called Code Composer Studio 3.1, study of this software was our basic requirement. This included complete know of Code Composer Studio 3.1 with all its features.
- The processing part involves comparison of input ECG signal with the database and heart rate detection of the input signal. Hence study of various correlation algorithms and heart rate detection algorithms was included in our survey.

### 3.2 FUNCTIONAL BLOCK DIAGRAM

The following figure shows the functional block diagram of the ECG processing unit.

The functional block diagram consists of

- Sensing network
- Signal conditioning circuit
- Filter networks
- Digital Signal Processor





**Figure 6. Functional Block Diagram**

The figure shows a Functional block diagram of Electrocardiogram (ECG) generation and processing. Here, the electrical activity of the heart is sensed and converted into equivalent electrical quantity by transducers. The electrical output of the transducers is applied to the input of signal conditioning circuit. Initially the signal is buffered to increase current driving capacity and provide isolation between conditioning circuit and

the sensors. The buffered signal is then applied to a set of instrumentation amplifiers with a gain of 40, where the difference signal is generated at the output called *leads*. We can have three such leads, out which the second lead gives the most stronger ECG signal. This three leads are then applied to the input of an analog multiplexer, out of which any one is selected at the output of multiplexer depending upon the select lines input. The selected lead (ECG) then passes through a set of filters to remove all the noise components present in the signal. The signal is then amplified by an inverting amplifier which provides a gain of 25. The output of the signal conditioning circuit is then applied to DSP processor through Line In input. This signal is then digitized by a **CODEC** (i.e., a **Coder-Decoder** which consists of ADC and DAC) at a sampling frequency of 8 kHz to 96 kHz. The TMS320CDSK runs a program which is written to calculate the heart rate, compress the Electrocardiogram signal, display it and store in a particular file.

The various components used are:

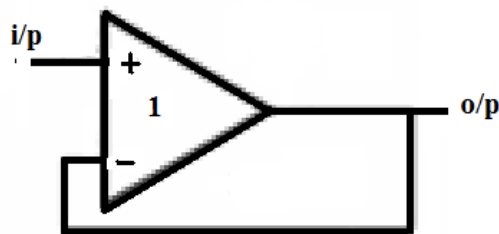
-  SENSORS
-  BUFFERS
-  INSTRUMENTATION AMPLIFIERS
-  MULTIPLEXER
-  FILTERS
-  AMPLIFIER
-  VOLTAGE REGULATORS
-  TMS320C6713 DSK

### **3.3 COMPONENT DESCRIPTION**

This topic provides the description about the various components used in the system with their respective specifications such as input voltage range, operating temperature range etc. Also the internal block diagram and pin configurations of various components are provided here.

#### **3.3.1 BUFFER**

A Buffer is a device which acts a unity gain amplifier, i.e., it's voltage gain is 1. A buffer can be used in two modes: inverting and non-inverting. For a non-inverting buffer using OP-AMP, the value of feedback resistor is  $R_F = 0$  as shown below:



**Figure 7. Non-inverting buffer**

Buffers are used to provide isolation between two stages. As it is an operational amplifier its input impedance is very high and output impedance is very low. Also they increase the current driving capacity of the circuit.

#### **3.3.2 SENSORS**

A sensor is a device which measures a physical quantity and converts it into a signal which can be read by an instrument. The sensitivity of a sensor indicates how much the sensor's output changes when the measured quantity changes. Sensors measure the electric current that moves through the body with the help of electrodes attached to the skin.



The ECG tabs used are as shown below:



**Figure 8. ECG sensors**

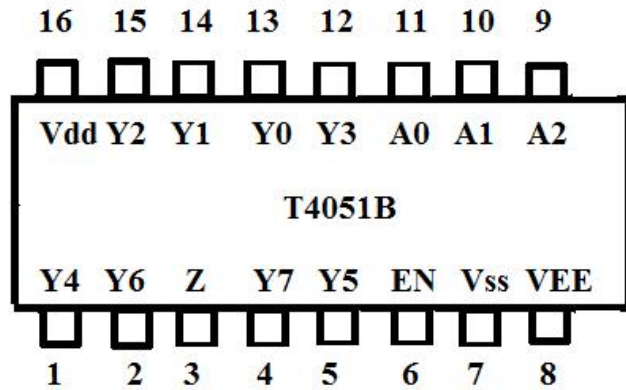
While using sensors it must be ensured that:

1. The ground lead or reference lead (black) on the body is connected before connecting other leads.
2. ECG sensor should not be placed in water or other liquids.
3. The main cause of problem with the sensor is a poor connection, either between the tabs and the skin or between the clips and the tabs. Hence it must be ensured that the patient is seated or lying down and is relaxed.

### **3.3.3 MULTIPLEXER**

A multiplexer is a device which has many inputs and one output as shown in the figure. Depending upon the select input one of the input lines is connected to the output line. The figure shows an 8:1 multiplexer which has eight input lines (Y0 – Y7) and one output line (Z). Also there are three select lines used (A0 A1 & A2) for selecting any one out of the

eight inputs. There is one control input to the IC called Enable input (E\_Bar) which is an active low input. When E\_Bar=0, the multiplexer works normally and when E\_Bar=1 the Multiplexer IC is disabled and the output is tri-stated independent of select inputs.



**Figure 9. IC T4051B Pin Configuration**

VDD and VSS are the supply voltage Connections for the digital control Inputs (A0 to A2, and E). The VDD to VSS range is 3 to 15 V. The analog inputs/outputs (Y0 to Y7 and Z) can swing between VDD as a positive limit and VEE as a negative limit. VDD and VEE may not exceed 15 V. For operation as a digital Multiplexer / demultiplexer, VEE is Connected to VSS (typically ground).

The truth table of the 8:1 multiplexer is given below Here, H indicates High state, L indicates low state, Y0 to Y7 are independent inputs and E is Enable input.

INPUTS				Channel ON
E	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	
L	L	L	L	Y <sub>0-Z</sub>
L	L	L	H	Y <sub>1-Z</sub>
L	L	H	L	Y <sub>2-Z</sub>
L	L	H	H	Y <sub>3-Z</sub>
L	H	L	L	Y <sub>4-Z</sub>
L	H	L	H	Y <sub>5-Z</sub>
L	H	H	L	Y <sub>6-Z</sub>
L	H	H	H	Y <sub>7-Z</sub>
H	X	X	X	None

**Figure 10. 8:1 Multiplexer Truth Table**

### **3.3.4 FILTERS**

A filter is a device used to reject the unwanted signal from the input. In a measurement system, it is seldom that the transducer used measures the measurand precisely and presents the information in a standard form eliminating the need for further filtering and analysis. Excepting for simple measurements, the measurement engineer soon finds that for high accuracy the main factors that must be considered are signal to noise ratio, response time and the bandwidth over which the measurements are desired. Among these the signal to noise ratio is perhaps the most important parameter that needs to be considered and the use of signal filters becomes a necessity when low measurements or high resolution measurements are attempted. The availability of operational amplifiers in the integrated form has changed the situation significantly, leading to emergence of “active filters”. Today, a majority of low frequency filters are necessarily of these types, particularly for frequencies below about 100 kHz. The special advantage of the active circuitry for use in low-frequency filters is the interesting fact that inductors can be totally avoided. In addition, active capacitance multiplication enables use of capacitors of low practical values to be used even for cut-off frequencies down to fraction of one hertz. However, due to the limited gain bandwidth products of ICs and their effect on the filter characteristics, and due to the advantages of the inductor in the high frequency range, passive filters are preferred for frequencies above few kilohertz.

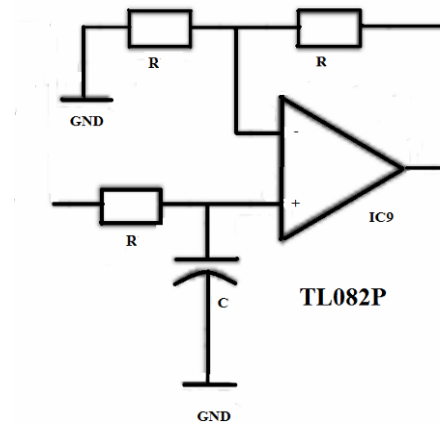
The general classification of filters are, Low pass filter, High pass filter, band-pass filter, and band-rejection filters types lend themselves to several subclasses, depending on the order of the filter.

### 3.3.4.1 Low Pass Filter

A low-pass filter is a filter that passes low-frequency signals but attenuates signals with frequencies higher than the cutoff frequency. It is also called as high-cut filter, or treble cut filter when used in audio applications. An ideal low-pass filter completely eliminates all frequencies above the cut-off frequency while passing those below unchanged. The transition region present in practical filters does not exist in an ideal filter. An ideal low-pass filter can be realized mathematically by multiplying a signal by the rectangular function in the frequency domain or, equivalently, convolution with a sinc function in the time domain.

Real filters for real-time applications approximate the ideal filter by truncating and windowing the infinite impulse response to make a finite impulse response; applying that filter requires delaying the signal for a moderate period of time. This delay is manifested as phase shift. Greater accuracy in approximation requires a longer delay.

The following figure shows a low-pass electronic filter realized by an RC circuit. An OP-AMP IC TL082P is used for construction of filter.



**Figure 11. Low Pass Filter**

The cut-off frequency (in Hertz) is determined by the time constant:

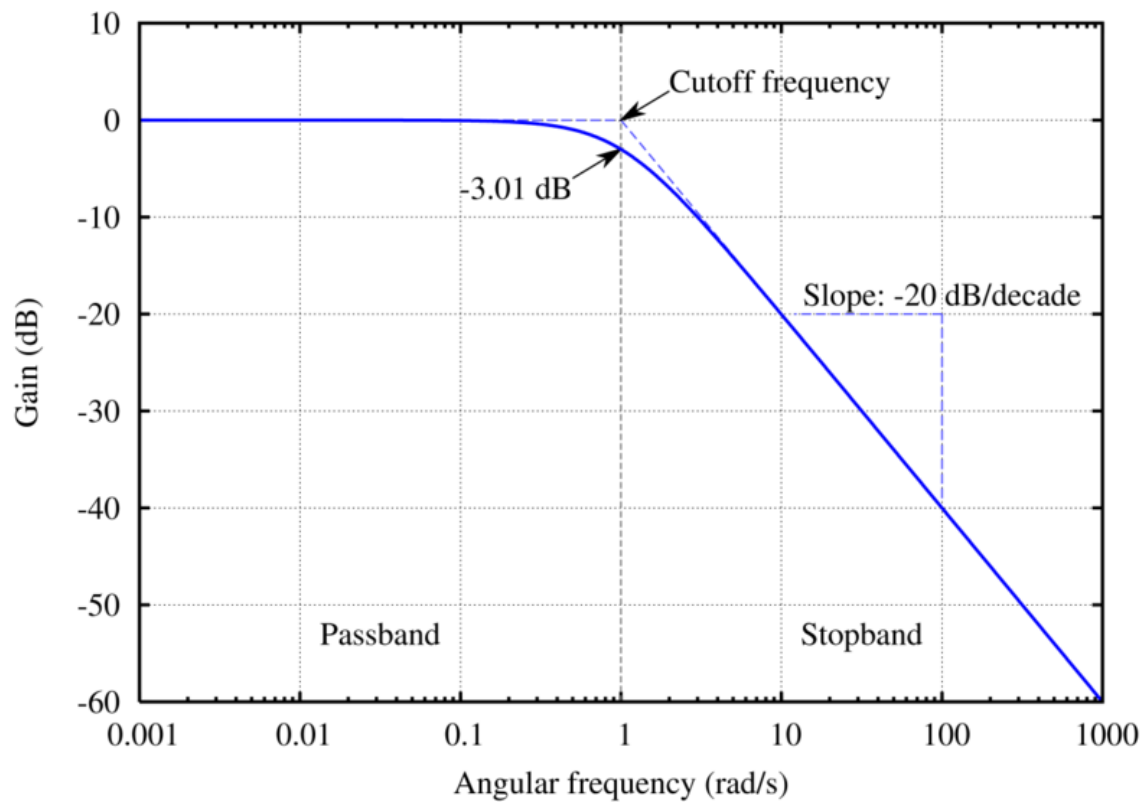
$$f_c = \frac{1}{2\pi\tau} = \frac{1}{2\pi RC}$$

Where:  $f_c$  = cut-off frequency of low pass filter

R = value of resistor used

C = value of capacitor used

The frequency response of a first order filter is as shown below:



**Figure 12** Frequency response of low pass filter

### 3.3.4.2 High Pass Filter

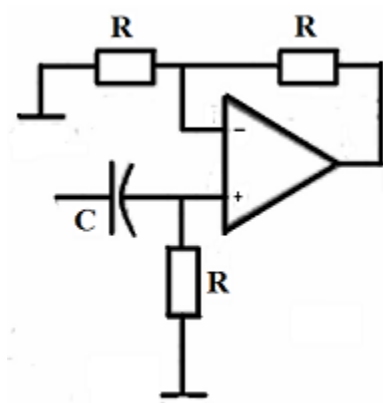
A high-pass filter is a filter that passes high frequencies well, but attenuates frequencies lower than the cutoff frequency. The actual amount of attenuation for each frequency varies from filter to filter. It is also called a low-cut filter; the terms bass-cut filter or rumble filter are also used in audio applications. The simplest electronic high-pass filter

consists of a capacitor in series with the signal path in conjunction with a resistor in parallel with the signal path. The resistance times the capacitance ( $R \times C$ ) is the time constant ( $\tau$ ); it is inversely proportional to the cutoff frequency, at which the output power is half the input ( $-3$  dB):

$$f_c = \frac{1}{2\pi\tau} = \frac{1}{2\pi RC}$$

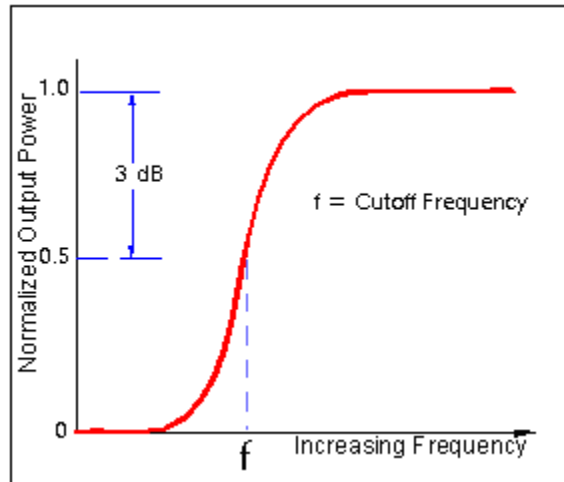
Where  $f$  is in hertz,  $\tau$  is in seconds,  $R$  is in ohms, and  $C$  is in farads.

The following figure shows the circuit diagram for high pass filter implemented using OP-AMP IC TL082P



**Figure 13. A Passive analog first order high pass filter realized by RC circuit**

The frequency response of a high pass filter is as shown below:

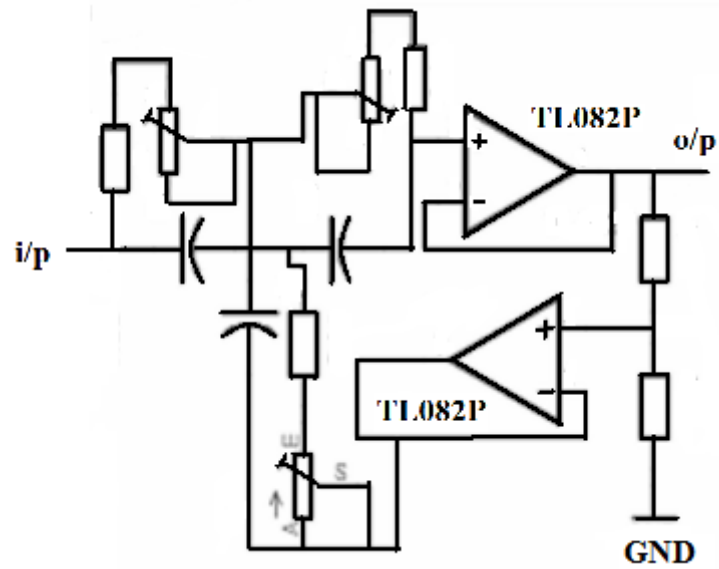


**Figure 14. frequency response of high pass filter**

### **3.3.4.3 Notch Filter**

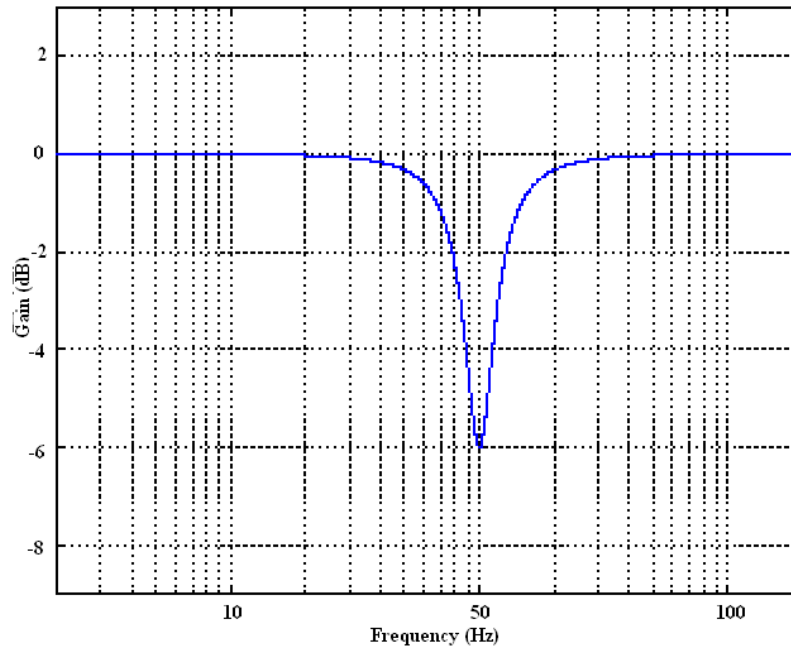
In signal processing, a band-stop filter or band-rejection filter is a filter that passes most frequencies unaltered, but attenuates those in a specific range to very low levels. It is the opposite of a band-pass filter. A notch filter is a band-stop filter with a narrow stop band and a high Q factor. Notch filters are used in instrument amplifier to reduce or prevent feedback, while having little noticeable effect on the rest of the frequency spectrum. Other names include 'band limit filter', 'T-notch filter', 'band-elimination filter', and 'band-reject filter'

The circuit diagram for Notch filter is as shown below:



**Figure 15. Notch Filter**

The frequency response of a notch filter is as shown below:

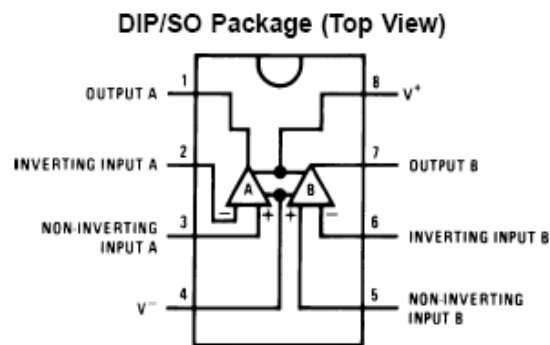


**Figure 16. Frequency response of a notch filter**



### 3.3.5 TL082 OPERATIONAL AMPLIFIER

The Operational amplifiers are low cost, high speed, dual JFET input operational amplifiers with an internally trimmed input offset voltage (BI-FET technology). They require low supply current yet maintain a large gain bandwidth product and fast slew rate. In addition, well matched high voltage JFET input devices provide very low input bias and offset currents. The TL082 is pin compatible with the standard LM1558 allowing designers to immediately upgrade the overall performance of existing LM1558 and most LM358 designs. These amplifiers may be used in applications such as high speed integrators, fast D/A converters, sample and hold circuits and many other circuits requiring low input offset voltage, low input bias current, high input impedance, high slew rate and wide bandwidth. The devices also exhibit low noise and offset voltage drift. The pin configuration of the op-amp is as shown in the figure above. The TL082 IC is an 8-Pin DIP Package IC. Each IC contains two op-amps within it as shown.

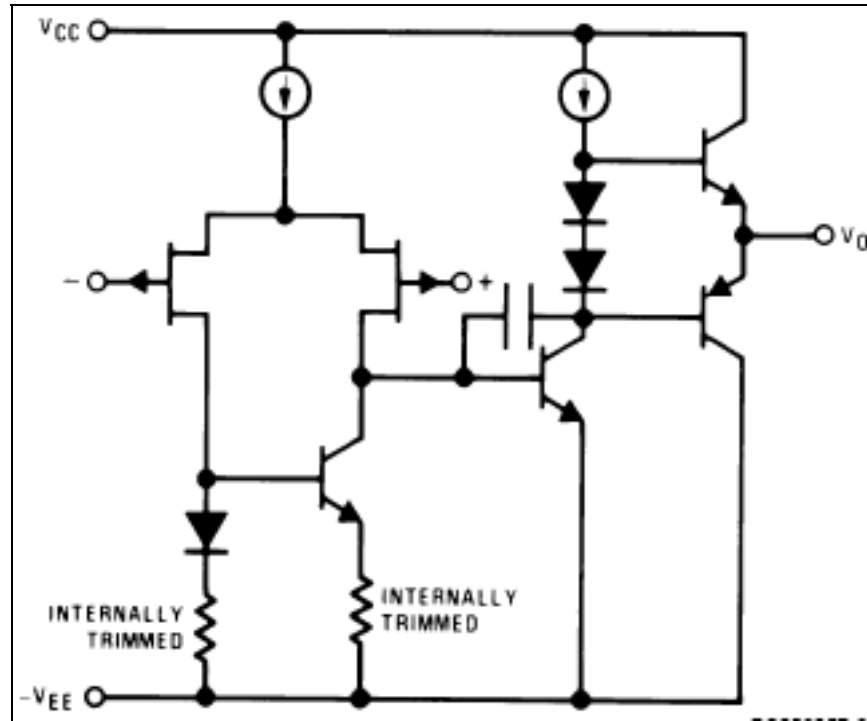


**Figure 17. OP-AMP connection diagram**

Precautions should be taken to ensure that the power supply for the integrated circuit never becomes reversed in polarity or that the unit is not inadvertently installed backwards in a socket as an unlimited current surge through the resulting forward diode within the IC could cause fusing of the internal conductors and result in a destroyed unit. Because these amplifiers are JFET rather than MOSFET input op-amps they do not require special handling. As with most amplifiers, care should be taken with lead dress, component placement and supply decoupling in order to ensure stability. For example, resistors from the output to an input should be placed with the body close to the input to

minimize “pick-up” and maximize the frequency of the feedback pole by minimizing the capacitance from the input to ground.

The internal schematic of the TL082 IC is shown below:



**Figure 18. Internal Schematic of IC TL082B**

These devices are op amps with an internally trimmed input offset voltage and JFET input devices (BI-FET II). These JFET's have large reverse breakdown voltages from gate to source and drain eliminating the need for clamps across the inputs. Therefore, large differential input voltages can easily be accommodated without a large increase in input current. The maximum differential input voltage is independent of the supply voltages. However, neither of the input voltages should be allowed to exceed the negative supply as this will cause large currents to flow which can result in a destroyed unit. Exceeding the negative common-mode limit on either input will cause a reversal of the phase to the output and force the amplifier output to the corresponding high or low state. Exceeding the negative common-mode limit on both inputs will force the amplifier output to a high state. In neither case does a latch occur since the rising input back within

the common-mode range again puts the input stage and thus the amplifier in a normal operating mode. Each amplifier is individually biased by a zener reference which allows normal circuit operation on  $\pm 6V$  power supplies. Supply voltages less than these may result in lower gain bandwidth and slew rate.

The features of OP-AMP TL082B are listed below:

- ✦ Internally trimmed offset voltage: 15mV
- ✦ Low input bias current: 50 pA
- ✦ Wide gain bandwidth: 4MHZ
- ✦ High slew rate: 13 V/  $\mu$ s
- ✦ Low supply current: 3.6mA
- ✦ High input impedance:  $10^{12} \Omega$
- ✦ Low total harmonic distortion  $A_v = 10$ : <0.02%
- ✦  $R_L = 10k$ ,  $V_o = 20$  Vp-p
- ✦  $BW = 20HZ - 20KHZ$
- ✦ Low 1/f noise corner: 50 HZ
- ✦ Fast setting time to 0.01%: 2  $\mu$ s

The Specifications for OP-AMP IC TL082P are as given below:

Symbol	Parameter	Conditions	TL082			Units
			Min	Typ	Max	
$V_{OS}$	Input offset voltage	$R_S = 10k$ , $T = 25\text{ C}$		5	15	mV
$I_{os}$	Input offset current	$T_J = 25\text{ C}$ $T_J \leq 25\text{ C}$		25	200 4	pA nA
$I_B$	Input Bias Current	$T_J = 25\text{ C}$ $T_J \leq 70\text{ C}$		50	400 8	pA nA
$R_{in}$	Input resistance	$T_J = 25\text{ C}$		$10^{12}$		$\Omega$
$A_{VOL}$	Large signal voltage gain	$V_S = \pm 15\text{ V}$ , $T_A = 25\text{ C}$ $V_O = \pm 10\text{ V}$ , $R_L = 2k\Omega$	25 15	100		V/Mv
$V_o$	Output voltage	$V_S = \pm 15\text{ V}$ , $R_L = 10k\Omega$	$\pm 12$	$\pm 13.5$		V
CMRR	Common Mode Rejection Ratio	$R_S \leq 10k\Omega$	70	100		dB
$V_{CM}$	Input common mode voltage range	$V_S = \pm 15\text{ V}$	$\pm 11$	+15 -12		V V
PSRR	Supply voltage rejection ratio	$V_S = \pm 6\text{ to }15\text{ V}$	70	100		dB
$I_S$	Supply current			3.6	5.6	mA
SR	Slew Rate	$V_S = \pm 15\text{ V}$ , $T_A = 25\text{ C}$	8	13		V/ $\mu$ s
GBW	Gain Bandwidth Product	$V_S = \pm 15\text{ V}$ , $T_A = 25\text{ C}$		4		MHz
$e_n$	Equivalent input noise voltage	$T_A = 25\text{ C}$ , $R_S = 100\Omega$ $f = 1\text{ kHz}$	25			nV/ $\sqrt{\text{Hz}}$
$i_n$	Equivalent input noise current	$T_J = 25\text{ C}$ , $f = 1\text{ kHz}$	0.01			pA/ $\sqrt{\text{Hz}}$

Figure 19. OP-AMP IC TL082P specifications

### 3.3.6 Inverting and Non-Inverting Amplifier

An Inverting amplifier using op-amp is used to invert and amplify the signal applied at its input. The circuit using op-amp in inverting mode configuration is as shown below:

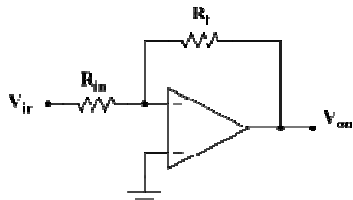


Figure 20. Inverting Amplifier

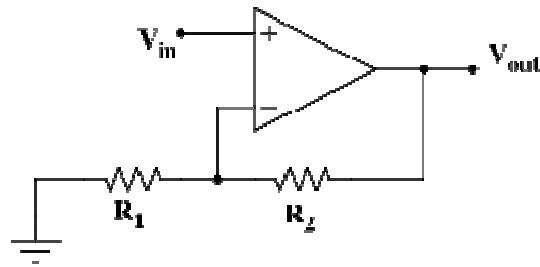
The output voltage is given by the equation:

$$V_{\text{out}} = -V_{\text{in}}(R_f/R_1)$$

A third resistor, of value,  $R_f \parallel R_{\text{in}} = R_f R_{\text{in}} / (R_f + R_{\text{in}})$  added between the non-inverting input and ground, minimizes errors due to input bias currents.

A Non-Inverting amplifier using op-amp is used to amplify the signal applied at its input.

The circuit using op-amp in non-inverting mode configuration is as shown below:



**Figure 21. Non-inverting amplifier**

The output voltage is given by the equation:

$$V_{\text{out}} = V_{\text{in}} \left( 1 + \frac{R_2}{R_1} \right)$$

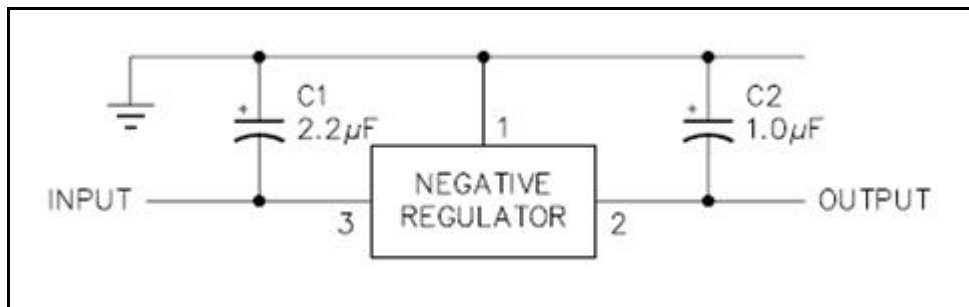
A third resistor, of value  $R_1 \parallel R_2$ , added between the  $V_{\text{in}}$  source and the non-inverting input, while not necessary, minimizes errors due to input bias currents.

### 3.3.7 VOLTAGE REGULATORS

A regulator IC 7805 is used as a voltage positive voltage regulator and IC 7905 is used as a negative voltage regulator.

#### 3.3.7.1 IC 7905 Regulator

The 79xx (also sometimes known as LM79xx) series of devices is a family of self-contained fixed linear voltage regulator integrated circuits. The 79xx family is a very popular choice for many electronic circuits which require a regulated power supply, due to their ease of use and relative cheapness. When specifying individual ICs within this family, the xx is replaced with a two-digit number, which indicates the output voltage the particular device is designed to provide (for example, the 7905 has a 5 volt output, while the 7912 produces 12 volts). The 79xx lines are negative voltage regulators, meaning that they are designed to produce a voltage that is positive relative to a common ground. There is a related line of 78xx devices which are complementary positive voltage regulators. 79xx and 78xx ICs can be used in combination to provide both negative and positive supply voltages in the same circuit, if necessary. 79xx ICs have three terminals and are most commonly found in the TO220 form factor, although smaller surface-mount and larger TO3 packages are also available from some manufacturers. These devices typically support an input voltage which can be anywhere from a couple of volts over the intended output voltage, up to a maximum of -35 or -40 volts, and can typically provide up to around 1 or 1.5 amps of current (though smaller or larger packages may have a lower or higher current rating). The figure below shows 7905 IC in a fixed negative regulator configuration.



**Figure 22. 7905 fixed voltage configuration**

As shown in the figure above IC 7905 regulator uses two decoupling capacitors C1 and C2. The capacitor C1 is required only if regulator is separated from rectifier filter. The values of two capacitors are selected appropriately; also both capacitors C1 and C2 should be low E.S.R. types such as solid Tantalum. If aluminum electrolytic is used, at least 10 times Values shown should be selected.

The features of IC 7905 are as follows:

- Excellent Line and Load regulation
- Fold back current limiting
- Thermal overload protection
- Voltages available: -5V, -12V, -15V
- Available in surface mount package

The parameters specifications for IC 7905 are

### **3.3.8 TMS320C6713**

Texas Instruments TMS320C6713 DSK kit includes DSK board with TMS320C6713 DSP chip, USB cable, Power supply, CD with Code composer studio IDE (v3.1) and electronic documentation, DSK technical reference manual, DSK quick start installation guide, Matlab/Simulink trial CD and other promotional material. The 225 MHz TMS320C6713 floating point DSP kit has AIC23 stereo codec (ADC and DAC) and is ideal for audio as well as real-time applications. It samples at the rate of 8 kHz - 96 kHz. It has a 16 MB dynamic RAM, 512kB non-volatile FLASH memory, General purpose I/O, 4 LEDs, 4 DIP switches, USB interface to PC, Enhanced Harvard Architecture, Rich Addressing modes, Two general purpose Register files (A0-A15 & B0-B15), 32/64- Bit Data Word, Rich Instruction set, Eight 32-Bit Instructions/Cycle, 32/64-Bit Data Word, 4.4- 6.7 ns Instruction Cycle Time, Rich Peripheral Set, Optimized for Audio and Highly Optimized C/C++ Compiler. The block diagram of the processor is as shown below:

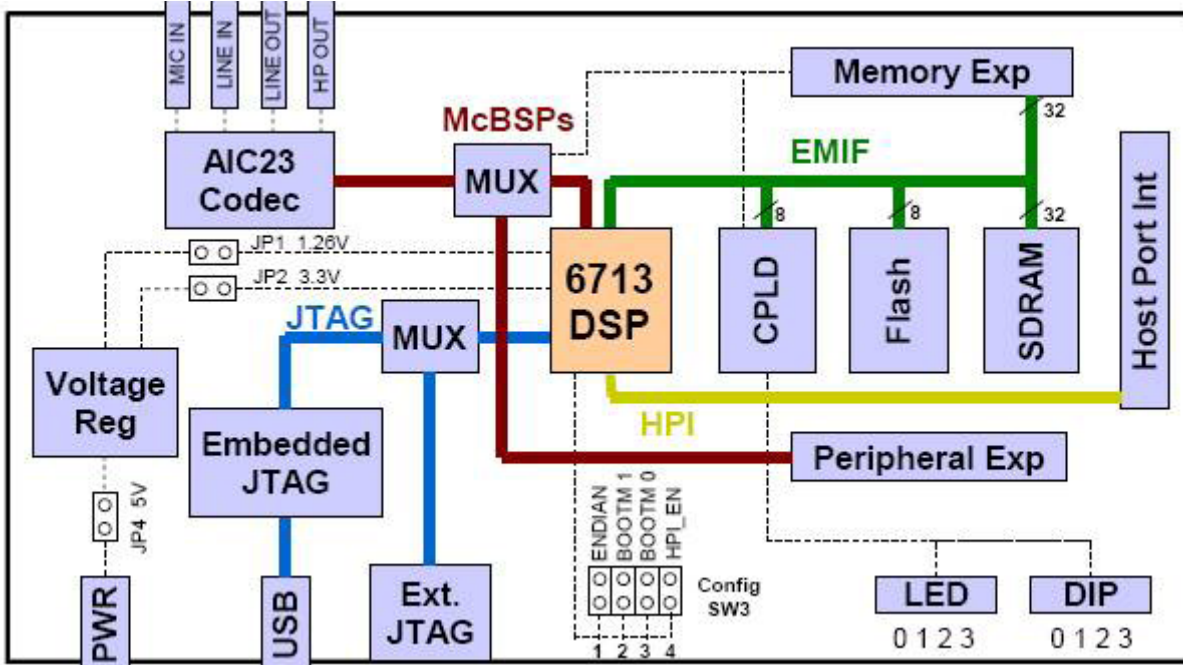


Figure 23. TMS320C6713 DSK internal block diagram

The DSP on the 6713 DSK interfaces to on-board peripherals through a 32-bit wide EMIF (External Memory Interface). The SDRAM, Flash and CPLD are all connected to the bus.

EMIF signals are also used for daughter cards. The DSP interfaces to analog audio signals through an on-board AIC23 codec and four 3.5 mm audio jacks (microphone input, line input, line output, and headphone output). The codec can select the microphone or the line input as the active input. The analog output is sent to both the line out and headphone out connectors. The line out has a fixed gain, while the headphone out allows for an adjustable gain connectors. A programmable logic device called a CPLD (Complex Programmable Logic Device) is used to implement logic that ties the board components together. The DSK includes 4 LEDs and a 4 position DIP switch which allow for interactive feedback.

### 3.3.9 PATTERN CORRELATION

The technique is based on the hypothesis that those ECGs whose signal waveforms are in best agreement in their leads can in all probability be assigned to the same diagnosis. The



assessment of the similarity of the ECG signal patterns of equal ECG leads is carried out by calculation of correlation functions whose function values are formed from the correlation coefficients  $K$ :

$$K = \frac{\sum_{n=1}^N x_n y_n - \frac{1}{N} \sum_{n=1}^N x_n \sum_{n=1}^N y_n}{\sqrt{\left[ \sum_{n=1}^N x_n^2 - \frac{1}{N} \left( \sum_{n=1}^N x_n \right)^2 \right] \left[ \sum_{n=1}^N y_n^2 - \frac{1}{N} \left( \sum_{n=1}^N y_n \right)^2 \right]}}$$

In dependence on the temporal shift of the signal window with the time series  $x_n$ , (unknown ECG) and  $y_n$ , (reference ECG) - with  $1 \leq n \leq N$ .  $N$  = number of function values – the correlation functions form maxima according to the periodicity of the ECG if the patterns are similar. Figs. 1 and 2 show two examples of correlation functions in the case of similarity of the ECG beats as well as in the case of strong differences of the waveforms from one another. According to the periodicity of the ECG signals, the correlation functions are also periodical. The different maxima are due to the variance between the ECG beats. The examples show that it is permissible for ECG without dysrhythmias or strong anomalies between the beats to limit the correlation analysis to only one beat typical of the ECG in question.

In pattern correlation, the unknown ECG is compared with the ECGs of the database in all 3 leads. The result of each reference ECG comparison is a 3-dimensional vector whose elements give the correlation values for the respective lead. The range of values corresponds to the definition range of the correlation coefficient  $K$  with  $-1 < K < +1$ . For the further considerations, only the positive correlations are taken into account. The correlation measure 100% ( $K=1$ ) thus means identity of the patterns. A correlation statement towards  $K=0$  indicates completely different signal patterns in the waveforms of the lead considered.

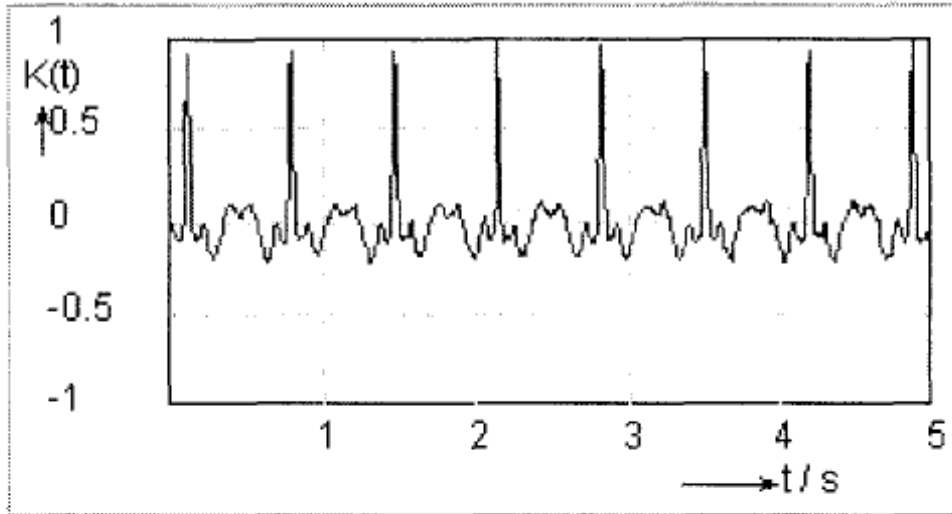


Fig. 1 Correlation function for very similar signal patterns

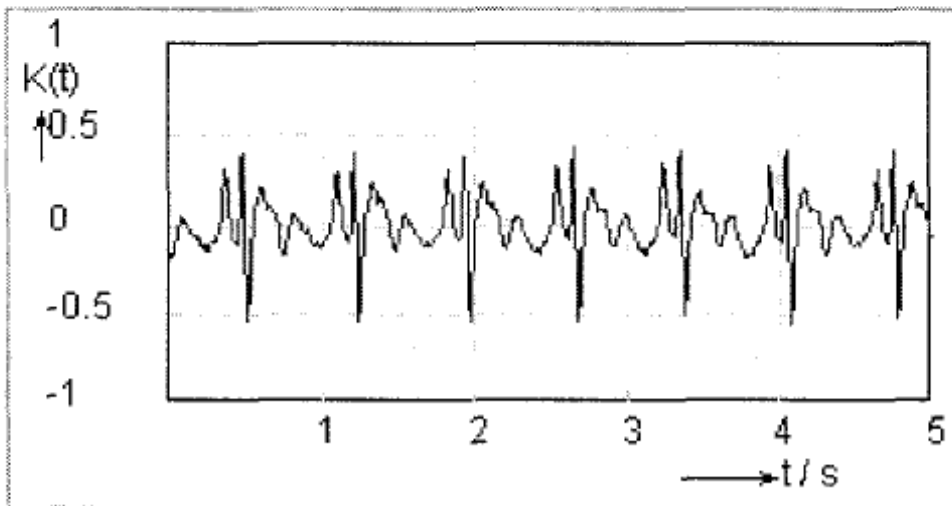
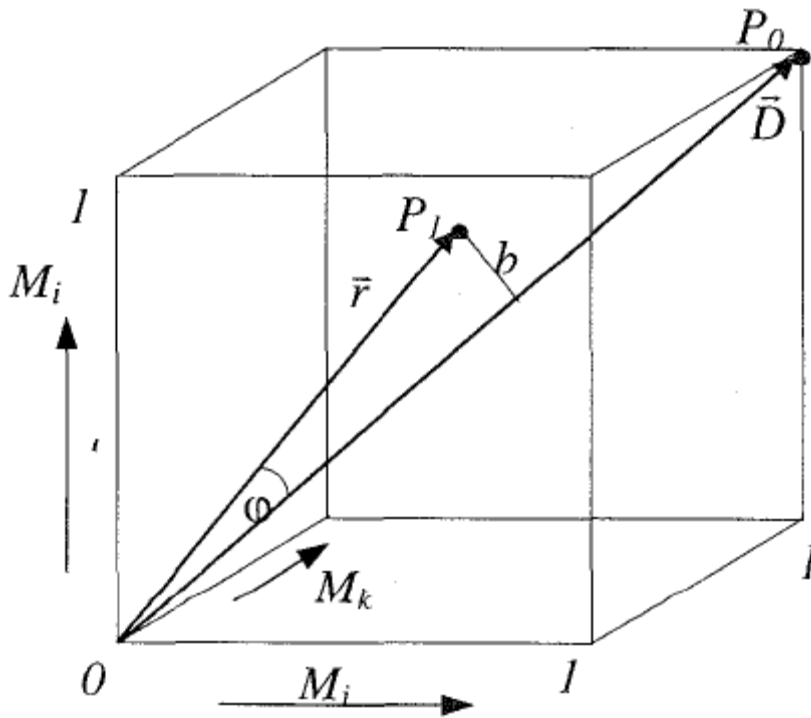


Fig. 2 Correlation function with distinct differences of the signal patterns

#### Figure 24. Correlation Patterns

It is the aim of classification to summarily assess the agreement of the lead-related correlation results obtained from the comparison of the unknown ECG with any reference ECG. It must be taken into consideration that the result of two ECG pattern comparisons may be multivariable and available as a 3-dimensional vector of the correlation results of the individual leads and the number of comparisons with the reference ECG may be

very great. To solve this task, a modified distance technique of the multivariate signal analysis is used. As the method can be applied independently of the dimension of the vector of the correlation results and thus independently of the number of leads to be taken into account, the classification step is illustrated by the example of the dimension  $n = 3$ .



**Figure 25. Graphical interpretation of the multichannel correlation results**

Fig. 3 shows a three-dimensional Cartesian system of coordinates which is formed by the correlation results of three ECG leads. According to the used range of values for the correlation coefficient  $K$ , the axial ranges from 0 to 1 are obtained. Any comparison of signal patterns of the unknown ECG with those of an ECG from the database leads to a point inside the cube formed for the three dimensional case. If the signal patterns are identical in all leads ( $K = 1$ ), point  $P_0 (1,1,1)$  will be obtained. The length of the diagonal in this cube represents the maximum similarity measure in the case of three leads. All other points are associated with vectors whose magnitude are smaller compared with the length of the diagonal and thus indicate less similarity of the signal patterns. When all leads of the 12-channel ECG are allowed for, the dimension  $m = 12$  is obtained. The normalized modulus vector as similarity measure then reads

$$|\bar{r}| = \frac{\sqrt{\sum_{i=1}^m (M_i)^2}}{\sqrt{m}}.$$

$M_i$ : maxima of the  
 correlations of a lead  
 $m$ : dimension

A second value should allow for the uniformity of the correlations of the leads. This is possible by indicating the angle  $\varphi$  or the distance  $b$ :

$$\varphi = \arccos \left( \frac{1}{\sqrt{m}} \frac{\sum_{i=1}^m M_i}{\sqrt{\sum_{i=1}^m (M_i)^2}} \right), \quad b = |r| \sin \varphi.$$

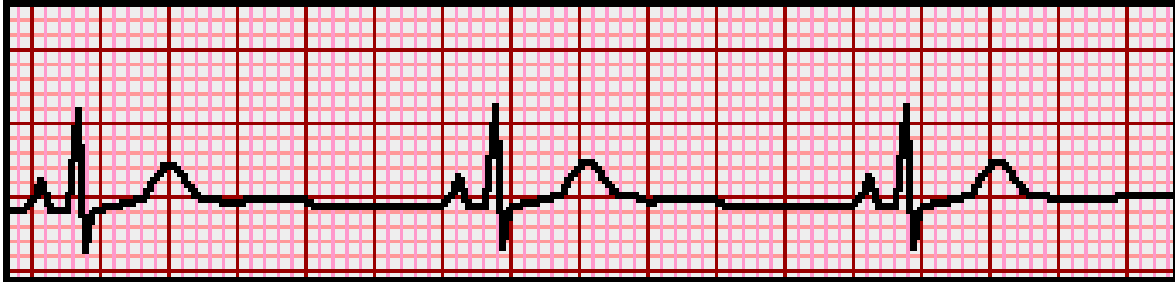
The value  $b = 0$  means here that equal correlation values were calculated for all leads. This does not allow conclusions to be drawn for the similarity of the signal patterns of all leads. It is, however, possible, for example, to take in the influence of noise-affected signals, or other technical disturbances in a lead, on the overall result. The two values  $r$  and  $b$  can be calculated for any comparison between an ECG of unknown diagnosis with the reference ECG.

### **3.3.10 HEART RATE DETECTION**

The heart rate of the patient is the one of the most important parameter for analysis of cardiac disorders of a patient. An ECG wave consisting of P wave, QRS complex, S wave and T wave is generated for every heart beat. The number of heart beats per minute can be measured by measuring number of QRS complexes that were observed in one minute duration. However for practical purposes the ECG wave is not observed for entire one

minute duration. Instead it is observed for a 10 second duration and then the heart rate is calculated using 10 second rule or a rule of 300.

### **Rule of 300**



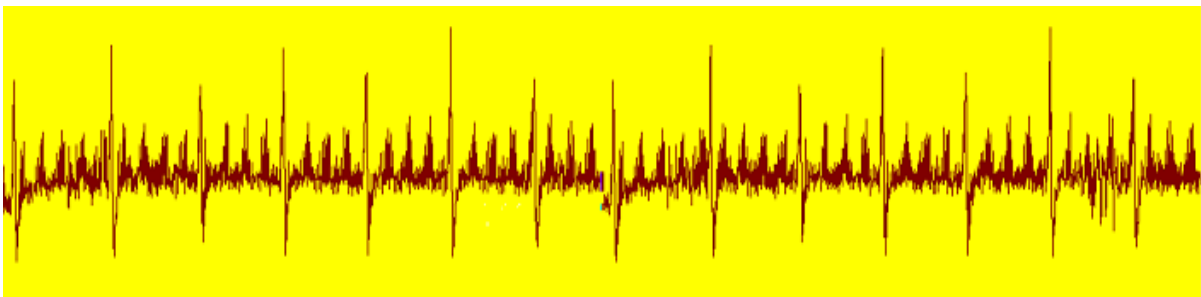
**Figure 23 An ECG wave**

Here 5 big boxes represents time duration of 1 second. To measure heart rate, count the number of such big boxes between consecutive QRS complexes and divide the number of big boxes by 300. The result will be approximately equal to heart rate. In the above figure, there are 6 boxes between consecutive QRS complexes. Therefore the heart rate is given by

$$\text{HEART RATE} = 300 / 6 = 50 \text{ BPM}$$

### **10 Second Rule**

As most ECGs record 10 seconds of rhythm per page, one can simply count the number



**Figure 24 ECG wave 10 second rule**

of beats present on the EKG and multiply by 6 to get the number of beats per 60 seconds. This method works well for irregular rhythms. In the above figure there are 14 QRS complexes in the duration of 10 seconds. Hence the heart rate is given by

$$\text{HEART RATE} = 14 \times 6 = 84 \text{ BPM}$$

## **4. HARDWARE IMPLEMENTATION**

## 4.1 CIRCUIT DIAGRAM

The following figure shows the detailed circuit diagram for ECG signal conditioning.

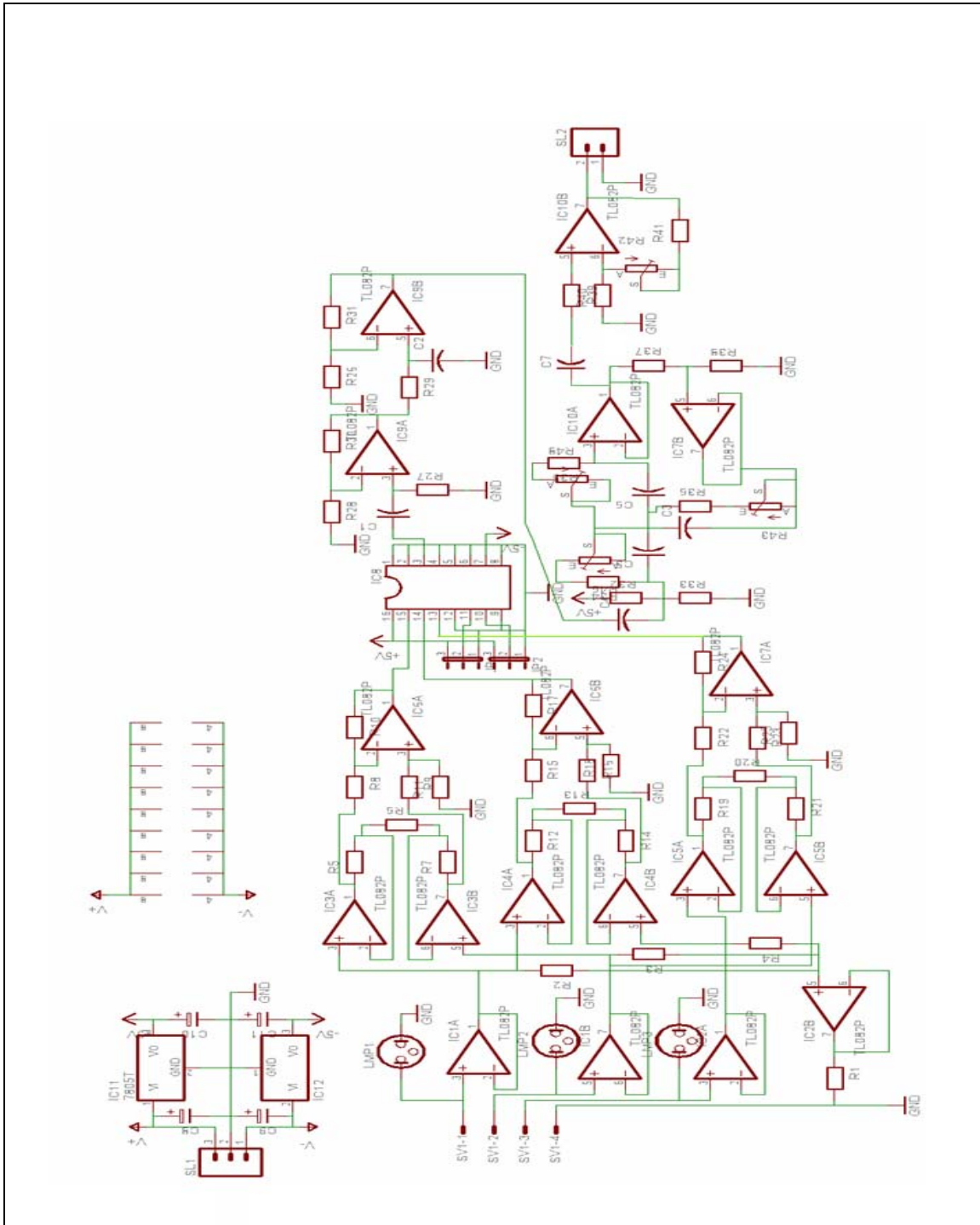


Figure 25 ECG signal conditioning circuit



## **4.2 WORKING**

The Electrocardiogram signal is sensed using sensors connected to Human body. The sensors were connected to Right Arm (RA), Left Arm (LA), Left Leg (LL) and Right Leg (RL) with RL as a reference. The ECG signals obtained at input are very weak signals with a voltage in the range of mill volts. So initially the input signals from body sensors are buffered to increase current driving capacity. The buffered signals are applied to Instrumentation Amplifiers to generate a difference signal with RL signal as a reference. The Instrumentation Amplifier used in the circuit is a 3 OP-Amp instrumentation amplifier which provides a gain of 40. There are three such instrumentation amplifiers in the circuit for obtaining leads I, II and III simultaneously. Out of the three leads, the lead II connected to Left Arm (LA), Left Leg (LL) and Right Leg (RL) is the strongest lead. An analog Multiplexer TC4051BP is used to select one lead out of the three leads obtained at the output of three instrumentation amplifier. The lead selected by the Multiplexer is then applied as an input to set of Filters. Initially, the signal passes through a High Pass Filter (HPF). The high pass filter filters out the high frequency noise and Electromagnetic Interference from the Electrocardiogram signal. It is also used to remove Baseline Wandering caused by factors such as the instability of the electrodes, the change in electrode-skin interface conductivity, the modulating effect of subject respiration, etc. The filtered signal is then passed through a Low Pass Filter (LPF). The rejection of DC and very low-frequency components has a main role to remove the noise from body electrodes and related electro-chemical potentials. It is ensured that the Low Pass Filter does not remove the significant low-frequency components of the ECG signal. This filtered Electrocardiogram signal is finally sent to a notch filter to remove 50Hz noise due to interference from power line. The 50 Hz power line interference frequency is within the useful bandwidth of Electrocardiogram signal; hence the 50 Hz rejection filter is a very narrow filter. The ECG signal is now free from major sources of interference. This signal is then amplified using a non-inverting amplifier, which provides a gain of 25. The output of non-inverting amplifier is up to 1.2 volts and is sent to line input of TMS320C6713 DSK for further processing. The overall gain provided by the circuit is 40 (by Instrumentation amplifier) X 25 (by non-inverting amplifier) = 1000.

### **4.3 COMPONENT LIST**

R1 – 100  $\Omega$ ; R2,3,4 – 10 M $\Omega$ ; R6,8,9,13,15,16,20,22,23,42 – 10 K $\Omega$

R5,7,12,14,19,21 – 120 K $\Omega$ ; R10,17,24 – 20 K $\Omega$

R28,30 – 20 K $\Omega$ ; R27 – 1 M $\Omega$ ; R29 – 56 K $\Omega$ ;

R26,31 – 390 K $\Omega$ ; R34,48 – 27 K $\Omega$ ;

R35,46 – 4.8 K $\Omega$ ; R36 – 12 K $\Omega$ ;

R43 – 3.9 K $\Omega$ ; R37 – 27 K $\Omega$ ;

R38 – 100 K $\Omega$ ; R39,40 – 1 K $\Omega$ ;

R41 – 15K $\Omega$ ;

C1 – 220 nF/100V

C2 – 100nF/63V

C3 – 2000nF

C4,7 – 470nF

C5,6 – 100nF

C8,9,410,11 - 10 $\mu$ F/63V

U1 – LM7805 (3 Pin Voltage Regulator)

U2 – LM7905 (3 Pin Voltage Regulator)

U3 – TL082CN

U4 – TC4051BP

U5 – TMS320C6713 DSK

## **5. SOFTWARE IMPLEMENTATION**

## **5.1 OVERVIEW OF CODE COMPOSER 3.1**

Code Composer Studio (CCS) allows us to write a program in C language that can be used to initialize the DSK. Through CCS, we can initialize various ports and registers of the DSK. Code Composer provides a rich debugging environment that allows stepping through the code, set breakpoints, and examining the registers as code is getting executed.

The Code Composer Studio (CCS) application provides an integrated environment with the capabilities like Integrated development environment with an editor, debugger, project manager, and profiler, C/C++ compiler, assembly optimizer and linker, Simulator, Real-time operating system (DSP/BIOS™), Real-Time Data Exchange (RTDX™) between the Host and Target, and Real-time analysis and data visualization.

CCStudio integrated development environment includes host tools and target software that slashes development time and optimizes the performance for all real-time embedded DSP applications.

Some of the Code Composer Studio's host side tools include TMS320 DSPs and OMAP Code, Drag and Drop CCStudio setup utility, Component manager support for multiple versions of DSP/BIOS and code generation tools within the IDE, Source Code Debugger common interface for both simulator and emulator targets, Connect/Disconnect; robust, resilient host to target connection, Application Code Tuning Dashboard, RTDX™ data transfer for real time data exchange between host and target, Data Converter Plug-in to auto configure support for Texas Instruments Mixed Signal products, Quick Start tutorials and Help.

Code Composer Studio's target software includes DSP/BIOS™ Kernel for the TMS320 DSPs, TMS320 DSP Algorithm Standard to enable software reuse, Chip Support Libraries to simplify device configuration, and DSP Libraries for optimum DSP functionality.

## 5.2 ALGORITHMS

### INITIALISATION

- a) Initialize DSK codec aic23 using appropriate header file.
- b) Initialize the buffers for storing the original and compressed ECG data
- c) Set the gain for input samples from aic23 line input
- d) Set the sampling frequency of the ADC to minimum (8kHz)

### DATA ACQUISITION

- a) Read the input from the line input
- b) The sampling frequency required for ECG is less than the sampling frequency of codec (about 400 Hz). Hence scale down the sampling rate by appropriate value (20)
- c) Store these samples in a .dat file
- d) Convert the dat files into header files.

### CORRELATION

- a) Initialize an array k to 0. Initialize a variable n to 0.
- b) Using the array elements of the database as well as input files calculate the values of array k using the formula given in the literature survey
- c) Also use an if statement to find out the maximum value of the entire array
- d) Repeat the above procedure for all the lead arrays
- e) Use the maximum values of all the arrays for computing r, b and phi
- f) Display the values of r, b and phi

### HEART RATE CALCULATION

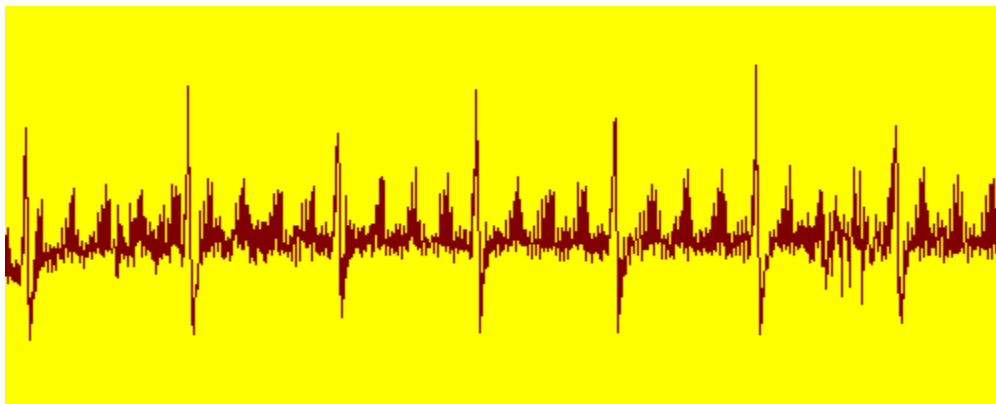
- a) Store the input .dat file into an array called ECG
- b) Set the threshold value to be 80% of peak value of ECG sample.
- c) Increment heart rate each time ECG sample value rises above threshold value
- d) Continue this procedure for 10 seconds
- e) At the end of 10 seconds, actual heart rate is obtained by multiplying the heart rate value obtained above by 6.

## **6. RESULT ANALYSIS AND CONCLUSION**

## 6.1 RESULT ANALYSIS

1. The hardware circuit designed to amplify the ECG signal provided a gain of 1000 and the output ECG voltage was observed to be up to 2V.
2. The noise due to baseline wandering and contact electrodes was successfully removed by using High Pass and Low Pass filters respectively.
3. Noise corresponding to 50 Hz power line frequency was still observed to be present in the signal.
4. The correlation program classifies the input ECG signal into normal, bradycardia and tachycardia signals by correlating the input signal with signals from the database and determines the signal according to the best match.
5. The program was tried with 10 input files with different heart conditions. The program could successfully identify 9 input conditions correctly. Thus the code for correlation gives around 90% accuracy.
6. The heart rate of the different patients varies from 30 to 110. However, since the calculation were made by multiplying the QRS complexes observed in 10 sec by 6, the heart rate always displayed as a multiple of 6. Hence, minor changes in heart rate were undetected.

The following figure shows the ECG signal displayed using the graphical display feature of Code Composer Studio v3.1



**Figure 26. ECG displayed using CCS v3.1**

## **6.2 CONCLUSION**

Thus, ECG correlation was successfully implemented DSP processing kit TMS320C6713 DSK. The ECG signal was generated using standard limb leads for obtaining normal signals and an ECG signal simulator was used for generating abnormal ECG signals. The signal was stored and correlated using the pattern comparison algorithm and an accuracy of 90% was obtained. The heart rate was successfully measured and displayed.



## **7. FUTURE SCOPE**

## **7.1 FUTURE SCOPE**

The interactive possibilities of this project are intended to enable the doctor - in contrast to conventional methods of computer aided ECG interpretation - to separately evaluate the results including the patient information available. The result of the signal pattern comparison is not a diagnostic statement which can be accepted or rejected but a probability statement on the basis of similar cases of an ECG database. Accordingly, the quality of this database as regards the extent and the validation is a decisive prerequisite for the function of the technique presented.

The database can be extended for other abnormalities without any change in the algorithm. The procedure followed for the processing of electrocardiogram signal can be extended for processing of other biomedical signal like electroencephalogram as well as non-biomedical signals. A graphical user interface (GUI) can be built to enhance the flexibility and user friendliness of processing system.

## **APPENDIX**

# APPENDIX I – CCS CODE FOR BIOMEDICAL SIGNAL PROCESSING

```
/*
 * ===== DSK6713_loop.c =====
 *
 * This example uses the AIC23 codec module of the 6713 DSK Board Support
 * Library to process left and right samples from input to output without
 * modification. Two 256 point buffers are included so that captured signal
 * samples can be observed using the CCS plotting tools.
 */

/*
 * DSP/BIOS is configured using the DSP/BIOS configuration tool. Settings
 * for this example are stored in a configuration file called tone.cdb. At
 * compile time, Code Composer will auto-generate DSP/BIOS related files
 * based on these settings. A header file called DSK6713_loopcfg.h contains the
 * results of the autogeneration and must be included for proper operation.
 * The name of the file is taken from DSK6713_loop(.cdb) and adding cfg.h.
 */

// Include header file auto-generated by the configuration tool
#include "DSK6713_loopcfg.h"

/*
 * The 6713 DSK Board Support Library is divided into several modules, each
 * of which has its own include file. The file dsk6713.h must be included
 * in every program that uses the BSL. This example also includes
 * dsk6713_aic23.h because it uses the AIC23 codec module.
 */

/*
 * Note the BSL has defined custom data types in csl_stdinc.h,
 * popular among DSP developers:
 *
 * typedef unsigned char   Uint8;
 * typedef unsigned short  Uint16;
 * typedef unsigned int    Uint32;
 * typedef unsigned long   Uint40;
 * typedef char           Int8;
 * typedef short          Int16;
 * typedef int            Int32;
 * typedef long           Int40;
```

```
*
*/

#include "dsk6713.h"
#include "dsk6713_aic23.h"
#include "stdio.h"
#include "math.h"
#include "string.h"

#include "input11.h"
#include "input12.h"
#include "input13.h"

#include "input21.h"
#include "input22.h"
#include "input23.h"

#include "input31.h"
#include "input32.h"
#include "input33.h"
/*
#include "input41.h"
#include "input42.h"
#include "input43.h"
*/
#include "input51.h"
#include "input52.h"
#include "input53.h"

#include "input61.h"
#include "input62.h"
#include "input63.h"

#include "input71.h"
#include "input72.h"
#include "input73.h"

#include "input81.h"
#include "input82.h"
#include "input83.h"

#include "input91.h"
#include "input92.h"
#include "input93.h"

#include "input101.h"
```

```
#include "input102.h"
#include "input103.h"
```

```
#include "normal11.h"
#include "normal12.h"
#include "normal13.h"
#include "normal21.h"
#include "normal22.h"
#include "normal23.h"
#include "normal31.h"
#include "normal32.h"
#include "normal33.h"
```

```
#include "brady11.h"
#include "brady12.h"
#include "brady13.h"
#include "brady21.h"
#include "brady22.h"
#include "brady23.h"
#include "brady31.h"
#include "brady32.h"
#include "brady33.h"
```

```
#include "tachy11.h"
#include "tachy12.h"
#include "tachy13.h"
#include "tachy21.h"
#include "tachy22.h"
#include "tachy23.h"
#include "tachy31.h"
#include "tachy32.h"
#include "tachy33.h"
```

```
int rand_int(void);
```

```
/* Codec configuration settings */
DSK6713_AIC23_Config config = { \
    0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Left line input channel volume */ \
    0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume */ \
    0x01f9, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */ \
    0x01f9, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */ \
```

```

0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */ \
0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */ \
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */ \
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */ \
0x0081, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */ \
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};

```

```
// Global data buffers
```

```

/*
 * main() - Main code routine, initializes BSL and connects input samples
 *          to output samples in an infinite while loop.
 */

```

```

##pragma DATA_SECTION(phiT5,".EXTRAM")
##pragma DATA_SECTION(phiB1,".EXTRAM")
##pragma DATA_SECTION(phiB2,".EXTRAM")
##pragma DATA_SECTION(phiB3,".EXTRAM")
##pragma DATA_SECTION(phiB4,".EXTRAM")
##pragma DATA_SECTION(phiB5,".EXTRAM")
##pragma DATA_SECTION(k44,".EXTRAM")
##pragma DATA_SECTION(k45,".EXTRAM")

```

```

float
k1[4800],k2[4800],k3[4800],k4[4800],k5[4800],k6[4800],k7[4800],k8[4800],k9[4800],k
10[4800];

```

```

float
k16[4800],k17[4800],k18[4800],k19[4800],k20[4800],k21[4800],k22[4800],k23[4800],k
24[4800];

```

```

float
k31[4800],k32[4800],k33[4800],k34[4800],k35[4800],k36[4800],k37[4800],k38[4800],k
39[4800];

```

```

int s1[4800],s2[4800],s3[4800],d1,d2,d3;
int ECG[4800];

```

```

int HEART_RATE = 0,flag = 0,MAX=0,m=0,n=0,i = 0;
float THRESH = 0;

```

```
void main()
```

```

{
float
RN1,RN2,RN3,RT1,RT2,RT3,RB1,RB2,RB3,BN1,BN2,BN3,BT1,BT2,BT3,BB1,BB2,
BB3,phiN1,phiN2,phiN3,phiT1,phiT2,phiT3,phiB1,phiB2,phiB3,BN,BB,BT;

float
maxnormal11=0,maxnormal12=0,maxnormal13=0,maxnormal21=0,maxnormal22=0,ma
xnormal23=0,maxnormal31=0,maxnormal32=0,maxnormal33=0;
float
maxtachy11=0,maxtachy12=0,maxtachy13=0,maxtachy21=0,maxtachy22=0,maxtachy23
=0,maxtachy31=0,maxtachy32=0,maxtachy33=0;
float
maxbrady11=0,maxbrady12=0,maxbrady13=0,maxbrady21=0,maxbrady22=0,maxbrady
23=0,maxbrady31=0,maxbrady32=0,maxbrady33=0;

long double a=0,b=0,c=0,d=0,e=0;

/* Initialize the board support library, must be called first */
DSK6713_init();

printf("enter the number of the signal that you want to input: input11, input21, input31,
input51, input61, input71, input81, input91, input101");
scanf("%d",&d1);
switch(d1)
{
case 11:
{
for(i=0;i<4800;i++)
s1[i]=input11[i];
break;
}
case 21:
{
for(i=0;i<4800;i++)
s1[i]=input21[i];
break;
}
case 31:
{
for(i=0;i<4800;i++)
s1[i]=input31[i];
break;
}
}

```



```

case 51:
{
for(i=0;i<4800;i++)
s1[i]=input51[i];
break;
}
case 61:
{
for(i=0;i<4800;i++)
s1[i]=input61[i];
break;
}
case 71:
{
for(i=0;i<4800;i++)
s1[i]=input71[i];
break;
}
case 81:
{
for(i=0;i<4800;i++)
s1[i]=input81[i];
break;
}
case 91:
{
for(i=0;i<4800;i++)
s1[i]=input91[i];
break;
}
case 101:
{
for(i=0;i<4800;i++)
s1[i]=input101[i];
break;
}
printf("invalid number");
}

```

```

printf("enter the number of the signal that you want to input: input12, input22, input32,
input52, input62, input72, input82, input92, input102");
scanf("%d",&d2);
switch(d2)
{
case 12:
{

```

```
for(i=0;i<4800;i++)
s2[i]=input12[i];
break;
}
case 22:
{
for(i=0;i<4800;i++)
s2[i]=input22[i];
break;
}
case 32:
{
for(i=0;i<4800;i++)
s2[i]=input32[i];
break;
}
case 52:
{
for(i=0;i<4800;i++)
s2[i]=input52[i];
break;
}
case 62:
{
for(i=0;i<4800;i++)
s2[i]=input62[i];
break;
}
case 72:
{
for(i=0;i<4800;i++)
s2[i]=input72[i];
break;
}
case 82:
{
for(i=0;i<4800;i++)
s2[i]=input82[i];
break;
}
case 92:
{
for(i=0;i<4800;i++)
s2[i]=input92[i];
break;
}
}
```

```
case 102:
{
for(i=0;i<4800;i++)
s2[i]=input102[i];
break;
}
printf("invalid number");
}
```

```
printf("enter the name of the signal that you want to input: input13, input23, input33,
input53, input63, input73, input83, input93, input103");
scanf("%d",&d3);
switch(d3)
{
case 13:
{
for(i=0;i<4800;i++)
s3[i]=input13[i];
break;
}
case 23:
{
for(i=0;i<4800;i++)
s3[i]=input23[i];
break;
}
case 33:
{
for(i=0;i<4800;i++)
s3[i]=input33[i];
break;
}
case 53:
{
for(i=0;i<4800;i++)
s3[i]=input53[i];
break;
}
case 63:
{
for(i=0;i<4800;i++)
s3[i]=input63[i];
break;
}
case 73:
```

```

{
for(i=0;i<4800;i++)
s3[i]=input73[i];
break;
}
case 83:
{
for(i=0;i<4800;i++)
s3[i]=input83[i];
break;
}
case 93:
{
for(i=0;i<4800;i++)
s3[i]=input93[i];
break;
}
case 103:
{
for(i=0;i<4800;i++)
s3[i]=input103[i];
break;
}
printf("invalid number");
}

while(1)
{

// Read a 32 bit codec sample from the serial port.
// The reads alternate between left and right channels.
// The value read is held as a Uint32, but the valid data
// is in a Int16 format in the 16 MSBs
//while (!DSK6713_AIC23_read(hCodec, &xL));
//while (!DSK6713_AIC23_read(hCodec, &xR));

// Extract ECG samples in a 2000 byte buffer

// Threshold

if((s1[m]>MAX)&&(m<=800))
{
MAX=s1[m];
THRESH=0.65*MAX;
}
}

```

```

    }

    // Heart Rate Detection
    if((s1[m]<THRESH)&&(s1[m-
1]>=THRESH)&&(flag==0)&&(m>800))
    {
        ++HEART_RATE;
        printf("\nHEART_RATE=%d",HEART_RATE);
        flag=200;
    }

    if(flag>0)
    flag--;

    if(m>4800)
    {
        HEART_RATE *=6;
        printf("\nHEART_RATE=%d",HEART_RATE);
        break;
    }

m++;

//while (!DSK6713_AIC23_write(hCodec, xL));
//while (!DSK6713_AIC23_write(hCodec, xR));

}

```

```

//normal11
a=0;
b=0;
c=0;
d=0;
e=0;
for(i=0;i<=4800-1;i++)
{
a+=normal11[i]*s1[i];
b+=normal11[i];
c+=s1[i];
d+=normal11[i]*normal11[i];
e+=s1[i]*s1[i];
k1[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));
}

```

```

if(k1[i]>maxnormal11)
{
maxnormal11=k1[i];
}

}
printf("\nmaxnormal11=%f",maxnormal11);

//normal12
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=normal12[i]*s2[i];
b+=normal12[i];
c+=s2[i];
d+=normal12[i]*normal12[i];
e+=s2[i]*s2[i];
k2[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k2[i]>maxnormal12)
{
maxnormal12=k2[i];
}

}

printf("\nmaxnormal12=%f",maxnormal12);

//normal13
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=normal13[i]*s3[i];
b+=normal13[i];
c+=s3[i];

```

```

d+=normal13[i]*normal13[i];
e+=s3[i]*s3[i];
k3[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*((e-((c*c)/4800))));

if(k3[i]>maxnormal13)
{
maxnormal13=k3[i];
}

}

```

```
printf("\nmaxnormal13=%f",maxnormal13);
```

```

//normal21
a=0;
b=0;
c=0;
d=0;
e=0;
for(i=0;i<=4800-1;i++)
{
a+=normal21[i]*s1[i];
b+=normal21[i];
c+=s1[i];
d+=normal21[i]*normal21[i];
e+=s1[i]*s1[i];
k4[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*((e-((c*c)/4800))));

if(k4[i]>maxnormal21)
{
maxnormal21=k4[i];
}

}

```

```
printf("\nmaxnormal21=%f",maxnormal21);
```

```

//normal22
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)

```

```

{
a+=normal22[i]*s2[i];
b+=normal22[i];
c+=s2[i];
d+=normal22[i]*normal22[i];
e+=s2[i]*s2[i];
k5[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k5[i]>maxnormal22)
{
maxnormal22=k5[i];
}
}
printf("\nmaxnormal22=%f",maxnormal22);

//normal23
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=normal23[i]*s3[i];
b+=normal23[i];
c+=s3[i];
d+=normal23[i]*normal23[i];
e+=s3[i]*s3[i];
k6[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k6[i]>maxnormal23)
{
maxnormal23=k6[i];
}
}
printf("\nmaxnormal23=%f",maxnormal23);

//normal31
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)

```



```

{
a+=normal31[i]*s1[i];
b+=normal31[i];
c+=s1[i];
d+=normal31[i]*normal31[i];
e+=s1[i]*s1[i];
k7[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k7[i]>maxnormal31)
{
maxnormal31=k7[i];
}
}
printf("\nmaxnormal31=%f",maxnormal31);

//normal32
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=normal32[i]*s2[i];
b+=normal32[i];
c+=s2[i];
d+=normal32[i]*normal32[i];
e+=s2[i]*s2[i];
k8[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k8[i]>maxnormal32)
{
maxnormal32=k8[i];
}
}
printf("\nmaxnormal32=%f",maxnormal32);

//normal33
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)

```

```

{
a+=normal33[i]*s3[i];
b+=normal33[i];
c+=s3[i];
d+=normal33[i]*normal33[i];
e+=s3[i]*s3[i];
k9[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k9[i]>maxnormal33)
{
maxnormal33=k9[i];
}
}
printf("\nmaxnormal33=%f",maxnormal33);

```

```
//brady11
```

```

a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=brady11[i]*s1[i];
b+=brady11[i];
c+=s1[i];
d+=brady11[i]*brady11[i];
e+=s1[i]*s1[i];
k16[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k16[i]>maxbrady11)
{
maxbrady11=k16[i];
}
}
printf("\nmaxbrady11=%f",maxbrady11);

```

```
//brady12
```

```

a=0;
b=0;
c=0;

```

```

d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=brady12[i]*s2[i];
b+=brady12[i];
c+=s2[i];
d+=brady12[i]*brady12[i];
e+=s2[i]*s2[i];
k17[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*((e-((c*c)/4800))));

if(k17[i]>maxbrady12)
{
maxbrady12=k17[i];
}
}
printf("\nmaxbrady12=%f",maxbrady12);

```

```
//brady13
```

```

a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=brady13[i]*s3[i];
b+=brady13[i];
c+=s3[i];
d+=brady13[i]*brady13[i];
e+=s3[i]*s3[i];
k18[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*((e-((c*c)/4800))));

if(k18[i]>maxbrady13)
{
maxbrady13=k18[i];
}
}
printf("\nmaxbrady13=%f",maxbrady13);

```

```
//brady 21
```

```
a=0;
```

```

b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=brady21[i]*s1[i];
b+=brady21[i];
c+=s1[i];
d+=brady21[i]*brady21[i];
e+=s1[i]*s1[i];
k19[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k19[i]>maxbrady21)
{
maxbrady21=k19[i];
}
}
printf("\nmaxbrady21=%f",maxbrady21);

```

```

//brady22
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=brady22[i]*s2[i];
b+=brady22[i];
c+=s2[i];
d+=brady22[i]*brady22[i];
e+=s2[i]*s2[i];
k20[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k20[i]>maxbrady22)
{
maxbrady22=k20[i];
}
}
printf("\nmaxbrady22=%f",maxbrady22);

```

```

//brady23
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=brady23[i]*s3[i];
b+=brady23[i];
c+=s3[i];
d+=brady23[i]*brady23[i];
e+=s3[i]*s3[i];
k21[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k21[i]>maxbrady23)
{
maxbrady23=k21[i];
}
}
printf("\nmaxbrady23=%f",maxbrady23);

//brady31
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=brady31[i]*s1[i];
b+=brady31[i];
c+=s1[i];
d+=brady31[i]*brady31[i];
e+=s1[i]*s1[i];
k22[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k22[i]>maxbrady31)
{
maxbrady31=k22[i];
}
}
printf("\nmaxbrady31=%f",maxbrady31);

```

```

//brady32
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=brady32[i]*s2[i];
b+=brady32[i];
c+=s2[i];
d+=brady32[i]*brady32[i];
e+=s2[i]*s2[i];
k23[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k23[i]>maxbrady32)
{
maxbrady32=k23[i];
}
}
printf("\nmaxbrady32=%f",maxbrady32);

```

```

//brady33
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=brady33[i]*s3[i];
b+=brady33[i];
c+=s3[i];
d+=brady33[i]*brady33[i];
e+=s3[i]*s3[i];
k24[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k24[i]>maxbrady33)
{
maxbrady33=k24[i];
}
}

```

```
printf("\nmaxbrady33=%f",maxbrady33);
```

```
//tachy11
```

```
a=0;
```

```
b=0;
```

```
c=0;
```

```
d=0;
```

```
e=0;
```

```
for(i=0;i<=4800-1;i++)
```

```
{
```

```
a+=tachy11[i]*s1[i];
```

```
b+=tachy11[i];
```

```
c+=s1[i];
```

```
d+=tachy11[i]*tachy11[i];
```

```
e+=s1[i]*s1[i];
```

```
k31[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));
```

```
if(k31[i]>maxtachy11)
```

```
{
```

```
maxtachy11=k31[i];
```

```
}
```

```
}
```

```
printf("\nmaxtachy11=%f",maxtachy11);
```

```
//tachy12
```

```
a=0;
```

```
b=0;
```

```
c=0;
```

```
d=0;
```

```
e=0;
```

```
for(i=0;i<=4800-1;i++)
```

```
{
```

```
a+=tachy12[i]*s2[i];
```

```
b+=tachy12[i];
```

```
c+=s2[i];
```

```
d+=tachy12[i]*tachy12[i];
```

```
e+=s2[i]*s2[i];
```

```
k32[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));
```

```
if(k32[i]>maxtachy12)
```

```
{
```

```

maxtachy12=k32[i];
}
}
printf("\nmaxtachy12=%f",maxtachy12);

//tachy13
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=tachy13[i]*s3[i];
b+=tachy13[i];
c+=s3[i];
d+=tachy13[i]*tachy13[i];
e+=s3[i]*s3[i];
k33[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k33[i]>maxtachy13)
{
maxtachy13=k33[i];
}
}

printf("\nmaxtachy13=%f",maxtachy13);

//tachy21

a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=tachy21[i]*s1[i];
b+=tachy21[i];
c+=s1[i];
d+=tachy21[i]*tachy21[i];
e+=s1[i]*s1[i];
k34[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

```



```

if(k34[i]>maxtachy21)
{
maxtachy21=k34[i];
}
}
printf("\nmaxtachy21=%f",maxtachy21);

//tachy22
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=tachy22[i]*s2[i];
b+=tachy22[i];
c+=s2[i];
d+=tachy22[i]*tachy22[i];
e+=s2[i]*s2[i];
k35[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k35[i]>maxtachy22)
{
maxtachy22=k35[i];
}
}
printf("\nmaxtachy22=%f",maxtachy22);

//tachy23
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=tachy23[i]*s3[i];
b+=tachy23[i];
c+=s3[i];
d+=tachy23[i]*tachy23[i];
e+=s3[i]*s3[i];
}
}

```

```
k36[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));
```

```
if(k36[i]>maxtachy23)
```

```
{  
maxtachy23=k36[i];  
}  
}
```

```
printf("\nmaxtachy23=%f",maxtachy23);
```

```
//tachy31
```

```
a=0;
```

```
b=0;
```

```
c=0;
```

```
d=0;
```

```
e=0;
```

```
for(i=0;i<=4800-1;i++)
```

```
{  
a+=tachy31[i]*s1[i];  
b+=tachy31[i];
```

```
c+=s1[i];
```

```
d+=tachy31[i]*tachy31[i];  
e+=s1[i]*s1[i];
```

```
k37[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));  
if(k37[i]>maxtachy31)
```

```
{  
maxtachy31=k37[i];  
}
```

```
}
```

```
printf("\nmaxtachy31=%f",maxtachy31);
```

```
//tachy32
```

```
a=0;
```

```
b=0;
```

```
c=0;
```

```
d=0;
```

```
e=0;
```

```
for(i=0;i<=4800-1;i++)
```

```
{  
a+=tachy32[i]*s2[i];  
b+=tachy32[i];
```

```
c+=s2[i];
```

```

d+=tachy32[i]*tachy32[i];
e+=s2[i]*s2[i];
k38[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k38[i]>maxtachy32)
{
maxtachy32=k38[i];
}
}
printf("\nmaxtachy32=%f",maxtachy32);

//tachy33
a=0;
b=0;
c=0;
d=0;
e=0;

for(i=0;i<=4800-1;i++)
{
a+=tachy33[i]*s3[i];
b+=tachy33[i];
c+=s3[i];
d+=tachy33[i]*tachy33[i];
e+=s3[i]*s3[i];
k39[i]=(a-((b*c)/4800))/sqrt((d-((b*b)/4800))*(e-((c*c)/4800)));

if(k39[i]>maxtachy33)
{
maxtachy33=k39[i];
}
}
printf("\nmaxtachy33=%f",maxtachy33);

```

```
// calculations for R,phi,B
```

```

RN1=sqrt(((maxnormal11)*(maxnormal11)+(maxnormal12)*(maxnormal12)+(maxnormal13)*(maxnormal13))/3);
RN2=sqrt(((maxnormal21)*(maxnormal21)+(maxnormal22)*(maxnormal22)+(maxnormal23)*(maxnormal23))/3);

```

```
RN3=sqrt(((maxnormal31)*(maxnormal31)+(maxnormal32)*(maxnormal32)+(maxnormal33)*(maxnormal33))/3);
```

```
RB1=sqrt(((maxbrady11)*(maxbrady11)+(maxbrady12)*(maxbrady12)+(maxbrady13)*(maxbrady13))/3);
```

```
RB2=sqrt(((maxbrady21)*(maxbrady21)+(maxbrady22)*(maxbrady22)+(maxbrady23)*(maxbrady23))/3);
```

```
RB3=sqrt(((maxbrady31)*(maxbrady31)+(maxbrady32)*(maxbrady32)+(maxbrady33)*(maxbrady33))/3);
```

```
RT1=sqrt(((maxtachy11)*(maxtachy11)+(maxtachy12)*(maxtachy12)+(maxtachy13)*(maxtachy13))/3);
```

```
RT2=sqrt(((maxtachy21)*(maxtachy21)+(maxtachy22)*(maxtachy22)+(maxtachy23)*(maxtachy23))/3);
```

```
RT3=sqrt(((maxtachy31)*(maxtachy31)+(maxtachy32)*(maxtachy32)+(maxtachy33)*(maxtachy33))/3);
```

```
printf("\nRN1=%f",RN1);  
printf("\nRN2=%f",RN2);  
printf("\nRN3=%f",RN3);
```

```
printf("\nRB1=%f",RB1);  
printf("\nRB2=%f",RB2);  
printf("\nRB3=%f",RB3);
```

```
printf("\nRT1=%f",RT1);  
printf("\nRT2=%f",RT2);  
printf("\nRT3=%f",RT3);
```

```
phiN1=acos((maxnormal11+maxnormal12+maxnormal13)/sqrt(((maxnormal11)*(maxnormal11)+(maxnormal12)*(maxnormal12)+(maxnormal13)*(maxnormal13))*3));
```

```
phiN2=acos((maxnormal21+maxnormal22+maxnormal23)/sqrt(((maxnormal21)*(maxnormal21)+(maxnormal22)*(maxnormal22)+(maxnormal23)*(maxnormal23))*3));
```

```
phiN3=acos((maxnormal31+maxnormal32+maxnormal33)/sqrt(((maxnormal31)*(maxnormal31)+(maxnormal32)*(maxnormal32)+(maxnormal33)*(maxnormal33))*3));
```

```
phiB1=acos((maxbrady11+maxbrady12+maxbrady13)/sqrt(((maxbrady11)*(maxbrady11)+(maxbrady12)*(maxbrady12)+(maxbrady13)*(maxbrady13))*3));
```

```
phiB2=acos((maxbrady21+maxbrady22+maxbrady23)/sqrt(((maxbrady21)*(maxbrady21)+(maxbrady22)*(maxbrady22)+(maxbrady23)*(maxbrady23))*3));
```

```
phiB3=acos((maxbrady31+maxbrady32+maxbrady33)/sqrt(((maxbrady31)*(maxbrady31)+(maxbrady32)*(maxbrady32)+(maxbrady33)*(maxbrady33))*3));
```

```
phiT1=acos((maxtachy11+maxtachy12+maxtachy13)/sqrt(((maxtachy11)*(maxtachy11)+(maxtachy12)*(maxtachy12)+(maxtachy13)*(maxtachy13))*3));
```

```
phiT2=acos((maxtachy21+maxtachy22+maxtachy23)/sqrt(((maxtachy21)*(maxtachy21)+(maxtachy22)*(maxtachy22)+(maxtachy23)*(maxtachy23))*3));
```

```
phiT3=acos((maxtachy31+maxtachy32+maxtachy33)/sqrt(((maxtachy31)*(maxtachy31)+(maxtachy32)*(maxtachy32)+(maxtachy33)*(maxtachy33))*3));
```

```
printf("\nphiN1=%f",phiN1);  
printf("\nphiN2=%f",phiN2);  
printf("\nphiN3=%f",phiN3);
```

```
printf("\nphiB1=%f",phiB1);  
printf("\nphiB2=%f",phiB2);  
printf("\nphiB3=%f",phiB3);
```

```
printf("\nphiT1=%f",phiT1);  
printf("\nphiT2=%f",phiT2);  
printf("\nphiT3=%f",phiT3);
```

```
BN1=RN1*sin(phiN1);  
BN2=RN2*sin(phiN2);  
BN3=RN3*sin(phiN3);
```

```
BT1=RT1*sin(phiT1);  
BT2=RT2*sin(phiT2);  
BT3=RT3*sin(phiT3);
```

```
BB1=RB1*sin(phiB1);  
BB2=RB2*sin(phiB2);  
BB3=RB3*sin(phiB3);
```

```
printf("\nBN1=%lf",BN1);  
printf("\nBN2=%lf",BN2);  
printf("\nBN3=%lf",BN3);
```

```
printf("\nBT1=%lf",BT1);
printf("\nBT2=%lf",BT2);
printf("\nBT3=%lf",BT3);
```

```
printf("\nBB1=%lf",BB1);
printf("\nBB2=%lf",BB2);
printf("\nBB3=%lf",BB3);
```

```
BN=(BN1+BN2+BN3)/3;
BB=(BB1+BB2+BB3)/3;
BT=(BT1+BT2+BT3)/3;
```

```
printf("\nBN=%f",BN);
printf("\nBB=%f",BB);
printf("\nBT=%f",BT);
```

```
if((BN<BT)&&(BN<BB))
printf("\nInput ECG signal is a normal signal");
else
{if((BB<BN)&&(BB<BT))
{
printf("\nInput ECG signal is a bradycardia signal");
}
}
else
printf("\nInput ECG signal is a tachycardia signal");
}
```

```
printf("\nHeart Rate=%d",HEART_RATE);
```

```
exit();
}
```

## **REFERENCES**

## REFERENCES

- [1] Dr. Ralf Bousseljot and Dr. Dieter Kreiseler, "ECG Signal Analysis by Pattern Comparison", IEEE Transactions on Computers in Cardiology Vol25, 1998
- [2] Benny P L Lo and Guang-Zhong Yang, "Key Technical Challenges and Current Implementations of Body Sensor Networks", Department of Computing, Imperial College London, UK, September, 2006.
- [3] Dejan Raskovic, Emil Jovanov, Thomas Martin, Shuaib Hanief and Pedro Gelabert , "Energy Profiling of DSP Applications, A Case Study of an Intelligent ECG monitor", October, 2002.
- [4] Borivoje Furht and Alex Perez, "An Adaptive real-time ECG Compression Algorithm with Variable Threshold", IEEE transactions on Biomedical Engg., Vol.35, No.6, June 1988.
- [5] Jalaeddine, Sateh M. S., 1990, "ECG Data Compression Techniques--A Unified Approach", IEEE Transactions on Biomedical Engineering, Vol. 37, No. 4., pp. 329-343
- [6] Suleyman Canan, Yuksel Ozbay and Bekir Karlik, "A method for removal of low varying frequency trend from ECG signal", presented in 2<sup>nd</sup> International Biomedical Engineering Days, 1998.
- [7] John Stevenson, "Code Composer Studio IDE v3 White Paper", Texas Instruments, July 2004.
- [8] *TMS320C6000 Programmer's Guide*, Texas Instruments, August 2002.
- [9] Victor M. dePinto, "Muscle Artifacts noise detectors for ECG signals", Quinton Instrument Company, December 1999.
- [10] <http://www.cis.hut.fi/Opinnot/T-61.6010/s04/local/aksela.ppt>
- [11] <http://medstat.med.utah.edu/kw/ecg/>
- [12] [www.cis.hut.fi/Opinnot/T-61.6010/s04/local/Model\\_based\\_spectral\\_analysis.ppt](http://www.cis.hut.fi/Opinnot/T-61.6010/s04/local/Model_based_spectral_analysis.ppt)
- [13] <http://www.patentstorm.us/patents/5762068.html>
- [14] [medresidents.stanford.edu/TeachingMaterials/EKG%20Basics/EKG%20Basics%20-%20Long.ppt](http://medresidents.stanford.edu/TeachingMaterials/EKG%20Basics/EKG%20Basics%20-%20Long.ppt)