

March 3, 2005

Prof. Lucky One  
Simon Fraser University  
8888 University Drive  
Burnaby, BC  
V5A 1S6

**RE: ENSC 440 Design Specification for a Pill Alerting System**

Dear Prof. One,

The attached document, *Design Specification for a Pill Alerting System (PAS)*, defines the various attributes of project idea for ENSC 440: Capstone Project.

We are currently in the process of designing and implementing a medical pill container that can remind patients when to take their pills and when to refill the container. The PAS consists of the interaction of three systems. The first one is the pharmacists' computer, which consists of software that presents the pharmacists with a simple-to-use GUI interface that accesses a database and interfaces with the second system which is called the "Dock". The Dock is where we place the cap of the container in order to program it with the timing information. The Dock connects to the pharmacists' computer by means of a USB cable. The information is downloaded from the computer to this system and from there to the third system, which is the circuitry beneath the cap of the container hereafter referred to as the "Cap".

The attached document outlines and discusses the functionality of our PAS in terms of the design specifications and implementation of its subsystems. This document in conjunction with the requirements specified in the *Functional Specification for a Pill Alerting System* ensures a smooth development for our final prototype of the system. Furthermore, by including a user interface specification, we facilitate the usability of our PAS with the goal of providing a user friendly interface for our users.

Our project team, MedSmart Inc., consists of three innovative, enthusiastic and industrious fourth and fifth year engineering students. The team members include Najla Dadmand (CEO), Tiffany Ho (CFO) and Michael Fang (CTO). Each member brings their diverse knowledge and talents to the team. If you have any questions or concerns regarding to our project proposal, please feel free contact us at ensc440-medsmart@sfu.ca.

Sincerely,



Michael Fang  
Chief Technology Officer  
MedSmart Inc.

Enclosure: *Design Specification for a Pill Alerting System*



# Design Specification for a Pill Alerting System

---

**Project Team:** Najla Dadmand  
Michael Fang  
Tiffany Ho

**Contact Person:** Michael Fang  
ensc440-medsmart@sfu.ca

**Submitted to:** Prof. Lucky One – ENSC 440  
Prof. Mike Sjoerdsma – ENSC 305  
School of Engineering Science  
Simon Fraser University

**Issued Date:** March 11, 2005

**Revision:** 1.0

### Executive Summary

Although medicine is advancing more and more day by day, there is the critical issue of forgetting to take them. The elderly have always suffered the most from this matter, but with the busy lifestyles of the twenty-first century, it is common for people of all ages to forget to take their pills.

Currently there are products that aim to solve this problem, however, they have disadvantages such as cost and complicated settings. We at MedSmart, believe that our product, the Pill Alerting System (PAS), will contribute significantly to this vital issue by presenting a user-friendly product that is also low in cost.

The PAS is designed such that it requires minimal effort from the users. The users do not need to learn a new technology or even program anything. All that the users see is a modified pill container that has the LED on the cap. Inside the cap, however, is carefully designed circuitry that sets off music and lights an LED when it is time to take the pills. The programming is done at the time of purchase by the pharmacists' computer.

The design specifications describe the design requirements of the overall system, ensuring a smooth transition between the functionality of the system and its development.

The PAS consists of the interaction of three systems. The first one is the pharmacists' computer, which consists of software that presents the pharmacists with a simple-to-use GUI interface that accesses a database and interfaces with the second system which is called the "Dock". The Dock is where we place the cap of the container in order to program it with the timing information. The Dock connects to the pharmacists' computer by means of a USB cable. The information is downloaded from the computer to this system and from there to the third system, which is the circuitry beneath the cap of the container hereafter referred to as the "Cap".

Each system has its own defined design specifications in terms of its component description, architectural overview and implementation. These specifications represent the requirements that the PAS prototype should follow. We have also prepared user interface specifications to ensure a user-friendly product.

## Table of Contents

Executive Summary .....	v
List of Figures .....	v
List of Tables .....	v
Glossary .....	vi
<b>1. Introduction .....</b>	<b>1</b>
1.1. The Scope .....	1
1.2. Intended Audience .....	1
<b>2. System Overview .....</b>	<b>2</b>
<b>3. Hardware Design .....</b>	<b>4</b>
3.1. The Dock Unit .....	4
3.1.1. The MC68HC908JB16 Microcontroller .....	5
3.2. The Cap Unit .....	6
3.2.1. The MC68HC908MR8 Microcontroller .....	7
3.2.2. Buzzer Circuit .....	7
<b>4. Firmware Design .....</b>	<b>9</b>
4.1. Firmware overview .....	9
4.2. Firmware in the Dock .....	9
4.2.1. Interface Code .....	9
4.2.1.1 <i>Interface Code for Computer</i> .....	9
4.2.1.2 <i>Interface Code for the Cap</i> .....	10
4.2.2. Application Code .....	10
4.3. Firmware in the Cap .....	11
4.3.1. Interface Code .....	12
4.3.2. Application Code .....	12
<b>5. Application Software Design .....</b>	<b>14</b>
5.1. Software Design Overview .....	14
5.2. Software Flowchart .....	14
5.3. Software Programming .....	16
5.4. Software Screens .....	18
<b>6. Test Plan .....</b>	<b>20</b>
6.1. Software Test Plan .....	20
6.1.1. Low Level Testing .....	20
6.1.2. High Level Testing .....	20
6.2. Firmware Testing .....	20
6.3. Hardware Testing .....	21



6.3.1.	Component Connection.....	21
6.3.2.	DC Power .....	21
6.3.3.	Piezoelectric Buzzer .....	21
<b>7.</b>	<b>Conclusion.....</b>	<b>23</b>
<b>8.</b>	<b>References .....</b>	<b>24</b>

## List of Figures

Figure 1: Overall System Diagram of the PAS .....	2
Figure 2: Hardware Block Diagram of the PAS .....	4
Figure 3: Schematic Diagram of the Dock Unit.....	5
Figure 4: Schematic Diagram of the Cap Unit .....	6
Figure 5: Typical Characteristic of the Buzzer.....	8
Figure 6: Firmware Overview .....	9
Figure 7: Flow Chart for the Dock Application Code.....	11
Figure 8: Flow Chart for the Cap Application Code .....	13
Figure 9: Software Design Overview .....	14
Figure 10: Flow Chart of the Application Software .....	15
Figure 11: Search Screen .....	18
Figure 12: Patient Profile.....	18
Figure 13: Medication Profile .....	19
Figure 14: Confirmation Message.....	19
Figure 15: Error Message .....	19
Figure 16: Inverter Push-Pull Circuit .....	21

## List of Tables

Table 1: Format of Data.....	12
Table 2: Message Protocol for Outgoing Packet .....	16
Table 3: Format of Data .....	16
Table 4: Order of Outgoing Packet .....	17
Table 5: Message Protocol for Incoming Packet .....	17
Table 6: Incoming Confirmation Packets .....	17

## Glossary

<b>CGM:</b>	Clock Generator Module
<b>DIP:</b>	Dual in line pin device or component having two rows of leaded pins for terminating to circuit conductors through holes in a substrate
<b>ENSC:</b>	Engineering Science
<b>FLASH:</b>	Flash Erasable Programmable Read-Only Memory
<b>GUI:</b>	Graphical User Interface
<b>LED:</b>	Light-Emitter Diode
<b>LVI:</b>	Low-Voltage Inhibit
<b>MTBF:</b>	Mean Time Before Failure
<b>PAS</b>	Pill Alerting System
<b>RAM:</b>	Random Access Memory.
<b>SCI:</b>	Serial Communications Interface Module
<b>SOIC:</b>	Small-Outline Integrated Circuit
<b>UL:</b>	Underwriters Laboratories
<b>USB:</b>	Universal Serial Bus

## **1. Introduction**

Our product, the Pill Alerting System (PAS), will remind individuals when it is time to take their pills by setting off music and flashing a LED on a slightly modified cap of a pill container. The PAS is very simple-to-use and requires no extra effort from the users. The design of this device is such that age and technical background will not limit its use. The PAS is also designed to accommodate hearing and visual disabilities. The visually-impaired can hear the music and the hearing-impaired can see the LED light. The intended completion date for this product is April 2005.

### **1.1. The Scope**

The purpose of this document is to describe in detail the various system attributes that our product must satisfy by the end of our development. The three functional systems are the pharmacy software, the Dock and the Cap. The overall system requirements outline many design specifications which explain how the functionality of the entire system is implanted. Furthermore, for each of the subsystems, the component description, architectural overview and implementation will be examined. There is also a section for the user interface specifications for the pharmacy software.

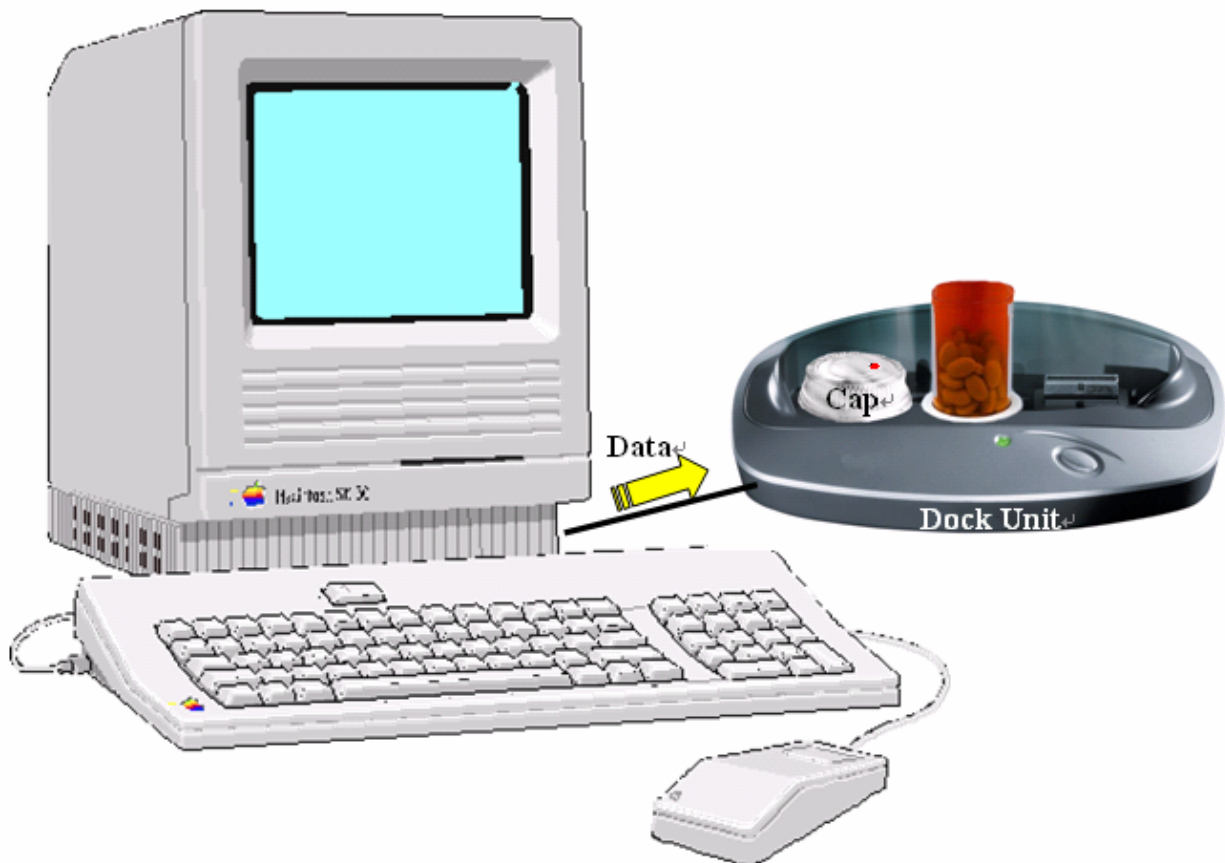
### **1.2. Intended Audience**

This document is intended for both the project manager and the field application engineers' reference throughout the process of manufacturing the functional subsystems of the PAS. This document used in conjunction with the *Functional Specifications* of the PAS will serve as an aid for the engineers to build the product in accordance with the design and functional requirements. Moreover, the project manager will use this document as a means of evaluation for potential technical difficulties associated with the schematic design and to ensure that each of the subsystems is implanted with the required design specifications.



## 2. System Overview

The Pill Alerting System reminds individuals to take their pills when it is time. The reminding mechanism is by means of setting off music from the Cap and flashing the LED which is located on top of the Cap. If the patient opens the container, the music stops and the LED is turned off. Otherwise, the music will continue to play for one minute but the LED will flash until the container has been opened.



**Figure 1: Overall System Diagram of the PAS**

The process of programming the PAS requires three systems:

### 1) Pharmacy Software:

The first system is the pharmacists' computer which has simple-to-use GUI software. The software allows the pharmacists to access the patients' information and medication history and has a database including the timing requirements for pills. With a simple press of a button, this information is transferred through a USB cable to the Dock.

### 2) The Dock:

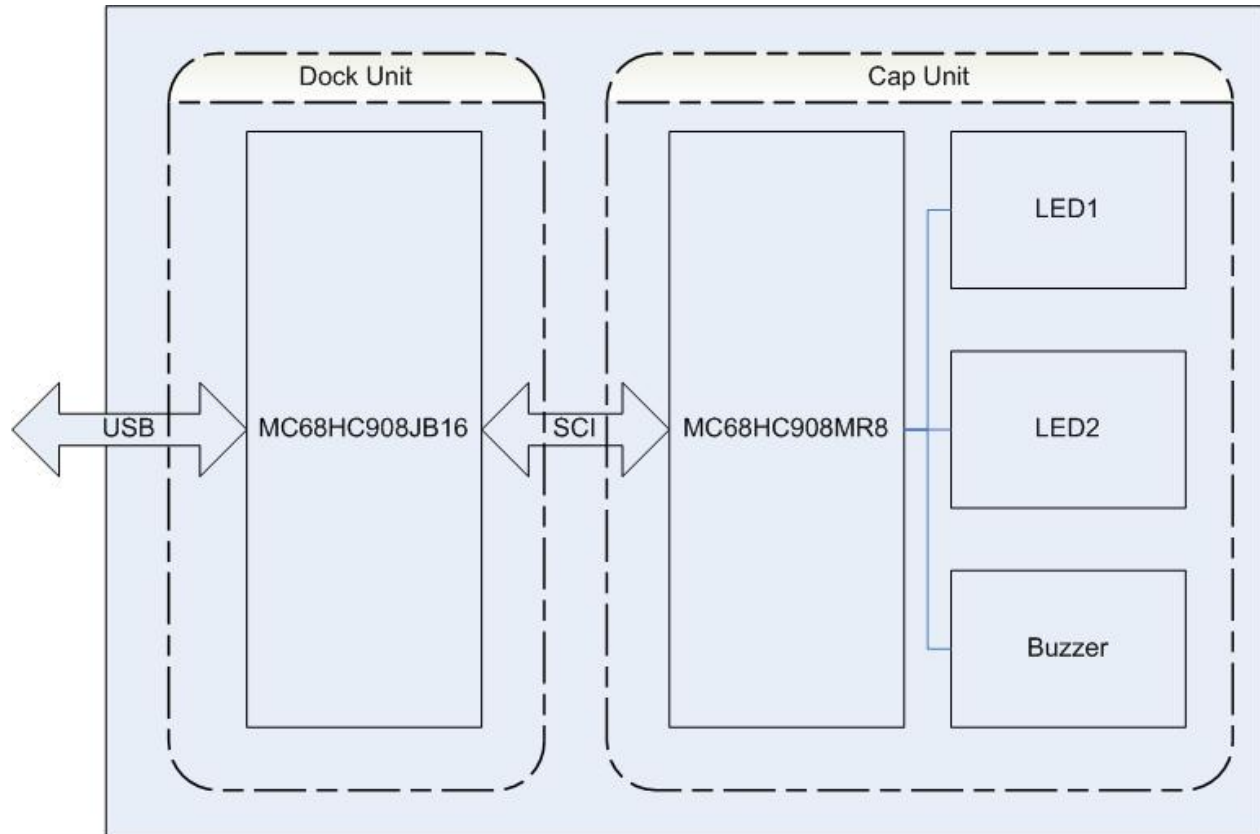
The second system, the Dock, is a container that connects to the pharmacists' computer and has an opening that can hold the Cap. There is also an opening on the Dock to place the bottle so that while the information is being transferred from the Dock to the Cap, the pharmacists can still fill the container with the appropriate pills.

### 3) The Cap:

The Cap is a simple pill container cap with the addition of circuitry underneath, so that it can store timing information, set off the music and flash the LED placed on top when it is the time to take the pills

### 3. Hardware Design

The block diagram of our hardware system which shows the major components is shown below.



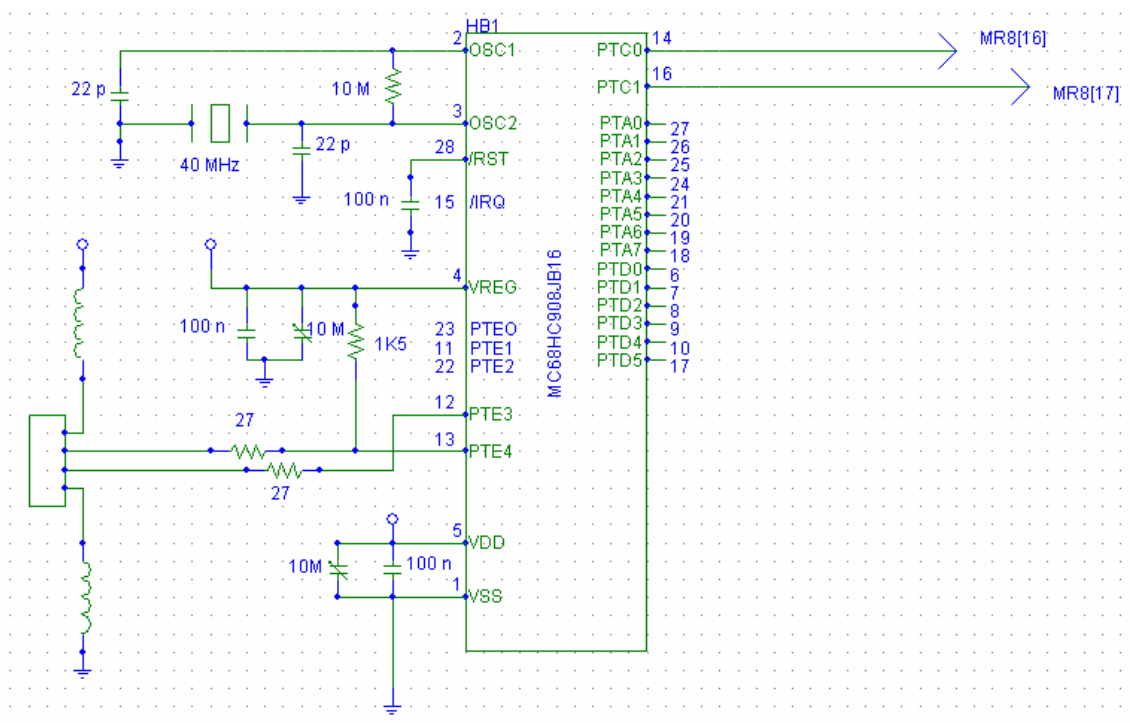
**Figure 2: Hardware Block Diagram of the PAS**

In our design, it is obvious that only the Dock and Cap need hardware components. The main hardware component of these two systems is the microcontroller. Two different microcontrollers from the same family are used; they are in the same family, however. The microcontrollers are both in Motorola MC68HC08 series, one is the MC68HC908JB16, and the other is the MC68HC908MR8. The reason that we choose the Motorola MC68HC08 series is that we only need 8 bits of datapath, and the MC68HC08 series is an 8-bit microcontroller. Also, Prof. One suggests that we have used the MC68HC12 microcontroller in previous courses, (ENSC 150 and ENSC 151) it is easier for us to pick up the Assembly language (Assembly) by using the MC68HC08 series, since the syntax is similar. In addition, the MC68HC08JB16 microcontroller (for the Dock) supports USB, which is a requirement of our system.

#### 3.1. The Dock Unit

The Dock is the hardware platform for the USB design. The Dock serves the application, which is contained in the integrated FLASH memory of the M68HC08 microcontroller.

The detailed schematic diagram is shown below.



**Figure 3: Schematic Diagram of the Dock Unit**

The Dock unit consists of the MC68HC908JB16 microcontroller, resistors, capacitors, crystal oscillator and ferrite beads.

### 3.1.1. The MC68HC908JB16 Microcontroller

Some of the features of this microcontroller taken from its data book are shown below. [2]

Low-power design; fully static with stop and wait modes

- 6-MHz internal bus frequency
- 16,384 bytes of on-chip FLASH memory with security1 feature
- 384 bytes of on-chip random access memory (RAM)
- Up to 21 general-purpose input/output (I/O) pins

Universal Serial Bus specification 2.0 low-speed functions:

- 1.5Mbps data rate
- On-chip 3.3V regulator
- Endpoint 0 with 8-byte transmit buffer and 8-byte receive buffer
- Endpoint 1 with 8-byte transmit buffer
- Endpoint 2 with 8-byte transmit buffer and 8-byte receive buffer

Serial communications interface module (SCI)

In-circuit programming capability using USB communication or standard serial link on PTA0 pin

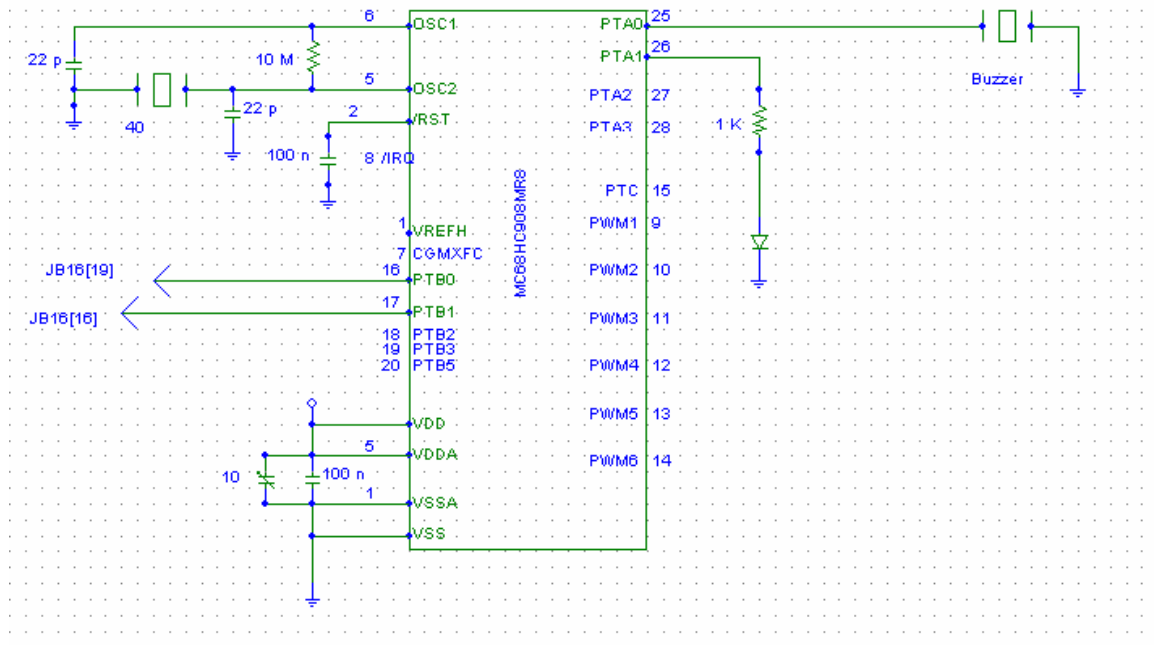
From the above features, we are particularly interested in its USB and SCI communication function, since we can use the USB function to communicate with Pharmacy's Software and the SCI function to communicate with the Cap.

The MC68HC908JB16 microcontroller is offered in several packages. For our design, the 20-pin DIP version was chosen instead of the 28-pin dual in-line package (SOIC) because the DIP package fits the breadboard.

### 3.2. The Cap Unit

The Cap is the hardware platform for the actual application. The Cap serves the application, which is contained in the integrated FLASH memory of the M68HC08 microcontroller.

The detailed schematic diagram is shown below.



**Figure 4: Schematic Diagram of the Cap Unit**

The cap unit consists of the MC68HC908MR8 microcontroller, resistors, capacitors, crystal oscillator, LED and the piezoelectric buzzer.

### 3.2.1. The MC68HC908MR8 Microcontroller

Some of the features of this microcontroller taken from its data book are shown below. [2]

- 8-MHz internal bus frequency
- 8 Kbytes of on-chip FLASH
- On-chip programming firmware for use with host personal computer
- 256 bytes of on-chip random-access memory (RAM):
- 12-bit, 6-channel center-aligned or edge-aligned pulse-width modulator (PWMMC)
- Serial communications interface module (SCI)
- Clock generator module (CGM)
- Digitally filtered low-voltage inhibit (LVI), software selectable for  $\pm 5$  percent or  $\pm 10$  percent tolerance
- Low-power design, fully static with stop and wait modes

Since the Cap does not need the USB connection, we choose to use this microcontroller since it does not have the USB connection. Also, it is under the same family as the MC68HC908JB16 microcontroller, which is easy to program/debug and can use SCI to communicate between them.

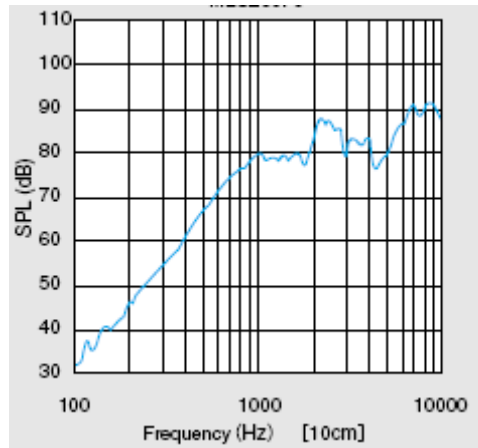
Likewise, the MC68HC908MR8 microcontroller is offered in several packages. For our design, the 20-pin DIP version was chosen instead of the 28-pin dual in-line package (SOIC) because the DIP package fits the breadboard.

### 3.2.2. Buzzer Circuit

The buzzer consists of a simple and inexpensive piezoelectric buzzer, which can play different tone. The buzzer is driven by the output of the MC68HC908MR8 microcontroller, and the features of the buzzer are shown as follow: [1]

- Super slim and possible to mount to the limited space.
- Low consumption, and prolong the battery life.
- No magnetic structure, no countermeasure to magnetic leakage.
- Simple structure
- Clear sound.

We can program the microcontroller such that it drives the buzzer circuit to have different tones. The typical characteristic of the buzzer is shown below.



**Figure 5: Typical Characteristic of the Buzzer**

## 4. Firmware Design

There are two pieces of the firmware involved in our project, one is in the Cap and the other one is in the Dock.

### 4.1 Firmware overview

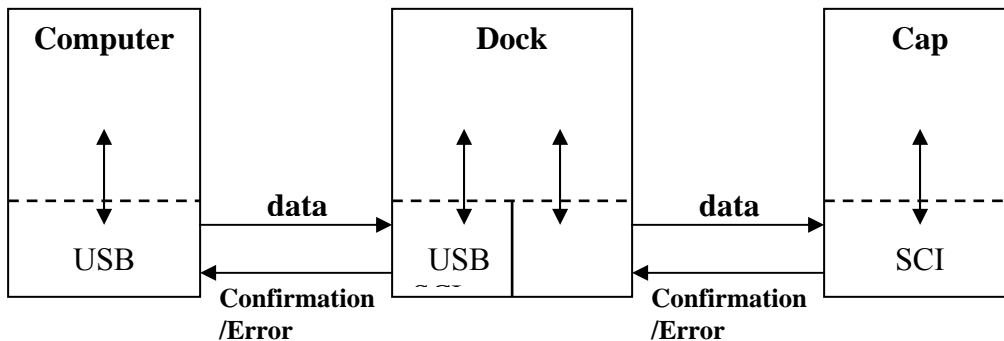


Figure 6: Firmware Overview

Solid Arrow  $\updownarrow$  represents communication within the unit.  
 Dashed Arrow  $\leftrightarrow$  represents communication with another unit.

### 4.2 Firmware in the Dock

The firmware in the Dock consists of two parts, one is the interface code for communicating to the computer and the Cap, and the other is the application code for translating information and sending data to the Cap. In addition, we have the header file for the microcontroller used in the Dock included from the firmware developing software.

#### 4.2.1. Interface Code

The interface code includes two modules, one for the Cap and the other for the USB communication with computer.

##### 4.2.1.1. Interface Code for Computer

The interface code consists of three files; *u08usb.h* defines the USB communication module according to the USB 1.1 Specification, *u08usb.c* is the USB implementation module, and *u08desc.c* includes constant for the USB descriptors.



*u08usb.h* defines the standard device descriptor, standard configuration descriptor, standard interface descriptor, standard endpoint descriptor, structure of setup packet, status code, standard device request code, and descriptor types.

*u08usb.c* implements clear feature function, set address function, set configuration function, get descriptor function, handle setup function, and interrupt handler function for Standard Device Request Handler

*u08desc.c* specifies constants needed for this project for the USB descriptor for the data structure that is defined in *u08usb.h*.

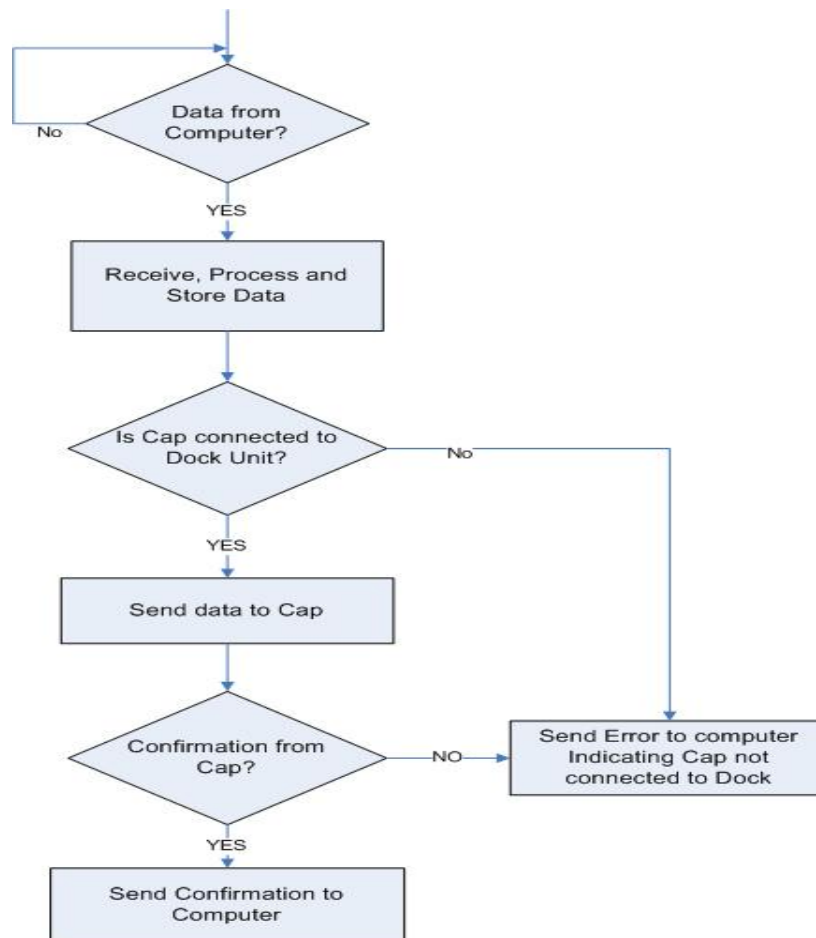
#### 4.2.1.2. Interface Code for the Cap

We will implement the interface code in Assembly since the instruction on the data sheet provided by Freescale Semiconductor (Freescale) is in Assembly.

The interface code for the Cap will include only one file in assembly, following the instruction given by Freescale and the memory location of each I/O port.

#### 4.2.2. Application Code

The following flow chart summarizes the functionality of this application code.



**Figure 7: Flow Chart for the Dock Application Code**

The application code will receive the data by packet via the USB communication. The data transfer is initiated by setting the start flag to 1, and end of data transfer is indicated by setting more flag to 0. After the end of the data transfer, the Dock will send a packet to the computer indicating if the data transfer process is successful by setting the confirmation flag to 1(successful) or 0(error). In addition, if an error has occurred in the data transfer, the Dock will indicate the source of the error and the type of error in the same packet. During the data transfer, information in each packet will be translated and stored in the Dock. The Dock will then send the data via the SCI interface to the Cap.

### 4.3 Firmware in the Cap

The firmware in the Cap includes the interface code for communicating with the Dock and application code for processing data and functioning based on the given data. In addition, we have the header file for the microcontroller used in the Dock included from the firmware developing software.

### 4.3.1. Interface Code

This interface code only communicates with the Dock, and the interface used is the SCI.

We will implement the interface code in Assembly, since the instruction for Assembly is provided by Freescale.

The interface code for the Cap will include only one file in Assembly, following the instruction given by Freescale and the memory location of each I/O port.

### 4.3.2. Application Code

The Cap will be taking the following data from the Dock:

**Table 1: Format of Data**

Description	Format
Real time	yyyy/mm/dd/hh/mm
Dosage	Number of pills
After/ before meal	1/0
Number of pills in total	Number of pills

The Cap will then process the data given above, the process is as follows: based on the real time and after/before meal value, the Cap will find the first pill time which the real time is closest to (breakfast/lunch/dinner) and calculate the next pill time based on the previous value. In addition, the expiration date for the prescription will be calculated by the number of pills in total.

For the functioning, after the data has been processed, the LED will flash at pill time until the cap has been opened, and the music will set off for exactly one minute and stop no matter the cap has been opened or not. All the reminding mechanism will stop once the cap has been opened.

We have also included the flow chart for the Cap application code in order to provide a clear picture of the process.

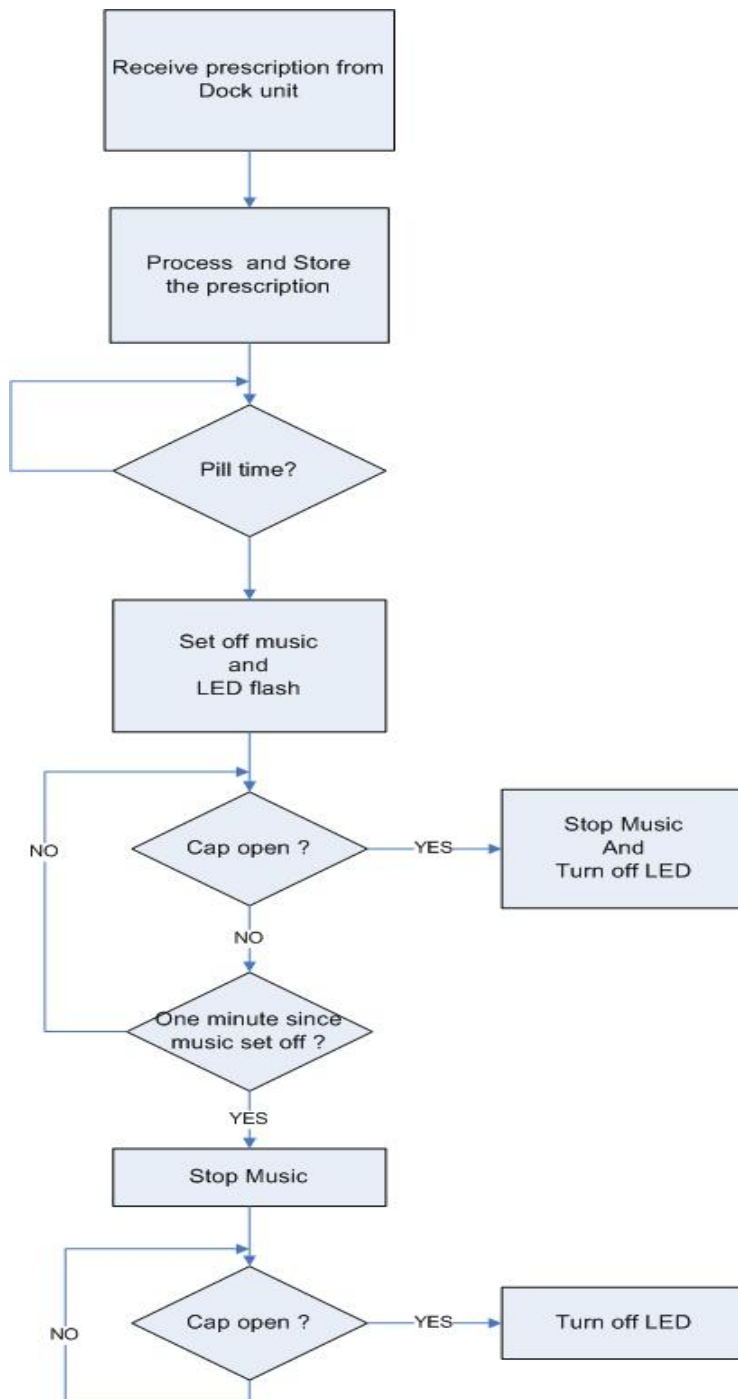


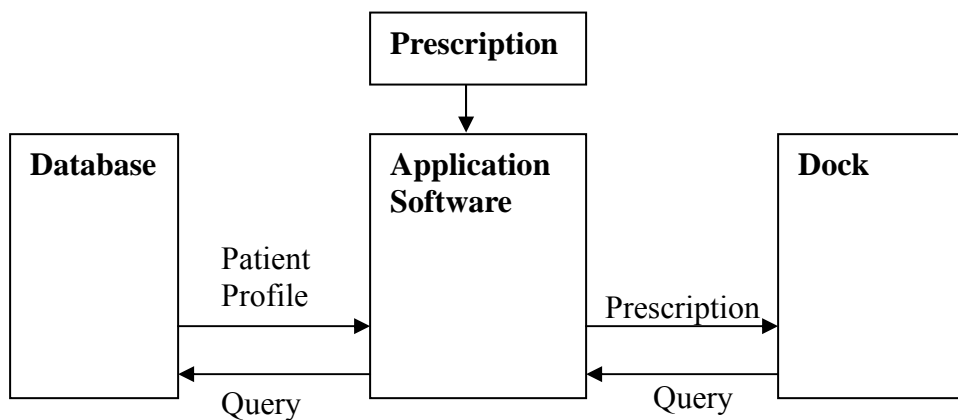
Figure 8: Flow Chart for the Cap Application Code

## 5 Application Software Design

In this section, we will discuss the design specifications for the software in the computer that communicates with the Dock via USB.

### 5.1 Software Design Overview

This application software is a prototype of the actual pharmacy software with added communication to the Dock unit via USB.



**Figure 9: Software Design Overview**

5.2 Software Flowchart

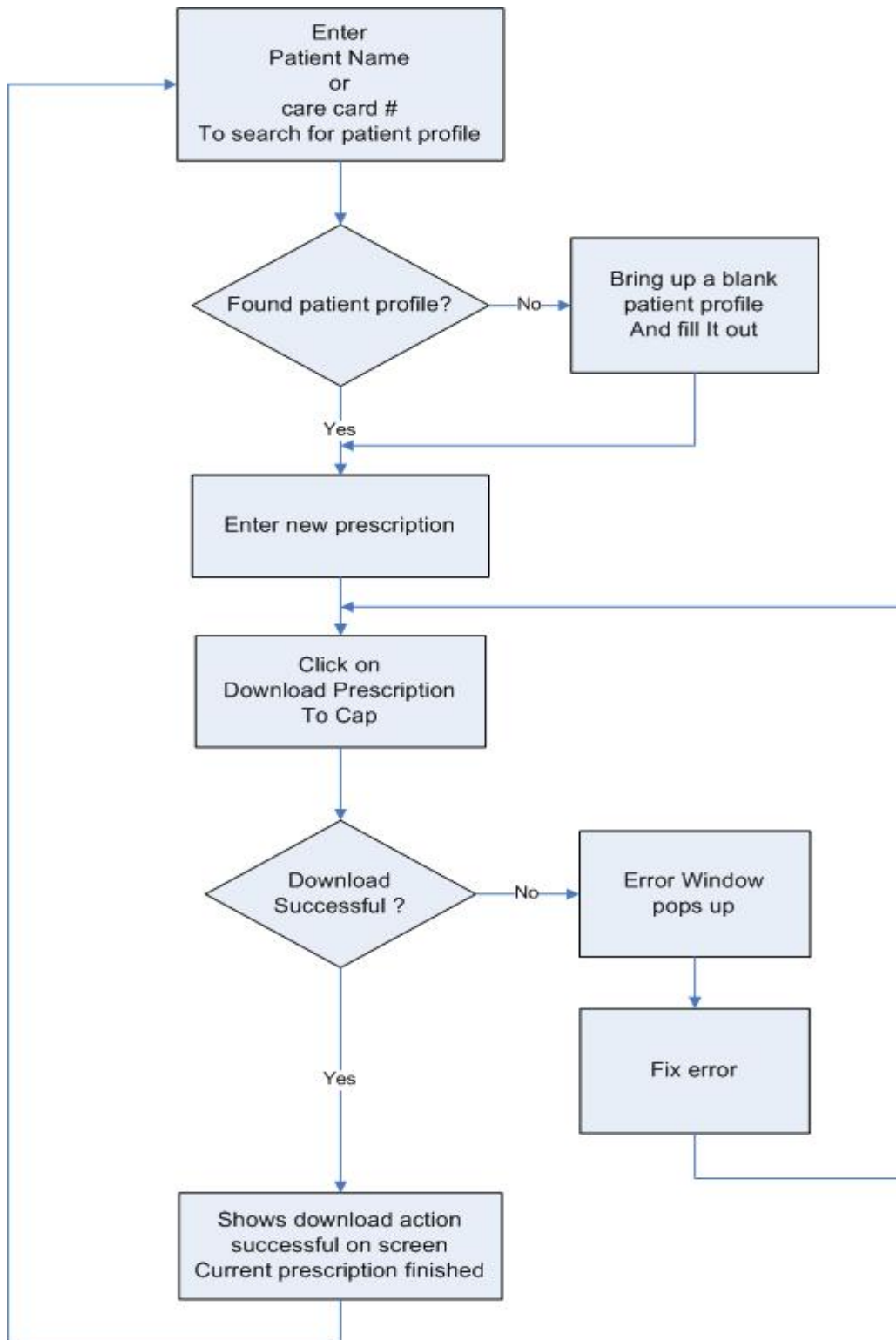


Figure 10: Flow Chart of the Application Software

### 5.3 Software Programming

The software checks the USB connection to the Dock, and the error message will be displayed at the bottom of the application screen if the connection is not detected. The software will display the search page which contains two fields, one is the care card number of the patient, and the other is the name of the patient. The software then queries the database for patient profile. If a matched patient profile was found, the user can then input the new prescription and download the prescription to the Cap via the Dock. If no matched patient profile is found, the software will then bring up a blank patient profile page and ask the user to fill it out. At the same time, the user will be able to fill out the new prescription in the blank patient profile and download it to the Cap as usual. The software will display an action successful message on the screen once it receives the confirmation message from both the Dock and Cap.

For the internal data processing, the software implements two functions: one is transmitting (unsigned char a, int b, unsigned char c) and the other is transmitting (unsigned char a, int b, unsigned char c, unsigned char d) for different data processing. The first and last parameter in these two transmit functions represent the start flag and the more flag.

We will be sending packets via USB, and each packet will contain 8 bits of information. The definition of each bit will be summarized in the following table,

**Table 2: Message Protocol for Outgoing Packet**

Bit	0	1	2	3	4	5	6	7
description	Start flag	Data						More flag

The following table shows the format of our data,

**Table 3: Format of Data**

Description	Format
Real time	yyyy/mm/dd/hh/mm
Dosage	Number of pills
After/ before meal	1=after/0=before
Total number of pills	Number of pills

The first packet sends to the Dock will have the start flag set to one, and the more flag set to one. The latter packets will have the start flag set to zero, and the more flag set to one except the last packet will have both the start and the more flag set to zero.

From the table above we can see that to send “year” of real time in one packet is not possible. Since dividing them into two numbers is not feasible, we will send the “year” of real time digit by digit.

The rest of the data of real time, we can send each of them in one packet. (i.e. send “month” of real time in one packet)

For the dosage and after/before meal, we will combine them into one packet so that the data transmission can be more efficient. In addition, we implement the transmit function with 4 parameters for this reason.

The total number of pills will be sent in one data packet with the more flag set to zero.

After all the packets are sent, the application code waits for two confirmations, one is from the Dock and the other is from the Cap. Once the confirmations are received, the download action has been completed and the software will go back to its initial state.

After the message protocol has been set up, we will show the order of data to be transmitted.

**Table 4: Order of Outgoing Packet**

Start Flag	Data	More Flag
1	First digit of year in real time	1
0	Second digit of year in real time	1
0	Three digit of year in real time	1
0	Four digit of year in real time	1
0	Month in real time	1
0	Date in real time	1
0	Hour in real time	1
0	Minute in real time	1
0	Dosage                      Before/after meal	1
0	Total number of pills	0

We will define the incoming message protocol as follows,

**Table 5: Message Protocol for Incoming Packet**

Bit	0	1	2	3	4	5	6	7
Description	Confirmation flag (1-confirm/0-error)	Source (0-dock/1-cap)	Error Type					

To confirm the download is successful, we need to receive two confirmation packets, one is from the Dock, and the other is from the Cap.

The two confirmation packets will look like the following:

**Table 6: Incoming Confirmation Packets**

0	1	2	3	4	5	6	7
1	0	X	X	X	X	X	X
1	1	X	X	X	X	X	X



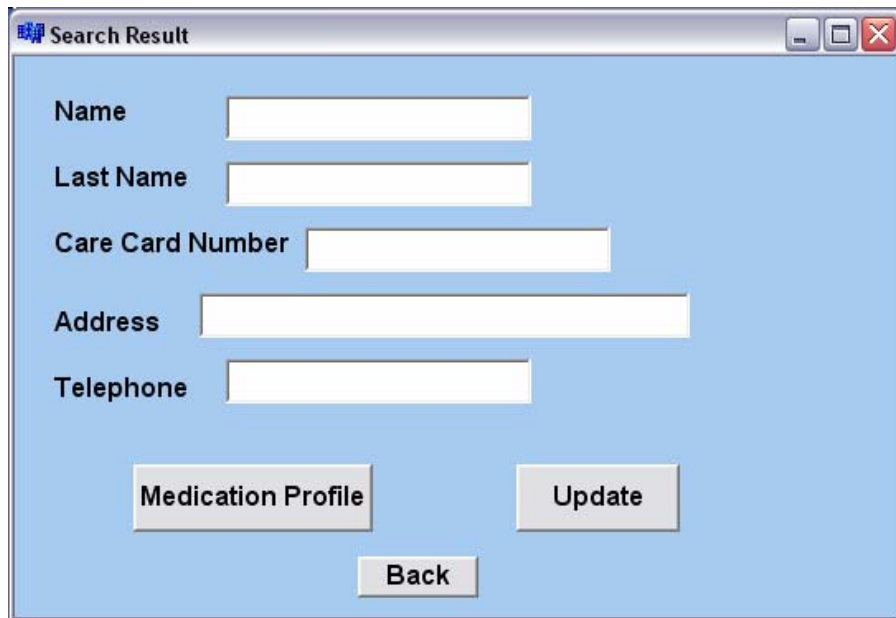
## 5.4 Software Screens

We have also included the software screens to give the reader a better understanding of what this software will look like. The order of the following screens follows the simple path of the flowchart in Figure 10.



The Search screen is a window titled "Search" with a light blue background. It contains three input fields: "Name", "Last Name", and "Care Card Number". Below the input fields are two buttons: "Search" and "Add".

**Figure 11: Search Screen**



The Patient Profile screen is a window titled "Search Result" with a light blue background. It contains five input fields: "Name", "Last Name", "Care Card Number", "Address", and "Telephone". Below the input fields are three buttons: "Medication Profile", "Update", and "Back".

**Figure 12: Patient Profile**

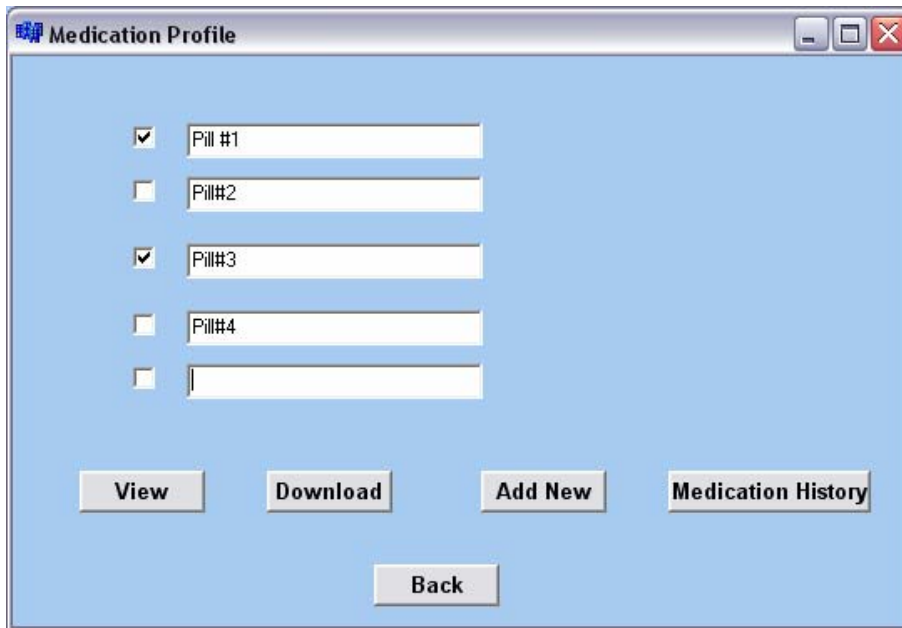


Figure 13: Medication Profile



Figure 14: Confirmation Message



Figure 15: Error Message

## **6 Test Plan**

All subsystems will be tested individually according to the criteria specified below. We will then test the system as a whole.

### **6.1. Software Test Plan**

The software test plan is composed of two levels of testing.

#### **6.1.1 Low Level Testing**

The low level routines outlined in Sections 5 will be tested by first creating a test database containing the patients' personal and medical information. We will then access this database through the pharmacy software by using a patients' name (first or last name) or their CareCard number. The result of this test will take us to a screen that shows the patients' personal information such as his/her address, phone number, etc. and will also have a link to the patients' medication profile. Once we request the medication information, simply by pressing a button, we will see the medication profile of the patient. When presented with this information, we will choose the download option so that this information will be sent through the USB to the Dock. The acknowledgement will be with a confirmation screen. It is expected that this level of testing will remove any bugs related to database access and retrieval of information as well as sending the information to the Dock.

#### **6.1.2. High Level Testing**

In this stage of testing the user interface software will be tested to ensure that all screens act as specified in the functional specification. This phase of testing is intended to remove bugs that might arise in the user interface logic routines.

### **6.2. Firmware Testing**

As mentioned in previous sections, we will use two microcontrollers, the MC68HC908JB16 for the Dock and MC68HC908MR8 for the Cap. The testing of the firmware will be done by using Code Warrior which is software we have obtained from the Metrowerk website. [2] The circuit debugger in Code Warrior will be used to verify that the data was received correctly to the Dock and verify the value of the processed data in the Dock. The same approach will be taken with respect to the Cap.

## 6.3. Hardware Testing

This section highlights the testing plan for the major components in the hardware.

### 6.3.1. Component Connection

This phase of testing is the primary stage of hardware testing and all requirements must be met in order to continue. The test engineer will perform the following tasks:

- 1) Confirm that component connections match the schematics
- 2) Make sure resistor values are within 5% of the nominal values
- 3) Verify the connection of pins to the board and socket

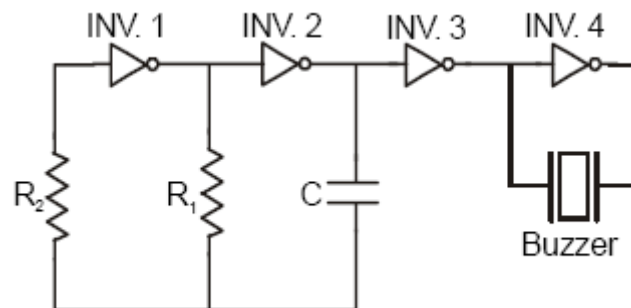
### 6.3.2. DC Power

In this stage, it must be verified that correct DC voltages are present at all power nodes before continuing with subsequent tests. The test engineer will perform the following tasks:

- 1) Apply the correct DC voltages required to power connections
- 2) Confirm that the correct DC voltage is applied at all of the power nodes

### 6.3.3. Piezoelectric Buzzer

In order to test this unit, we need a test circuit for it, the circuit is shown below.



**Figure 16: Inverter Push-Pull Circuit**

$$\text{where } f_{osc} = \frac{1}{2.2R_1C} \text{ and } R_2 \cong 10R_1$$

Resistor values can range from about 3kΩ to about 10MΩ. Capacitor values can range from 50pF up, though below 1000pF, the frequency will be somewhat lower than predicted by the equation.

The input resistor,  $R_2$ , is normally made 10 times the timing resistor,  $R_1$ , to minimize the output curving effect of the protection diodes in the inverter. By adding a fourth inverter between the leads of the external drive buzzer, a push-pull circuit is made. If one terminal of the buzzer were connected to ground and the other to the output of the inverter, the buzzer would see a voltage only on one terminal. The element would only deflect in one direction. By adding the fourth inverter, a voltage can be applied to both pins and the element will deflect in both directions. This doubles the voltage across the buzzer. Using this circuit, we can test to see if this unit works properly or play tone correctly.

## 7 Conclusion

The requirements and design specifications that are discussed in this document provide a clear picture of how each of the three subsystems must be implemented. We have provided the architectural layout, implementation procedures and the components that are needed in this document. In addition, we have presented the user interface specifications that provide a user-friendly interface for the users of the software. Therefore, by means of these design specifications, we at MedSmart, are confident that the prototype version of our PAS will at least satisfy the minimum requirements outlined in the *Functional Specification* document. Moreover, we will simulate our completed prototype in accordance with the test plan discussed in the *Functional Specification* to ensure that all subsystems function correctly and are able to achieve the desired performance. This careful approach shows our well defined requirements for the system as a whole and each individual subsystem. MedSmart will use these design and functional specifications for the development of the PAS.

### 8 References

- [1] Digi-Key Corporation. <<http://dkc1.digikey.com/ca/digihome.html>>. 2004
- [2] Freescale semiconductor <<http://www.freescale.com/>>. 2005
- [3] TechWeb The Business Technology Network <<http://content.techweb.com/>>. 2005
- [4] The USB08 Project Page <<http://hc08web.de/usb08/>>. 2005
- [5] Underwriters Laboratories Inc. <<http://www.ul.com/>>. 2005
- [6] USB Implementers Forum Inc. <<http://www.usb.org/>>. 2005