March 3, 2004

Lakshman One
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

Re: ENSC 440 Project Design Specifications –Voice Recognition System in MP3 Players

Dear Mr. One:

The attached document, *design specification for a voice recognition system in MP3 player,* describes design requirements for the voice recognition system we are developing. We are currently working with Start Labs Inc. on controlling their MP3 players via voice commands. Our project is to design the voice recognition module of the MP3 player. We will design the module in accordance with Start Labs Inc.'s needs and expectations.

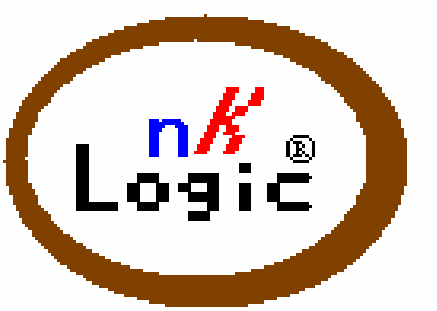
This document illustrates the design considerations that are taken into account in the process of development. It consists of a system overview, hardware and software specifications, user interfaces, and test procedures.

The nK Logic Group consists of two experienced senior engineering students: Won Kang and Garet Kim. We look forward to your feedback and suggestions. Please feel free to contact me by phone at (604) 785-5933 or by e-mail at gkim@sfu.ca. Thank you for you attention.

Sincerely,

Garet Kim
nK Logic

Enclosure: *Design Specification for a voice recognition system in MP3 players*



Design Specification for a

Voice Recognition System in MP3 Players

Project Team: Won Kang
Garet Kim

Contact Person: Garet Kim
gkim@sfu.ca

Submitted to: Lakshman One – ENSC 440
Nakul Verma – ENSC 440
Mike Sjoerdsma – ENSC 305
School of Engineering Science
Simon Fraser University

Issued date: March 3, 2004

Revision: 1.1

Executive Summary

We, the nK Logic Group, are committed to building an advanced Voice Control Unit (VCU), for the Start Labs' MP3 player. After careful examination, we have narrowed down our possible solutions and purchased a Voice Extreme™ Development Toolkit from Sensory Inc. in order to shorten the development time. This development board is divided into two main boards: the Voice Extreme™ development module and the Voice Extreme™ IC module. The functionality of the VCU will be described in terms of the custom IC version of this Voice Extreme™ Module.

The design requirements mainly focus on the software since we are implementing on a development board. However, these requirements should be independent of the platform so that the codes and algorithms can be imported into the actual ICs.

Table of Contents

Table of Figures and Tables

1. Introduction

The aim of this project is to develop a prototype Voice Control Unit (VCU) for an MP3 player. The system is built on Sensory Inc.'s Voice Extreme development board. Therefore, this document focuses on the structure and flow of the software while briefly explains main hardware components and interfaces of the board.

1.1 Scope

This document summarizes the design requirements for the VCU. It also outlines various design aspects of the system by providing flow charts of the software and detailed explanations on the interfaces between functions.

This document begins with a system overview explaining mainly the Voice Extreme development board and technologies. Next, main components of the hardware are described, followed by the software design of the system. The following sections are dedicated to the User Interface design and the test plan.

1.2 Glossary/Acronyms

Glossary

Voice Extreme™	Sensory Inc.'s application specific IC that enables intuitive programming of interactive speech application
Weight	Samples of voice data required for speech synthesis and pattern comparison
Hit Ratio	Ratio that a voice recognition unit successfully recognizes the user's commands.

Acronyms

CL	Continuous Listening
GUI	Graphic User Interface
LED	Light Emitting Diode
MP3	Moving Picture Experts Group Layer-3 Audio (audio file format/extension)
PCB	Printed Circuit Board
PCM	Pulse-Code Modulation
SD	Speaker Dependent
SI	Speaker Independent
SNR	Signal to Noise Ratio
SS	Speaker Synthesis
SV	Speaker Verification
WS	WordSpot

1.3 References

[1] Voice Extreme™ Module, Sensory Inc. 2004, www.sensoryinc.com

1.4 Intended Audience

Engineers at Start Labs Inc. will use this document when they integrate this voice control unit into their MP3 player. Engineers at nK Logic should use this as a tool for evaluation and verification .of the VCU to ensure that the VCU performs all the system functions in a proper manner.

2. System Overview

The voice recognition module, developed by the nK Logic Group, allows the user to control the MP3 player by speaking aloud into the microphone. Since the Start Labs' MP3 decoder unit is still under development, the nK Logic Group will simply output messages to the host computer that identify which voice control command has been recognized.

The voice recognition can be viewed as a series of sequential processes. This is illustrated in Figure 1.

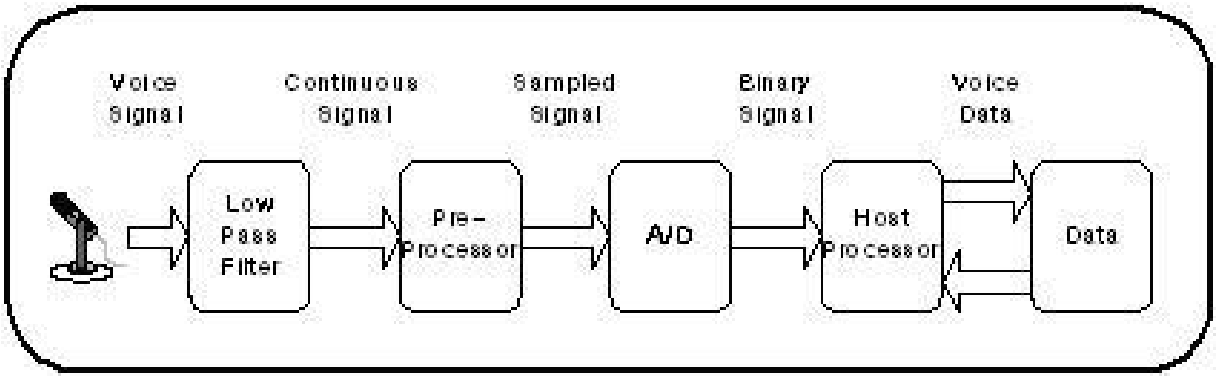


Figure 1: Configuration of Voice Recognition Process

The nK Logic will design an efficient algorithm for the host processor and the interaction between the host processor and data.

2.1 Voice Extreme™

There are several reasons why Voice Extreme™ module is ideal for this project. Beside the fact that this chip is suitable for portable devices, (i.e. small package and low power consumption) Voice Extreme™ has sufficient process power for this particular the development. The 8-bit CPU with 64 KB maskable ROM is adequate to implement the functionalities required here. Voice Extreme™ also supports various voice recognition technologies in order to improve the success ratio of the system. These technologies will be discussed further in following sections. Lastly, the development toolkit of the Voice Extreme™ is reasonably priced. One can purchase the board, accessories, and software tools in C language for under US $200.

2.2 Speech Technologies

In a VCU, accuracy of the recognition measures integrity of the system. To improve it, Voice Extreme™ supports various speech technologies, such as Speech Synthesis (SS), Speaker Independent (SI), Speaker Dependent (SD), Continuous Listening (CL), WordSpot (WS), and Speaker Verification (SV). For our application, both SD and CL technologies are employed to achieve the two-level grammar structure with high success ratio.

2.2.1 Speech Synthesis

SS is a process which produces words, phrases, or sentences based on the pre-recorded utterances. The VCU must be able to prompt the user in text and sound. The sound can be either a beep or double beeps to alert an error condition or acknowledgement whereas a specific phrase can notify different events. SS can be efficient for the later case.

2.2.2 Speaker Independent Technology

Along with SD, SI is one of the two mainstreams of voice recognition techniques. SI uses a WEIGHTS file to guide neural-net processing to help find the matching patterns. A WEIGHTS file contains the characteristics of the linguistic word to be recognized instead of the characteristics of word spoken by a specific speaker. It can be used regardless of the speakers, but has lower accuracy compared to SD.
SI can be very costly to developers since they need to purchase the prerecorded word files (i.e. SI wave files) from Sensory Inc. If the word is in their database, each word can be purchased at around US $250. If not, the price can go up to over US $1,000.

2.2.3 Speaker Dependent Technology

SD is another voice recognition technique which can be used only for single specified speaker. SD has better accuracy compared to SI, but it requires the training process before it can be applied. The speaker must train the system to recognize his/her voice prior to the recognition process.

2.2.4 Continuous Listening Technology

CL is a variation of speech recognition technology that provides the capability to listen continuously for a “trigger” word or phrase to be spoken. CL is main scheme to understand a short command sequence. The VCU must incorporate this feature as much as possible so as not to make the user press buttons before speaking.

2.2.5 WordSpot Technology

WS is similar to CL in the sense that it searches for the specific words continuously, but it is different that the words can be embedded in continuous speech. However, it is not useful for selecting a command word from a list of commands.

2.2.6 Speaker Verification Technology

SV is generally used to discriminate between two speakers. The VCU can optionally enable this feature to lock the MP3 player. When it is enabled, the only person who owns the MP3 player may have access to the control.

3. System Hardware

There is no additional hardware required other than the Voice Extreme Development Board. The rest of this section discusses main components of the development board.

3.1 Voice Extreme Development Board

Although it is difficult to modify the development board directly, the choice of Flash memory, configuration of input/output port, and the usage of buttons can be modified to suit the purpose of the software.
The following two figures show the pictures of the development board.

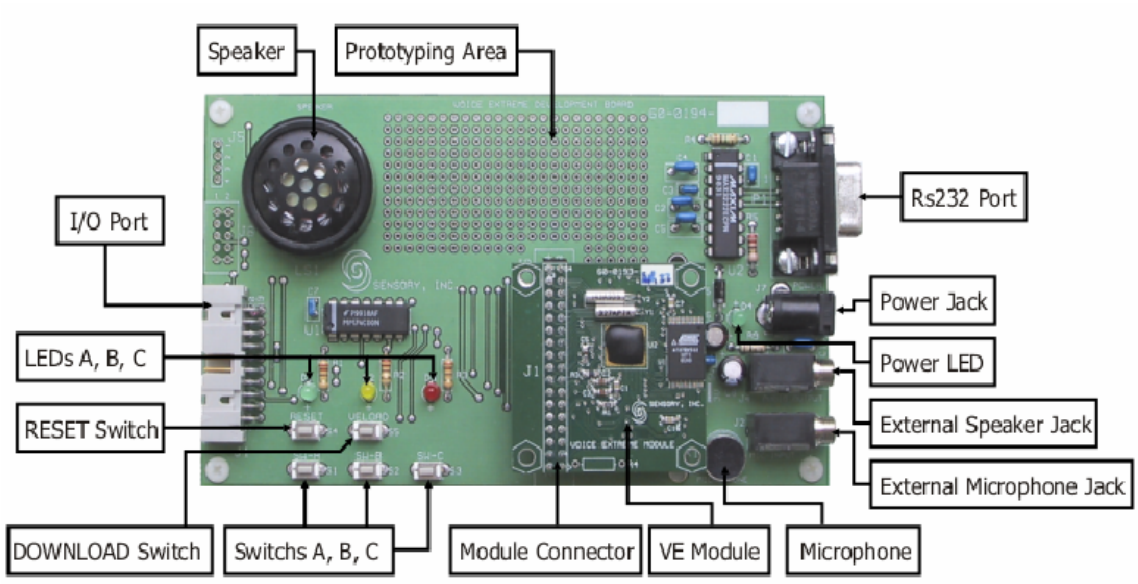


Figure 2: Voice Extreme Development Board

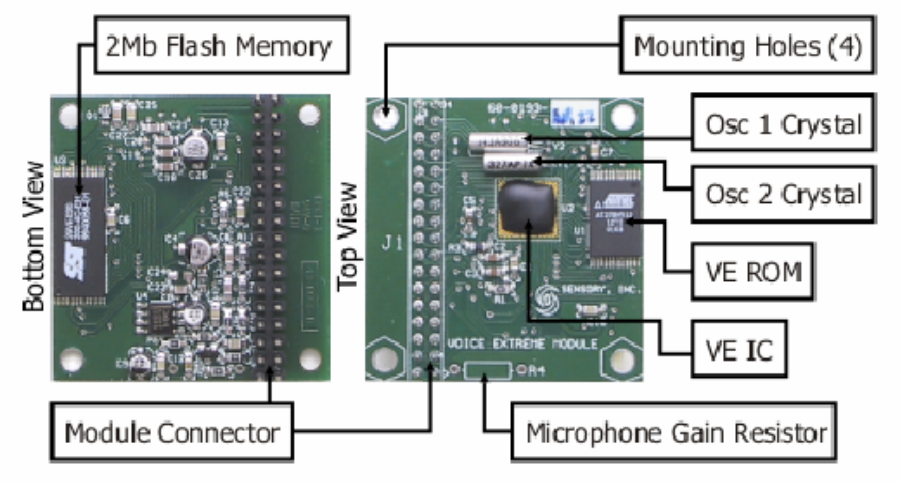


Figure 3: Voice Extreme Module

The development board's specifications provided by the manufacturer are summarized in Table 1.

Table 1: Specifications of Voice Extreme™

Manufacturer	Sensory Inc.
Product	Voice Extreme™
Core	8-bit CPU
Add. Memory Reqd	2MB Flash
Maskable ROM	64KB
Internal ROM	N/A
Speech Duration (Max.)	100 sec. (ext. flash)
RAM	2.5 KB
I/O	14
SI words on chip	350 (ext. flash)
SD words on chip	1900 (ext. flash)
Packages	TQFP-64
100k die price	<$1.50
Power dissipation	3.0V, 10mA
Notes	ASSP of RSC-3X
Operating Temperature	0 to 70 degrees C

3.2 Microphone

Microphone is the front end of the system and gets the user input. As specified in the datasheet, a microphone with a good quality may increase the SNR significantly and improve the reliability of the system consequently.

3.3 Audio Codec

Audio Codec is an indispensable requirement for every digital audio device. It converts analog voice signal to digital signal so that the CPU can perform various functions on it. It also converts digital signal into the analog audio signal. Due to the frequent usages, manufacturers often integrate it into the ASIC for convenience.

3.4 Micro-Processor Unit (MPU)

Micro-processor unit is the heart of the VCU. It integrates all other sub-systems and decides on what actions should take place at a certain time. It has a high level language interpreter that understands the voice recognition programs; operates the various IOs and peripherals; and executes multiple tasks. Enough processing power to perform the action in real time is essential for the MPU, and low power consumption is another key requirement.

3.5 Memory

There are two types of memory on the board: RAM and ROM. RAM is the memory space required for the MPU to process the software programs. ROM is the memory space for the source code and voice data.

Type and size of the RAM depend on the specific microprocessor chosen, but typically it is integrated into the IC. The size of the ROM depends on the number of commands rather than the code since the voice weights occupy much more space than the code. Flash ROM is appropriate for the development stage due to the repeated usage, but One Time Programmable (OTP) type may be good enough for the actual production. An alternative is to share a portion of the Flash in the MP3 encoder module. However, the maximum size of the ROM is limited by the number of address lines of the chosen MPU.

3.6 Speaker

The speaker in a voice recognition system is mostly used to indicate errors and user prompts. The message can be short sentences or beeps.

3.7 Interface / IO port

The VCU requires a number of interfaces to interact with other devices. Table 2 is a brief summary of the interfaces.

Table 2: Interfaces

Interface	Bit	Function
SPI	3	Serial Downloading of source code Connection with real time debugging terminal
Input	1 1	Toggle the activation of VCU System reset
Output	3	Output LEDs for simple result
RS232	3	Interaction with the debugger on the host computer
TBD	TBD	Interface with MP3 player
Reserved	6	Future enhancement

4. System Software

A Speaker Dependent VCU requires three basic programming modules: Training, recognition, and debugging modules. First, the training module prompts the user to say each command twice. When those two commands sound similar, the module averages them, and saves the average into the memory. In the recognition module, it receives the command input and compares with saved commands to find a match. Once the recognition module finds a match, it tells to the debugging module which command has been received and recognized (Debugging module is to be explained in the later sections). Figure 4 illustrates the operation of this system in a flow chart.

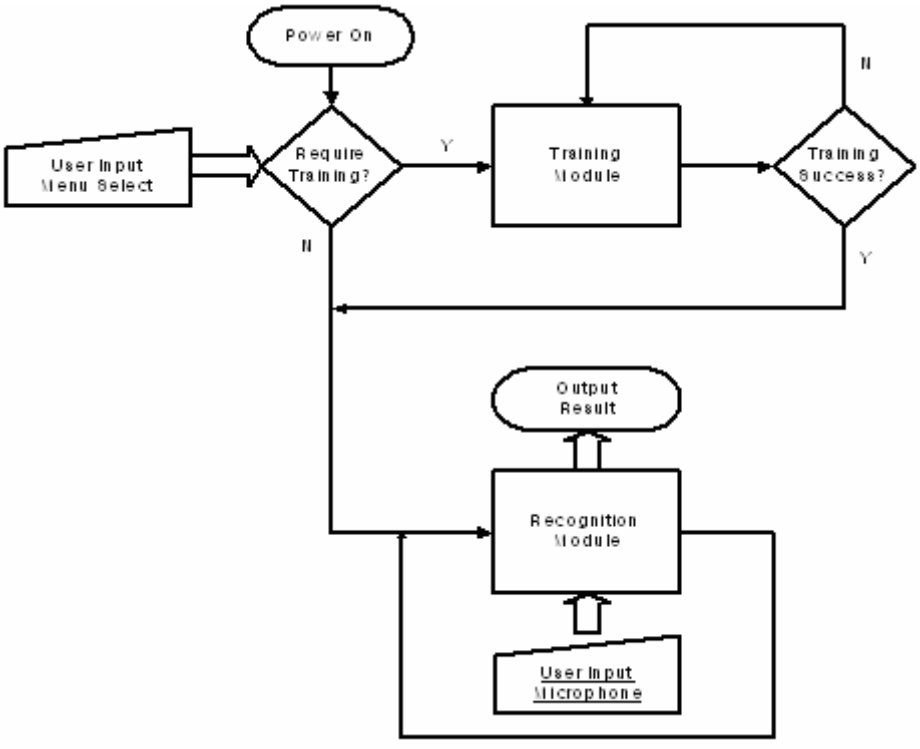


Figure 4: Overview of a voice recognition system

4.1 Command Set

The command set used in this voice recognition system is listed in Table 3. An ID Code is assigned to each command to identify and easily locate the command in the event of communication between the board and the debugger. ID Codes are also used as a part of the user prompt (e.g. the system can prompt the user to train the word "Play" by saying, "Please say 11").

Table 3: Command Set

Command List for the MP3 Player		ID Code	Recog. word
Playback (1)			
"Play"	Play	11	Play
"Pause"	Toggle pause (Also say "play" to resume)	12	Pause
"Stop"	Stop the current playing song.	13	Stop
"Rewind [0~9]"	Rewind by [N] seconds (0 = until "play")	14	Rewind
"Forward [0~9]"	Forward by [N] seconds (0 = until "play")	15	Forward
"Previous [1~9]"	Play previous [N]-th tracks	16	Previous
"Next [1~9]"	Play next [N]-th tracks	17	Next
Volume (2)			
"Volume [0~9]"	Volume scaled from 0 to 9. (0 : mute)	20	Volume
"Up"	Volume up by 1 level, un-mute.	21	Up
"Down"	Volume down by 1 level	22	Down
"Mute"	Toggle Mute. (Also say "Volume Up" to unmute	23	Mute
Option (3)			
"Clear [option]"	Clear "repeat", "shuffle", "hold"	31	Clear
"Repeat [menu]"	Repeat "single, "all"	32	Repeat
"Repeat Single"	Repeat the currently playing song	33	Single
"Repeat All"	Repeat the entire playlist	34	All
"Shuffle"	Shuffle the playlist	35	Shuffle
"Timer [1~9]"	Set up the power-off timer in units of 30 min (0=off)	36	Timer
"Hold"	Disable all other commands but clear hold	37	Hold
Equalizer (4)			

Command List for the MP3 Player		ID Code	Recog. word
"Equalizer [mode]]"	Change the equalizer's settings	40	Equalizer
"Classic"		41	Classic
"Latin"		42	Latin
"Rock"		43	Rock
"Dance"		44	Dance
"Club"		45	Club
"Bass"		46	Base
"Treble"		47	Treble
"Hall"		48	Hall
"Live"		49	Live
"Party"		50	Party
"Reggae"		51	Reggae
"Soft"		52	Soft
"Techno"		53	Techno
"Pop"		54	Pop
Power (6)			
"Shutdown"	Power off	61	Shutdown
"Reset [or Reboot]"	Reboot the software	62	Reset
Skin (7)			
"Skin [1~9]"	Change the background of the LCD display	71	Skin
Status (8)			
"Status"	Report status of the MP3 player	81	Status

The commands are carefully chosen so that none of them sound similar that may cause possible wrong detection.

In order to implement the two-level (i.e. two-word) command structure, all commands are categorized into 3 different types: Standalone type, trigger type, and parameter type. Standalone type commands are simple one-level (i.e. one word) commands. On the other hand, trigger type commands always expect a second word after the initial command word to form a complete command. Parameter type commands are “shortcut” commands for trigger type commands. This type of commands must be masked in the level-one recognition stage to improve the speed and accuracy of the system. Table 4 summarizes the command types.

Table 4: Command Types

ID Code	Recog word	Type	Second Level	Note
11	Play	Standalone	None	
12	Pause	Standalone	None	Effective only during playing
13	Stop	Standalone	None	
14	Rewind	Trigger	0~9	
15	Forward	Trigger	0~9	
16	Previous	Trigger	1~9	0 ignored
17	Next	Trigger	1~9	0 ignored
20	Volume	Trigger	0-9	
21	Up	Standalone	None	
22	Down	Standalone	None	
23	Mute	Standalone	None	“Stop” and “Up” clear mute
31	Clear	Trigger	repeat, shuffle, timer, hold	Mask other parameters
32	Repeat	Trigger	single, all	Mask other parameters
33	Single	Parameter		
34	All	Parameter		
35	Shuffle	Standalone		
36	Timer	Trigger	0-9	
37	Hold	Standalone		Only “hold” can follow.
40	Mode	Trigger	all the mode	
61	Shutdown	Standalone	None	
62	Reset	Standalone	None	
71	Skin	Trigger	0-9	
81	Status	Standalone	None	can be changed to trigger

4.2 Training

Using SD requires training of each command initially. The following flow chart shows the algorithm of the training session.

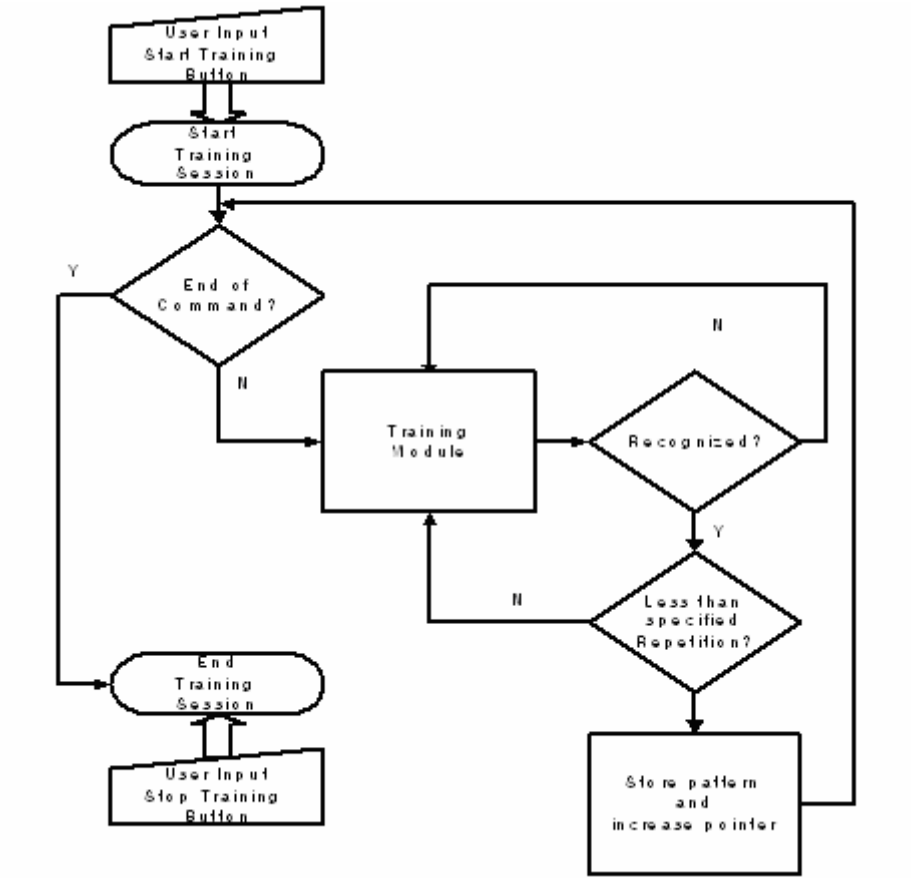


Figure 5: Flow chart for Training Session

The length of the training session is directly proportional to the number of commands to be recorded. To provide effective and fast training session, the followings are to be realized during the training.

1) Training session should start at anytime by pressing a push button.
2) The number of the training for each command should be asked.
3) The user should be able to skip the commands that he/she does not wish to use.
4) The user must be able to abort the session any time.
5) The user should be notified of the result of the training after recording each

command.

The nK Logic group will program the system to accomplish all five tasks.

4.3 Recognition

The success ratio of the voice recognition is basically the measure of success. Carefully designed algorithms and implementation methods are essential to a successful VCU. Figure 6 illustrates the flow chart for the voice recognition module.

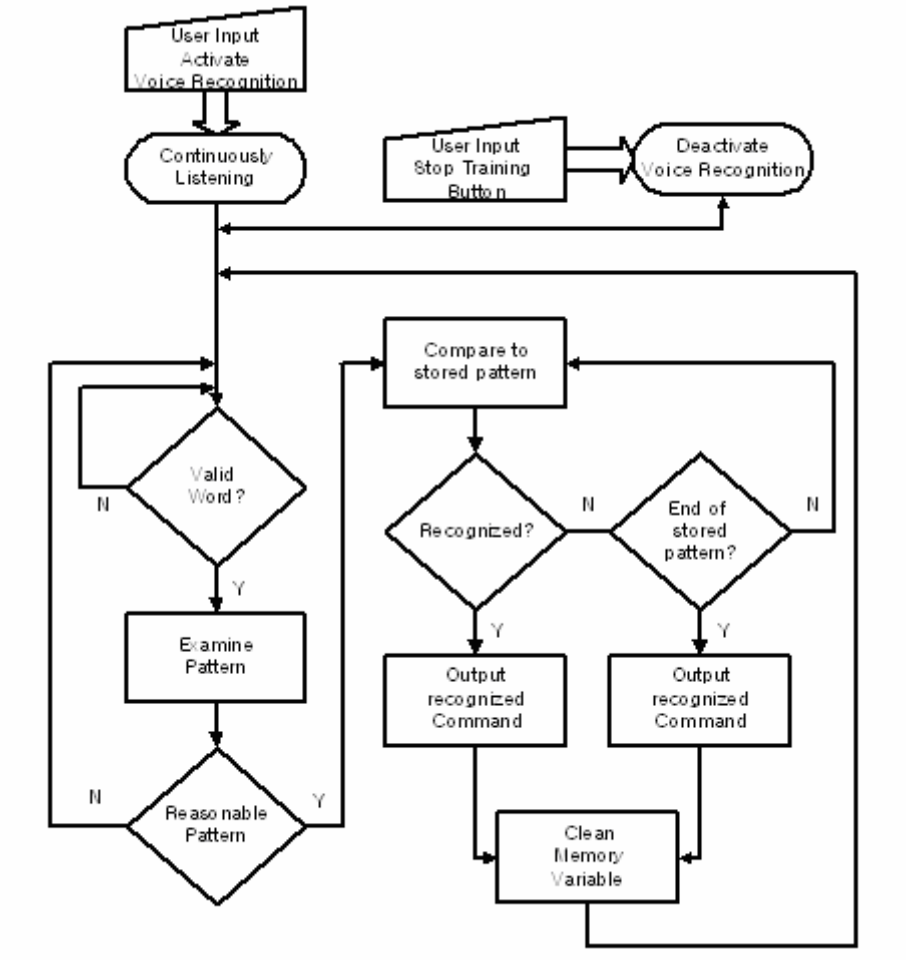


Figure 6: Flow Chart for Recognition Session

4.4 Interrupt

The user must be able to abort any sessions at any time. In order words, the microprocessor should be capable of handling interrupt signals. The push buttons on

the board are sources of interrupt signals in this case. This is particularly useful when the user attempts to stop the training session. This is discussed more in later sections.

5. User Interface

5.1 Input

The VCU receives signals from two sources: Push buttons on the board and the microphone. Push buttons are used for system settings while the microphone accommodates the user's voice commands. The following table lists these inputs.

Table 5: Input and the use

Input	Use
Button A	Start the training session
Button B	Skip the training command
Button C	Abort the training session
Microphone	User input

5.2 User Prompt

A voice recognition system must be able to prompt or alert the user in voice. The development board has "record and play" speech synthesis function embedded to do this. One must ensure that the prompt is simple and precise to minimize the interaction and process time during the training session.

5.3 Debugger

A debugger is an important tool to monitor the overall operation of the system. Although a text-based debugger was suggested at the proposal stage, the nK Logic Group has decided to create a GUI version of the debugger to enhance the development and testing processes.

5.3.1 Debugger Features

The host computer is connected to the board through RS232. Once the development board acknowledges the voice commands, it tells the host computer (debugger) which voice command has been accepted. Then, the debugger outputs the corresponding

features. Since the VCU will be a part of the Start Labs' MP3 player, the nKLogic designed a debugger that imitates an mp3 player display.

The nKLogic Group designed the debugger using Microsoft Visual C language. The following figure shows the GUI.

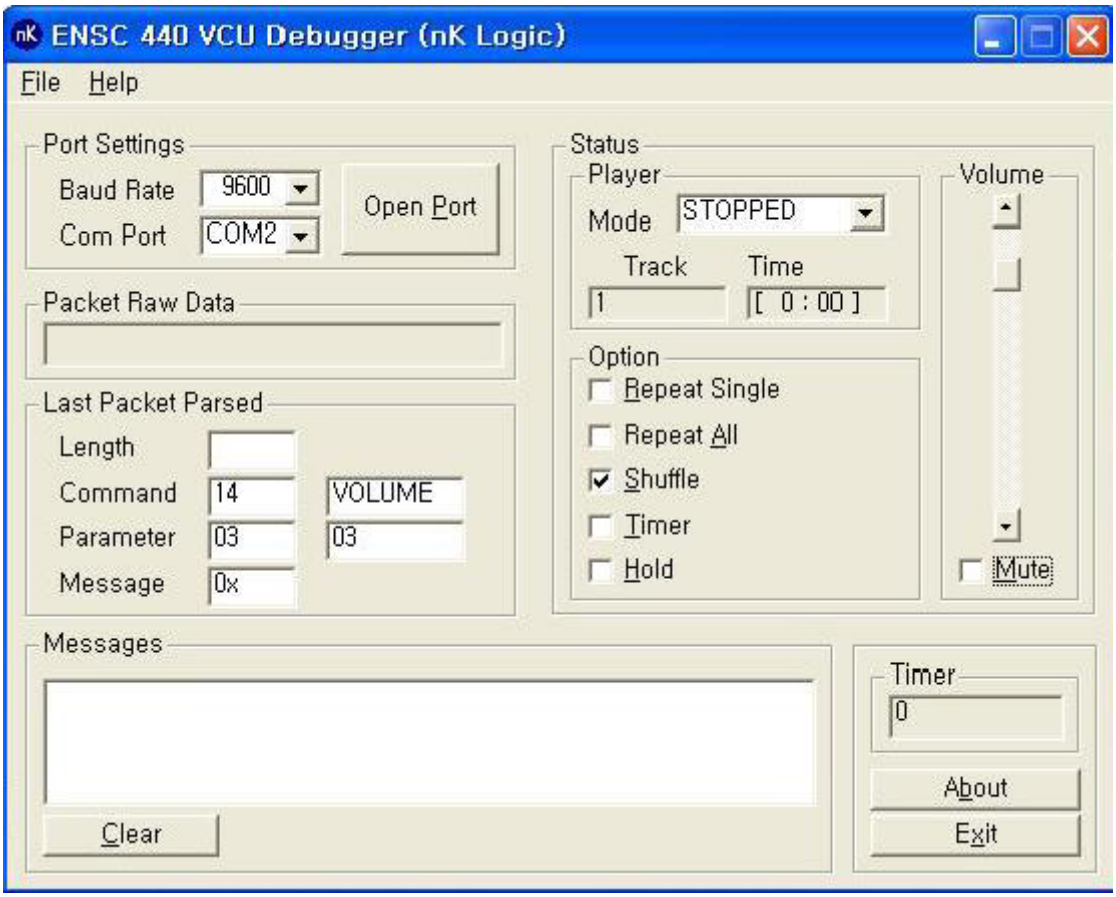


Figure 7: Screen Shot of the Debugger

Here is a summary of the features:

Table 6: Features of the Debugger

Features	Display	Details
Player Mode	Play, stop, rewind, forward, pause	
Player Status	Track number, track time	

Features	Display	Details
Player Option	Repeat	Repeat mode (single or all)
	Shuffle	
	Timer	Include timer value
	Hold	
	Volume	Include Mute option
Messages	Acknowledgement	
	Warnings	
	Errors	

In addition, the debugger let the developer to select communication port properties such as port number, baud rate, parity bit, stop bit, and handshaking.

5.3.2 Debugger Design

In data exchange between the board and the debugger, the standard Type-Length-Value (TLV) data field is used. The following tables clarify this data format:

Table 7: Data Exchange Format

Size	N bytes	1 byte	N bytes	……	N bytes	1 byte
Content	Preamble	Packet Length	TLV	……	TLV	Check Sum

Table 8: Description of Data Field

Content	Description
Preamble	A packet always starts with the preamble of 0xFF. First non 0xFF value is the valid packet length.
Packet Length	Packet length from the first TLV to the Check Sum
TLV	Type-Length-Value field. (See below for more)
Check Sum	The check sum byte. The sum of all data values modulo 256.

Table 9: Description of TLV Field

Field	Description
Type	Specifies the property of the value. Possible Type fields are Command, Parameter, and Message.
Length	Length of the TLV field, excluding the size of Type field
Value	Value field depends on the type field, as defined by the pre-defined grammar structure.

5.4 LED's

LED's on the board are utilized to indicate the status of the system. The following table summarizes the statuses:

Table 10: Output and Its Use

Output	Use
Green LED On	Successful operation / Prompt user
Yellow LED On	Warning / Processing
Red LED On	Error
All LED On	Successful training or recognition
Speaker	Prompt/alert the user with beep or voice
RS232 (Debug port 3 bits)	Connect to the debugger

7. Test Plan

This section outlines test procedures for developers to ensure the proper operation of the VCU. Since the only hardware is Sensory Inc.'s development board, the nKLogic group believes that hardware testing is unnecessary. Thus, assuming all power and cable connections are in place, the test focuses on the functionalities of the unit defined by its software.

The test plan consists of 3 parts: The training test, recognition test, and debugger test. These tests should take place in the order to guarantee the successful completion of the test. To maximize the efficiency of the test plan, the nKLoic group has generated a check-list for testers to follow. When the test gets stuck, upgrade the software and resume the test.

Checklist

1. Training Test
1) Does Button 'A' allow the user to start training process at any time? 2) During the training, does Button A allow the user to repeat the process for the command currently being trained? 3) During the training, does Button B allow the user to skip the command currently being trained? 4) During the training, does Button C allow the user to abort the training session? 5) Does the training session provide sufficient instructions to the user in voice? 6) Does the training session provide the training result correctly?
2. Recognition Test
1) Does the VCU recognize the command in a reasonable time? 2) Does the VCU accept the parameters only when allowed? 3) Does the VCU report the status back to the user?
3. Debugger Test
1) Does the debugger communicate with the VCU properly? 2) Does the debugger reports the status of the VCU properly?

8. Conclusion

This document explains design specifications of the nKLogic's voice recognition unit implement on Sensory Inc.'s Voice Extreme. The hardware specifications of the development board have been discussed. Also, the software design has been detailed out with flow charts and tables.

The nKLogic is confident that the integration of this voice recognition unit with the Start Labs' MP3 player will make the product to stand out in the market.

Appendix A

The following table summarizes the system built-in functions. This table is included so that the designers take advantage of utilizing the best appropriate functions without referring to the manual.

Table 11: Built-in Functions

Function	Description
Configuration	
BOOL SetOutput(UINT8 technology, UINT8 device)	Selects the output device to be used for the sound output technologies.
BOOL SetDebug(UINT8 type, UINT8 voice)	Selects the output voice and level of detail of destination for debug output information. The VE-C default is "NONE" for all types except GENERAL; "SPEECH_OUTPUT" for GENERAL.
BOOL SetStopCondition(UINT8 technology, UINT8 handler)	Selects the stop (jumpout) condition for a given technology.
BOOL SetIOStopCondition(UINT8 port, UINT8 bits, UINT8 states)	Used in conjunction with SetStopCondition to further specify a stop condition based on an IO event by specifying which bit(s) in which port need to reach which state(s) to cause an abort.
BOOL SetKeypadStopCondition(UINT8 key, UINT8 state)	Used in conjunction with SetStopCondition to further specify a stop condition based on a keypad event by specifying which key needs to reach which state to cause an abort.
Speech Synthesis	
void Talk (UINT8 messageNumber, SPEECH *speechData)	Speaks the utterance at index messageNumber in the speechData vocabulary.
SINT8 SenTalk (UINT8 sentenceNumber, SENTENCES *sentenceTable)	Speaks the sentence at index sentenceNumber in sentenceTable.
Pattern Generation	
SINT8 PatGen(UINT8 runHow)	Generates a pattern for SD/SV recognition or training.
SINT8 PatGenW(UINT8 runHow, WEIGHTS *weightTable)	Generates a pattern for SI recognition.
SINT8 GetPatGenResult(void)	Returns the result from the most recent call to Patgen, PatgenW, PatGenWS, CLPatgen or CLPatGenW functions.

Function	Description
void DebugPatGen(void)	Outputs debug information from the most recent call to Patgen, PatgenW, PatGenWS, CLPatgen or CLPatGenW functions.
BOOL SetPatGenMaxWords(UINT8 maxWords)	Controls the maximum number of words allowed by PatGen, PatGenW and PatGenWS.
BOOL SetPatGenSepSil(UINT8 sepSil)	Used in conjunction with SetPatGenMaxWords to control the maximum amount of word separation when >1 words are recorded in PatGen, PatGenW or PatGenWS.
BOOL SetPatGenPreSil (UINT16 preSil)	Controls the amount of time before a PatGen, PatGenW or PatGenWS no data (timeout) result. If nothing is spoken before the PreSil timeout duration, PatGen, PatGenW and PatGenWS will return a value of 1 (no data).
BOOL SetPatGenNoErrors (BOOL OnOff)	Controls whether errors due to “too soft“, “too loud“ or “too soon“ are ignored by PatGen, PaGenW and PatGenWS.
SI Recognition	
UINT8 Recog(WEIGHTS *weights)	Performs speaker independent recognition against a given weights set.
UINT8 Prior(UINT8 favorite, BOOL emphasize)	Postprocesses the results of the most recent Recog to emphasize or remove a specific favorite answer.
UINT8 GetRecogMatch1(void)	Returns the index number of the best match for the most recent Recog or Prior function call.
UINT8 GetRecogLevel1(void)	Returns confidence level of the best match for the most recent Recog or Prior function call.
UINT8 GetRecogMatch2(void)	Returns the index number of the 2nd best match for the most recent Recog or Prior function call.
UINT8 GetRecogLevel2(void)	Returns confidence level of the 2nd best match for the most recent Recog or Prior function call.
UINT8 GetRecogMatch3(void)	Returns the index number of the 3rd best match for the most recent Recog or Prior function call.
UINT8 GetRecogLevel3(void)	Returns confidence level of the 3rd best match for the most recent Recog or Prior function call.
UINT8 GetRecogSetSize(void)	Returns the number of members in the SI recognition set for the most recent Recog or Prior function call.
Void DebugRecog(void)	Outputs debug information from the most recent call to Recog.
SD Recognition	
UINT8 PutTemplate(UINT8 sourceTemplate, UINT8 index, TEMPLATE *baseTemplate)	Copies a PatGen type pattern from an internal Voice Extreme IC buffer to a TEMPLATE array in flash memory.

Function	Description
UINT8 GetTemplate(UINT8 destTemplate, UINT8 index, TEMPLATE *baseTemplate)	Copies a PatGen type pattern from a TEMPLATE array in flash memory to an internal Voice Extreme IC buffer.
Void MaskTemplate (BOOL enable, UINT8 index, TEMPLATE *baseTemplate)	Disables or enables a PatGen type pattern stored in a TEMPLATE array in flash memory.
BOOL SetSDPerformance(UINT8 performanceLevel)	Controls the tradeoff between recognition speed and accuracy for RecogSD.
UINT8 TrainSD(UINT8 srcTemplateA, UINT8 srcTemplateB, UINT8 dstTemplate)	Compares two patterns recorded with PatGen or PatGenWS and averages them into a third template suitable for use with RecogSD or Wordspot.
UINT8 GetTrainSDScore(void)	Returns the comparison score from the most recent call to TrainSD
UINT8 RecogSD(UINT8 numPatterns, TEMPLATE *baseTemplate)	Performs speaker dependant recognition.
UINT8 GetRecogSDResult(void)	Returns the result for the most recent RecogSD function call.
UINT8 GetRecogSDClass1(void)	Returns the index number of the best match for the most recent RecogSD function call.
UINT8 GetRecogSDScore1(void)	Returns the best recognition score from the most recent call to RecogSD
UINT8 GetRecogSDClass2(void)	Returns the index number of the 2nd best match for the most recent RecogSD function call.
UINT8 GetRecogSDScore2(void)	Returns the 2nd best recognition score from the most recent call to RecogSD
UINT8 GetRecogSDDiff(void)	Returns difference between the two best scores from the most recent call to RecogSD.
UINT8 GetRecogSDSetSize(void)	Returns the number of patterns argument (numPatterns) from the most recent call to RecogSD.
void DebugRecogSD(void)	Outputs debug information from the most recent call to RecogSD.
Speaker Verification	
BOOL SetSVSecurityLevel(UINT8 level)	Controls the tradeoff between false accepts (FA) and false rejects (FR) for RecogSV.
UINT8 TrainSV(UINT8 srcTemplateA, UINT8 srcTemplateB, UINT8 dstTemplate)	Compares two patterns recorded with PatGen and averages them into a third template suitable for use with RecogSV.
UINT8 GetTrainSVScore(void)	Returns the comparison score from the most recent call to TrainSV

Function	Description
SINT8 RecogSV(UINT8 element, UINT8 size, UINT8 classes, TEMPLATE *baseTemplate)	Performs speaker verification type recognition.
SINT8 GetRecogSVResult(void)	Returns the result for the most recent RecogSV function call.
UINT8 GetRecogSVWordResult(void)	Returns the current word result for the most recent RecogSV function call.
UINT8 GetRecogSVClass(void)	Returns the index number of the best match for the most recent RecogSV function call.
UINT8 GetRecogSVScore(void)	Returns the best recognition score from the most recent call to RecogSV
UINT8 GetRecogSVSetSize(void)	Returns the set size argument (size) from the most recent call to RecogSV.
void DebugRecogSV(void)	Outputs debug information from the most recent call to RecogSV.
Continuouse Listening	
SINT8 CLPatGen(UINT8 word)	Generates a pattern for SD/SV recognition using continuous listening.
SINT8 CLPatGenW(INT8 word, WEIGHTS *weightsTable)	Generates a pattern for SI recognition using continuous listening.
UINT8 CheckDuration(UINT8 duration)	Checks the duration of the pattern most recently recorded with CLPatGen or CLPatGenW to see if it's reasonable.
BOOL SetCLPerformance(UINT8 performanceLevel)	Controls the tradeoff between recognition speed and accuracy for CLPatGen and CLPatGenW.
BOOL SetCLPreSil(UINT16 period)	Controls the amount of time before a CLPatGen or CLPatGenW no data (timeout) result. If nothing is spoken before the period timeout duration, CLPatGen and CLPatGenW will return a value of 1 (no data).
WordSpot	
SINT8 PatGenWS(UINT8 runHow)	Generates a pattern for Wordspot training.
SINT8 WordSpot(UINT8 timeout, UINT8 index, TEMPLATE *baseTemplate)	Performs speaker dependant recognition with wordspotting.
BOOL SetWSPerformance(UINT8 performanceLevel)	Controls the tradeoff between recognition speed and accuracy for WordSpot.
Record and Play	

Function	Description
SINT8 RecordRP(UINT8 maxTime, UINT8 threshType, UINT8 recordingNumber)	Records digital audio and stores it in the flash memory.
UINT8 PlayRP(UINT8 recordingNumber)	Plays back digital audio stored in the flash memory.
UINT8 PlayFastRP(UINT8 recordingNumber)	Plays back digital audio stored in the flash memory at an accelerated rate.
SINT8 PostRP(UINT8 recordingNumber)	Postprocesses speech from a previous RecordRP to trim and adjust gain.
SINT8 CompressRP(UINT8 destLevel, UINT8 recordingNumber)	Compresses speech from a previous RecordRP to use less flash memory.
UINT8 EraseRP(UINT8 recordingNumber)	Erases speech from the flash memory.
UINT8 GetAvailableMemory(void)	Returns the amount of free memory available for recording in the flash memory.
DTMF	
UINT8 TTone(UINT8 toneNumber)	Generates a single DTMF tone
BOOL SetTToneDur(UINT8 duration)	Specifies the tone duration for subsequent calls to TTone
BOOL SetTToneSil(UINT8 duration)	Specifies the silence duration following DTMF output for subsequent calls to TTone
Music	
SINT8 PlayMusic(UINT8 tune, NOTEDATA *noteData, TUNEDATA *tuneData)	Plays MIDI type music.
BOOL SetMusicFilter(UINT8 filter)	Controls the filter used by subsequent calls to Music
RS232	
void Init232(void)	Initializes the serial communication software drivers and I/O hardware.
void Idle232(void)	Disables serial communication software drivers and I/O hardware.
SINT8 GetPacket (UINT8 MAX_SIZE, CHAR *dataBuffer)	Receives a serial packet from an external serial source.
SINT8 SendPacket(UINT8 bufferSize, CHAR *dataBuffer)	Transmits a serial packet to an external serial destination.
UINT8 PutByte232(INT8 value)	Sends one character to the serial port
UINT8 WaitByte232(void)	Waits indefinitely to receive one character from the serial port

Function	Description
UINT16 WaitByteTimeout232(void)	Waits one second to receive one character from the serial port
UINT8 WriteString232(CHAR *string)	Sends a string of characters to the output port.
Debug Output	
void DebugH4(UINT8 value)	Output the values of VE-C variables or constants
void DebugH8(UINT8 value)	
void DebugH16(UINT16 value)	
void DebugH24(UINT24 value)	
void DebugD8(UINT8 value)	
void DebugD16(UINT16 word)	
void DebugD100(UINT8 value)	
I/O	
void ConfigurePort0(UINT8 controlA, UINT8 controlB)	Allows configuration of all pins in a single IO port.
void ConfigurePort1(UINT8 controlA, UINT8 controlB)	
UINT8 ConfigureIO (INT8 port, INT8 bit, INT8 function)	Allows configuration of a single IO pin.
UINT8 ReadPort0(void)	Reads the value of the input pins of the specified I/O port
UINT8 ReadPort1(void)	
UINT8 ReadOutputPort0(void)	Reads the value last output to the specified I/O port
UINT8 ReadOutputPort1(void)	
void WritePort0(UINT8 value)	Writes to the output pins of the specified I/O port
void WritePort1(UINT8 value)	
UINT8 SleepIO(UINT8 port, UINT8 bits, UINT8 states)	Places the hardware into a low power sleep mode until a specific IO event happens.
UINT8 WaitForIO(UINT8 port, UINT8 bits, UINT8 states)	waits until a specific IO event happens.
Keypad Functions	
SINT8 ScanKeypad(void)	Scans an external 4x5 keypad one time.

Function	Description
SINT8 WaitForKeypadPress(UINT8 key, UINT8 debounceCount)	Waits until a keypad button has been pressed for a specified time.
SINT8 WaitForKeypadRelease(UINT8 key, UINT8 debounceCount)	Waits until a keypad button has been released for a specified time.
Timing Functions	
void DelayMilliSeconds(UINT16 milliseconds)	Delays program execution for a specified time.
void DelaySeconds(UINT16 seconds)	
UINT16 ReadTime(void)	Returns the time since the last system reset.
void SetCrystalTimer2(void)	Allows the application program to specify that the seconds counter (OCS2) is configured with a crystal rather than with the standard RC setup.
UINT8 SleepT2(UINT16 seconds)	Places the hardware into a low power sleep mode for a specified amount of time.
Utility Functions	
UINT8 CopyMemory(UINT16 numberOfBytes, void *source, void *dest)	Copies a block of memory from one place to another.
UINT8 EraseFlash(UINT16 numberOfBytes, void *start)	Erases a block of User flash memory.
UINT8 FillMemory(UINT16 numberOfBytes, UINT8 value, void *dest)	Fills a block of memory with a specified constant.
UINT24* GetApplicationText(void)	Returns a pointer to the application text string.
BOOL GetFirstTime(void)	Returns TRUE the first time an application program is run and FALSE all other times.
UINT8 GetVersion(UINT8 type)	Returns one of four version numbers from the application.
UINT8 Random(UINT8 seed)	Returns a pseudo-random 8-bit number
void ResetSystem(BOOL firstTime)	Performs a soft reset of the application program
BOOL SetLEDOutput(BOOL onOff)	Enables or disable the error output normally sent to the Development Board LEDs
BOOL SetMicDistance(UINT8 Distance)	Affects the internal amplifier gain setting. Choose the value most appropriate for the application.