VitalStatis Medical Solutions
School of Engineering Science
Simon Fraser University
Burnaby, BC, V5A 1S6
vitalstatis-med@sfu.ca

December 18, 2002

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, BC, V5A 1S6

Re: ENSC 340 Project Palm™ Bio-Reader Project Post Mortem

Dear Dr. Rawicz,

Attached you will find VitalStatis' *Palm™ Bio-Reader Project Post Mortem*. After the 13-week development period, we have completed the product and have documented the changes and advancements in the document. The post mortem also includes the personal statements from each member.

Please feel free to contact us should you have any questions, comments or concerns. We can be reached by email through vitalstatis-med@sfu.ca or by phoning our CEO at 604-274-1888 (home) or 604-818-7899 (cell). Thank you for your time.

Sincerely,

See-Ho Tsang

See-Ho Tsang
CEO
VitalStatis Medical Solutions

Enclosure: Palm ™ Bio-Reader Project Post Mortem

Palm™ Bio-Reader Project Post Mortem

Project Team: See-Ho Tsang, Bob Wai, Cory Jung, Jason Yu, James Hu, David Poon

Contact Email: vitalstatis-med@sfu.ca

Submitted to: Dr. Andrew Rawicz — ENSC 340
Steve Whitmore — ENSC 305
School of Engineering Science
Simon Fraser University

Date Issued: December 18, 2002

Revision: 1.0

Table of Contents

List of Figures

Glossary

A/D	Analog to digital conversion, conversion of continuous analog signals into digital binary format
Baud rate	Measurement of speed in data transmission, equals one bit per second
bps	Bits per second, unit of measurement of baud rate
DSP	Digital signal processing, computer manipulation of analog signals that have been converted to digital form
EKG	Electrocardiogram, a graphical record of the cardiac cycle
HotSync™	Palm™ Proprietary PC to Palm synchronization software
GUI	Graphical user interface, the screens of the software that the user interacts with
Multiplex	Merging of two or more signals for transmission on the same wire
OS	Operating system, software to interface between application and hardware
Parity	The even or odd quality of the number of 1's or 0's in a binary code
PCB	Printed circuit board, a thin board to which electronic components are fixed by solder
PDA	Personal digital assistant, also called handheld computers
PDB	Palm™ Database file type
PRC	Palm™ Resource file format, file format used for Palm™ PDA applications
Protocol	Standard procedure for regulating data transmission
QRS	QRS complex, name of the electrocardiogram waveform. The various peaks and troughs of a waveform are named P, Q, R, S, and T
RAM	Random Access Memory, a kind of storage device that can be stored or accessed in any order
RS-232	A communication interface standard, also named EIA-232
USART	Universal Synchronous/Asynchronous Receiver/Transmitter,
USB	Universal Serial Bus, a serial communication standard

1. Introduction

Our product, VitalChart, has come a long way in the past four months. It has progressed from an idea, to a design, and finally to reality. This document will describe the current status of VitalChart and reflect upon a number of important issues that VitalStatis has encountered during the product development cycle. Much has changed with VitalChart during this time and the deviations from our design specification, as well as what we have all learned as a result of this project, are included. Our budget, product timeline, and group dynamics will also be assessed to evaluate our performance for this project.

Lastly, a personal statement from each of our group members is included because even though we all worked diligently on the same project, we each had our own unique viewpoints, challenges, and triumphs.

2. Current State of VitalChart

The overall system overview of VitalChart has changed very little since the writing of the design specification. VitalChart measures a patient's EKG signal and respiratory rate utilizing a hardware attachment to a Palm™ PDA. Electrodes and sensors are attached to the patient's body and the signals are sent through the hardware to the Palm™ PDA, where they are displayed on the screen by the software. The system can be divided into three main components: measurement hardware, interface hardware, and Palm™ software. The system block diagram in Figure 1 illustrates how the components in the system interact with each other.

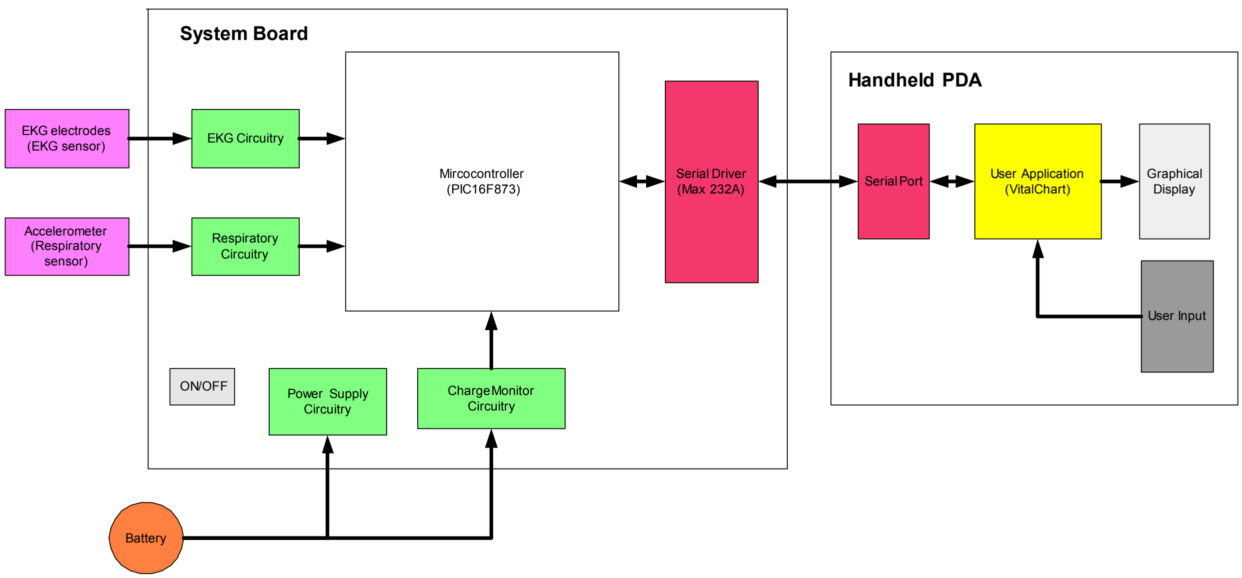

Figure 1: System block diagram

Input to the board is provided by the following components:

- Analog EKG sensor signal
- Analog respiratory sensor signal
- Battery terminals
- On/Off button

Output from the board is:

- RS-232 standard serial signal

The EKG electrodes and respiratory rate sensor continuously measure heart and respiratory signals. The data is captured by the microcontroller, a PIC16F873A, and sent periodically to the Palm™ PDA via the MAX232A communications driver. Once the data is acquired by the PDA, VitalChart software interprets the data and displays the EKG signal and respiratory rate in a GUI. The PDA displays the patient's EKG signal, pulse rate, and respiratory rate in a user-friendly format.

A simple database system has been implemented in the VitalChart software to store patient records on the Palm™ PDA. Each record contains the patient's information, a 15 second sample of the patient's EKG signal, the respiratory rate, and pulse rate. This database can be exported to a PC, where the records can be extracted to view the EKG signal or for interfacing with a full-blown database system.

Most of the circuitry is implemented on PCBs, allowing us to fit the VitalChart product into a plastic box that clips onto the Palm™ PDA and is held in place by Velcro. The battery that powers the entire system is also enclosed in the package. Leads from the EKG and respiratory units come out from the box and can be attached to the patient while viewing the Palm™.

3. Deviations From Design Specification

3.1. Overall System Design

The overall system design did not deviate very much from our design specifications. The handheld PDA and environmental considerations were completed without deviation. Aspects of the system design that were changed or added will be discussed in the following sections.

3.1.1. System Power

The power supply circuitry was absent in our design specification as we did not know what the power requirements of our device would be.

The power supply circuitry of the device consists of two main halves. The first half of the circuitry converts the single input voltage supply into a dual supply for the analog circuitry. The second half takes the input voltage supply and creates a 5V regulated voltage supply to power the digital side of the device. The single input voltage supply is a 9V alkaline battery.

The schematic of the power supply circuitry is shown in Figure 2.

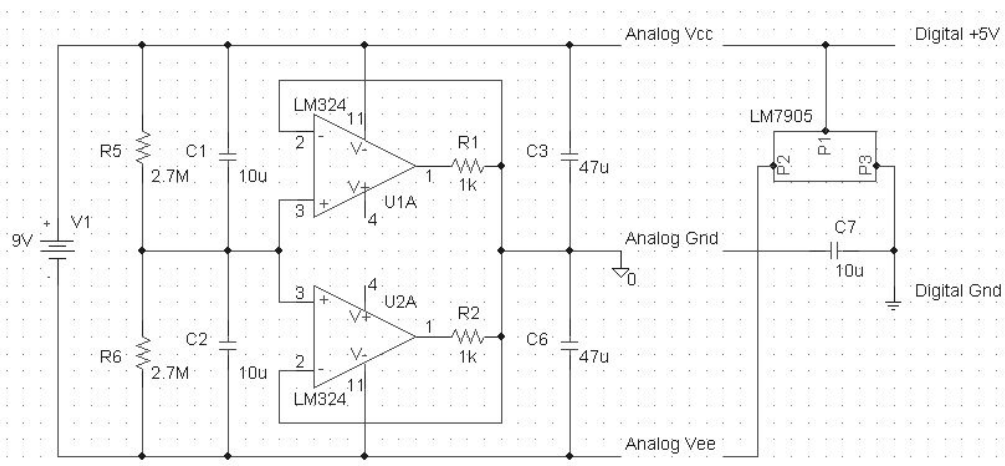

Figure 2: Schematic of Power Circuitry

The circuit is essentially a voltage divider with two voltage buffers. The resistors at the output of the voltage buffers allow the positive and negative side of the dual supplies to draw different amount of currents while still maintaining the ground reference created between the two rails constant. The -5V voltage regulator creates a -5V voltage with respect to the most positive rail. The 5V voltage difference is used to power the digital circuitry. The analog and digital grounds are connected to each other through a capacitor. This is to ensure the grounds of the analog and digital circuitry are connected together, so the digital circuitry will receive clean voltage levels when performing A/D conversion on the analog signals.

The charge monitoring circuitry monitors the voltage level of the battery supply. It compares the voltage level of the analog ground to that of the digital ground using A/D conversion every few seconds. Since the digital ground is always –5V with respect to the positive rail, and the analog ground is halfway between the positive and negative rails, the analog ground would rise in potential relative to the digital ground as the voltage level of the supply lowers. The converted A/D value is compared against a threshold of 1.5V. When the value exceeds the threshold, the low battery indicator LED would turn on.

3.1.2. Printed Circuit Boards

The EKG circuit, respiratory circuit, hardware interface, and part of the power circuit were implemented on PCBs. Protel XP was used to draw the schematics and PCB layout for each circuit. The PCB layouts were printed on transparencies which were placed on top of photosensitive copper boards and exposed to ultraviolet light. After exposing the board to the light for approximately ten minutes, the boards were put into a developer solution to remove the photoresist that was exposed to the light. Next, the boards were etched in ferric chloride solution, and all of the copper that did not have photoresist on it was etched away leaving our traces. Finally, holes were drilled for our components and the boards were populated.

3.1.3. Package

The package for our project is a clear plastic box which we modified for use as our enclosure. All leads, LEDs, and sockets are accessible from the outside while all sensitive circuitry is located safely inside the box. Velcro adheres the Palm™ PDA to the enclosure so that our project and the PDA become essentially a single unit.

3.2. Electrocardiogram

The EKG unit detects the voltage potential between two medical pads, then filters and amplifies the signal. This output signal is the EKG waveform read by the microcontroller.

The EKG unit outputs an analog signal by measuring the voltage potential between two nodes: right chest (RC), and left chest (LC). The placements of these nodes are as shown in Figure 3.

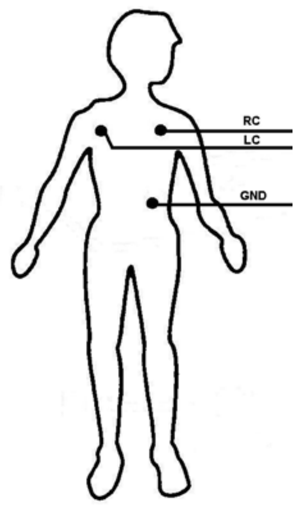

Figure 3: Pad Locations for the EKG

The nodes were originally located at the right arm, left arm, and left ankle in our initial design. As you can see, the final design represents an improvement, as the pad locations are the same as hospital convention.

The potentials between the right chest and left chest with respect to the ground node are amplified and filtered to produce the desired analog signal from microcontroller. The EKG circuit is shown in Figure 4.

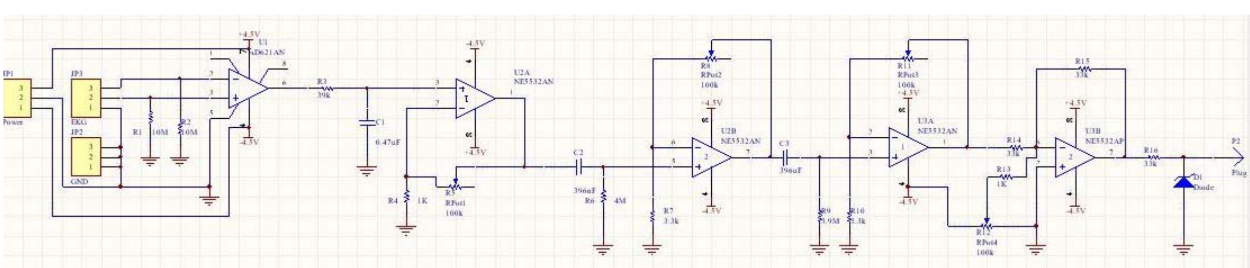

Figure 4: EKG Circuit

U1 (AD621AN) in Figure 4 is an instrumentation amplifier which takes the voltage difference between the inputs. A low-pass filter is connected to remove most of the high frequency noise. This is followed by three level of amplification that provides us with the EKG signal. All voltage amplifiers have variable gain that helps us to locate the signals for different people. The signal is then added with a variable DC offset which keeps the signals in the positive range as required by the A/D module on the microcontroller. A diode is also used to clamp the circuit to keep the signals from going negative.

The variable gain amplifier was not included in the initial design. Since with different pad locations, and different persons, the signals can vary in both amplitude and DC offset. Therefore, the variable gain amplifiers were added to help alleviate the problem. But as a result, the unit will sometimes require tuning to locate the appropriate signal.

3.3. Respiratory Measuring Device

The finalized respiratory measuring device uses an ADXL202E accelerometer which is mounted on pieces of foam with a cross rod underneath as shown in Figure 5.

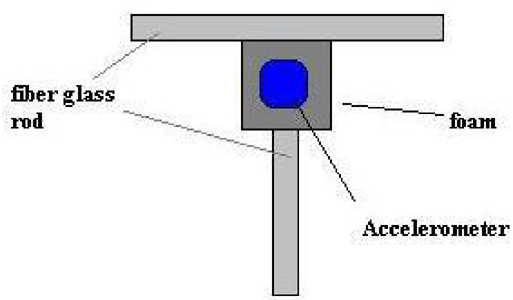

Figure 5: Respiratory Sensor

The design of the respiratory sensor was changed from what was documented in the design specification. After experimenting with the previous sensor design, we discovered placing the accelerometer on the abdomen without additional attachment did not generate a signal large enough for calculation of the respiratory rate. Thus, fiberglass rods connected in a T-shape is used to increase the tilting angle when the patient's abdomen expands and contracts when the patient takes a breath. This accelerometer is a good tilting sensor especially when it operates perpendicular to gravity. However, size and weight of the sensor increase proportionally because of the extra attachments. The length and the weight of the sensor are longer than 10cm and heavier than 150grams than the desired values in our design specification.

The output signal of the accelerometer is generated by pin Y-Filt (vertical movement). Although wires connecting between the sensor and the circuitry are not shielded, the signal is not significantly affected by noise. The circuitry was changed from the original design because a differential amplifying circuit was not useful in measuring the respiratory rate. Figure 6 shows the schematic of the respiratory measuring device.

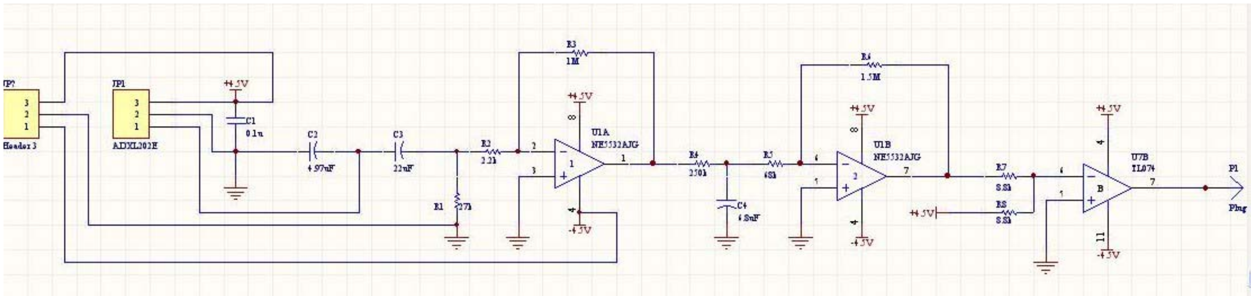

Figure 6: Respiratory Measuring Device Circuit

A high-pass filter is used to remove the DC offset of the input sensor signal. Afterwards, the signal passes through two amplifier stages. Between each stage of the amplifying circuits is a low-pass filter utilized to remove noise for a clearer output signal. A DC offset is then added to the amplified signal to keep the output above 0V. A clamping diode is also used to avoid a negative output voltage. The output voltage range is from 0V to 4.5V which is our desired range.

3.4. Hardware Interface

3.4.1. Overview

The hardware interface is located between the EKG and respiratory rate sensor circuitry and the Palm™ PDA. Its functions are to convert the analog sensor outputs to digital, perform some simple signal processing algorithms on the data, and send it to the Palm™ PDA. RS-232 serial communication protocol is used for data transmission. The main components used in the hardware interface are identical to the design specification, namely we used a PIC16F873 microcontroller and a MAX232 RS-232 Driver/Receiver. Figure 7 shows the schematic of the hardware interface.

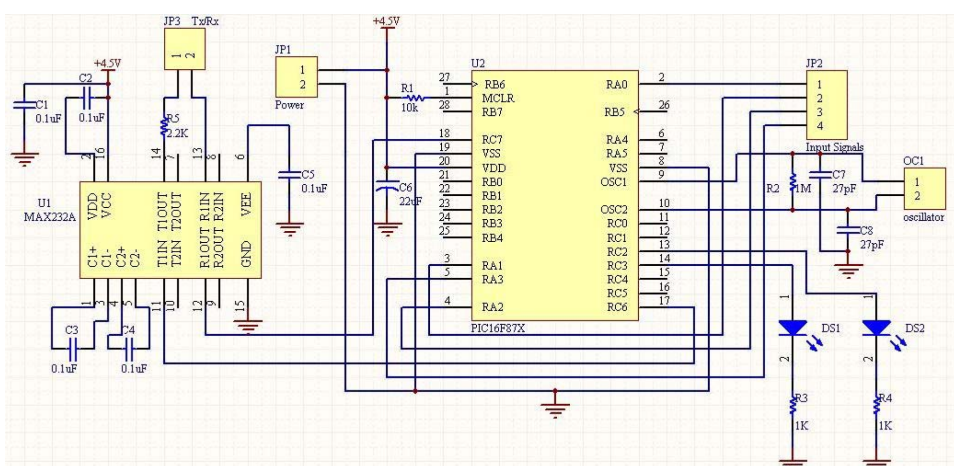

Figure 7: Schematic of the Hardware Interface

The most significant changes compared to the original design occurred in the RS-232 communication component of the interface, since we did not know the performance of the serial port on the Palm™ PDA before we actually built the circuit and began transferring data. We ran into a number of problems due to synchronization between the Palm™ PDA and the hardware interface, especially because RS-232 is an asynchronous transmission protocol. The modifications we made to the initial design allowed us to meet or even exceed all the functional specification of the device. The changes we made from the design specification and possible improvements are detailed in the following sections.

3.4.2. System Connection

The signal connections between the hardware interface and the Palm™ PDA serial port are the same as initially designed, except the RTS/CTS flow control signals were removed in the communication bus. As discussed in the design specification, these signals were not being used in our design, but we planned to connect them in case we later find a need for them. However, after further research, we discovered that the operation of these signals is not standardized, and every device uses them differently. Furthermore, our implementation of the communication scheme ensures the device will operate without the flow control signals. Hence they were removed in the final design. Figure 8 illustrates the modified signal connections.

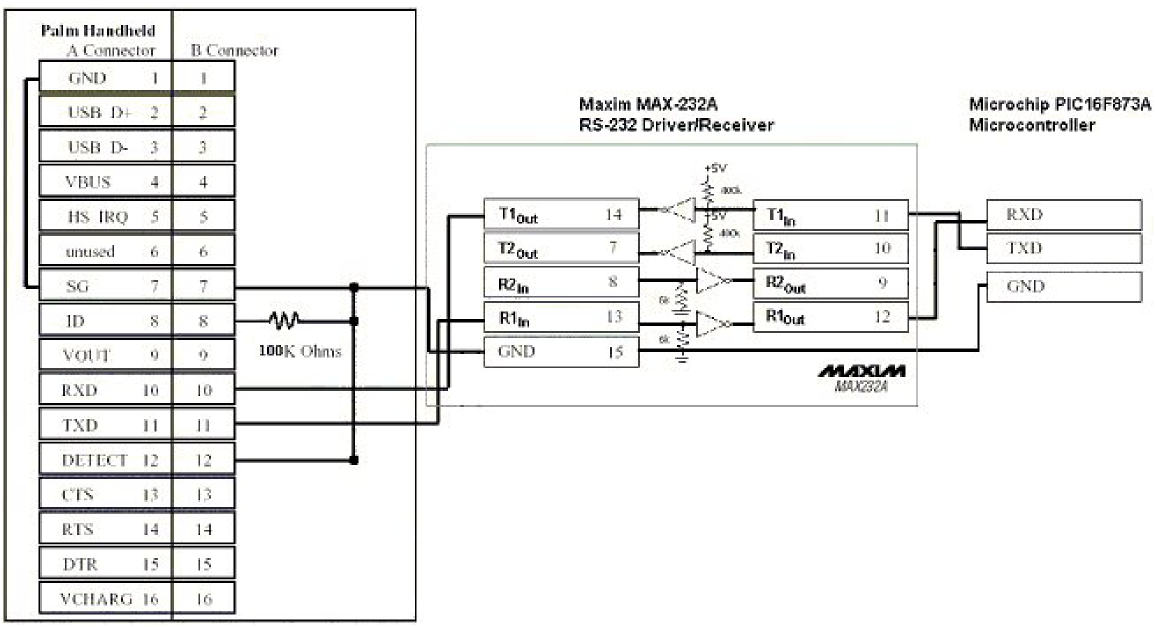

Figure 8: System Connection Interface

3.4.3. Communication Methodology

The most significant change in the hardware interface from design specification is the communication between the interface and the Palm™ PDA was made bi-directional instead of unidirectional, from the interface to the Palm™ PDA. The original design had the microcontroller continuously transmitting the acquired data samples at fixed intervals so that the Palm™ PDA could have its serial port open all the time to continuously receive data. In the final design, data samples are only transmitted to the PDA when they are requested. This modification was the result of several difficulties we encountered in transmitting the data samples to the PDA.

We realized shortly after we began programming the Palm™ PDA communication code that we could not continuously open the serial port on the PDA because it drains a significant amount of the PDA's battery, greatly shortening lifespan of the PDA on a single charge, and as a result, our project's operational period. Therefore, we needed to periodically open and close the serial port. This complication led us to synchronization problems of when to open and close the port so that the PDA would have its serial port open when data is transmitted from the hardware interface, and hence, receive the data correctly. After much thought and experimentation with different schemes of synchronization, we decided to instead make the PDA the active side of the data transaction with a request based communication scheme.

In our revised communication methodology, the Palm™ PDA software requests for data periodically, leaving the serial port open for a short interval after the request. When the hardware interface detects the transmission request, it immediately sends the requested data back to the Palm™ PDA which can then close the serial port and process the data. Extra resources required for this communication scheme is we needed a relatively large buffer on the hardware interface to buffer the data samples it acquired while waiting for a transmission request. This is unavoidable since the Palm™ PDA cannot send requests very frequently or else the operating system has no time to service other software functions and the whole operating system becomes unresponsive.

3.4.4. Microcontroller Configuration and Firmware

The PIC16F873 microcontroller in the hardware interface operates on a 20MHz crystal, which is the maximum operating frequency of the device. The high frequency is to ensure the EKG signal transmitted has a high enough resolution and miscellaneous operations on the microcontroller can be completed fast enough such that the EKG signal obtained will be as accurate as possible. To accomplish this, we also turned off the watchdog timer on the device.

The differences in the firmware compared to the design specification were mostly made to accommodate the new communication scheme. The complete assembly source code of the microcontroller firmware can be found in Appendix A.

3.4.5. A/D Conversion

Early in our integration, we decided that we wanted to increase the sampling rate of the EKG signal to allow for better resolution for display on the Palm™ PDA. This would allow the user to better see the EKG waveform. A timer is used to set a flag which causes the microcontroller to perform an A/D conversion subroutine. To increase the sampling rate, all we had to do was shorten the amount of time it takes for the timer to go off to 0.01 seconds.

For the A/D conversion we need to sample from the EKG and respiratory unit at the same time, so we had to implement multiplexing to our sampling. The sampling period of the EKG is 0.01 seconds and the sampling period of the respiratory unit is 0.05 seconds. In the A/D conversion subroutine, a sample from the EKG is read and saved in the EKG data buffer and a counter is used to determine if a sample from the respiratory unit should be read. Every fifth time the subroutine is run, the A/D is reconfigured to read data from the respiratory unit and a sample is read and saved into the respiratory data buffer. Additional DSP algorithms are performed periodically in the A/D subroutine and will be discussed later in the next section.

Overall, functionality of the A/D conversion has not been altered, however the increased sampling rate of 100 Hz for the EKG data meant that we needed to increase the speed of the A/D conversion. We accomplished this by setting the A/D conversion clock to the fastest possible rate, 10 MHz.

3.4.6. Respiratory Rate Digital Signal Processing

The DSP algorithm is performed on the microcontroller to calculate the respiratory rate to be sent to the Palm™ PDA. The simple assembly code performs DSP on the breathing data acquired from the A/D conversions. The algorithm is performed every 0.2 seconds and involves averaging every fourth sample and saving the average into a data buffer reserved for respiratory rate data. The DSP utilizes the entire buffer, which contains 12 seconds of data, to calculate the breathing rate. In our initial design, we counted the number of peaks after a minimum was found. In order to make the system more robust, we added threshold values to search for the peaks and the troughs. Once the number of peaks is found, we send the number of breaths the user has taken in 12 seconds. The Palm™ PDA multiplies this number by five and displays a value that represents the number of breaths the user has taken in a minute.

3.4.7. RS-232 Communication

The configuration we used in our final design differs from our design specification in that it did not utilize the parity bit of the RS-232 communication protocol. Since the hardware interface is directly wired to the Palm™ PDA's serial port, the chances of the signal becoming distorted during transmission through a few centimeters of wires is minimal, hence parity bit error checking is not necessary.

The baud rate of the RS-232 communication between the hardware interface and the Palm™ PDA is 62,500kbps, up from our initial design of 19,200kbps. This non-standard baud rate is the result of error in the microcontroller's baud rate generator when dividing down the clock frequency.

The configuration of the RS-232 communication used in the final design is 8 data bits, no parity bit, and 1 stop bit.

3.5. VitalChart Software

3.5.1. Software to Hardware Communication Interface (SHCI)

All minimum requirements as specified in the functional specification have been met. In addition to the minimum requirements, the software is able to discriminate between the proper peripherals and other peripherals upon connection of the hardware to detect abnormal termination of data transfer and indicate the status to user on the screen.

The design of the software has deviated slightly from the original. The software currently operates at a baud rate of 62,500 bps, more than three times faster than the original proposed baud rate of 19,200 bps. This increase in speed increases the resolution of the EKG and respiratory data and response time between the Palm™ PDA and external circuits.

Another modification is that the Palm™ can now send data to the microcontroller. The software to hardware communication interface (SCHI) was designed using the Palm™ 4.0 SDK serial manager. This manager communicates with the microcontroller by sending a start byte (0x05) and capturing up to 26 bytes of data every 0.1 seconds. During this process, the manager asserts control of the operating system so a timeout sequence is required. In the design, the manager allocates 0.02 seconds to retrieve the entire data stream before it times out. Using this timeout sequence, the application can determine whether not the hardware device is connected. The SCHI also performs the appropriate decoding of the data stream to differentiate between EKG, respiration data, and null data.

3.5.2. Database

The database section of the software is implemented as one of the ideal requirements from our functional specification. The VitalChart database is implemented using the Palm™ PDA's PDB format. The user can record the patient's name, age, and 15 seconds of EKG data. A unique ID for each patient will be generated automatically whenever a new record is created and the user can stop the recording sequence anytime during the 15 seconds recording cycle by pressing the stop button. Records will be retained in the PDA's memory unless the user explicitly deletes the database file. Whenever the user uses the HotSync operation with their desktop PC, the database file will be automatically transferred to the desktop. A database conversion utility has been programmed so that user can extract all records contained in the PDB database file to a text file. The EKG plot can then be displayed on the desktop PC using software such as Excel and Matlab and then printed. The PDB database conversion utility is programmed using C and Perl programming languages.

3.5.3. Graphical User Interface Design

3.5.3.1. Overview

The GUI has met all the minimum functional requirements and we even implemented several ideal requirements from our functional specification. The ideal requirements we added include the menu function for interacting with the database, a check box to initiate a zoom feature, and a heartbeat detection indicator with sound. The database interface, once initiated by the menu, allows a user to label and record patient records.

As well as meeting the minimal requirements, several areas of the design specification were modified to improve the overall usability.

3.5.3.2. Main Interface Window

The following figure shows the application start-up form that was implemented.

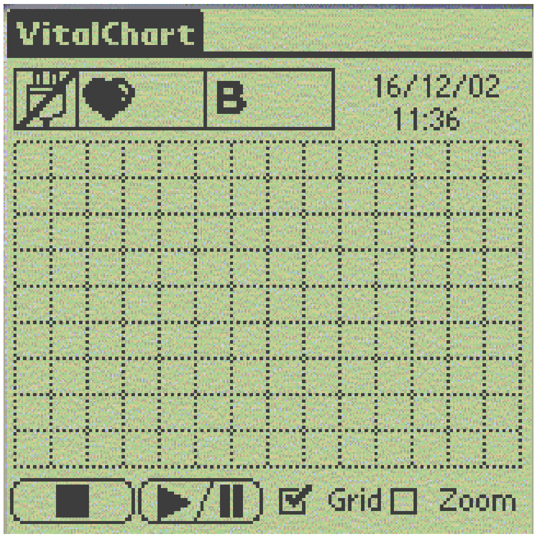

Figure 9: The VitalChart Interface

To improve the usability, the hardware connection indicator has been implemented using an electrical plug icon instead of a lighting bolt. The reason for this change is that a lightning bolt cannot be resolved very well due to the low resolution of the Palm ™ PDA display and became unrecognizable. In addition to this change, the *Grid On* button has been modified into a simple check box for user simplicity where the check mark indicates whether or not the grid is activated. Furthermore, a *Zoom* check box is now included to toggle a newly implemented 2x zoom feature which allows for a higher resolution of the QRST complex to be displayed. The details of this feature will be discussed in the next section.

3.5.3.3. Database User Interface

In addition to the modifications made to the GUI, a second, Database User Interface (DUI) was added. The following figures show the menu used to activate the DUI and then the DUI itself.

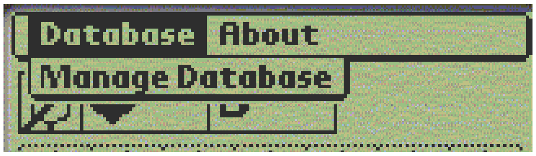

Figure 10: The VitalChart Menu

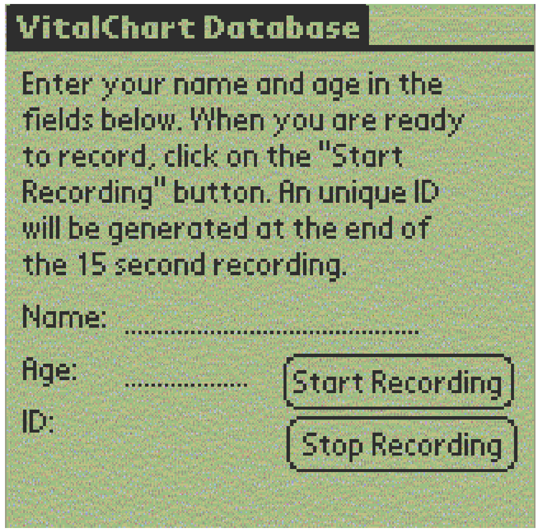

Figure 11: The VitalChart Database User Interface

The DUI has two fields for entering the name and age of the patient whose vitals are to be recorded. Once the user taps on the *Start Recording* button, the application will record 15 seconds of continuous EKG data from the patient and store it in the *VitalChartDB.pdb* file. The *Stop Recording* button allows the user to quit recording immediately and store less data into the database if they wish. By clicking on the menu bar, the user can exit the DUI and return to the GUI.

3.5.3.4. About Menu

The menu of the VitalChart application allows for the activation of the about form as shown in Figure 12.

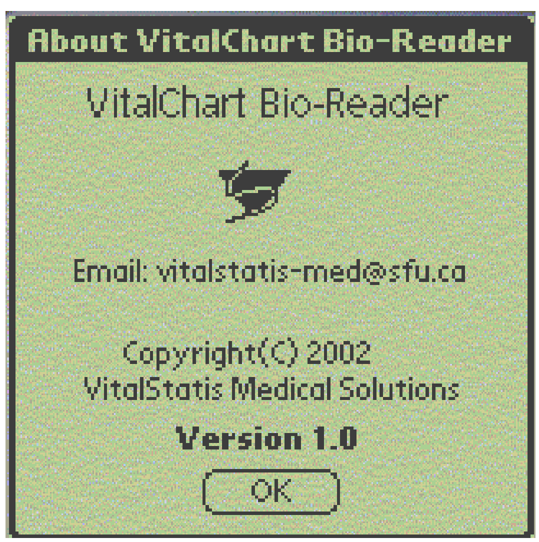

Figure 12: The About VitalChart Form

The about form has been modified from the design specification such that the size of the form is increased. The address of the company, as well as the serial number for the software, have been removed. The serial number and company address were moved because they have not yet been decided upon.

3.5.4. Graphics Engine Design

3.5.4.1. Overview

The VitalChart Graphics Engine (VGE) has met the minimal functional specification and has been modified to incorporate extra features including a zoom feature and performance improvements.

For the basic design, the VGE follows the design flowchart as described in the design specification except for one change. The graphics buffer has now been removed and the graphics engine updates the display immediately upon gathering the data. The reason for this change is that the graphics buffer did very little in improving the display of the data points but caused a great deal of processing lag between each point. Therefore, this step was removed in favor of opting for a quicker plot time.

3.5.4.2. System Timing Flow

The greatest obstacle in this section was caused by buffer overrun and underrun issues. While communicating with the hardware, the communications manager needs to collect data much faster than the plot in order to provide real-time operation. Although the minimal requirement of plotting the data within one second of the real-world time was met, plotting the data much faster was highly desired since our device targets a medical application. After several trials with "real" EKG data (the software was developed using waveforms generated from the function generator). It appeared that the sampling rate at 0.05s was not high enough to resolve the entire ST section of the QRST complex. Therefore, the sampling rate was increased to 0.01s and the bottleneck of our system became the slow plotting speed of the Palm ™ PDA. In order to resolve this problem, the data rate was increased to 62,500kbps as stated in Section 3.4.7. Using the higher data rate, more data can be gathered at once on the hardware side and the communications manager can request data less frequently and still acquire more data. This allowed more computing time for the data conversion, peak locator, heart rate calculation and plotting. The increase in resources for these functions allowed the Palm™ OS to plot data within 0.2 seconds of when the data is received at a 100 Hz sampling rate.

In order to protect against buffer overrun and underrun at this data rate, the system utilizes a large buffer for storing data and initiates a buffer pointer synchronization every after plotting an entire screen. Once the pointers synchronize, the plotting is delayed for 10 samples to allow the data buffer to fill ahead of the plot. This prevents buffer overrun and the synchronization prevents underrun.

3.5.4.3. Peak Locator and Heart Rate Calculation

In order to detect the peaks of the QRST complex, calculate the heart rate, and toggle the heartbeat indicator, a peak detection algorithm is utilized. This algorithm takes the difference between each data sample as they enter the plotting algorithm. When the difference is lower than a set threshold, a trigger flag is set, and the algorithm looks for a difference that is higher than a set threshold while the trigger is set and records the location of the peak. Once the plot reaches the end of the display, the heart rate is calculated by averaging all the plot locations and converting it to beats per minute.

3.5.4.4. Zoom

In addition to the minimum requirements, a zoom feature has been implemented for the Palm™ PDA display to resolve the QRST complex. At the regular resolution, the system displays several QRST complexes at a once on each screen wipe whereas when the system is zoomed in, the QRST complex is displayed at a much higher resolution with less samples per screen.

3.6. Future Plans

3.6.1. Hardware Interface

The most valuable improvement to the hardware interface, as with the other hardware components of the device, would be to lower the power requirements. We found low voltage substitutes for both the microcontroller and the RS-232 Driver/Receiver. The PIC16LF873 is a low voltage version of the same microcontroller that can operate at a voltage supply down to 2.0V. The tradeoff for the low voltage is a decreased operational frequency, which is a maximum of 4MHz. The MAX3224E is another RS-232 Driver/Receiver that can operate at supply voltages of 3.3V to 5V. It also features an autoshutdown mode that can shut down the chip when no traffic occurs on the signal lines. We hope to incorporate these components into our device in a future revision of the design.

We found during our testing that the current bottleneck in the microcontroller firmware is the serial data transmission, as many bytes are being sent at a time in a loop. Therefore, to achieve the lower operating frequency, the algorithm must be optimized to increase its speed. Lastly, we can utilize a smaller microcontroller in the design, as many output pins as well as much of the program memory in our PIC16F873 microcontroller remains unused. This improvement can cut down on both the size and cost of the device, two important factors in a portable instrument design.

3.6.2. Database

An important area to expand in the future will be the development of a web-based SQL server system for the data collected by the VitalChart. In this system, hospital personnel will be able to synchronize the data collected on to the PDA with the central database every time the Palm™ PDA is HotSync'ed. This database of patient information and records would eventually be developed such that it can be accessible by other health care professionals by logging on to the system and downloading the data. This ability is a vital part to increase the value of our system by simplifying patient charting and information sharing. In addition, we will develop VitalChart to use USB connectivity such that it will become compatible will all PDAs. Furthermore, the implementation of EKG recognition and diagnostic software may be developed.

4. Budget

The following table shows the final budget for our project. The actual costs are not exact because when we bought electrical components, we often used the same parts in more than one module.

Table 1: Final Project Budget

Module	Estimated Cost	Actual Cost
EKG	$150	$50
Respiratory Rate	$225	$20
Hardware Interface	$225	$140
Palm™	$345	$340
Development Software	$0	$0
Power Supply	$20	$20
Miscellaneous	$100	$220
Total Expenses	$1165	$790

As can be seen in Table 1, we were able to stay within our total budget and most sections of the project were completed below budget. The reason we overestimated our costs for the EKG module, breathing rate monitor, and interface is because we were expecting to have to buy all of our parts. In actuality, we managed to obtain a significant number of the parts we needed through free samples from various chip manufacturers.

Our miscellaneous costs were over our expected budget mostly because we did not budget for the materials we needed to make our PCBs. We needed to buy chemicals for developing and etching, photosensitive boards, a UV lamp to expose the boards, and a drill press stand for the dremel we used to drill the holes in our PCBs. We are satisfied that we spent our money wisely and managed to complete our project under budget.

5. Project Timeline

Our initial project timeline is illustrated below in Figure 13.

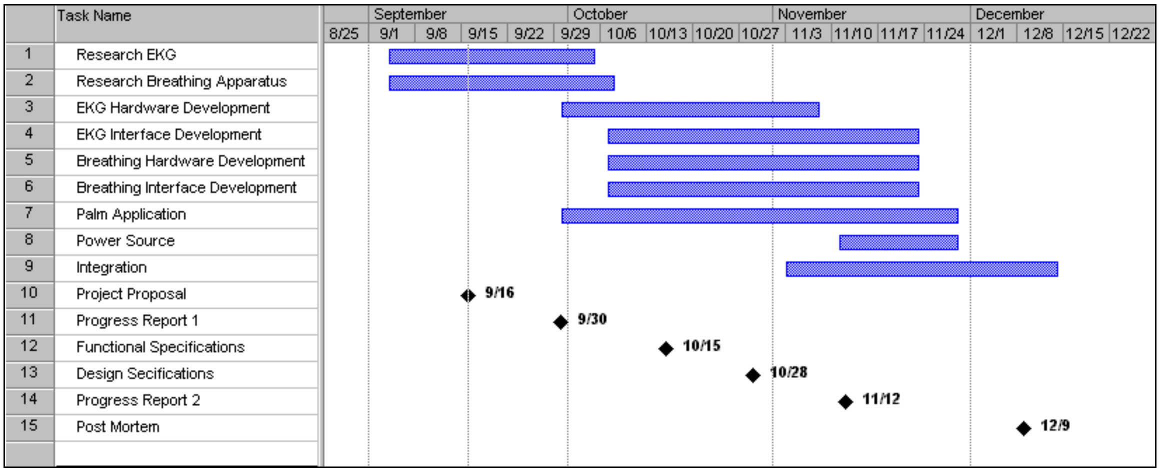

Figure 13: Initial Gant Chart for VitalChart Development

We managed to meet all of our milestones except for the post mortem which was finished just before our project demonstration on Dec. 17th. While we managed to start almost all of our tasks when we planned, we underestimated the time required for almost every task. Research for the EKG and breathing apparatus was not completed until the end of October, when we had to write our design specifications. This occurred because when we started developing each module, additional problems or concerns would arise causing us to do more research to gain greater insight into possible solutions. Once we had defined what we wanted to do in the design specifications, we had researched most of the relevant information we would need.

Development of the EKG and breathing apparatus started as scheduled, but also unfortunately took longer than expected. While the major development of the EKG was completed by the end of November and the breathing apparatus by the beginning of December, we did not freeze our design until the end of the second week of December. The reason the development was extended was because we had to make changes to each module during integration. We should have anticipated that integration would involve continued development of both modules.

The actual development of the EKG and breathing apparatus interface was delayed until the beginning of November, however we began research of the interface at the time scheduled for the start of interface development. Interface development was similar to the development of the EKG and breathing apparatus in that the major development was finished by the end of November, but minor changes were made until the second week of December. The reasoning for the extended development is the same and, again, something we should have anticipated.

Like the interface, the Palm™ VitalChart software application development was delayed while research was done. Actual development of the software commenced at the beginning of November and continued until the second week of December, for the same reasons as the development of the EKG, breathing apparatus, and interface. However, an addition reason for the extended development of the VitalChart software was implementation of a database which was a function not part of the minimum requirements of our project in our design specification.

Power source development was delayed until December because it was not part of our minimum requirements so the other tasks took precedence over it. Our expected development time was about right for the power source and was completed during the second week of December.

Integration was delayed for two weeks as the other modules had not been developed to the point where they could be integrated. We did try to integrate as soon as possible as it was important for us to know that we would be able to communicate from one module to the other. To this end, we first performed an initial integration as soon as the individual modules had enough functionality that we can try combine them together. This was done to foresee any particularly difficult problems that we may encounter during the process, and solve them before the final project is to be combined. System integration was completed on schedule, just in time for our demonstration.

Overall, we underestimated how long each task would take, but we also did not incorporate two other factors into our scheduling. The first factor was documentation, which took up most of the time we allocated for ENSC 340/305 for the first two months of our project. We thought we would have more time to develop our project so that we would not have to change as much during integration. We might have had that time, if not for the other factor we underestimated: the dreaded sixth engineering semester. As each of our six group members was taking 17 or more credits, we simply could not find as much time to work on our project as we wanted. This resulted delays and the extension of our research and development times.

6. Group Dynamics

Without a doubt, any successes or failures we have had were impacted by our group dynamics, the most important aspects being organization and communication. Our organization was established from Day 1 when we began forming our team in the summer of 2002, three months before the start of ENSC 340/305. We were a group of six talented engineering students who had all worked together before and who knew what to expect from each other. This was a very significant because we knew what the strengths and weaknesses of each group member was and were able to organize our workload so that each member could work effectively. We began having meetings in the summer, keeping meeting minutes and conducting ourselves like a startup company would, to ensure that each member would have an input on any decision we made.

Once our project official began at the beginning of September, we had weekly meetings on Wednesdays to update everyone on the status of each other and to make sure everyone had a clear idea of what our goals were. Our meeting minutes were being posted on our group website so that everyone could see if they were on track. While these meetings were a little long near the beginning of the semester, we quickly learned how to keep the meetings going at a good pace. These meetings were particularly effective when we needed to brainstorm ideas or debate management-type decision, such as choosing a company name and establishing the scope of our project. However, as the semester progressed we found these meetings became less valuable as we more often had engineering decisions to discuss where only a few group members were needed at a time. Near the end of October we had split ourselves into smaller workgroups and no longer had weekly meetings. Informal meetings were scheduled with the relevant group members to avoid wasting the time of the other group members. Our communication worked out quite well due to the fact that we are all friends and see each other on a consistent basis. Had this situation been different, we would have adjusted our methods of communication.

For the most part, we managed to cooperate fairly well with each other. While there were some heated conversations, they were not a result of any personality conflicts and we are all still friends with each other. In fact, we are probably a stronger group now than when we started, having suffered and succeeded together. This project has been a valuable experience for all of us; each of us has learned how to be a more effective team member and we even still like each other.

7. Personal Challenges

See-Ho Tsang – Chief Executive Officer

This course has been an extremely challenging but exciting experience for me. I enjoyed developing our product throughout the 13+ weeks but not-so-enjoyed sleeping only 5 and half hours in a 110 hr period. One thing that I will take away from this experience is the confidence that we can design and build a product that is definitely non-trivial and difficult. I now feel that we can tackle anything.

For the development of VitalChart, I was assigned to build the Graphics Interface and the Graphical User Interface on the Palm™ side. For these tasks, I had to develop my skills as a Palm™ OS programmer and think up some creative DSP algorithms to extract the heart rate from the EKG waveform. However, the most difficult assignment was to create a system that would plot the data in real-time. This proved to be the biggest headache since the Palm™ m500 has limited capabilities. This task brought on the occasional profanity coming from the back of lab when things started to break down. However, once the program came together, I felt the greatest feeling of satisfaction. For the future, I will probably use a more powerful PDA to achieve our goal.

Another job I faced was assuming the role of CEO and help my team believe in my idea during the planning process, and continue to believe when times were tough where it appeared that this project might not be realizable. In addition to this, I also had to research the market and make contact with nurses and other health care professionals to see whether or not the product is useful.

If I were to do this project again, I would still work with the same group. When things were tough we were all there toughing it out – as one. The one thing that kept our group as together as it did was the fact that we mostly worked side by side. Even for the other courses, our group would remain as one team. For this semester, we lived as one team in all the courses not just 340. Therefore, the amount of understanding and help that we gave each other occurred right across the board.

Since we have decided to take this project to market, I think our team will keep a long lasting friendship.

Cory Jung – VP Engineering

ENSC 340/305 has been both a pleasure and a pain-in-the-ass for me. On one hand, I am happy with what I have learned from this course and I think it has been a valuable experience for me. On the other hand, I have had to work extremely long and hard hours under the influence of sleep deprivation, the health effects of which have probably shortened my life span.

The technical tasks I was assigned were the A/D conversion and subsequent multiplexing of the sampling and producing the PCBs. While not trivial, these tasks were not especially challenging to me. I like playing around with microcontrollers and was able to draw upon past co-op experiences. I was well equipped to face these tasks. In fact, my most challenging tasks were not technical at all.

The hardest task I was faced with was my role as VP Engineering. My tasks were to tasks to each group member and generally act as the project manager. This proved to be quite difficult for me as I had to be the one to make some of the tougher decisions. This meant increased responsibility and as a result, increased stress. While I am generally pretty good at making decisions, my inexperience in managing a project of this size was a challenging one. This role has really allowed me to explore a different facet of project work, one I am glad to have had. The experience I have gained is valuable and I hope to get another opportunity to work as a project manager.

I am very grateful that I got to work with this great group of guys on this project. While it has been painful at times, there have been moments I will never forget. After getting through all of the obstacles that we have encountered in this project, I feel a real sense of accomplishment and camaraderie. I guess in the end, despite the perhaps shortened lifespan, it was worth it.

Bob Wai - VP Finance

From ENSC 305/340, I learned the importance of dividing workload evenly for a big project and how to stay awake for 48 hours. Sleeping becomes optional after taking this project course. Although I suffered in this course, I believe it is a valuable experience for me.

In the process of developing the respiratory circuit, the magnitude of the output signal was the major obstacle because a ±2g accelerometer is not very sensitive to small vertical movement. The magnitude of the sensor signal was almost the same level as the noise, which was not a useful signal. To choose a suitable accelerometer for our respiratory sensor was difficult because the current smallest g accelerometer is still not sensitive enough to detect the respiratory movement and the z-axis accelerometer is not available in the market. As a result, we spent a long time doing research on all possible respiratory sensors. I also face difficulty when making PCBs. The pads and traces on the board could be ruined fairly easily since they are thin and small. Since the PCB is small, it is complicated to troubleshoot the circuit when there are problems.

I learned to use Protel™ Design Explorer to draw schematics and layout PCBs, as well how to make a single layer PCB. This was a valuable experience for me because it could help me to find a job in PCB industry and will help if I make PCBs in future projects. I also understand how an accelerometer operates and how it is used to detect movement and rotation.

If I could redo this project, I would build an oxygen saturation circuit to measure the respiratory rate because it gives a more accurate respiratory signal for further analysis.

Everyone contributed greatly in this project to achieve the goal of completing our project.

Jason Yu – VP Communications

During the project development cycle, I was mostly occupied in developing the hardware interface together with Cory. The hardware interface includes the microcontroller, RS-232 Driver/Receiver circuitry, and the microcontroller firmware. Within the firmware, my responsibility was the communication between the Palm™ PDA and the hardware interface. I felt that particular section is one of the trickier parts of the project and we ran into many difficulties throughout the project development. I discovered that synchronizing communication between multiple devices (in this case, only two) is quite a difficult task, and a robust methodology is required to ensure good results. I enjoyed developing and experimenting with the different communication methodologies to find one that worked well for our requirements and participating in a larger scale project development.

Having seen through the development of a complete product from beginning to end, I learned much about working on a big project in a team environment. Frequently we would have questions about each other's work, and it is very difficult to continue if we cannot get quick responses from team members. Collaboration between the hardware interface personnel with the Palm™ PDA software developers was especially critical to work on the communication. Any changes to the communication required changes on both ends, and the modifications had to be evaluated to see if any further tweaks were needed.

Working in a small group in a closed environment, under stress, and under sleep deprivation was also an interesting experience that I gathered during the last 72 hours before the project demonstration. I am happy to say I did not have to sleep in the lab, although most of my team members did, in sleeping bags and inflatable mats on the floor. When everybody in the group has had less than 5 hours of sleep within a 48-hour period, we definitely have to be more tolerant towards each other.

David Poon – Senior Engineer

I have been quite amazed at how our project has come together during the past 13 weeks. We have completed and met every minimal requirement and even had extra time to implement some extra features such as database and PCB. Our software, firmware, and hardware integrated very smoothly as the project progressed, even though the software and the hardware parts are done independently.

This ENSC 340 project provided me an opportunity to practically apply the skills that I have acquired in the past few years. The C programming skill that I have developed helped me tremendously in creating the VitalChart Bio-Reader software. During the project, I programmed the serial communication interface for the Palm™ device, a database system for storing patient's EKG data and a desktop database conversion utility for extracting the Palm™ database into a text file. I have learned the project engineering cycle, from project idea brainstorming, project planning, writing documentation, and doing a presentation. I have also learned how to work with other people to create software, how to stay awake for a few days with minimum sleep, and how to control my temper when I get cranky or annoyed by other members. I have also learned the proper research methodology and how to utilize online resources when I get stuck on software problems.

Finally, I enjoyed working with my fellow teammates and hopefully we can make VitalChart Bio-Reader a successful commercial product.

James Hu – Senior Engineer

ENSC 340/305 has been a great pleasure for me. I have a personality trait that never lets me give up so the project seemed like a big challenge to it. Throughout the project, these following skills were constantly required: application of the knowledge and experience I have gained, co-operation between team members, and problem solving.

Application of my knowledge was the most important for my duty. Since I am the senior engineer in the group, the main part of my job was to design and fix circuits, program in assembly, and other technical tasks. The EKG design was the main part of my duty. Applications of filters and amplifiers were the very core design of the EKG unit. I also helped out with programming the hardware interface where I designed and wrote the respiratory DSP code.

Problem solving was also very important in my role; I learned a lot as our product was developed. For example, the EKG design was developed and tested at the beginning of the cycle, however, several problems were encountered, such as the voltage of each person is different and the microcontroller can only input positive voltages. These were all solved by the engineering intuition I have earned as I study in Simon Fraser engineering.

Co-operation between team members was the key to our success. Even though there were a lot of arguments and disagreements, we were always working toward one important goal: the completion of the project. As the deadline approached, the amount of sleep we got was close to none. At this time, I learned it's important to keep an open mind and make sure we work in a humorous environment. I tried to keep all of us going by telling sick jokes once a while. Of course, being friends with all the group members kept me from crossing the line and pissing someone off.

Finally, I would like to thank all my group members for being able to stand me and being such a great bunch with diversity and amazing talents. We've done great, guys!!!

8. Conclusion

There is no doubt that everyone in our group is extremely proud of what we have accomplished for VitalChart. While the process has not been without frustration or suffering, it has also been an on-going learning experience, and in the end, a rewarding one. We have managed to accomplish the product goals we set out in our specifications and more.

While we certainly encountered our share of technical problems, we also contended with group dynamics, organization, and project management difficulties. Despite these obstacles, our group of six engineers has learned from our mistakes and we are confident we could continue to improve VitalChart, whatever troubles should arise.

9. References

ePanorama.net, (2001) “Power supplies.” http://www.epanorama.net/links/psu.html (Accessed December 14, 2002).

Elliot Sound Products (1999) “Project 43 – Simple DC Adapter Power Supply.” http://sound.westhost.com/project43.htm (Accessed December 14, 2002).

EDN Access (1997) “EDN Access—07.17.97 Op amp makes precise 9V-battery splitter.” http://www.e-insite.net/ednmag/archives/1997/071797/15di_05.htm (Accessed December 14, 2002).

Appendix A: PIC Microcontroller Firmware Source Code

```
;**********************************************************************
;   This file is a basic code template for assembly code generation   *
;   on the PICmicro PIC16F873. This file contains the basic code      *
;   building blocks to build upon.                                    *
;                                                                     *
;   If interrupts are not used all code presented between the ORG     *
;   0x004 directive and the label main can be removed. In addition    *
;   the variable assignments for 'w_temp' and 'status_temp' can       *
;   be removed.                                                       *
;                                                                     *
;   Refer to the MPASM User's Guide for additional information on     *
;   features of the assembler (Document DS33014).                     *
;                                                                     *
;   Refer to the respective PICmicro data sheet for additional        *
;   information on the instruction set.                               *
;                                                                     *
;   Template file assembled with MPLAB V4.00 and MPASM V2.20.00       *
;                                                                     *
;**********************************************************************
;                                                                     *
;    Filename:          vcht_10.asm                                   *
;    Date:          Dec. 15, 2002                                     *
;    File Version:  1.0                                               *
;                                                                     *
;    Author:        James Hu, Cory Jung, Jason Yu                     *
;    Company:       VitalStatis Medical Solutions                     *
;                                                                     *
;                                                                     *
;**********************************************************************
;                                                                     *
;    Files required:                                                  *
;                                                                     *
;                                                                     *
;                                                                     *
;**********************************************************************
;                                                                     *
;    Notes: Implemented the following:                                *
;           - Added low power detection code                          *
;                                                                     *
;**********************************************************************


      list      p=16f873            ; list directive to define processor
      #include <p16f873.inc>        ; processor specific variable definitions

      __CONFIG _CP_OFF & _WDT_OFF & _BODEN_ON & _PWRTE_ON & _HS_OSC & _WRT_ENABLE_ON
& _LVP_ON & _CPD_OFF
; '__CONFIG' directive is used to embed configuration data within .asm file.
; The lables following the directive are located in the respective .inc file.
; See respective data sheet for additional information on configuration word.



;***** VARIABLE DEFINITIONS
; Variable          EQU    Data address

w_temp              EQU     0x20        ; variable used for context saving
status_temp         EQU     0x21        ; variable used for context saving
```

```
fsr_temp            EQU         0x22        ; variable used for context saving
isr_temp            EQU         0x23        ; variable used for context saving

AD_wait_count EQU         0x24
AD_dL               EQU         0x25        ; temporary data storage, must be
AD_dH               EQU         0x26        ; consecutive and low byte before high
sub_temp            EQU         0x27        ; subroutine temporary variable
sub_temp2           EQU         0x28        ; subroutine temporary variable
prog_flags          EQU         0x29        ; store program flags
parity              EQU         0x2a        ; temporary parity check
rcv_byte            EQU         0x2b        ; storage for RCREG
rcv_status          EQU         0x2c        ; storage for RCSTA
rcv_parity          EQU         0x2d        ; temporary storage for receive parity
AD_BR_count         EQU         0x2e        ; variable for A/D multiplexing
power_chk_cnt EQU         0x2f

; USART send buffer
send_buf_add  EQU         0x30        ; data address pointer
send_buf_psnd EQU         0x31        ; buffer send pointer, points to first byte to
send
send_buf_pdat EQU         0x32        ; buffer data pointer, points to next empty
location
send_buf_start      EQU         0x50        ; start address of send buffer
send_buf_end  EQU         0x80        ; constant, location AFTER last buffer space

BR_addr                   EQU         0xA2        ; Breathing A/D Data
BR_DATA_ST          EQU         0xC0        ; Breathing A/D buffer
BR_DATA_END         EQU         0xFC
BR_rate                   EQU         0xAA        ; Breathing Rate

;--------- part 1 data ------
BR_count            EQU         0xA3
BR_temp                   EQU         0xA4
BR_carry            EQU         0xA5

;--------- part 2 data ------
BR_flag                   EQU         0xA6
BR_max_count  EQU         0xA7
BR_DSP_addr         EQU         0xA8
BR_DSP_data         EQU         0xA9


;***** PROGRAM CONSTANTS
AD_delay            EQU         0x80
data_length         EQU         0x02
send_window         EQU         0x1A        ; sends 26 bytes at a time
dummy_byte          EQU         0x1F
power_th_H          EQU         0x02
power_th_L          EQU         0xCC        ; 3.5V power threshold
power_chk_const     EQU         0x14

; Breathing rate constants
BR_HI_const         EQU         0x90
BR_LOW_const  EQU         0x70
BR_HI_flag          EQU         0x00

; Flags stored in prog_flag
go_ad_flag          EQU         0x00        ; flag to start A/D
go_send_flag  EQU         0x01        ; flag to start sending
go_receive_flag     EQU         0x02        ; flag to start receive decode

; Transmission data ID (upper 3 bits)
ID_EKG_L            EQU         0x00        ; ID: 000
```

```
ID_EKG_H            EQU         0x20         ; ID: 001
ID_RES_L            EQU         0x40         ; ID: 010
ID_RES_H            EQU         0x60         ; ID: 011

; Instruction constants
TMT_SYNC            EQU         0x04
TMT_SEND            EQU         0x05
TMT_END                    EQU         0xbb         ; not used yet


;**********************************************************************
             ORG     0x000            ; processor reset vector
             clrf    PCLATH           ; ensure page bits are cleared
             goto    main             ; go to beginning of program

;**********************************************************************

             ORG     0x004            ; interrupt vector location
             movwf    w_temp          ; save off current W register contents
             movf    STATUS,w        ; move status register into W register
             bcf      STATUS,RP0      ; ensure file register bank set to 0
             movwf   status_temp     ; save off contents of STATUS register
             movf    FSR,w
             movwf   fsr_temp                ; save off address in FSR


;-------------------------
; Check interrupt occurred
;-------------------------

             btfsc   PIR1,RCIF               ; check if receive int occurred
             goto    receive_int
             btfsc   PIR1,TMR1IF             ; check if timer1 overflow occurred
             goto    timer1_int
             goto    end_isr                        ; failsafe to exit ISR


;------------------------
; USART receive interrupt
;------------------------

receive_int
             movf    RCSTA,W                        ; copy RX9D bit
             movwf   rcv_status
             movf    RCREG,W                        ; copy received data
             movwf   rcv_byte

receive_cont
             btfss   rcv_status,OERR         ; check receive overrun
             goto    decode_inst             ; instruction decode

oerr_handle
             bcf           RCSTA,CREN
             bsf           RCSTA,CREN
             goto    end_isr

;--------------------------------
; Decode and process instructions
;--------------------------------

decode_inst
             movf    rcv_byte,W
             sublw   TMT_SEND                ; start transmission instruction
```

```
            btfsc  STATUS,Z
            goto   inst_do_transmit
            goto   end_isr                          ; failsafe to end ISR

; Start transmission instruction
inst_do_transmit
            bsf           prog_flags,go_send_flag           ; set flag to send data
            btfss  PORTC,2                          ; toggle send LED
            goto   set_send_led
clr_send_led
            bcf           PORTC,2
            goto   end_isr
set_send_led
            bsf           PORTC,2
            goto   end_isr

;------------------
; Timer 1 interrupt
;------------------

timer1_int
            bsf           prog_flags,go_ad_flag             ; set A/D flag
            call   set_timer
            bcf           PIR1,TMR1IF                              ; re-enable timer
interrupt

            bsf           ADCON0,GO_DONE             ; perform single A/D acquisition

AD_done_loop
            btfsc  ADCON0,GO_DONE             ; wait for A/D to complete
            goto   AD_done_loop

            movf   ADRESH,W
            movwf  AD_dH                      ; store upper A/D bits
            bsf           STATUS,RP0                 ; select bank 1
            movf   ADRESL,W                   ; retrieve lower A/D bits
            bcf           STATUS,RP0                 ; select bank 0
            movwf  AD_dL
            goto   end_isr

;----------------------------
; Restore pre-interrupt state
;----------------------------

end_isr
            movf   fsr_temp,w                 ; restore FSR contents
            movwf  FSR
            bcf     STATUS,RP0        ; ensure file register bank set to 0
            movf    status_temp,w     ; retrieve copy of STATUS register
            movwf  STATUS            ; restore pre-isr STATUS register contents
            swapf   w_temp,f
            swapf   w_temp,w          ; restore pre-isr W register contents
            retfie                    ; return from interrupt


;**********************************************************************
; Subroutine: set_timer
;
; Adjusts the timer counter for 0.05s period
;**********************************************************************

set_timer
            bcf           T1CON,TMR1ON        ; stop timer1
```



```
            movlw  0x58                     ; set timer counters
            movwf  TMR1L
            movlw  0x9E
            movwf  TMR1H
            bsf          T1CON,TMR1ON       ; start timer1

            return

;**********************************************************************


;--------------
; Main Program
;--------------

main

;-----------
; Setup A/D
;-----------

            bsf          STATUS,RP0               ; select data bank 1
            movlw  0xC9                     ; initialize ADCON1
            movwf  ADCON1

            bcf          STATUS,RP0               ; select data bank 0
            movlw  0x80                     ; initialize ADCON0
            movwf  ADCON0
            bsf          ADCON0,ADON              ; turn on A/D

            movlw  AD_delay
            movwf  AD_wait_count
AD_wait
            decfsz AD_wait_count,F
            goto   AD_wait


;-------------
; PORT config
;-------------

            bsf          STATUS,RP0               ; select data bank 1
            movlw  0xff
            movwf  TRISA
            movlw  0x7f
            movwf  TRISB                    ; make PORTB input
            movlw  0xB3
            movwf  TRISC                    ; set PORTC direction

;--------------
; USART config
;--------------

            movlw  D'4'                     ; baud rate = 62500
            movwf  SPBRG
            movlw  0x00                     ; configure serial tranmit
            movwf  TXSTA
            bcf          STATUS,RP0               ; select data bank 0
            movlw  0x10                     ; configure serial receive
            movwf  RCSTA
            bsf          RCSTA,SPEN               ; enable serial port

            movlw  send_buf_start           ; initialize send buffer
```



```
            movwf  send_buf_psnd
            movwf  send_buf_pdat       ; store start address to pointer
            clrf   AD_BR_count                ; initialize multiplexing counter

            bsf          STATUS,RP0                 ; select bank 1
            bsf          TXSTA,TXEN                 ; enable transmit

;------------
;Timer 1 Setup
;------------

            bcf          STATUS,RP0                 ; select data bank 0
            movlw  0x10                       ; 1:2 prescaler
            movwf  T1CON                      ; configure timer 1
            clrf   TMR1L                      ; clear timer 1
            clrf   TMR1H

;---------------------
; Configure interrupts
;---------------------

            bsf          STATUS,RP0                 ; select data bank 1
            bsf          PIE1,RCIE                  ; enable serial receive int
            bsf          PIE1,TMR1IE                ; enable timer 1 int
            bcf          STATUS,RP0
            movlw  0xC0                       ; enable interrupts
            movwf  INTCON

;---------------
; Process start
;---------------

            bsf          PORTC,2                          ; set transmit indicator
LED
            clrf   prog_flags
            bsf          STATUS,RP0                 ; select data bank 1
            movlw  BR_DATA_ST                 ; for initialization of EKG_data
            movwf  BR_addr
            clrf   BR_count                   ; for initializetion of DSP data
            clrf   BR_temp
            clrf   BR_carry
            clrf   BR_flag
            bcf          STATUS,RP0                 ; select data bank 0
            call   set_timer                  ; start timer

main_loop
            btfsc  prog_flags,go_ad_flag                ; wait for interrupt to
occur
            goto   go_AD                      ; process data
            btfsc  prog_flags,go_send_flag
            call   send_data
            movf   power_chk_cnt,W
            sublw  power_chk_const
            btfsc  STATUS,Z
            call   check_power
            goto   main_loop

;***********************************************************************

;---------------
; A/D process
;---------------
```



```
go_AD

            btfss  PORTB,7                      ; toggle an indicator pin
            goto   set_indicator
clr_indicator
            bcf          PORTB,7
            goto   indicator_cont
set_indicator
            bsf          PORTB,7

indicator_cont
            bsf          PIR1,TMR1IF            ; disable timer interrupt
            movlw  AD_dL                  ; copy address of AD_dL to REGW
            call   shift_data

            movf   AD_dL,W                      ; append EKG header to data
            iorlw  ID_EKG_L
            movwf  AD_dL
            movf   AD_dH,W
            iorlw  ID_EKG_H
            movwf  AD_dH

            movlw  data_length            ; select number of bytes to buffer
            call   buffer_data

; Breathing rate A/D cycle

            incf   AD_BR_count,F       ; check whether to perform breathing A/D
            movlw  0x05
            subwf  AD_BR_count,W
            btfss  STATUS,Z               ; do breathing A/D every 5 A/D cycles
            goto   end_AD

            clrf   AD_BR_count            ; re-initialize counter

            bcf          ADCON0,ADON            ; turn off A/D
            bsf          STATUS,RP0             ; select data bank 1
            movlw  0x49                   ; initialize ADCON1
            movwf  ADCON1                 ; switch to AN1, left justified

            bcf          STATUS,RP0             ; select data bank 0
            movlw  0x88                   ; initialize ADCON0
            movwf  ADCON0
            bsf          ADCON0,ADON            ; turn on A/D

            movlw  AD_delay
            movwf  AD_wait_count

AD_BR_wait1
            decfsz AD_wait_count,F        ; wait for A/D initialization
            goto   AD_BR_wait1

            bsf          ADCON0,GO_DONE         ; Start breathing rate A/D

AD_BR_done_loop
            btfsc  ADCON0,GO_DONE         ; wait for A/D to complete
            goto   AD_BR_done_loop

            movf   ADRESH,W               ; store A/D data
            movwf  AD_dH

BR_AVG_ST
```



```
            call   st_BR_AVG
            bsf           STATUS,RP0
            btfsc  BR_count,2
            call   DSP_BR

; Reconfigure A/D for EKG cycle

            bcf           STATUS,RP0
            bcf           ADCON0,ADON                ; turn off A/D
            bsf           STATUS,RP0                 ; select data bank 1
            movlw  0xC9                       ; initialize ADCON1
            movwf  ADCON1                     ; switch back to AN0, right justified
            bcf           STATUS,RP0                 ; select data bank 0
            movlw  0x80                       ; initialize ADCON0
            movwf  ADCON0
            bsf           ADCON0,ADON                ; turn on A/D

            movlw  AD_delay
            movwf  AD_wait_count

AD_BR_wait2
            decfsz AD_wait_count,F
            goto   AD_BR_wait2

end_AD
            bcf           prog_flags,go_ad_flag             ; clear A/D flag
            bcf           PIR1,TMR1IF                              ; re-enable timer
interrupt
            goto   main_loop



;-------------
; Subroutines
;-------------

;---------------------------------------------------------------------------
; Subroutine: Check power supply voltage
;
; Checks if the power supply voltage is below a threshold (implemented as
; ABOVE threshold detection) and turns on an indicator LED when it is
;---------------------------------------------------------------------------

check_power

            bsf           PIR1,TMR1IF                ; disable timer interrupt
            bcf           STATUS,RP0
            bcf           ADCON0,ADON                ; turn off A/D
            bsf           STATUS,RP0                 ; select data bank 1
            movlw  0xC9                       ; initialize ADCON1
            movwf  ADCON1                     ; switch to AN2, right justified
            bcf           STATUS,RP0                 ; select data bank 0
            movlw  0x90                       ; initialize ADCON0
            movwf  ADCON0
            bsf           ADCON0,ADON                ; turn on A/D

            movlw  AD_delay
            movwf  AD_wait_count

AD_power_wait
            decfsz AD_wait_count,F
            goto   AD_power_wait
```



```
              bsf    ADCON0,GO_DONE            ; perform single A/D acquisition

AD_power_done_loop
              btfsc  ADCON0,GO_DONE            ; wait for A/D to complete
              goto   AD_power_done_loop

              movf   ADRESH,W
              movwf  AD_dH                     ; store upper A/D bits
              bsf           STATUS,RP0                ; select bank 1
              movf   ADRESL,W                  ; retrieve lower A/D bits
              bcf           STATUS,RP0                ; select bank 0
              movwf  AD_dL

              movf   AD_dH,W
              andlw  0x03
              sublw  power_th_H
              btfss  STATUS,C
              goto   supply_low
              movf   AD_dL,W
              sublw  power_th_L
              btfss  STATUS,C
              goto   supply_low

              bcf           PORTC,3
              goto   end_check_power

supply_low
              bsf           PORTC,3

end_check_power
; Reconfigure A/D for EKG cycle
              clrf   power_chk_cnt

              bcf           ADCON0,ADON               ; turn off A/D
              bsf           STATUS,RP0                ; select data bank 1
              movlw  0xC9                      ; initialize ADCON1
              movwf  ADCON1                    ; switch back to AN0, right justified
              bcf           STATUS,RP0                ; select data bank 0
              movlw  0x80                      ; initialize ADCON0
              movwf  ADCON0
              bsf           ADCON0,ADON               ; turn on A/D

              movlw  AD_delay
              movwf  AD_wait_count

AD_power_wait2
              decfsz AD_wait_count,F
              goto   AD_power_wait2

              bcf           PIR1,TMR1IF                        ; re-enable timer
interrupt
              return

;---------------------------------------------------------------------------
; Subroutine: Breathing Average
; Updated: Nov 15th, 2002
;
;---------------------------------------------------------------------------

st_BR_AVG
              bcf           STATUS,RP0                ; select bank 0
              movf   AD_dH,W                          ; move data into buffer, W
              bsf           STATUS,RP0                ; select bank 1
```

```
              addwf  BR_temp,F                  ; W + Br_temp = BR_temp
              btfsc  STATUS,C                   ; check if is carry is set
              incf   BR_carry,F                 ; if true, increment BR_carry, and
BR_count
              incf   BR_count,F                 ; if false, Just increment BR_count

              movlw  0x04                       ; Check if four datas were saved
              subwf  BR_count,W                 ; =
              btfsc  STATUS,Z                   ; =
              goto   STR_BR_D                   ; If true, Divide the total by 4 (go
into STR_BR_D)
              goto   RET_BR_AVG                 ; If false, Return

STR_BR_D
              movf   BR_addr,W                  ; Find current buffer address (BR_addr)
              movwf  FSR                               ; =
              bcf          STATUS,C                    ; Init carry to be 0 (divide
total by 4)
              btfsc  BR_carry,0                 ; Check if 1st carry exist
              bsf          STATUS,C                    ; If exist, Set carry to 1 and
rotate
              rrf          BR_temp,F                   ; If NO exist, Just rotate
              bcf          STATUS,C                    ; Init carry to be 0
              btfsc  BR_carry,1                 ; Check if 2nd carry exist
              bsf          STATUS,C                    ; If exist, Set carry to 1 and
rotate
              rrf          BR_temp,W                   ; If NO exist, Just rotate
              movwf  INDF                       ; Move my average to the current buffer
address (BR_addr)
              clrf   BR_temp                           ; Reset my registers
              clrf   BR_carry                   ; =
              movlw  BR_DATA_END                ; Check if the current buffer address @
the end
              subwf  BR_addr,W                  ; =
              btfsc  STATUS,Z                   ; =
              goto   RESET_BR_ADDR       ; if true, Reset current address to the top
              goto   INC_BR_ADDR                ; if NOT, Increment current buffer
address

INC_BR_ADDR
              incf   BR_addr,F                  ; Increment current buffer address
              goto   RET_BR_AVG

RESET_BR_ADDR
              movlw  BR_DATA_ST                 ; Reset current address to the top
              movwf  BR_addr

RET_BR_AVG
              bcf          STATUS,RP0
              return


;----------------------------------------------------------------------------
; Subroutine: DSP for breathing rate
; Author: James Hu
; Updated: Nov. 15th, 2002
;
; Digital Signal process on digital signal data from accelerometer.
; PIC16F873 will process the data and provides us the breathing rate of the
; user.  The final value will be store in br_rate @ 0x_$)(*!#$_)!(*#$.
;
; Pre: A/D signals are buffered inside RAM space in Bank1
; post: final breathing rate will be stored inside br_rate
```

```
;---------------------------------------------------------------------------

DSP_BR
            bsf         STATUS,RP0
            clrf    BR_max_count
            movf    BR_addr,W
            movwf   BR_DSP_addr

DSP_P1
            movf    BR_DSP_addr,W       ; acquire data from current DSP pointer
position
            movwf   FSR                             ; =
            movf    INDF,W                    ; =
            movwf   BR_DSP_data               ; =
            goto    check_addr
check_addr_ret
            incf    BR_DSP_addr,F       ; increment current DSP pointer position
check_addr_ret2
            btfsc   BR_flag,BR_HI_flag  ; check if hi_flag is set or not
            goto    Check_low_flag            ; if already set, check low
            movlw   BR_HI_const               ; if NOT already set, Check if current
data higher than HI_THRESHOLD
            subwf   BR_DSP_data,W       ; =
            btfsc   STATUS,C                  ; =
            goto    set_flag                  ; if higher, goto set flag
            goto    DSP_P1                    ; if not go to the next data

check_addr
            movlw   BR_DATA_ST                ; Check Br_addr @ beginning (Find where
is the newest data)
            subwf   BR_addr,W                 ; =
            btfsc   STATUS,Z                  ; =
            goto    Wrap_around_check   ; If true, wrap around to the end
            goto    Dec_check                 ; If false, Just decrement Br_addr
check_ret
            subwf   BR_DSP_addr,W       ; Compare DSP current pointer with pointer @
newest data
            btfsc   STATUS,Z                  ; =
            goto    DSP_end                         ; If same, DSP end.
            movlw   BR_DATA_END               ; If not, Check if Current pointer @
the end
            subwf   BR_DSP_addr,W       ; =
            btfsc   STATUS,Z                  ; =
            goto    BR_wrap_around            ; If true, Current pointer wrap around
            goto    check_addr_ret            ; If not, return and increment current
DSP pointer position

Wrap_around_check
            movlw   BR_DATA_END
            goto    check_ret

Dec_check
            decf    BR_addr,W
            goto    check_ret

BR_wrap_around
            movlw   BR_DATA_ST                ; Current pointer wrap around to the
beginning
            movwf   BR_DSP_addr               ; =
            goto    check_addr_ret2           ; =

Check_low_flag
            movlw   BR_LOW_const        ; Check if data is lower than LOW THRESHOLD
```



```
            subwf  BR_DSP_data,W       ; =
            btfss  STATUS,C                  ; =
            goto   clr_flag                  ; if lower, clear flag and increment
Br_max_count
            goto   DSP_P1                    ; if not go to the next data

set_flag
            bsf           BR_flag,BR_HI_flag
            goto   DSP_P1

clr_flag
            bcf           BR_flag,BR_HI_flag
            incf   BR_max_count,f
            goto   DSP_P1

DSP_end
            movf   BR_max_count,W            ; stores BR_rate value
            movwf  BR_rate                          ; =
            clrf   BR_count                  ; reset BR_count = 0 from 4
            clrf   BR_flag                          ; reset BR_flag

; Package and buffer breathing rate data

            movf   BR_rate,W
            bcf           STATUS,RP0
            movwf  AD_dL
            clrf   AD_dH

            movlw  AD_dL
            call   shift_data                ; package data

            movf   AD_dL,W                          ; append RES header
            iorlw  ID_RES_L
            movwf  AD_dL
            movf   AD_dH,W
            iorlw  ID_RES_H
            movwf  AD_dH

            movlw  data_length
            call   buffer_data               ; buffer data

            incf   power_chk_cnt,F

            return

;---------------------------------------------------------------------------
; Subroutine: shift_data
;
; Shifts data into data transmission format: lower byte uses LS 5-bits,
; higher byte uses LS 5-bits. Two data bytes must be one after the other,
; with lower byte in lower address.
;
; Pre: Address of lower byte must be in REGW
;---------------------------------------------------------------------------

shift_data

            ; assemble lower data byte
            movwf  FSR                       ; store lower byte address to indirect
reg
            movf   INDF,W                    ; copy data in FSR location to W
            movwf  sub_temp                  ; make a copy of lower byte at sub_temp
            andlw  B'00011111'               ; clear top 3 bits of data
```

```
              movwf  INDF

              ; assemble upper data byte
              swapf  sub_temp,F                  ; move upper 3-bits of lower byte to
right hand side
              rrf           sub_temp,W
              andlw  B'00000111'                 ; keep only bottom 3 bits
              movwf  sub_temp

              incf   FSR,F                       ; move pointer to upper byte address
              swapf  INDF,F                      ; move upper byte to appropriate
position
              rrf           INDF,W
              andlw  B'00011000'
              iorwf  sub_temp,W
              movwf  INDF

              return

;----------------------------------------------------------------------------
; Subroutine: buffer_data
;
; Stores number of bytes into USART send buffer.
;
; Pre: - Number of bytes to buffer must be stored in REGW
;      - Data to be buffered must be stored in AD_dL
;       (and consecutive locations if buffer > 1 byte)
;----------------------------------------------------------------------------

buffer_data
              bcf           STATUS,RP0
              movwf  sub_temp                    ; use same temporary variable
              movlw  AD_dL                       ; start address of data
              movwf  send_buf_add

buffer_loop
              movf   send_buf_add,W
              movwf  FSR
              movf   INDF,W                      ; load data to be buffered
              movwf  sub_temp2                   ; store to temporary location
              movf   send_buf_pdat,W             ; move current buffer pointer to REGW
              movwf  FSR                               ; store buffer pointer to FSR
              movf   sub_temp2,W
              movwf  INDF                        ; store data bits
              incf   send_buf_pdat,F             ; increment buffer pointer
              incf   send_buf_add,F              ; increment data pointer

              movf   send_buf_pdat,W
              sublw  send_buf_end         ; check if buffer pointer at end
              btfsc  STATUS,Z
              goto   reset_pdat                  ; reset buffer pointer to top

buffer_cont
              movf   send_buf_pdat,W
              subwf  send_buf_psnd,W             ; check if buffer overflow
              btfsc  STATUS,Z                    ; beginning pointer = end pointer if
overflow
              bsf           PORTC,2                            ; set power on/transmit
indicator LED

              decfsz sub_temp,F
              goto   buffer_loop
              return
```

```
reset_pdat
            movlw  send_buf_start             ; buffer end reached
            movwf  send_buf_pdat              ; reset pointer to beginning
            goto   buffer_cont

;---------------------------------------------------------------------------
; Subroutine: send_data
;
; Sends number of bytes as specified by send_window through TX of USART.
; Send begins at send_buf_psnd. Terminates when USART send complete.
;
;---------------------------------------------------------------------------

send_data

send_finish
            bsf          STATUS,RP0                 ; Select bank 1
            btfss  TXSTA,TRMT                 ; Wait for previous send to complete
            goto   send_finish

            bcf          STATUS,RP0                 ; Deselect bank 1
            movlw  send_window
            movwf  sub_temp
            movf   send_buf_psnd,W            ; send from location of buffer pointer
            movwf  FSR

send_loop
            movf   send_buf_psnd,W            ; check if buffer underflow
            subwf  send_buf_pdat,W
            btfsc  STATUS,Z                   ; pointer beginning = pointer end if
underflow
            goto   und_paddata                ; run if buffer underrun

            call   send_byte
            incf   send_buf_psnd,W
            movwf  send_buf_psnd
            sublw  send_buf_end        ; check if pointer at buffer end
            btfsc  STATUS,Z
            goto   reset_send_psnd            ; reset if at buffer end

cont_send
            movf   send_buf_psnd,W
            movwf  FSR

send_wait
            btfss  PIR1,TXIF                  ; Check if data has been moved to TSR
            goto   send_wait                  ; Wait for TXREG to be empty

            decfsz sub_temp,F
            goto   send_loop

            bcf          prog_flags,go_send_flag
            return


reset_send_psnd
            movlw  send_buf_start                   ; wrap around pointer to
beginning
            movwf  send_buf_psnd              ; of buffer
            goto   cont_send
```



```
und_paddata
            movlw  dummy_byte                       ; send dummy byte if underrun
            movwf  sub_temp2
            movlw  sub_temp2
            movwf  FSR
            call   send_byte
            goto   send_wait

;---------------------------------------------------------------------------
; Subroutine: send_byte
;
; Sends byte through TX of USART. Returns immediately.
;
; Pre: Adress of data to be sent must be placed in FSR
;---------------------------------------------------------------------------

send_byte
            movf   INDF,W
            movwf  TXREG                ; actual data send
            return


; End of source code
            END
```

Appendix B: Palm™ OS Source Code

```
// VtChtGrph.h
//
// header file for VitalChartGraphics
//
// This wizard-generated code is based on code adapted from the
// stationery files distributed as part of the Palm OS SDK 4.0.
//
// Copyright (c) 1999-2000 Palm, Inc. or its subsidiaries.
// All rights reserved.

#ifndef VTCHTGRPH_H_
#define VTCHTGRPH_H_

// *********************************************************************
// Internal Structures
// *********************************************************************

typedef struct VtChtGrphPreferenceType
{
	UInt8 replaceme;
} VtChtGrphPreferenceType;

// *********************************************************************
// Global variables
// *********************************************************************

extern VtChtGrphPreferenceType g_prefs;

// *********************************************************************
// Internal Constants
// *********************************************************************

#define appFileCreator			'VSMS'
#define appName					"VitalChart"
#define appVersionNum			0x01
#define appPrefID				0x00
#define appPrefVersionNum		0x01

#endif // VTCHTGRPH_H_
// *********************************************************************
// External Function Prototypes
// *********************************************************************
void EncodeValue(UInt8 num_array, UInt16 *EKG, UInt16 *RES);
static void GetData();
UInt16 ScaleData(UInt16 UnformattedData);
static void ResetDataBuffer();
static void ClearLine(UInt16 xlocation, UInt16 width);
static void DrawBitmap(Int16 resID, Int16 x, Int16 y);
static void CheckConnect();
UInt16 ScaleData(UInt16 UnformattedData);
static void PlotData();
static void ShowDateTime();
static void ResetPlotArea();
static void GridSelect();
static void ZoomSelect();
static void PeakDetect(UInt16 Index);
static void CalcHeartRate();
static void InvertHeart();
```

```
static void PlotBRate();
Boolean DatabaseFormHandleEvent(EventType * eventP);
Boolean DatabaseFormDoCommand(UInt16 command);
void * GetObjectPtr(UInt16 objectID);
void DatabaseFormInit(FormType * /*frmP*/);
static void EncodeValueDB(UInt8 num_array, UInt16 *EKG, UInt16 *RES);
static void GetDataDB();
static void PlaySound();
static void PlayAlarm();
void EncodeDBValue(UInt8 num_array, UInt16 *EKG, UInt16 *RES);
static void GetDataBaseData();
static void StopAlarm();
void CreateRecord();
static void CompleteBar();
static void ClearCompleteBar();

//     Header generated by Constructor for Palm OS¨ 1.6
//
//     Generated at 12:23:27 PM on December 16, 2002
//
//     Generated for file: C:\Documents and Settings\See-Ho  Tsang\My Documents\2002-
03\ENSC 340\Software\VitalChartDataBaseNew\VitalChartGraphics\Rsc\VtChtGrph.rsrc
//
//     THIS IS AN AUTOMATICALLY GENERATED HEADER FILE FROM CONSTRUCTOR FOR PALM OS¨;
//     - DO NOT EDIT - CHANGES MADE TO THIS FILE WILL BE LOST
//
//     Palm App Name:           "VitalChartGraphics"
//
//     Palm App Version:        "1.0"


//     Resource: tFRM 1000
#define MainForm                                  1000      //(Left Origin = 0, Top
Origin = 0, Width = 160, Height = 160, Usable = 1, Modal = 0, Save Behind = 0, Help ID
= 0, Menu Bar ID = 1000, Default Button ID = 0)
#define MainZoomCheckbox                          1008      //(Left Origin = 115, Top
Origin = 144, Width = 45, Height = 12, Usable = 1, Selected = 0, Group ID = 0, Font =
Standard)
#define MainGridCheckbox                          1012      //(Left Origin = 81, Top
Origin = 144, Width = 45, Height = 12, Usable = 1, Selected = 1, Group ID = 0, Font =
Standard)
#define MainUnnamed1006BitMap                     1140      //(Left Origin = 2, Top
Origin = 40, Bitmap Resource ID = 1140, Usable = 1)
#define MainUnnamed1003BitMap                     1700      //(Left Origin = 2, Top
Origin = 18, Bitmap Resource ID = 1700, Usable = 1)
#define MainUnnamed1008BitMap                     1160      //(Left Origin = 21, Top
Origin = 18, Bitmap Resource ID = 1160, Usable = 1)
#define MainUnnamed1010BitMap                     1170      //(Left Origin = 60, Top
Origin = 18, Bitmap Resource ID = 1170, Usable = 1)
#define MainUnnamed1007BitMap                     1800      //(Left Origin = 22, Top
Origin = 19, Bitmap Resource ID = 1800, Usable = 1)
#define MainPlayButtonGraphicButton               1002      //(Left Origin = 41, Top
Origin = 144, Width = 36, Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-
bold Frame = 1, Bitmap Resource ID = 1100, Selected Bitmap Resource ID = 1101)
#define MainStopButtonGraphicButton               1004      //(Left Origin = 2, Top
Origin = 144, Width = 36, Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-
bold Frame = 1, Bitmap Resource ID = 1110, Selected Bitmap Resource ID = 1111)

//     Resource: tFRM 1100
#define AboutForm                                 1100      //(Left Origin = 2, Top
Origin = 2, Width = 156, Height = 156, Usable = 1, Modal = 1, Save Behind = 1, Help ID
= 0, Menu Bar ID = 0, Default Button ID = 0)
```

```
#define AboutOKButton                              1105      //(Left Origin = 58, Top
Origin = 137, Width = 40, Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-
bold Frame = 1, Font = Standard)
#define AboutUnnamed1101BitMap                     1001      //(Left Origin = 62, Top
Origin = 39, Bitmap Resource ID = 1001, Usable = 1)
#define AboutTitleLabel                            1102      //(Left Origin = 22, Top
Origin = 18, Usable = 1, Font = Large)
#define AboutAboutCopyrightTextLabel               1103      //(Left Origin = 21, Top
Origin = 95, Usable = 1, Font = Standard)
#define AboutText2Label                            1104      //(Left Origin = 49, Top
Origin = 121, Usable = 1, Font = Bold)
#define AboutUnnamed1106Label                      1106      //(Left Origin = 18, Top
Origin = 70, Usable = 1, Font = Standard)

//     Resource: tFRM 1200
#define DatabaseForm                               1200      //(Left Origin = 0, Top
Origin = 0, Width = 160, Height = 160, Usable = 1, Modal = 0, Save Behind = 0, Help ID
= 0, Menu Bar ID = 1100, Default Button ID = 0)
#define DatabaseCreateRecordButton                 1201      //(Left Origin = 85, Top
Origin = 108, Width = 68, Height = 15, Usable = 1, Anchor Left = 1, Frame = 1, Non-
bold Frame = 1, Font = Standard)
#define DatabaseEndRecordButton                    1202      //(Left Origin = 86, Top
Origin = 127, Width = 68, Height = 15, Usable = 1, Anchor Left = 1, Frame = 1, Non-
bold Frame = 1, Font = Standard)
#define DatabaseDatabaseNameFieldField             1206      //(Left Origin = 35, Top
Origin = 90, Width = 90, Height = 15, Usable = 1, Editable = 1, Underline = 1, Single
Line = 0, Dynamic Size = 0, Left Justified = 1, Max Characters = 40, Font = Standard,
Auto Shift = 0, Has Scroll Bar = 0, Numeric = 0)
#define DatabaseDatabaseAgeFieldField              1207      //(Left Origin = 35, Top
Origin = 106, Width = 38, Height = 14, Usable = 1, Editable = 1, Underline = 1, Single
Line = 0, Dynamic Size = 0, Left Justified = 1, Max Characters = 5, Font = Standard,
Auto Shift = 0, Has Scroll Bar = 0, Numeric = 0)
#define DatabaseDatabaseIdLabel                    1203      //(Left Origin = 5, Top
Origin = 122, Usable = 1, Font = Standard)
#define DatabaseDatabaseNameLabel                  1205      //(Left Origin = 5, Top
Origin = 90, Usable = 1, Font = Standard)
#define DatabaseDatabaseAgeLabel                   1208      //(Left Origin = 5, Top
Origin = 106, Usable = 1, Font = Standard)
#define DatabaseDatabaseTextLabel                  1209      //(Left Origin = 5, Top
Origin = 19, Usable = 1, Font = Standard)


//     Resource: Talt 1001
#define RomIncompatibleAlert                       1001
#define RomIncompatibleOK                          0


//     Resource: MBAR 1000
#define VtChtGrphMainMenuBar                       1000

//     Resource: MBAR 1100
#define DatabaseMainMenuMenuBar                    1100


//     Resource: MENU 1000
#define OptionsMenu                                1000
#define OptionsAboutVitalChartBioReader            1000

//     Resource: MENU 10000
#define EditMenu                                   10000
#define EditUndo                                   10000     //   Command Key: U
#define EditCut                                    10001     //   Command Key: X
#define EditCopy                                   10002     //   Command Key: C
```

```
#define EditPaste                                       10003     //   Command Key: P
#define EditSelectAll                                   10004     //   Command Key: S
#define EditKeyboard                                    10006     //   Command Key: K
#define EditGraffitiHelp                                10007     //   Command Key: G

//      Resource: MENU 1100
#define OptionsDatabaseMenu                             1100
#define OptionsDatabaseManageDatabase                   1100

//      Resource: MENU 1200
#define DatabaseMenu                                    1200
#define DatabaseExitToMainMenu                          1200


//      Resource: PICT 1001
#define Bitmap                                          1001

//      Resource: PICT 1002
#define Bitmap2                                         1002

//      Resource: PICT 1008
#define Bitmap3                                         1008

//      Resource: PICT 1011
#define Bitmap4                                         1011

//      Resource: PICT 1012
#define Bitmap5                                         1012

//      Resource: PICT 1018
#define Bitmap6                                         1018

//      Resource: PICT 1100
#define PlayPauseBitmap                                 1100

//      Resource: PICT 1101
#define InvPlayPauseBitmap                              1101

//      Resource: PICT 1110
#define StopBitmap                                      1110

//      Resource: PICT 1111
#define InvStopBitmap                                   1111

//      Resource: PICT 1140
#define GridBitmap                                      1140

//      Resource: PICT 1150
#define PlugBitmap                                      1150

//      Resource: PICT 1160
#define HeartBoxBitmap                                  1160

//      Resource: PICT 1170
#define BreathingBBitmap                                1170

//      Resource: PICT 1190
#define DataPointBitmap                                 1190

//      Resource: PICT 1200
#define GridLineStartBitmap                             1200

//      Resource: PICT 1300
```

```
#define GridDotsABitmap                          1300

//     Resource: PICT 1400
#define GridBlankBitmap                          1400

//     Resource: PICT 1500
#define GridLineStopBitmap                       1500

//     Resource: PICT 1600
#define GridDotsBBitmap                          1600

//     Resource: PICT 1700
#define NoPlugBitmap                             1700

//     Resource: PICT 1800
#define HeartBitmap                              1800

//     Resource: PICT 1900
#define DashBitmap                               1900


//     Resource: tbmf 1100
#define BitmapID1100BitmapFamily                 1100

//     Resource: tbmf 1101
#define SelectedBitmapID1101BitmapFamily         1101

//     Resource: tbmf 1110
#define BitmapID1110BitmapFamily                 1110

//     Resource: tbmf 1111
#define SelectedBitmapID1111BitmapFamily         1111

//     Resource: tbmf 1140
#define BitmapResourceID1140BitmapFamily         1140

//     Resource: tbmf 1150
#define BitmapResourceID1150BitmapFamily         1150

//     Resource: tbmf 1160
#define BitmapResourceID1160BitmapFamily         1160

//     Resource: tbmf 1170
#define BitmapResourceID1170BitmapFamily         1170

//     Resource: tbmf 1700
#define BitmapResourceID1700BitmapFamily         1700

//     Resource: tbmf 1800
#define BitmapResourceID1800BitmapFamily         1800

//     Resource: tbmf 1900
#define BitmapResourceID1900BitmapFamily         1900

//     Resource: tbmf 1001
#define BitmapResourceID1001BitmapFamily         1001


//     Resource: taif 1000
#define Largeicons12and8bitsAppIconFamily        1000

//     Resource: taif 1001
#define Smallicons12and8bitsAppIconFamily        1001
```

```
// VtChtGrphMain.c
//
// main file for VitalChartGraphics
//
// This wizard-generated code is based on code adapted from the
// stationery files distributed as part of the Palm OS SDK 4.0.
//
// Copyright (c) 1999-2000 Palm, Inc. or its subsidiaries.
// All rights reserved.

#include <PalmOS.h>

#include "VtChtGrph.h"
#include "VtChtGrphRsc.h"


// ********************************************************************
// Entry Points
// ********************************************************************

// ********************************************************************
// Global variables
// ********************************************************************
#define leftOrigin                 2
#define GridChangeCount    4
#define GridTopOrigin              40
#define GridHeight                 100
#define GridWidth                  155
#define EKG_ARRAYSIZE              10000
#define GridBottomOffset   138
#define MaxValue                   93
#define SamplingTime               0.1
#define MaxEdgeIndexSize   50
#define ThresholdPlus              0
#define ThresholdMinus          -40
#define MaxDiffArraySize    50
#define timeouttime                0.02
#define NumHRAve                   3
#define DelayBuffer                10
#define PlotTime1                  0.01
#define PlotTime2                  0.02
#define DB_EKG_Len                 1500
#define DB_RES_Len                 20
#define timeoutconst               250


// g_prefs
// cache for application preferences during program execution

VtChtGrphPreferenceType g_prefs;
Boolean Plot=false;
Boolean GridOn=true;
Int32 i=0;
UInt16 DataIndex = 0;
UInt16 EKG_ARRAY[EKG_ARRAYSIZE];
UInt16 RES_ARRAY[EKG_ARRAYSIZE];
UInt16 EKG_ARRAY_DB[DB_EKG_Len];
UInt16 RES_ARRAY_DB[DB_RES_Len];
UInt16 DataBuffer[155];
UInt16 EKG_ARRAY_P;
UInt16 EKG_ARRAY_P_DB;
UInt16 RES_ARRAY_P;
UInt16 RES_ARRAY_P_DB;
```

```
Boolean Connected;
Boolean Delay =false;
Boolean Sync = false;
UInt16 EdgeLocation[MaxEdgeIndexSize];
UInt16 EdgeIndex=0;
Int16 DiffArray[MaxDiffArraySize];
UInt16 SoundAmp;
Boolean trigger=false;
Boolean CheckConnectDisable = false;
UInt32 *watchdog_time;
UInt32 *elapsed_time, *plotelapsed_time;
Boolean WatchDog = false;
UInt16 BSum =0;
UInt16 HRate=0;
UInt16 PointDelta = 2;
UInt32 WatchDogTime = 10;
Boolean GetHRate = true;
Boolean Zoom = false;
Boolean GetDBData=false;
Boolean DataComplete = false;
UInt32 HRTimeOut = 0;
Boolean DatabaseStarted = false;


// global database reference, will be init in AppStart() and close db in AppStop()
DmOpenRef    gDatabase = 0;
// ********************************************************************
// Internal Constants
// ********************************************************************

// Define the minimum OS version we support
#define ourMinVersion    sysMakeROMVersion(3,0,0,sysROMStageDevelopment,0)
#define kPalmOS10Version sysMakeROMVersion(1,0,0,sysROMStageRelease,0)



// ********************************************************************
// Internal Functions
// ********************************************************************


// FUNCTION: GetObjectPtr
//
// DESCRIPTION:
//
// This routine returns a pointer to an object in the current form.
//
// PARAMETERS:
//
// formId
//     id of the form to display
//
// RETURNED:
//     address of object as a void pointer

void * GetObjectPtr(UInt16 objectID)
{
    FormType * frmP;

    frmP = FrmGetActiveForm();
    return FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP, objectID));
}
```

```
// FUNCTION: MainFormInit
//
// DESCRIPTION: This routine initializes the MainForm form.
//
// PARAMETERS:
//
// frm
//     pointer to the MainForm form.

static void MainFormInit(FormType * /*frmP*/)
{
}

// FUNCTION: MainFormDoCommand
//
// DESCRIPTION: This routine performs the menu command specified.
//
// PARAMETERS:
//
// command
//     menu item id



static Boolean MainFormDoCommand(UInt16 command)
{
    Boolean handled = false;
    FormType * frmP;
    Char     Send0x04[] = " \n";
    //Err           err;
    //UInt16 portId;

    switch (command)
    {
    case OptionsAboutVitalChartBioReader:

        // Clear the menu status from the display
        MenuEraseStatus(0);

        // Display the About Box.
        frmP = FrmInitForm (AboutForm);
        FrmDoDialog (frmP);
        FrmDeleteForm (frmP);

        handled = true;
        break;

    case OptionsDatabaseManageDatabase:

       //MenuEraseStatus(0);
       FrmGotoForm (DatabaseForm);
       WatchDog = false;
       handled = true;
       break;

    case MainPlayButtonGraphicButton:


       if(Plot == true)
          {
             Plot = false;
             WatchDog = false;
          }
```

```
        else if(Plot == false)
        {
        DataIndex = 0;
        EKG_ARRAY_P = 0;
        Sync=true;
        Delay=true;
               WatchDog = true;
               StopAlarm();
               HRTimeOut = 0;
               Plot = true;
        }

               //record our open status in global.
               else
               {
                      //SrmClose(portId);
               }

               break;

      case MainStopButtonGraphicButton:
        Plot=false;
        DataIndex=0;
        EKG_ARRAY_P=0;
        ResetPlotArea();
        ResetDataBuffer();
        WatchDog = false;
        StopAlarm();
            break;


    case MainGridCheckbox:
        GridSelect();

               break;

        case MainZoomCheckbox:
               ZoomSelect();
        }
        return handled;

}

// FUNCTION: MainFormHandleEvent
//
// DESCRIPTION:
//
// This routine is the event handler for the "MainForm" of this
// application.
//
// PARAMETERS:
//
// eventP
//     a pointer to an EventType structure
//
// RETURNED:
//     true if the event was handled and should not be passed to
//     FrmHandleEvent

static Boolean MainFormHandleEvent(EventType * eventP)
{
    Boolean handled = false;
```



```
    FormType * frmP;

    switch (eventP->eType)
        {
        case menuEvent:
            return MainFormDoCommand(eventP->data.menu.itemID);

        case frmOpenEvent:
            frmP = FrmGetActiveForm();
            MainFormInit(frmP);
            FrmDrawForm(frmP);
            handled = true;
            break;

        case frmUpdateEvent:
            // To do any custom drawing here, first call
            // FrmDrawForm(), then do your drawing, and
            // then set handled to true.
                    frmP = FrmGetActiveForm();
            FrmDrawForm(frmP);
            handled = true;
            break;

             case ctlSelectEvent:
             return MainFormDoCommand(eventP->data.ctlSelect.controlID);

             default:
            break;

        }

    return handled;
}

// FUNCTION: AppHandleEvent
//
// DESCRIPTION:
//
// This routine loads form resources and set the event handler for
// the form loaded.
//
// PARAMETERS:
//
// event
//     a pointer to an EventType structure
//
// RETURNED:
//     true if the event was handled and should not be passed
//     to a higher level handler.

static Boolean AppHandleEvent(EventType * eventP)
{
    UInt16 formId;
    FormType * frmP;

    if (eventP->eType == frmLoadEvent)
    {
        // Load the form resource.
        formId = eventP->data.frmLoad.formID;
        frmP = FrmInitForm(formId);
        FrmSetActiveForm(frmP);

        // Set the event handler for the form.  The handler of the
```

```
        // currently active form is called by FrmHandleEvent each
        // time is receives an event.
        switch (formId)
        {
        case MainForm:
             CheckConnectDisable = false;
             GridOn=true;
            FrmSetEventHandler(frmP, MainFormHandleEvent);
            break;

             case DatabaseForm:
             CheckConnectDisable = true;
             Plot = false;
             GridOn=false;
             ResetPlotArea();
             FrmSetEventHandler(frmP, DatabaseFormHandleEvent);
             break;

        default:
            break;

        }
        return true;
    }

    return false;
}

// FUNCTION: AppEventLoop
//
// DESCRIPTION: This routine is the event loop for the application.

static void AppEventLoop(void)
{
    UInt16 error;
    EventType event;

       //, *delaytime;
   // char EKGPointer1[20],EKGPointer2[20];
       //char numchar[20];
       Char   Send0x04[] = " \n";
       //Err         err;
   // UInt16 portId;

       // Initialize timer

    *elapsed_time = TimGetTicks();

    *plotelapsed_time = TimGetTicks();


    // delaytime=MemPtrNew(sizeof(UInt32));
       //*delaytime = TimGetTicks();

    do {
        // change timeout if you need periodic nilEvents
        EvtGetEvent(&event, 1);


        HRTimeOut++;
        if((HRTimeOut >= timeoutconst)&&(WatchDog ==true))
              PlayAlarm();
```

```
        if((i%50)==1)
        {
         PlotBRate();


        }

        if(DataComplete==true)
        {
              CreateRecord();
                             DataComplete=false;
              //EKG_ARRAY_P_DB=0;
              }

              if (i>=GridWidth)
              {
              i=0;
              DataIndex = 0;
              EKG_ARRAY_P = 0;
              Sync=true;
              Delay=true;
              CalcHeartRate();

              }

       //     err = SrmOpen(serPortCradleRS232Port /* port */, 19200, /* baud */
&portId);
       //     if (err == errNone)
       //            {
                      // display error message here.

       //                    SrmControl(portId, srmCtlSetDTRAsserted, (void *)false,
(UInt16 *)sizeof(Boolean));
         //                  Send0x04[0] = 0x04;
       //                    SrmSend(portId, Send0x04, 2, &err);
       //                    SrmSendWait(portId);
       //                    SrmClose(portId);
        //           }
       //     }

              else
              {
                     /*StrIToA(EKGPointer1,EKG_ARRAY_P);
                     StrIToA(EKGPointer2,EKG_ARRAY_P2);
                     WinDrawChars("       ",StrLen("       "),120,140);
                     WinDrawChars("       ",StrLen("       "),120,150);
                     WinDrawChars(EKGPointer1,StrLen(EKGPointer1),120,140);
                     WinDrawChars(EKGPointer2,StrLen(EKGPointer2),120,150);*/

                     if(Plot==true)
                     SysSetAutoOffTime(0);

                     else
                     SysSetAutoOffTime(60);

                     if(event.eType==nilEvent && CheckConnectDisable==false)
                     CheckConnect();
                      //if((Plot == true)&&(HaveData==true))


               if (! SysHandleEvent(&event))
               {
```



```
                    if (! MenuHandleEvent(0, &event, &error))
                    {
                       if (! AppHandleEvent(&event))
                        {

                            FrmDispatchEvent(&event);
                        }
                    }
                }

            //  StrIToA(EKGPointer,EKG_ARRAY_P%EKG_ARRAYSIZE);
           //   StrIToA(DataIndexChar,DataIndex);
           //   WinDrawChars("         ",StrLen("         "),120,130);
           //   WinDrawChars("         ",StrLen("         "),120,120);
          //    WinDrawChars(DataIndexChar,StrLen(DataIndexChar),120,120);
           //   WinDrawChars(EKGPointer,StrLen(EKGPointer),120,130);

                     if((Plot==true)&&(Delay==false)&&(Zoom==true))
                {
                     if(((double)(TimGetTicks()-
*plotelapsed_time)/(double)SysTicksPerSecond()) >= PlotTime1)
                     {
                                  *plotelapsed_time = TimGetTicks();
                                  PlotData();
                                  if(GetHRate ==true)
                                  PeakDetect(DataIndex);




                    }
                }
                else if((Plot==true)&&(Delay==false)&&(Zoom==false))
                     if(((double)(TimGetTicks()-
*plotelapsed_time)/(double)SysTicksPerSecond()) >= PlotTime2)
                     {
                                  *plotelapsed_time = TimGetTicks();
                                  PlotData();
                                  PeakDetect(DataIndex);




                    }


             // Watch dog timer for heart rate
           //  if(WatchDog == true)
           //  {
              //       if(((double)(TimGetTicks()-
*watchdog_time)/(double)SysTicksPerSecond()) >= WatchDogTime)
             //             {
                                  //Play Alarm
              //          PlayAlarm();
              //       }
           //  }

             if(((EKG_ARRAY_P%EKG_ARRAYSIZE)>=DelayBuffer)&&(Sync==true))

                     {
                            Delay = false;
                            Sync = false;
                     }
```

```
              //            WinDrawChars("                  ",StrLen("
"),120,144);
              //            Delay=false;
              //      }
              //      if((EKG_ARRAY_P%EKG_ARRAYSIZE)-DataIndex<=ResumeBuffer)
              //      {
              //            WinDrawChars("Bfr Udrn",StrLen("bfr Udrn"),120,144);
              //            Delay=true;
              //      }


                     // If an event is not received within 0.2s, receive data
                     if((event.eType == nilEvent) && (Plot == true))
                     {
                            if(((double)(TimGetTicks()-
*elapsed_time)/(double)SysTicksPerSecond()) >= SamplingTime)//&&(SendDelay==false));
                            {
                                   *elapsed_time = TimGetTicks();
                                   GetData();


                            }
              }
              if((event.eType == nilEvent) && (GetDBData == true))
                     {
                            if(((double)(TimGetTicks()-
*elapsed_time)/(double)SysTicksPerSecond()) >= SamplingTime)//&&(SendDelay==false));
                            {
                                   *elapsed_time = TimGetTicks();
                                   GetDataBaseData();



                            }
              }
              if(GetDBData == true)
                     {
                          CompleteBar();
                      }
       }
    }
    while (event.eType != appStopEvent);



  //  MemPtrFree(delaytime);
}

// FUNCTION: AppStart
//
// DESCRIPTION:  Get the current application's preferences.
//
// RETURNED:
//     errNone - if nothing went wrong

static Err AppStart(void)
{

    // database
       // declarations
       const UInt32 kType = 'DATA';
       const UInt32 kCreator = 'VSMS';
```

```
        const char *kName = "VitalChartDB";
        const int kCardNumber = 0;
        Err    err = 0;

     LocalID theLocalID;
     UInt16 theCardNum;
     UInt16 theAttributes, index;

     UInt16 prefsSize;
     Boolean Plot = false;


     plotelapsed_time = (UInt32 *)MemPtrNew(sizeof(UInt32));
        elapsed_time = (UInt32 *)MemPtrNew(sizeof(UInt32));
        ShowDateTime();


     // Read the saved preferences / saved-state information.
     prefsSize = sizeof(VtChtGrphPreferenceType);
     if (PrefGetAppPreferences(
         appFileCreator, appPrefID, &g_prefs, &prefsSize, true) !=
         noPreferenceFound)
     {
         // FIXME: setup g_prefs with default values
     }

     SoundAmp = PrefGetPreference(prefGameSoundVolume);
     watchdog_time = (UInt32 *)MemPtrNew(sizeof(UInt32));

     for(index=0; index < DB_EKG_Len; index++)
     {
        EKG_ARRAY_DB[index] = 0;
     }

     BSum = 0;
     //EKG_ARRAY_P_DB = 0;
/*
DataBuffer    [      0      ]       =     0      ;
DataBuffer    [      1      ]       =     0      ;
DataBuffer    [      2      ]       =     0      ;
DataBuffer    [      3      ]       =     0      ;
DataBuffer    [      4      ]       =     0      ;
DataBuffer    [      5      ]       =     0      ;
DataBuffer    [      6      ]       =     0      ;
DataBuffer    [      7      ]       =     0      ;
DataBuffer    [      8      ]       =     0      ;
DataBuffer    [      9      ]       =     0      ;
DataBuffer    [      10     ]       =     0      ;
DataBuffer    [      11     ]       =     0      ;
DataBuffer    [      12     ]       =     0      ;
DataBuffer    [      13     ]       =     0      ;
DataBuffer    [      14     ]       =     0      ;
DataBuffer    [      15     ]       =     0      ;
DataBuffer    [      16     ]       =     0      ;
DataBuffer    [      17     ]       =     0      ;
DataBuffer    [      18     ]       =     0      ;
DataBuffer    [      19     ]       =     0      ;
DataBuffer    [      20     ]       =     0      ;
DataBuffer    [      21     ]       =     0      ;
DataBuffer    [      22     ]       =     0      ;
DataBuffer    [      23     ]       =     0      ;
DataBuffer    [      24     ]       =     0      ;
DataBuffer    [      25     ]       =     600    ;
```



```
DataBuffer    [     26     ]     =     600    ;
DataBuffer    [     27     ]     =     600    ;
DataBuffer    [     28     ]     =     600    ;
DataBuffer    [     29     ]     =     600    ;
DataBuffer    [     30     ]     =     600    ;
DataBuffer    [     31     ]     =     600    ;
DataBuffer    [     32     ]     =     600    ;
DataBuffer    [     33     ]     =     600    ;
DataBuffer    [     34     ]     =     600    ;
DataBuffer    [     35     ]     =     600    ;
DataBuffer    [     36     ]     =     600    ;
DataBuffer    [     37     ]     =     600    ;
DataBuffer    [     38     ]     =     600    ;
DataBuffer    [     39     ]     =     600    ;
DataBuffer    [     40     ]     =     600    ;
DataBuffer    [     41     ]     =     600    ;
DataBuffer    [     42     ]     =     600    ;
DataBuffer    [     43     ]     =     600    ;
DataBuffer    [     44     ]     =     600    ;
DataBuffer    [     45     ]     =     600    ;
DataBuffer    [     46     ]     =     600    ;
DataBuffer    [     47     ]     =     600    ;
DataBuffer    [     48     ]     =     600    ;
DataBuffer    [     49     ]     =     600    ;
DataBuffer    [     50     ]     =     0      ;
DataBuffer    [     51     ]     =     0      ;
DataBuffer    [     52     ]     =     0      ;
DataBuffer    [     53     ]     =     0      ;
DataBuffer    [     54     ]     =     0      ;
DataBuffer    [     55     ]     =     0      ;
DataBuffer    [     56     ]     =     0      ;
DataBuffer    [     57     ]     =     0      ;
DataBuffer    [     58     ]     =     0      ;
DataBuffer    [     59     ]     =     0      ;
DataBuffer    [     60     ]     =     0      ;
DataBuffer    [     61     ]     =     0      ;
DataBuffer    [     62     ]     =     0      ;
DataBuffer    [     63     ]     =     0      ;
DataBuffer    [     64     ]     =     0      ;
DataBuffer    [     65     ]     =     0      ;
DataBuffer    [     66     ]     =     0      ;
DataBuffer    [     67     ]     =     0      ;
DataBuffer    [     68     ]     =     0      ;
DataBuffer    [     69     ]     =     0      ;
DataBuffer    [     70     ]     =     0      ;
DataBuffer    [     71     ]     =     0      ;
DataBuffer    [     72     ]     =     0      ;
DataBuffer    [     73     ]     =     0      ;
DataBuffer    [     74     ]     =     0      ;
DataBuffer    [     75     ]     =     600    ;
DataBuffer    [     76     ]     =     600    ;
DataBuffer    [     77     ]     =     600    ;
DataBuffer    [     78     ]     =     600    ;
DataBuffer    [     79     ]     =     600    ;
DataBuffer    [     80     ]     =     600    ;
DataBuffer    [     81     ]     =     600    ;
DataBuffer    [     82     ]     =     600    ;
DataBuffer    [     83     ]     =     600    ;
DataBuffer    [     84     ]     =     600    ;
DataBuffer    [     85     ]     =     600    ;
DataBuffer    [     86     ]     =     600    ;
DataBuffer    [     87     ]     =     600    ;
DataBuffer    [     88     ]     =     600    ;
```



```
DataBuffer    [    89    ]    =    600    ;
DataBuffer    [    90    ]    =    600    ;
DataBuffer    [    91    ]    =    600    ;
DataBuffer    [    92    ]    =    600    ;
DataBuffer    [    93    ]    =    600    ;
DataBuffer    [    94    ]    =    600    ;
DataBuffer    [    95    ]    =    600    ;
DataBuffer    [    96    ]    =    600    ;
DataBuffer    [    97    ]    =    600    ;
DataBuffer    [    98    ]    =    600    ;
DataBuffer    [    99    ]    =    600    ;
DataBuffer    [    100   ]    =    0      ;
DataBuffer    [    101   ]    =    0      ;
DataBuffer    [    102   ]    =    0      ;
DataBuffer    [    103   ]    =    0      ;
DataBuffer    [    104   ]    =    0      ;
DataBuffer    [    105   ]    =    0      ;
DataBuffer    [    106   ]    =    0      ;
DataBuffer    [    107   ]    =    0      ;
DataBuffer    [    108   ]    =    0      ;
DataBuffer    [    109   ]    =    0      ;
DataBuffer    [    110   ]    =    0      ;
DataBuffer    [    111   ]    =    0      ;
DataBuffer    [    112   ]    =    0      ;
DataBuffer    [    113   ]    =    0      ;
DataBuffer    [    114   ]    =    0      ;
DataBuffer    [    115   ]    =    0      ;
DataBuffer    [    116   ]    =    0      ;
DataBuffer    [    117   ]    =    0      ;
DataBuffer    [    118   ]    =    0      ;
DataBuffer    [    119   ]    =    0      ;
DataBuffer    [    120   ]    =    0      ;
DataBuffer    [    121   ]    =    0      ;
DataBuffer    [    122   ]    =    0      ;
DataBuffer    [    123   ]    =    0      ;
DataBuffer    [    124   ]    =    0      ;
DataBuffer    [    125   ]    =    600    ;
DataBuffer    [    126   ]    =    600    ;
DataBuffer    [    127   ]    =    600    ;
DataBuffer    [    128   ]    =    600    ;
DataBuffer    [    129   ]    =    600    ;
DataBuffer    [    130   ]    =    600    ;
DataBuffer    [    131   ]    =    600    ;
DataBuffer    [    132   ]    =    600    ;
DataBuffer    [    133   ]    =    600    ;
DataBuffer    [    134   ]    =    600    ;
DataBuffer    [    135   ]    =    600    ;
DataBuffer    [    136   ]    =    600    ;
DataBuffer    [    137   ]    =    600    ;
DataBuffer    [    138   ]    =    600    ;
DataBuffer    [    139   ]    =    600    ;
DataBuffer    [    140   ]    =    600    ;
DataBuffer    [    141   ]    =    600    ;
DataBuffer    [    142   ]    =    600    ;
DataBuffer    [    143   ]    =    600    ;
DataBuffer    [    144   ]    =    600    ;
DataBuffer    [    145   ]    =    600    ;
DataBuffer    [    146   ]    =    600    ;
DataBuffer    [    147   ]    =    600    ;
DataBuffer    [    148   ]    =    600    ;
DataBuffer    [    149   ]    =    600    ;
DataBuffer    [    150   ]    =    0      ;
DataBuffer    [    151   ]    =    0      ;
```

```
DataBuffer    [      152    ]       =     0      ;
DataBuffer    [      153    ]       =     0      ;
DataBuffer    [      154    ]       =     0      ;


*/

       // Database codes added!!!
       // open a database and create it if it does not exist
       //FrmAlert (OpenDBAlert); // alert of open DB
       gDatabase = DmOpenDatabaseByTypeCreator( kType, kCreator, dmModeWrite);
       if (!gDatabase) {
              //FrmAlert (DBCreatedAlert); // alert of create DB
              err = DmCreateDatabase(kCardNumber, kName, kCreator, kType, false);
              if (!err) {
                     gDatabase = DmOpenDatabaseByTypeCreator( kType, kCreator,
dmModeWrite);
                     if (!gDatabase)
                            err = DmGetLastErr();
              }
       }

       DmOpenDatabaseInfo(gDatabase, &theLocalID, NULL, NULL, &theCardNum, NULL);
       DmDatabaseInfo(theCardNum, theLocalID, NULL, &theAttributes, NULL, NULL, NULL,
NULL, NULL, NULL, NULL, NULL, NULL);
       theAttributes |= dmHdrAttrBackup;
       DmSetDatabaseInfo(theCardNum, theLocalID, NULL, &theAttributes, NULL, NULL,
NULL, NULL, NULL, NULL, NULL, NULL, NULL);


   return err;

    //return errNone;
}

// FUNCTION: AppStop
//
// DESCRIPTION: Save the current state of the application.

static void AppStop(void)
{
    // Write the saved preferences / saved-state information.  This
    // data will be saved during a HotSync backup.
    PrefSetAppPreferences(
        appFileCreator, appPrefID, appPrefVersionNum,
        &g_prefs, sizeof(VtChtGrphPreferenceType), true);

    // Close all the open forms.
    FrmCloseAllForms();

    // database codes added!!!
       // close database
       if (gDatabase)
              DmCloseDatabase(gDatabase);

       MemPtrFree(watchdog_time);
       MemPtrFree(elapsed_time);
    MemPtrFree(plotelapsed_time);
       StopAlarm();
}

// all code from here to end of file should use no global variables
#pragma warn_a5_access on
```

```
// FUNCTION: RomVersionCompatible
//
// DESCRIPTION:
//
// This routine checks that a ROM version is meet your minimum
// requirement.
//
// PARAMETERS:
//
// requiredVersion
//     minimum rom version required
//     (see sysFtrNumROMVersion in SystemMgr.h for format)
//
// launchFlags
//     flags that indicate if the application UI is initialized
//     These flags are one of the parameters to your app's PilotMain
//
// RETURNED:
//     error code or zero if ROM version is compatible

static Err RomVersionCompatible(UInt32 requiredVersion, UInt16 launchFlags)
{
    UInt32 romVersion;

    // See if we're on in minimum required version of the ROM or later.
    FtrGet(sysFtrCreator, sysFtrNumROMVersion, &romVersion);
    if (romVersion < requiredVersion)
    {
        if ((launchFlags &
            (sysAppLaunchFlagNewGlobals | sysAppLaunchFlagUIApp)) ==
            (sysAppLaunchFlagNewGlobals | sysAppLaunchFlagUIApp))
        {
            FrmAlert (RomIncompatibleAlert);

            // Palm OS 1.0 will continuously relaunch this app unless
            // we switch to another safe one.
            if (romVersion <= kPalmOS10Version)
            {
                AppLaunchWithCommand(
                    sysFileCDefaultApp,
                    sysAppLaunchCmdNormalLaunch, NULL);
            }
        }

        return sysErrRomIncompatible;
    }

    return errNone;
}

// FUNCTION: VtChtGrphPalmMain
//
// DESCRIPTION: This is the main entry point for the application.
//
// PARAMETERS:
//
// cmd
//     word value specifying the launch code.
//
// cmdPB
//     pointer to a structure that is associated with the launch code
//
```



```
// launchFlags
//     word value providing extra information about the launch
//
// RETURNED:
//     Result of launch, errNone if all went OK

static UInt32 VtChtGrphPalmMain(
    UInt16 cmd,
    MemPtr /*cmdPBP*/,
    UInt16 launchFlags)
{
    Err error;
       UInt16 k;
    error = RomVersionCompatible (ourMinVersion, launchFlags);
    if (error) return (error);

    switch (cmd)
    {
    case sysAppLaunchCmdNormalLaunch:
        error = AppStart();
        if (error)
            return error;

        // start application by opening the main form
        // and then entering the main event loop
        FrmGotoForm(MainForm);

        // assignment initial values to global variables
        EKG_ARRAY_P = 0;
             RES_ARRAY_P = 0;
             Connected = false;

             for(k=0;k<=MaxEdgeIndexSize-1;k++)
             {
             EdgeLocation[k]=0;
             }
        for(k=0;k<=MaxDiffArraySize-1;k++)
        {
        DiffArray[k]=0;
        }
        AppEventLoop();

        AppStop();
        break;

    default:
        break;
    }

    return errNone;
}

/***********************************************************************
 *
 * FUNCTION:        DrawBitmap
 *
 * DESCRIPTION:     Get and draw a bitmap at a specified location
 *
 * PARAMETERS:      resID        -- bitmap resource id
 *                              x, y         -- bitmap origin relative to current
window
 *
 * RETURNED:  nothing.
```

```
 *
 * REVISION HISTORY:
 *                  Name   Date          Description
 *                  ----   ----          -----------
 *                  vmk    10/9/95       Initial Revision
 *
 ***********************************************************************/
static void DrawBitmap(Int16 resID, Int16 x, Int16 y)
{
       MemHandle     resH;
       BitmapPtr     resP;


       resH = DmGetResource( bitmapRsc, resID );
       //ErrFatalDisplayIf( !resH, "Missing bitmap" );
       resP = MemHandleLock(resH);
       WinDrawBitmap (resP, x, y);
       MemPtrUnlock(resP);
       DmReleaseResource( resH );

}

// FUNCTION: GetData
//
// DESCRIPTION: This routine receives the data from the PIC.
//
// PARAMETERS:
//
// None
//
static void GetData()
{
              // Initialize local variables
              UInt16 *EKG;
              UInt16 *RES;
              UInt16 portId;
              UInt16 index;
              Err err, err2;
              Char recmsg[27];
              UInt32 lineStatus;
              UInt16 lineErrs;
              Char *error = "Error Opening Port\n";
              Char   Send0x05[] = " \n";
              UInt32 timeout=SysTicksPerSecond()*timeouttime;
              UInt8
test_value[26]={0x1F,0x3F,0x5F,0x7F,0x1F,0x3F,0x5F,0x7F,0x1F,0x3F,0x1F,0x3F,0x5F,0x7F,
0x1F,0x3F,0x5F,0x7F,0x1F,0x3F,
                                                 0x1F,0x3F,0x5F,0x7F,0x1F,0x3F};

//0x1F,0x3F,0x5F,0x7F,0x1F,0x3F,0x5F,0x7F,0x1F,0x3F,0x1F,0x3F};//,0x5F,0x7F,0x1F,0x3F,
0x5F,0x7F,0x1F,0x3F,

//0x1F,0x3F,0x5F,0x7F,0x1F,0x3F,0x5F,0x7F};
              UInt32 num_received;


              // allocates memory for storing EKG and RES data
              EKG = (UInt16 *)MemPtrNew(sizeof(UInt16));
              RES = (UInt16 *)MemPtrNew(sizeof(UInt16));

              err = SrmOpen(serPortCradleRS232Port /* port */, 62500, /* baud */
&portId);
              if (err == errNone)
```

```
            {
             // display error message here.

                  num_received = 0;

                  SrmControl(portId, srmCtlSetDTRAsserted, (void *)false, (UInt16
*)sizeof(Boolean));
                  Send0x05[0] = 0x05;
                  SrmSend(portId, Send0x05, 1, &err2);
                  SrmSendWait(portId);

                  num_received = SrmReceive (portId, recmsg, 26, timeout, &err2);

                  if(err2 == serErrTimeOut)
                  {
                        Connected = false;
                  }
                  else
                  {
                        Connected = true;
                  }

                  recmsg[26] = '\0';

                  if (err2 == serErrLineErr)
                  {
                  SrmGetStatus(portId, &lineStatus, &lineErrs);
                  // test for framing or parity error.
                  if (lineErrs & (serLineErrorFraming | serLineErrorParity))
                  {
                 //framing or parity error occurred. Do something.
                   }
                        SrmClearErr(portId);
                  }

                   for(index=0; index < num_received; index++)
                   {
                        test_value[index]=recmsg[index];
                   }

                   for(index=0; index < num_received; index++)
                   {
                        //throw away two consecutive 0x1F
                        if((test_value[index] == 0x1F) && (test_value[index+1] ==
0x1F))
                        {
                              index++;
                              continue;
                        }

                        EncodeValue(test_value[index],EKG,RES);

                   }
                  SrmClose(portId);
            }
```

```
            //record our open status in global.
            else
            {
                  SrmClose(portId);
            }

            MemPtrFree(EKG);
            MemPtrFree(RES);
            //Delay=false;

}
// FUNCTION: GetDataBaseData
//
// DESCRIPTION: This routine receives the data from the PIC.
//
// PARAMETERS:
//
// None
//
static void GetDataBaseData()
{
            // Initialize local variables
            UInt16 *EKG;
            UInt16 *RES;
            UInt16 portId;
            UInt16 index;
            Err err, err2;
            Char recmsg[27];
            UInt32 lineStatus;
            UInt16 lineErrs;
            Char *error = "Error Opening Port\n";
            Char   Send0x05[] = " \n";
            UInt32 timeout=SysTicksPerSecond()*timeouttime;
            UInt8
test_value[26]={0x1F,0x3F,0x5F,0x7F,0x1F,0x3F,0x5F,0x7F,0x1F,0x3F,0x1F,0x3F,0x5F,0x7F,
0x1F,0x3F,0x5F,0x7F,0x1F,0x3F,
                                                0x1F,0x3F,0x5F,0x7F,0x1F,0x3F};

//0x1F,0x3F,0x5F,0x7F,0x1F,0x3F,0x5F,0x7F,0x1F,0x3F,0x1F,0x3F};//,0x5F,0x7F,0x1F,0x3F,
0x5F,0x7F,0x1F,0x3F,

//0x1F,0x3F,0x5F,0x7F,0x1F,0x3F,0x5F,0x7F};
            UInt32 num_received;


            // allocates memory for storing EKG and RES data
            EKG = (UInt16 *)MemPtrNew(sizeof(UInt16));
            RES = (UInt16 *)MemPtrNew(sizeof(UInt16));

            err = SrmOpen(serPortCradleRS232Port /* port */, 62500, /* baud */
&portId);
            if (err == errNone)
            {
             // display error message here.


                  num_received = 0;


                  SrmControl(portId, srmCtlSetDTRAsserted, (void *)false, (UInt16
*)sizeof(Boolean));
                  Send0x05[0] = 0x05;
```

```
        SrmSend(portId, Send0x05, 1, &err2);
        SrmSendWait(portId);


        num_received = SrmReceive (portId, recmsg, 26, timeout, &err2);


        if(err2 == serErrTimeOut)
        {
               Connected = false;
        }
        else
        {
               Connected = true;
        }

        recmsg[26] = '\0';


        if (err2 == serErrLineErr)
        {
        SrmGetStatus(portId, &lineStatus, &lineErrs);
        // test for framing or parity error.
        if (lineErrs & (serLineErrorFraming | serLineErrorParity))
        {
       //framing or parity error occurred. Do something.
         }
               SrmClearErr(portId);
        }

         for(index=0; index < num_received; index++)
         {
               test_value[index]=recmsg[index];
         }


         for(index=0; index < num_received; index++)
         {
               //throw away two consecutive 0x1F
               if((test_value[index] == 0x1F) && (test_value[index+1] ==
0x1F))
               {
                      index++;
                      continue;
               }


               EncodeDBValue(test_value[index],EKG,RES);


         }
        SrmClose(portId);
}

//record our open status in global.
else
{
        SrmClose(portId);
}

MemPtrFree(EKG);
MemPtrFree(RES);
//Delay=false;
```

```
}

// FUNCTION: EncodeValue
//
// DESCRIPTION: This routine initializes the MainForm form.
//
// PARAMETERS:
//
// test_value
//     integer of the test value
//
// RETURNED:
//            a decoded msg string

void EncodeValue(UInt8 num_array, UInt16 *EKG, UInt16 *RES)
{
       UInt8 header;
       UInt8 value;
       UInt16 converted_value;

       header = num_array >> 5;
       value = (num_array << 3) >> 3;
       converted_value = value; // converts from UInt8 to UInt16

       // EKG_L 000
       // EKG_H 001
       // RES_L 010
       // RES_H 011



       if(header == 0x00)
              {
                     *EKG = (converted_value&(0x1F));
              }
              else if (header == 0x01)
              {
                     *EKG += ((converted_value << 5)&(0x3E0));
                            EKG_ARRAY[(EKG_ARRAY_P)%EKG_ARRAYSIZE] = (*EKG)&(0x3FF);
                            EKG_ARRAY_P++;
                            if(EKG_ARRAY_P>=EKG_ARRAYSIZE)
                                   EKG_ARRAY_P=0;
              }
              else if (header == 0x02)
              {
                     *RES = (converted_value&(0x1F));

              }
              else if (header == 0x03)
              {
                     *RES += ((converted_value << 5)&(0x3E0));

                            RES_ARRAY[(RES_ARRAY_P)%EKG_ARRAYSIZE] = (*RES)&(0x3FF);
                            RES_ARRAY_P++;
              }
}
// FUNCTION: EncodeValue
//
// DESCRIPTION: This routine initializes the MainForm form.
//
// PARAMETERS:
//
```

```
// test_value
//     integer of the test value
//
// RETURNED:
//            a decoded msg string

void EncodeDBValue(UInt8 num_array, UInt16 *EKG, UInt16 *RES)
{
       UInt8 header;
       UInt8 value;
       UInt16 converted_value;
       char eadb[20];
       header = num_array >> 5;
       value = (num_array << 3) >> 3;
       converted_value = value; // converts from UInt8 to UInt16

       // EKG_L 000
       // EKG_H 001
       // RES_L 010
       // RES_H 011



       if(header == 0x00)
       {
              *EKG = (converted_value&(0x1F));
       }
       else if (header == 0x01)
       {
              *EKG += ((converted_value << 5)&(0x3E0));
              EKG_ARRAY_DB[(EKG_ARRAY_P_DB)%(DB_EKG_Len)] = (*EKG)&(0x3FF);

              /*if(EKG_ARRAY_DB[(EKG_ARRAY_P_DB)%(DB_EKG_Len)] <= 20)
              {

              StrIToA(eadb,EKG_ARRAY_DB[EKG_ARRAY_P_DB%DB_EKG_Len]);
              WinDrawChars(eadb,StrLen(eadb),(EKG_ARRAY_P_DB*30)%150,40);
              }*/

              //StrIToA(eadb,EKG_ARRAY_DB[EKG_ARRAY_P_DB%(DB_EKG_Len+1)]);
              //WinDrawChars(eadb,StrLen(eadb),(EKG_ARRAY_P_DB*30)%150,60);

              EKG_ARRAY_P_DB++;


              if(EKG_ARRAY_P_DB>=DB_EKG_Len)
              {
                     DataComplete=true;
                     GetDBData=false;
                     EKG_ARRAY_P_DB=0;
              }
       }
              /*else if (header == 0x02)
              {
                     *RES = (converted_value&(0x1F));
              }
              /*else if (header == 0x03)
              {
                     *RES += ((converted_value << 5)&(0x3E0));

                            RES_ARRAY_DB[(RES_ARRAY_P)%DB_RES_Len] = (*RES)&(0x3FF);
                            RES_ARRAY_P_DB++;
```



```
                            if(RES_ARRAY_P_DB>=DB_RES_Len)
                            {
                                   RES_ARRAY_P_DB=0;
                                   DataComplete=true;
                                   GetDBData=false;
                            }
              }*/
}




// FUNCTION: PilotMain
//
// DESCRIPTION: This is the main entry point for the application.
//
// PARAMETERS:
//
// cmd
//     word value specifying the launch code.
//
// cmdPB
//     pointer to a structure that is associated with the launch code
//
// launchFlags
//     word value providing extra information about the launch.
//
// RETURNED:
//     Result of launch, errNone if all went OK

UInt32 PilotMain(UInt16 cmd, MemPtr cmdPBP, UInt16 launchFlags)
{
    return VtChtGrphPalmMain(cmd, cmdPBP, launchFlags);


}

// turn a5 warning off to prevent it being set off by C++
// static initializer code generation
#pragma warn_a5_access reset

/**************************************************************************
 *
 * FUNCTION:        ResetDataBuffer
 *
 * DESCRIPTION:     *
 * PARAMETERS:      resID        -- bitmap resource id
 *                               x, y         -- bitmap origin relative to current
window
 *
 * RETURNED:  nothing.
 *
 * REVISION HISTORY:
 *                  Name   Date          Description
 *                  ----   ----          -----------
 *                  vmk    10/9/95       Initial Revision
 *
 **************************************************************************/
 static void ResetDataBuffer()
 {
       UInt16 k;
       for(k=0;k<=EKG_ARRAYSIZE;k++)
```

```
        EKG_ARRAY[k]=0;
 }

/**********************************************************************
 *
 * FUNCTION:        ClearLine
 *
 * DESCRIPTION:     Clears a rectangle of width 5 for plotting new data
 *
 * PARAMETERS:      Nothing
 *
 * RETURNED: nothing.
 *
 * REVISION HISTORY:
 *                  Name   Date         Description
 *                  ----   ----         -----------
 *                  SHT        16/11/02 Initial Revision
 *
 **********************************************************************/
static void ClearLine(UInt16 xlocation, UInt16 width)
{
       RectangleType bounds;
       bounds.topLeft.x = xlocation;
       bounds.topLeft.y = GridTopOrigin;
       bounds.extent.x = width;
       bounds.extent.y = GridHeight;
       WinEraseRectangle(&bounds, 0);
}
/**********************************************************************
 *
 * FUNCTION:        CheckConnect
 *
 * DESCRIPTION:
 *
 * PARAMETERS:      Nothing
 *
 * RETURNED: nothing.
 *
 * REVISION HISTORY:
 *                  Name   Date         Description
 *                  ----   ----         -----------
 *                  SHT        16/11/02 Initial Revision
 *
 **********************************************************************/
static void CheckConnect()
{
       if(Connected==true)
              DrawBitmap(PlugBitmap,2,18);
       if(Connected==false)
              DrawBitmap(NoPlugBitmap,2,18);

}
/**********************************************************************
 *
 * FUNCTION:        ScaleData
 *
 * DESCRIPTION:     Collects Data Points and Plots them on the grid
 *
 * PARAMETERS:      Nothing
 *
 * RETURNED: nothing.
 *
 * REVISION HISTORY:
```

```
 *                  Name   Date          Description
 *                  ----   ----          -----------
 *                  SHT        16/11/02  Initial Revision
 *
 ***********************************************************************/
UInt16 ScaleData(UInt16 UnformattedData)
{
       UInt16 Scaleddata;

//     Scaleddata = UnformattedData///Maxof Data?
       Scaleddata = GridBottomOffset+((UnformattedData/11)*-1);

       return Scaleddata;
}
/***********************************************************************
 *
 * FUNCTION:        PlotData
 *
 * DESCRIPTION:     Collects Data Points and Plots them on the grid
 *
 * PARAMETERS:      Nothing
 *
 * RETURNED:  nothing.
 *
 * REVISION HISTORY:
 *                  Name   Date          Description
 *                  ----   ----          -----------
 *                  SHT        16/11/02  Initial Revision
 *
 ***********************************************************************/
//static void PlotData(UInt16 NumberSamples)
static void PlotData()
{
 UInt16 CurrentEKGdata, NextEKGdata;

Int16 j;
//Int16 k;
Int16 Delta;
Int16 p;
char ekgchar[20];

 UInt16 Diff;
 char DiffChar[20];


//CurrentLocation = i;
//for(k=0; k<=NumberSamples; k++)
//{
 if(i == GridWidth)
ClearLine(2,1);
 ClearLine(i+2,4);

// StrIToA(DiffChar,Diff);


       //WinDrawChars("Bfr Urn",StrLen("Bfr Urn"),120,144);
       //WinDrawChars(DiffChar,StrLen(DiffChar),140,140);
       if(GridOn==true)
       {
              if((i%2 == 0)&&(!(((i+1)%22==1)||((i+11)%22==0))))//Even Grid Points
              {
                     p=GridTopOrigin;
                     while(p<=GridHeight+GridTopOrigin)
```

```
                {
                        WinDrawPixel(i+leftOrigin,p);
                        p=p+22;
                }
        }
        else if((i%2 == 1)&&(!(((i+1)%22==1)||((i+11)%22==0))))
        {
                p=GridTopOrigin+11;
                while(p<=GridHeight+GridTopOrigin)
                {
                        WinDrawPixel(i+leftOrigin,p);
                        p=p+22;
                }
        }
        if((i+1)%22==1)
        {
        p=GridTopOrigin;
                while(p<GridHeight+GridTopOrigin)
                {
                        WinDrawPixel(i+leftOrigin,p);
                        p=p+2;
                }
        }

        if((i+11)%22==0)
        {
                p=GridTopOrigin+1;
                while(p<GridHeight+GridTopOrigin)
                {
                        WinDrawPixel(i+leftOrigin,p);
                        p=p+2;
                }
        }

}

//Interpolation Code
        if(DataIndex >= EKG_ARRAYSIZE)
          DataIndex = 0;
     if(DataIndex == EKG_ARRAYSIZE-1)
     {
        CurrentEKGdata = ScaleData(EKG_ARRAY[DataIndex]);
        Delta=ScaleData(EKG_ARRAY[0])-CurrentEKGdata;
        }
     else
     {
             CurrentEKGdata = ScaleData(EKG_ARRAY[DataIndex]);
             NextEKGdata = ScaleData(EKG_ARRAY[DataIndex+PointDelta]);
                Delta=NextEKGdata - CurrentEKGdata;
     }

        if(Delta < 0)
        {
                for(j=Delta+1; j<=-1; j++)

if((CurrentEKGdata+j<=GridTopOrigin+GridHeight)&&(CurrentEKGdata+j>=GridTopOrig
in))
                WinDrawPixel(i+leftOrigin,CurrentEKGdata+j);
        }
        if(Delta >0)
        {
                for(j=1; j<=Delta-1; j++)
```

```
	if((CurrentEKGdata+j<=GridTopOrigin+GridHeight)&&(CurrentEKGdata+j>=GridTopOrig
in))
			WinDrawPixel(i+leftOrigin,CurrentEKGdata+j);
		}


	if((CurrentEKGdata<=GridTopOrigin+GridHeight)&&(CurrentEKGdata>=GridTopOrigin))
		WinDrawPixel(i+leftOrigin,CurrentEKGdata);
		//StrIToA(ekgchar,EKG_ARRAY[DataIndex]);
		//WinDrawChars(ekgchar,StrLen(ekgchar),i+leftOrigin,CurrentEKGdata);

	DataIndex=DataIndex+PointDelta;
	//DataIndex++;

	i++;


	//}

}
/***********************************************************************
 *
 * FUNCTION:		ShowDateTime
 *
 * DESCRIPTION:	Computes the Date and Time of Application start and displays it on
the GUI
 *
 * PARAMETERS:		Nothing
 *
 * RETURNED:	nothing.
 *
 * REVISION HISTORY:
 *			Name	Date		Description
 *			----	----		-----------
 *			SHT		16/11/02	Initial Revision
 *
 ***********************************************************************/

static void ShowDateTime()
{
	UInt32 seconds;
	DateTimeType TheDate;
	char DateString[dateStringLength];
	char TimeString[timeStringLength];

	seconds=TimGetSeconds();
	TimSecondsToDateTime(seconds,&TheDate);
	DateToAscii(TheDate.month,TheDate.day,TheDate.year,dfDMYWithSlashes,DateString)
;
	TimeToAscii(TheDate.hour,TheDate.minute,tfColon24h,TimeString);

	WinDrawChars(TimeString,StrLen(TimeString),117,28);
	WinDrawChars(DateString,StrLen(DateString),111,18);

}
/***********************************************************************
 *
 * FUNCTION:		ResetPlotArea
 *
 * DESCRIPTION:	Clears the Plot Area and Redraws the Grid
 *
 * PARAMETERS:		Nothing
```

```
 *
 * RETURNED: nothing.
 *
 * REVISION HISTORY:
 *                 Name   Date          Description
 *                 ----   ----          -----------
 *                 SHT        16/11/02  Initial Revision
 *
 ***********************************************************************/
static void ResetPlotArea()
{
       RectangleType bounds;
       bounds.topLeft.x = 2;
       bounds.topLeft.y = 40;
       bounds.extent.x = GridWidth;
       bounds.extent.y = GridHeight;
       WinEraseRectangle(&bounds, 0);
       if(GridOn==true)
       DrawBitmap(GridBitmap,2,40);
       i=0;
}
/***********************************************************************
 *
 * FUNCTION:       GridSelect
 *
 * DESCRIPTION:
 *
 * PARAMETERS:     Nothing
 *
 * RETURNED: nothing.
 *
 * REVISION HISTORY:
 *                 Name   Date          Description
 *                 ----   ----          -----------
 *                 SHT        16/11/02  Initial Revision
 *
 ***********************************************************************/
static void GridSelect()
{

       ControlType *checkP;
       Int16 GridFlag;

       checkP=GetObjectPtr(MainGridCheckbox);
       GridFlag=CtlGetValue(checkP);
       if(GridFlag == 0)
       {
       GridOn=false;
       ResetPlotArea();
       }
       if(GridFlag ==1)
       {
       GridOn=true;
       ResetPlotArea();
       }

}
/***********************************************************************
 *
 * FUNCTION:       ZoomSelect
 *
 * DESCRIPTION:    *
 * PARAMETERS:     Nothing
```



```
 *
 * RETURNED: nothing.
 *
 * REVISION HISTORY:
 *                 Name   Date         Description
 *                 ----   ----         -----------
 *                 SHT        16/11/02 Initial Revision
 *
 ***********************************************************************/
static void ZoomSelect()
{

       ControlType *checkP;
       Int16 ZoomFlag;

       checkP=GetObjectPtr(MainZoomCheckbox);
       ZoomFlag=CtlGetValue(checkP);
       if(ZoomFlag == 1)
       {
       PointDelta = 1;
       ResetPlotArea();
       DataIndex = 0;
       EKG_ARRAY_P = 0;
       Sync=true;
       Delay=true;
       Zoom = true;

       }
       if(ZoomFlag ==0)
       {
       PointDelta = 2;
       ResetPlotArea();
       DataIndex = 0;
       EKG_ARRAY_P = 0;
       Sync=true;
       Delay=true;
       Zoom = false;

       }

}
/***********************************************************************
 *
 * FUNCTION:        PeakDetect
 *
 * DESCRIPTION:     Takes the difference (ie derivative) of the data and stores
 *                        the location of the peaks
 *
 * PARAMETERS:      Nothing
 *
 * RETURNED: nothing.
 *
 * REVISION HISTORY:
 *                 Name   Date         Description
 *                 ----   ----         -----------
 *                 SHT        16/11/02 Initial Revision
 *
 ***********************************************************************/
 static void PeakDetect(UInt16 Index)
 {
        Int16 Diff=0;
        char countchar[20];
        char diffchar[20];
```



```
        Diff = (EKG_ARRAY[Index+1]-EKG_ARRAY[Index]);//PlotTime2;
        //Diff=DataBuffer[Index+1]-DataBuffer[Index];
          if((Diff<=ThresholdMinus)&&(trigger==false))
         //if((EKG_ARRAY[Index+1]<ThresholdMinus)&&(trigger==false))
         {
               EdgeLocation[EdgeIndex%MaxEdgeIndexSize] = i;

               EdgeIndex++;
               trigger = true;
               InvertHeart();
               HRTimeOut =0;
         }
         if((Diff>=ThresholdPlus)&&(trigger==true))
         //if((EKG_ARRAY[Index+1]>=ThresholdPlus)&&(trigger==true))
         {
            trigger=false;
               InvertHeart();
               PlaySound();
               HRTimeOut =0;

         //    count = count +1;
         }

        /*StrIToA(countchar,EdgeIndex);
        if(Diff <=0)
        {
               StrIToA(diffchar,Diff);
               WinDrawChars("       ",StrLen("       "),115,1);
               WinDrawChars(diffchar,StrLen(diffchar),115,1);
        }
        else
        {
               StrIToA(diffchar,Diff);
               WinDrawChars("       ",StrLen("       "),135,5);
               WinDrawChars(diffchar,StrLen(diffchar),135,5);
        }
        WinDrawChars("          ",StrLen("          "),120,140);
        WinDrawChars(countchar,StrLen(countchar),120,140);*/

}
/***********************************************************************
 *
 * FUNCTION:        CalcHeartRate
 *
 * DESCRIPTION:
 *
 * PARAMETERS:      Nothing
 *
 * RETURNED:  nothing.
 *
 * REVISION HISTORY:
 *                  Name   Date          Description
 *                  ----   ----          -----------
 *                  SHT        16/11/02  Initial Revision
 *
 ***********************************************************************/

static void CalcHeartRate()
{
        UInt16 k;
        double Sum=0;
        UInt32 Rate;
```



```
        char HrtRt[20];

        if(EdgeIndex>=2)
        {
                for(k=0;k<=EdgeIndex-2;k++)
                {
                        DiffArray[k]=EdgeLocation[k+1]-EdgeLocation[k];
                }
                for(k=0;k<=EdgeIndex-2;k++)
                {
                        Sum=DiffArray[k]+Sum;
                }
                if(Zoom==true)
                Rate=60/((Sum/(EdgeIndex-1))*PlotTime1);

                if(Zoom==false)
                Rate=60/((Sum/(EdgeIndex-1))*PlotTime2);

                if(Rate<=300)
                {
                        StrIToA(HrtRt,Rate);
                        WinDrawChars("        ",StrLen("        "),44,21);
                        WinDrawChars(HrtRt,StrLen(HrtRt),44,21);
                }
        }

        else
        {
        WinDrawChars("        ",StrLen("        "),44,21);
        WinDrawChars("NA",StrLen("NA"),44,21);
        }
        EdgeIndex=0;

}
/***********************************************************************
 *
 * FUNCTION:        InvertHeart
 *
 * DESCRIPTION:     Inverts the Heart Bitmap
 *
 * PARAMETERS:      Nothing
 *
 * RETURNED:  nothing.
 *
 * REVISION HISTORY:
 *                  Name   Date         Description
 *                  ----   ----         -----------
 *                  SHT        16/11/02  Initial Revision
 *
 ***********************************************************************/
 static void InvertHeart()
 {
        RectangleType bounds;
        bounds.topLeft.x = 22;
        bounds.topLeft.y = 19;
        bounds.extent.x = 19;
        bounds.extent.y = 17;
        WinInvertRectangle(&bounds, 0);
}
/***********************************************************************
 *
 * FUNCTION:        PlotBRate()
 *
```



```
 * DESCRIPTION:     *
 * PARAMETERS:      Nothing
 *
 * RETURNED: nothing.
 *
 * REVISION HISTORY:
 *                  Name   Date          Description
 *                  ----   ----          -----------
 *                  SHT        16/11/02  Initial Revision
 *
 ***********************************************************************/
 static void PlotBRate()
 {
       char BRatechar[20];
       UInt16 BRate=0;
       UInt16 Sum,AveB;
       UInt8 s;
       //if(RES_ARRAY_P>=3)
       //{
       //      for (s=1;s<=3;s++)
       //      {
                     BRate = RES_ARRAY[(RES_ARRAY_P)%EKG_ARRAYSIZE-1]*5;
              //     Sum = Sum+BRate;
       //      }

        //    AveB = (Sum/3)*5;
              if((BSum != BRate)&&(BRate<=99))
              //{
              {
                     StrIToA(BRatechar,BRate);
                     WinDrawChars("         ",StrLen("         "),79,21);
                     WinDrawChars(BRatechar,StrLen(BRatechar),79,21);
              }
        //    }
              BSum = BRate;
       //}
 }


 /***********************************************************************
 *
 * FUNCTION:     PlaySound
 *
 * DESCRIPTION:
 *
 * PARAMETERS:   nothing
 *
 * RETURNED:     nothing
 *
 * REVISION HISTORY:
 *                  Name   Date          Description
 *                  ----   ----          -----------
 *
 *
 ***********************************************************************/
static void PlaySound()
{
       SndCommandType             sndCmd;

              sndCmd.cmd = sndCmdFrqOn;
              sndCmd.param1 = 880;
              sndCmd.param2 = 1;
              sndCmd.param3 = SoundAmp;
```

```
            SndDoCmd( 0, &sndCmd, true);

}

 /**********************************************************************
 *
 * FUNCTION:     PlayAlarm
 *
 * DESCRIPTION:
 *
 * PARAMETERS:   nothing
 *
 * RETURNED:     nothing
 *
 * REVISION HISTORY:
 *                  Name   Date          Description
 *                  ----   ----          -----------
 *
 *
 **********************************************************************/
static void StopAlarm()
{
      SndCommandType             sndCmd;

            sndCmd.cmd = sndCmdFrqOn;
            sndCmd.param1 = 880;
            sndCmd.param2 = 60000;
            sndCmd.param3 = SoundAmp;

            SndDoCmd( 0, &sndCmd, true);

}
 /**********************************************************************
 *
 * FUNCTION:     PlayAlarm
 *
 * DESCRIPTION:
 *
 * PARAMETERS:   nothing
 *
 * RETURNED:     nothing
 *
 * REVISION HISTORY:
 *                  Name   Date          Description
 *                  ----   ----          -----------
 *
 *
 **********************************************************************/
static void StopAlarm()
{
      SndCommandType             sndCmd;

            sndCmd.cmd = sndCmdQuiet;
            sndCmd.param1 = 0;
            sndCmd.param2 = 0;
            sndCmd.param3 = 0;

            SndDoCmd( 0, &sndCmd, true);

}
 /**********************************************************************
 *
```

```
 * FUNCTION:     CompleteBar
 *
 * DESCRIPTION:
 *
 * PARAMETERS:   nothing
 *
 * RETURNED:     nothing
 *
 * REVISION HISTORY:
 *                  Name   Date          Description
 *                  ----   ----          -----------
 *
 *
 ***********************************************************************/
static void CompleteBar()
{

       RectangleType bounds;
       bounds.topLeft.x = 5;
       bounds.topLeft.y = 150;
       bounds.extent.x = EKG_ARRAY_P_DB/10;
       bounds.extent.y = 10;
       WinDrawRectangle (&bounds, 0);



}
```