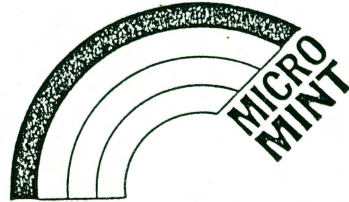# LIS'NER 1000

VOICE RECOGNITION

USER & ASSEMBLY MANUAL

APPLE VERSION

MICRO MINT

THE MICROMINT, INC. 561 Willow Avenue, Cedarhurst, N.Y. 11516

Rev 2.0

TABLE OF CONTENTS

********* Conditions of Sale and PRODUCT WARRANTY *********

MICROMINT, INC., and the Buyer agree to the following terms and conditions of the Sale and Purchase.

1. MICROMINT, INC. extends the following warranty; a factory manufactured circuit board or assembly carries with it a 90 day warranty covering both parts and labor. Any unit which is found to have a defect in materials or workmanship shall at the option of MICROMINT, INC. be repaired or replaced.

2. For repair of units which have expired their warranty, a minimum inspection fee must be prepaid. Contact MICROMINT, INC. for information on current minimum charges.

3. NO WARRANTY is extended on USER ASSEMBLED systems or kits. However, assembled kits will be inspected and repaired with charges based on the current minimum one hour charge. MICROMINT, INC. retains the right to refuse to repair any USER ASSEMBLED item. This right is at the sole discretion of MICROMINT, INC.. However, in the event that repair charges would exceed a reasonable amount, the user may be consulted for a determination. Repairs on user assembled items must be PREPAID. Return authorization must be obtained prior to any return.

4. MICROMINT, INC. shall not be responsible for repair or replacement of any units which become defective through user modification, negligence, abuse and/or mishandling, or improper installation.

5. MICROMINT, INC. shall not be responsible to the Buyer for any loss or claim of special or consequential damages.

6. All units returned for repair must have prior authorization from MICROMINT, INC.. A return authorization number may be obtained by phone or letter. Please retain a record of the return authorization number as most subsequent correspondence will reference the number. Under no circumstances is any product to be returned to MICROMINT, INC. without prior authorization. MICROMINT, INC. will assume no responsibility for unauthorized returns. All returns must be shipped prepaid. Insurance is recommended as losses by a shipping carrier are not the responsibility of MICROMINT, INC.. Repaired units will be returned with postage paid.

7. MICROMINT, INC. reserves the right to change any feature or specification at any time as well as the minimum charges and other condition or warranty contained herein.

********************* REV. 2/84 ************************

======================== DISCLAIMER ========================

THE MICROMINT INC. makes no representations or warranties
with respect to the contents hereof. Further, changes are
periodically made to the information contained herein.
THE MICROMINT INC reserves the right to incorporate these
changes in new editions of this publication without
obligation to notify any person of such revision or changes.

Mention in this document of specific product(s) does not
constitute an endorsement of the product(s); rather, the
information regarding specific products(s) is given
for illustrative purposes. Description of other
manufacturer's interface or technical data is not
intended to supercede information provided by such
manufacturer.

GENERAL INSTRUMENT CORPORATION does not warrant, nor will be
responsible or liable for, the accuracy of information, its
use or, any infringement of patents or other rights of third
parties. GENERAL INSTRUMENT CORPORATION reserves the right
to incorporate changes into the information contained herein
without obligation to notify any person of such revision or
changes.

The GENERAL INSTRUMENT software supplied as a courtesy to
MICROMINT for a source of theoretical data as well as for
distribution by THE MICROMINT is not supported by GENERAL
INSTRUMENT or MICROMINT. It is included on the distribution
diskette merely as educational information.

==

INTRODUCTION

MICROMINT's LIS'NER 1000 is a speaker dependent, voice recognition board for the APPLE II designed around the General Instruments SP1000 Voice Recognition/Synthesis device. Software is provided to support 64 words in eight groups of eight words each. LIS'NER 1000 is also capable of a fixed vocabulary voice output using the LPC speech output from the SP1000 and has provisions for an optional SSI-263 phonetic speech synthesizer chip for unlimited speech with a text to speech algorithm.

The LIS'NER 1000 is "speaker dependent" and must be trained to recognize specific words from a specific person's voice. This may seem to be a more primitive system than a "speaker independent" system, but consider the following points:

(1) Most currently available "speaker independent" systems have recognition accuracy for a vocabulary of only ten words!

(2) "Speaker independent" recognizers can be trained ONLY by the manufacturer. The addition of a single word to a vocabulary usually necessitates restructuring the entire vocabulary.

(3) "Speaker dependent" recognizers (such as the LIS'NER 1000) offer a second level of input flexibility in that an individual user can train the system to recognize commands that make sense to that user. The most obvious example of this power occurs with different languages. The commands to run programs can be trained by each user in his/her own language. Even within a single language, YOU choose the verbal commands which make sense to you. This is not currently possible with a speaker independent system.

(4) Since the LIS'NER 1000 performs its recognition task primarily from software, it can be upgraded or even converted to a "speaker independent" system as improved algorithms become available.

Voice recognition, or Automatic Speech Recognition (ASR) as it is more properly termed, is the use of a machine to identify words spoken by a human being. It should not be confused with Voice Verification, which is the use of a machine to identify a particular speaker. We are concerned with "WHAT was said?", not "WHO said it?".

When the transistor was invented, it was thought that ASR was just around the corner. As investigators began to attack the problem, they learned how little they really knew about human speech. Now, thirty years later, when hundreds of thousands of transistors can be put on a single chip, SOME types of ASR have become practical for SOME applications.

The LIS'NER 1000 brings you what we feel is the most viable implementation of ASR for the present and near future. It combines a low price with high performance and a flexible design philosophy which will enable a graceful upgrade process as ASR techniques improve.

The algorithm provided with the LIS'NER 1000 is called a "discrete utterance" or "isolated word" recognizer. This means that it requires a pause or silence before and after each word. The pause required is only about 200 milliseconds long, and enables the algorithm to adjust itself to the current level of background noise in the user's area. After some practice, you will become used to its operation, but it is worth being conscious of at first.

All speech recognition systems are sensitive to noise. The LIS'NER 1000 features three means of protection against the effects of background noise or aural pollution.

First, the LIS'NER 1000 comes equipped with a headset type microphone to insure that it will be close to the user's mouth. Thus, the user's voice should be much louder and easier to hear than the background noise.

Second, the constant adjustment to the level of noise heard between words (mentioned above) also helps the system to successfully pick out the beginnings and ends of actual words.

Finally, the LIS'NER 1000 has a 'Rejection Threshold' which can be programmed by the user. This feature is particularly useful to prevent reaction to typical room noises such as doors closing, telephones ringing, etc.

Even with these protections, operating the LIS'NER 1000 in a quiet area will provide the best results. If this is not possible, observe the following precaution:

If the unit will be used in an area with sharp intermittent noises (doors closing, telephones ringing), train the unit in a 'QUIET' area and use the Rejection Threshold to protect against the intermittent noise.

HOW TO SPEAK TO THE LIS'NER 1000

Changes in mood or emotional state can heavily "color" our speaking patterns. Differences in context or intent alter the vocalization of a spoken word so that it may convey both gross and subtle differences in meaning. These differences in inflection, pitch, and accentuation can make a word "unmatchable" for a discrete utterance voice recognition system.

All recognizers will perform better with a consistent tone of voice as input. The best way to train and command the LIS'NER 1000 is to adopt the firm tone of voice that might be used in training a dog. Picture a puppy waiting for your command, and you will get fine results when using the LIS'NER 1000.

The LIS'NER 1000 accepts words or phrases up to 2 seconds in length. For best results, use multisyllable words, as they are easier to differentiate than single syllable words. Letters of the alphabet will be more recognizable with words like "Able, Baker, Charlie" rather than the letter names themselves.

It's not necessary to command your LIS'NER in a flat or artificial voice. Just speak a command approximately the same way each time you use it.

================MINIMUM SYSTEM REQUIREMENTS================

1.   APPLE II, APPLE II PLUS or, APPLE IIe COMPUTER

2.   APPLESOFT BASIC in ROM

3.   48K RAM

4.   1 DISK DRIVE

INSTALLING THE CIRCUIT BOARD

SWITCH OFF THE POWER TO YOUR APPLE II COMPUTER!
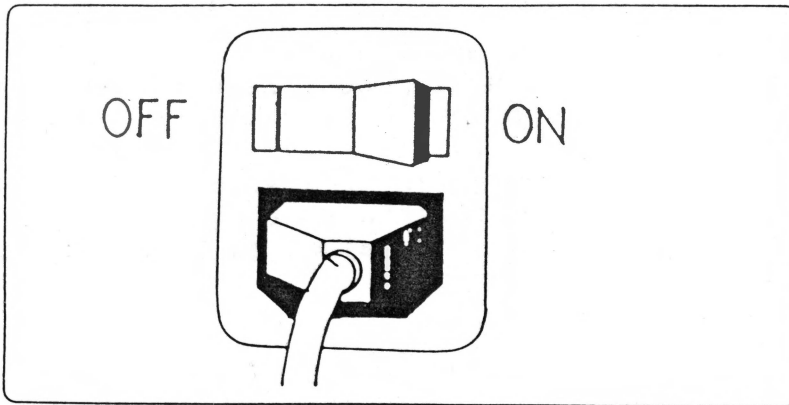
Remove the cover of your Apple and insert the circuit board in any slot (preferably slot 4). Plug the microphone into the mini jack on the board. If your board has the optional SSI-263 Speech Synthesizer chip installed, or you wish to use the SP1000's synthesized voice output, attach an external 8 ohm speaker to the phono jack on the board.
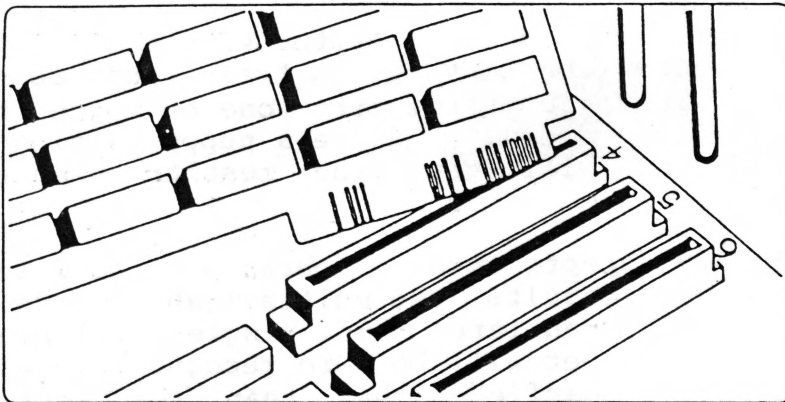
Run the cables through a slot at the rear of the computer, replace the cover, and your computer is ready to listen.
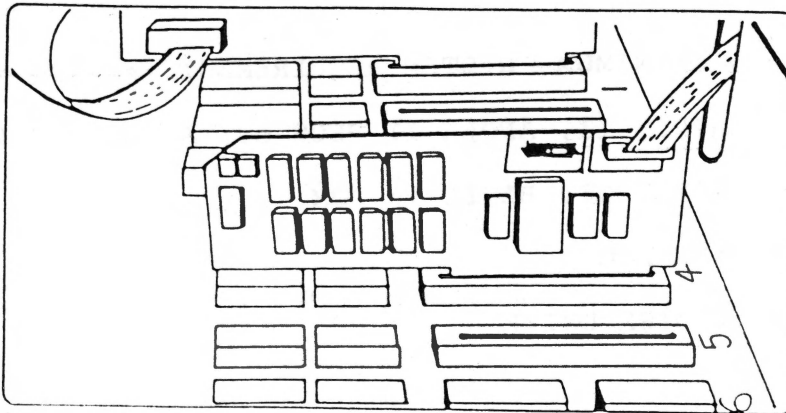
SOFTWARE BACKUP

The software for LIS'NER 1000 is too extensive to fit on a single disk. Rather than increase the cost of the LIS'NER 1000, the machine language source files for recognition are supplied on the "FLIP" side of the MASTER DISKETTE. "FLIPPY" diskettes should be used only for copy masters. The wear caused by the head load pad in the drive will eventually destroy the recording surface on that side of the disk.

```
*******************************************************
* Please copy your disk. File the MASTER for safekeeping *
*              ALWAYS work with a copy.              *
*******************************************************
```

COPYING YOUR LIS'NER 1000 MASTER DISKETTE

1. Place the Apple II System Master diskette in the currently selected drive. Type 'RUN COPYA'(Any file copier may be used to move individual files or any commercial BIT copiers can bulk copy your MASTER DISKETTE. This diskette is not copy-protected in any way.)

2. Answer the prompts to use Drive 1 for the original and for the duplicate. Remove the System Master.

3. Insert the LIS'NER 1000 MASTER DISKETTE into Drive 1 and follow the prompts to make a copy on a blank diskette.

4. Answer the prompt for another copy with 'YES'.

5. Turn the LIS'NER 1000 MASTER DISKETTE over, insert it in Drive 1, and follow the prompts to make a copy on a second blank diskette.

6. Store the LIS'NER 1000 MASTER DISKETTE in a safe place.

PROGRAM DESCRIPTION

The programs on the MASTER DISKETTE for the LIS'NER 1000 are divided into three groups: BASIC utilities, machine language recognition utilities, and machine language synthesis utilities and data files. Exerciser and demonstration software is written in Floating Point APPLESOFT (TM) BASIC so modification is simplified and there is no requirement for additional development software (assemblers, etc). Much of the utility code which supports the BASIC programs is written in machine language to provide the required speed for voice template matching and acquisition.

The two separate functions of the board divide the support software into recognition utilities and speech generation utilities groupings.

Entry and control points in the machine language utilities are documented in the APPLESOFT tm BASIC programs.

LIS'NER 1000 MENU is the 'HELLO' program which executes on power up to simplify selection of the other utility programs. One of the selections on the menu is a utility to change the slot assignment for the board. Slots 1 thru 7 may be used.

LIS'NER 1000 TRAINER & EDITOR is the largest and most useful development program. Because the LIS'NER 1000 is a speaker dependent system, each person who uses the system must train a set of word templates. Once the vocabulary is trained, the **TEST** function within the **EDITOR** can be used to give visual feedback of recognition matching.

Words are trained in groups of one to eight words. As many as four groups may be trained together. In anticipation of changes in vocabulary requirements as well as changes in voice characteristics even for a single speaker, the **EDITOR** portion of the TRAINER/EDITOR allows the user to retrain, add, and delete words from the vocabulary lists.

The **TEST** procedure within the editor allows an option to generate statistics to indicate the recognition accuracy of the vocabulary you have selected.

THE GAME is intended to demonstrate the superiority of voice mode input over keyboard input in applications requiring a short vocabulary list. This game may be used as an easily modified prototype for more advanced games or educational applications.

TEMPLATE LOADER is a sample of the techniques required to initialize the LIS'NER 1000 in your computer so that voice input is active and the correct template vocabulary is loaded.

TEMPLATE LINKER provides a method of relinking word template groups. Like the EDITOR, it generates a "HEADER FILE" that contains the linkage data. Unlike the EDITOR, which is restricted to 32 words, the linker allows for 64 words (8 groups).

DISPLAY COEFFICIENTS gives a real time visual display of the nine coefficient frames as they are generated by the SP1000. The display is updated each time a new set of coefficients is ready. Each of the 8 filter coefficents and the energy factor are shown as decimal and binary values as they are read from the processor. The current software will refresh the screen 50 times a second.

LIS'NER LPC SPEECH DEMO allows the user to hear the high quality voice synthesis capabilities of the SP1000 speech processor. Several sample words and phrases are available as binary data files within the synthesis program support files.

NOTE:- At the time of this writing, only the limited demonstration vocabulary supplied by General Instrument is available for the SP1000. Preparation of LPC (Linear Predictive Coding) speech synthesis data files requires a large scale minicomputer with specialized software and, in most cases, is regarded as an artistic endeavor rather than an exact science. We do hope in the future to be able to prepare a set of phonemes or allophones similar to the SSI-263 phoneme set for text to speech synthesis. Consult the latest MICROMINT catalog for additional LIS'NER utility software.

SLOT is a sequential text file which contains the number of the slot where LIS'NER 1000 is installed. Each of the BASIC programs reads this file and uses the slot number to point the machine language programs to the LIS'NER 1000 board. The main menu allows you to change the slot when you install or move LIS'NER 1000.

Supporting the recognition software are several machine language drivers and binary data files.

RECOGNIZE.OBJ0 is the machine language utility which interfaces the LIS'NER 1000 hardware to BASIC language programs. Routines provide for initialization of the SP1000 processor, determination of silence-speech, compression of data samples, and all of the subroutines needed for matching sample words to templates. This file is provided in object code form only. Source code for the MICROMINT recognition routines are available by license only.

TRAIN.OBJ0 is used by the EDITOR for creation and storage of templates during the training process.

KDISPLAY.OBJ0 is used by the coefficient display program to pre-process the coefficient data from the SP1000 for display on the screen.

DOSWORDS.HEADER is a binary data file used to index and link the templates files.

DOSWORDS1-DOSWORDS4 are the actual template files for the DOS control words.

DOSWORDS.WORDS contain the spoken words used in training. This file isn't used for recognition, but must be present if any further editing or linking of the templates is to be done.

Speech synthesis for the SP1000 also requires several binary file support programs.

SYNTHESIS.S is the text file source for the synthesis driver. It is a modified version of the SYNTH.S text file provided by General Instruments. The major modification relates to the way we chose to integrate the machine language/BASIC interface to make the code slot independent. This file can be assembled using the APPLE ASSEMBLER supplied by APPLE as part of their DOS TOOLKIT (TM) product.

SYNTHESIS.OBJ0 is a short machine language driver which sends the LPC synthesis parameters to the SP1000.

PROGRAMMING WARNING When this driver initializes the SP1000 for synthesis, it enables interrupts from the SP1000. Do not attempt disk access during speech synthesis operations as the result on the disk may be disasterous.

GI SPEECH PROMO is one of the LPC data files used by the LIS'NER LPC SPEECH DEMO BASIC program.

NUMBERS is the LPC data files for the numbers 0 thru 9.
This file is actually a concatenation of ten smaller files.
Inspect the LIS'NER LPC SPEECH DEMO BASIC program to see how
the starting pointer and length are passed to SYNTHESIS.OBJ
driver prior to calling the driver.

READY is the LPC data file for the phrase "I am ready to
record."

WASHER is the LPC data file for the phrase "The washer fluid
is low".

TIME1 is the LPC data file for the phrase "Good morning.
It's 8 o'clock". This phrase is spoken in a female voice.

TIME2 is the LPC data file for the same phrase "Good
morning. It's 8 o'clock" spoken with a slightly different
inflection.

A space is provided for user template files. The files for
the game vocabulary are in fact dummy files which will need
to be trained before playing the game. It is not necessary
to use the dummy file names for template names.

A complete listing of the program files on each diskette
side can be found in APPENDIX -.

On the next page, we will begin to train the LIS'NER 1000 to
recognize your voice.

Insert your copy of MASTER DISKETTE Side 1 into the Apple's disk drive No. 1 and turn the power on. The LIS'NER 1000 MENU program will run automatically, and display a list of options. If option 7 does not indicate the correct slot for your board, select 7 **CHANGE SLOT ASSIGNMENT** (followed by ‹return›) and enter the proper number. The LIS'NER 1000 MENU will then verify the slot number you entered. Since this is the first time you've used the recognition system, the computer doesn't know your voice, so it must be trained. Choose 1 **RECOGNITION EDITOR** (followed by ‹return›). The first set of words you will train is a set of control words for DOS/BASIC.

Adjust the headset microphone comfortably on your head with the microphone toward your mouth, but out of the direct path of your breath. Hit any key, and the computer will begin prompting you to speak individual command words. Say each word after the computer highlights the word. (Each word is repeated twice in order to get a good average sample. From this sample, a 'TEMPLATE' is created. The template consists of a number of parameters unique to this word only and is used later in the recognition process.) After the entire list has been passed through twice, the templates are saved to disk for later use under the file DOSWORDS.HEADER. To pause during training, hit any key. **To resume, hit any key again.**

If, in the future, you want to retrain all the DOS templates, simply delete the file DOSWORDS.HEADER from the disk. When the EDITOR can't find this file, it will retrain the words automatically. When you decide to use other words at the DOS/BASIC control level, the DOSWORDS files may be edited and trained just like any of the other TEMPLATE files.

The computer now asks for a list of words to be trained or to enter the TEMPLATE EDITOR. Notice the word **'YES'** is displayed under the cursor. This is the default response to the question. You may simply hit ‹return› to select the default response, or enter a 'NO' response followed by ‹return›.

NOTE - You may now SPEAK your responses as well as type them as long as they are among the DOS words you just trained. Say 'RETURN' to enter the EDITOR using voice input, the default = 'YES'.

During the initial training, you may have noticed the cursor changed from solid to flashing, and with and without an **inverse** V. The following explanations will prove helpful:

Voice Active Indicator - When the recognition routines are ready to accept a spoken word, the cursor is a **solid block**. If recognition has been turned off, the cursor is a **flashing block**.

Voice Active / Inactive Toggle - Typing CTRL-V toggles the recognition on and off. The keystroke isn't passed through to the application.

Voice Busy Indicator - When a word has been spoken and the recognition routines are busy trying to find a match, no further words or keystrokes can be accepted. To show that this is happening, the cursor becomes an **inverse V**.

Backspace - Any command sequence beginning with CTRL-B is interpreted as a backspace. If the last characters on the screen are from a spoken word, a backspace will move the cursor to the beginning of that word. If a keystroke put the last character on the screen, a backspace moves the cursor back one space.

2 Output Modes - The recognize routines can return the matched word by either inserting it into the input stream (**BUFFER MODE**), which is the default, or can return the group and word numbers (**INDEX MODE**) which the user can then use in other programs. (All of the above features are available in the buffer mode only.) These features are explained in another section.

Editor - The EDITOR in the software package allows the user to train a vocabulary and save it for easy integration into application software.

```
┌──────────────────────────────────────────────────────┐
│  ┌────────────────────────────────────────────────┐  │
│  │                                                │  │
│  │                                                │  │
│  │                 Q & A AREA                     │  │
│  │                                                │  │
│  └────────────────────────────────────────────────┘  │
│     GROUP 1   T R SRCH        GROUP 2   T R SRCH     │
│  1│        /\              │ 1│                    │  │
│  2│       /  \             │ 2│                    │  │
│  3│      /    \            │ 3│                    │  │
│  4│     /|    |\           │ 4│                    │  │
│  5│   GROUP #              │ 5│                    │  │
│  6│                        │ 6│                    │  │
│  7│  ----WORD #            │ 7│                    │  │
│  8│                        │ 8│                    │  │
│     GROUP 3   T R SRCH        GROUP 4   T R SRCH     │
│  1│                        │ 1│                    │  │
│  2│                        │ 2│                    │  │
│  3│                        │ 3│                    │  │
│  4│                        │ 4│                    │  │
│  5│                        │ 5│                    │  │
│  6│                        │ 6│                    │  │
│  7│                        │ 7│                    │  │
│  8│                        │ 8│                    │  │
│        Command Currently Being Executed              │
└──────────────────────────────────────────────────────┘
```
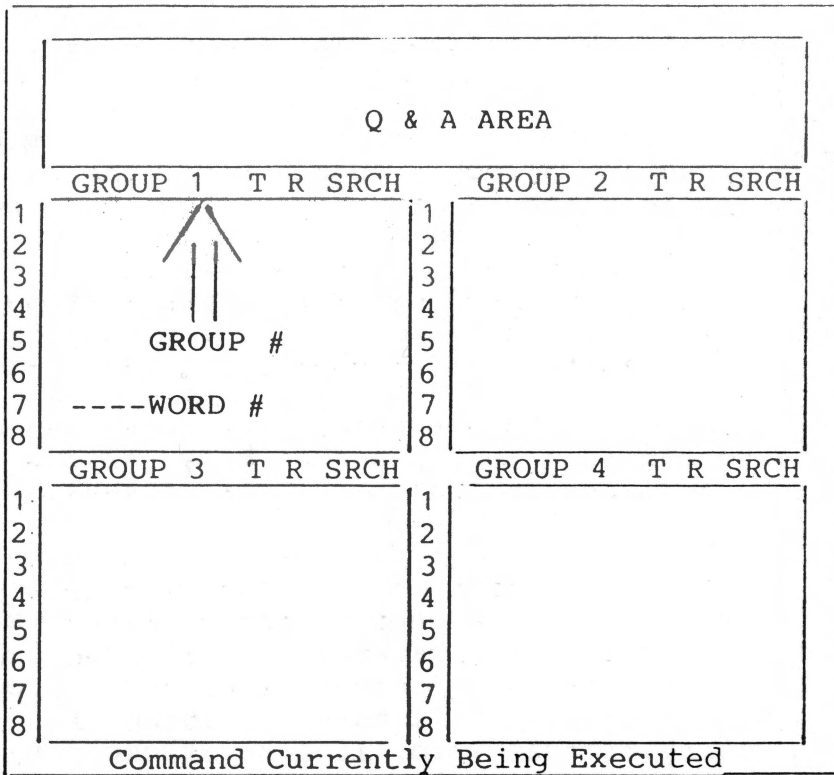
Q & A AREA The top four lines of the screen are used as a question and answer area. This is where the computer prompts you for information and you respond with the appropriate answers.

GROUPS Below that is an area divided into four groups. The group number is found at the top of each group.

WORDS Each group may contain up to eight words. The word number is displayed to the left of each word. When using the EDITOR, each word is referred to by its coordinates (group #, word #) rather than the word itself.

To the right of each word are three columns:

TEST The column, labeled 'T', is used to display the results in the test section. This will be explained shortly.

REJECTION PARAMETER The column labeled 'R', contains the rejection parameter set chosen for the entire group.

Search Flag Byte This column (may actually contain four numbers) labeled 'SRCH' contains the Search Flag Byte for the entire group.

Command Currently Being Executed The bottom line is used to display the command currently being executed.

TRAINING AND TESTING TEMPLATES

The EDITOR is entered from the MAIN MENU, and is used to enter and test your list of words to be recognized. Up to four groups of eight words, or 32 words, may be edited at any time.

Let's take a moment to clarify a few concepts.

First, there are spelled words, in groups, on the screen. These words are used as visual prompts for speech training.

Second, in the template buffer (off screen), you will create a representation of a sound pattern for the prompting word in the displayed word list. This template pattern can be created from a whistle, a grunt, or any other word or sound made near the microphone as the spelled word is highlighted on the screen.

Third, associated with this utterance template pattern is a sequence of characters which will control the computer. This is refered to as the command sequence. There need be no apparent relationship between the word displayed on the screen, the sound made for the template pattern, and the characters in the command sequence. You may type '<return> on the screen list, say "ENTER", and have the computer turn off all the lights in your house when it recognizes the word. The command sequence may contain any combination of letters, numbers, punctuation marks, and control characters. To enter a control character, hit 'ESC' followed by the character. The control character that you type will be displayed in inverse video to show you it's there.

The first question you will be asked is, 'How many words do you wish to put in the first group (0-8)'. Enter a number from zero to eight, or just return for the default value of eight. (Entering '0' will allow you to go to the command mode: more on that later.)

Next, you are asked to enter the spelling for each spoken word, followed by its corresponding command sequence. If you want to make the list of words shorter than you originally planned, just hit '<return>' when asked for the next spoken word and you will enter the next section.

After all the words in a group have been entered, you are asked a number of questions. First, you are asked to choose a rejection parameter set for this group of words, a number between 0 and 3. The higher the number, the closer the spoken word must be to the template to be called a match (for now just enter '1'). Next, you are asked for the numbers of the groups which may follow this word. (In the future, this will be referred to as a 'SEARCH BYTE').

Recognition time is a function of the total number of words in the list to be searched. Fewer words will result in a faster response. The 'SEARCH BYTE' is used to shorten recognition time by eliminating words which may not normally follow the last word used. For example, input to an adventure game is normally **verb** followed by **noun** (i.e. get food).   If verbs are in group 1 and nouns are in group 2 and 3, the syntax byte for group 1 would be '23' (the nouns) and the syntax byte for group 2 and 3 would be '1' (the verbs). If just '**<return>** is pressed, the default value is '1234', so all four groups are searched every time.

You will then be prompted to say each of the words a total of two times. (If the phone should ring during training, for example, hit any key to pause the training process. Hit any key again to resume). After training, the computer will now recognize your new word list, and command mode is entered. Below the command prompt, the first letter of each command is listed. If a more complete explanation of the possible commands is desired, hit '<return>. The following is a list and description of each of the commands. As before, each may be spoken as well as typed.

TEST - This option allows you to test how well each word was trained. When this command is selected, you are asked to enter the number of groups you wish to test ('<return> is the default for testing all groups '1234'). Next, you are asked if you want 'accuracy statistics' ('<return>' is the default for no statistics. Now you are prompted to say any word on the list. Either an 'A' or an 'R' is displayed next to the word. If the 'A' is displayed, the word has been accepted and everything is all right. However, if the 'R' is displayed, it means the test routines have rejected the match. Consistent rejections indicate that the word won't be chosen often by the recognize routines and either it should be retrained, or a new rejection parameter set point selected. As before, hit any key to return to command mode.

EDIT - 'Edit' includes options to modify the spoken word and command sequence assigned to individual entries, modify the syntax flag byte and rejection parameter set for the entire group, and modify the values in each rejection parameter set. When each question is asked, the present value is displayed as the default. Just hit return to leave the parameter the same, or enter a new value. If this option is entered accidentally, simply enter an illegal response for the first question (such as '0,0' for 'group #, word #').

ADD - 'Add' allows you to add an entry to the end of a group or start a new group. If you decide not to add a word after the command has been chosen, just hit return when asked for the spoken word.

DELETE - 'Delete' allows you to delete an entry from a group. Use this entry with caution. Once an entry has been deleted, there is no way to get it back. Also note that if, for example, there are words in group 1, 2 and 3 and all the words in group 2 are deleted, group 3 is shifted down into group 2. When this happens, be sure to update all the syntax flag bytes to reflect the change. If this mode is entered accidentally, just hit return when asked if you really want to delete the chosen entry.

RETRAIN - 'Retrain' an individual word template. This is useful if unsatisfactory results are obtained in the test section for a particular entry. If this command is entered by error, just enter an illegal entry for the "which word" question (such as '0,0').

LOAD - 'Load'enters templates from disk into memory so further editing and testing may be performed. You are asked for the name of the templates you want to load. This is the same name that they were saved under. If the DOS templates are to be loaded, use the name 'DOSWORDS'. It is possible to create sets of templates containing up to eight groups using the linker program. However, the editor will only load the first four groups of a set and will ignore any remaining groups. Use this command with caution. <u>Any templates in memory are lost when the new templates are loaded.</u> To exit this command without loading anything, just hit <return> when asked for a file name.

SAVE - 'Save' the templates being worked on for later use. This is used to save all the work you have done up to now. These templates may be loaded by the template loader at some later date for use in your application programs. Remember, the editor supports only four groups of templates at one time, use the linker, after saving them to restore the complete sets. To exit this command without saving anything, just hit <return> when asked for a file name.

QUIT - This option allows you to leave the EDITOR and return to the main menu. Once all your editing and saving is done, you may return to the main menu with the recognize routine and DOS templates still active. (The templates being worked on in memory are lost. Be sure to save all work before leaving the EDITOR!).

EDITING NOTE When you are prompted for a 'group,word' you may enter the two digits with or without a comma separator. i.e. 2,4 and 24 both designate group 2, word 4.

PROGRAMMING NOTE If you decide to modify the EDITOR after you have become familiar with its use, do so with CAUTION. The program is written in BASIC to allow for ease of modification, but you do not have much memory left for additions. After the program is run and variable storage has been allocated, you have a little more than 1K in the BASIC program space for additions.

Demonstration Game for Two Players

From the MAIN MENU, select option 4 for THE GAME. The following notice will be displayed:

```
BEFORE RUNNING THE GAME, TRAIN THE
WORD LIST FOR BOTH PLAYERS

THEN LINK THE LISTS WITH THE
TEMPLATE LINKER

USE THE TEMPLATE LOADER TO LOAD THE
TEMPLATES FIRST, THEN -THE GAME-

REFER TO THE MANUAL FOR MORE EXTENSIVE
INSTRUCTIONS



ANY KEY WILL RETURN YOU TO THE MENU
```

This may seem complicated at first, but these three steps are essential for linking recognition templates to your application software. Let's go through each step together.

Train the Word List for a Player

We've just read the notice and typed any key to get the MAIN MENU again. Now choose the EDITOR (option 1) and we will begin training.

The EDITOR screen shows:

```
DO YOU HAVE A LIST OF WORDS TO BE
TRAINED? YES
```

You type <return> the default = YES
The EDITOR screen then shows:

```
┌─────────────────────────────────────────────────┐
│          LIS'NER 1000 VOICE TRAINER               │
│ ┌───────────────────────────────────────────────┐ │
│ │ ENTER NUMBER OF TEMPLATES IN                   │ │
│ │   GROUP #1 (0-8): 8                            │ │
│ ├───────────────────────────────────────────────┤ │
│ │  GROUP 1   T R SRCH  │  GROUP 2   T R SRCH    │ │
│ │1┌─────────────────┐ 1┌──────────────────────┐  │ │
│ │2│                 │ 2│                      │  │ │
│ │3│                 │ 3│                      │  │ │
│ │4│                 │ 4│                      │  │ │
│ │5│                 │ 5│                      │  │ │
│ │6│                 │ 6│                      │  │ │
│ │7│                 │ 7│                      │  │ │
│ │8└─────────────────┘ 8└──────────────────────┘  │ │
│ │  GROUP 3   T R SRCH  │  GROUP 4   T R SRCH    │ │
│ │1┌─────────────────┐ 1┌──────────────────────┐  │ │
│ │2│                 │ 2│                      │  │ │
│ │3│                 │ 3│                      │  │ │
│ │4│                 │ 4│                      │  │ │
│ │5│                 │ 5│                      │  │ │
│ │6│                 │ 6│                      │  │ │
│ │7│                 │ 7│                      │  │ │
│ │8└─────────────────┘ 8└──────────────────────┘  │ │
│ └───────────────────────────────────────────────┘ │
└─────────────────────────────────────────────────┘
```

You type '6<return>' 6 = SIX WORDS (If by mistake you typed
<return> and the selected default (8 words) don't worry; you
can bail out later.)
Now the EDITOR screen shows:

```
┌─────────────────────────────────────────────────┐
│ ┌───────────────────────────────────────────────┐ │
│ │                                                │ │
│ │ ENTER THE SPOKEN WORD #1:                      │ │
│ │                                                │ │
│ └───────────────────────────────────────────────┘ │
└─────────────────────────────────────────────────┘
```

The label, 'GROUP 1', will be blinking indicating the
present group you're adding a word to.

You type '1<return>' (shorthand for the word 'ONE')
The EDITOR screen shows:

```
┌─────────────────────────────────────────────────┐
│ ┌───────────────────────────────────────────────┐ │
│ │                                                │ │
│ │ ENTER CORRESPONDING COMMAND                    │ │
│ │ SEQUENCE (PRECEDE CTRL CHARACTERS              │ │
│ │ WITH 'ESC'):                                   │ │
```

You type '1<esc><return><return>' (longhand for 1<return>)
The EDITOR screen shows:

```
┌─────────────────────────────────────────────────┐
│ ┌───────────────────────────────────────────┐   │
│ │                                           │   │
│ │ ENTER THE SPOKEN WORD #2:                 │   │
│ │                                           │   │
│ └───────────────────────────────────────────┘   │
└─────────────────────────────────────────────────┘
```

You type '2<return> (shorthand for 'TWO')

Then continue on with   '2<esc><return><return>'

   '3<return>'   '3<esc><return><return>'

   '4<return>'   '4<esc><return><return>'

   '5<return>'   '5<esc><return><return>'

   'CANCEL<return>'   '<esc><backarrow><return>'

The EDITOR screen shows:

```
┌───────────────────────────────────────────────────────┐
│ ┌───────────────────────────────────────────────────┐ │
│ │ CHOOSE A REJECTION PARAMETER SET FOR              │ │
│ │  THIS GROUP (0-3): 1                              │ │
│ └───────────────────────────────────────────────────┘ │
│    GROUP 1   T R SRCH      GROUP 2   T R SRCH         │
│ 1│1                     │1│                        │  │
│ 2│2                     │2│                        │  │
│ 3│3                     │3│                        │  │
│ 4│4                     │4│                        │  │
│ 5│5                     │5│                        │  │
│ 6│CANCEL                │6│                        │  │
│ 7│                      │7│                        │  │
│ 8│                      │8│                        │  │
│    GROUP 3   T R SRCH      GROUP 4   T R SRCH         │
│ 1│                      │1│                        │  │
│ 2│                      │2│                        │  │
│ 3│                      │3│                        │  │
│ 4│                      │4│                        │  │
│ 5│                      │5│                        │  │
│ 6│                      │6│                        │  │
│ 7│                      │7│                        │  │
│ 8│                      │8│                        │  │
└───────────────────────────────────────────────────────┘
```

NOTE:If the screen is still prompting you to enter spoken word #7 it means that you accidently selected 8 instead of 6 words in the list. Just hit <return> to get the prompt shown above and recover.

Regarding the rejection paramter, you type '<return>' the default = 1
The EDITOR screen shows:

```
┌──────────────────────────────────────┐
│  ┌──────────────────────────────┐    │
│  │ ENTER THE NUMBER(S) OF THE GROUP(S) │
│  │ WHICH MAY FOLLOW THIS GROUP IN A    │
│  │ COMMAND STRING: 1234                │
│  └──────────────────────────────┘    │
└──────────────────────────────────────┘
```

You could type '<return>' to select the default = groups 1234. Such a default selection allows the software to search all 4 groups in response to any utterance input. While technically OK (in this case since the other three groups are blank), it will extend the recogniton search time and should not be selected unless desired. Instead type '1<return>' to insure that only group 1 is searched for a match during the game.
The EDITOR screen shows:

```
┌──────────────────────────────────────┐
│  ┌──────────────────────────────┐    │
│  │ ENTER THE NUMBER OF TEMPLATES │    │
│  │ IN GROUP #2 (0-8): 8          │    │
│  └──────────────────────────────┘    │
└──────────────────────────────────────┘
```

You type '0<return>' 0 = no more templates
The EDITOR screen shows:

```
┌──────────────────────────────────────────────┐
│         LIS'NER 1000 VOICE TRAINER            │
│                                               │
│  HIT ANY KEY TO BEGIN TRAINING...             │
│                                               │
│  ┌─ GROUP 1   T R SRCH ── GROUP 2   T R SRCH ─┐
│ 1│ 1            1  1234  │1                    │
│ 2│ 2                     │2                    │
│ 3│ 3                     │3                    │
│ 4│ 4                     │4                    │
│ 5│ 5                     │5                    │
│ 6│ CANCEL                │6                    │
│ 7│                       │7                    │
│ 8│                       │8                    │
│  ┌─ GROUP 3   T R SRCH ── GROUP 4   T R SRCH ─┐
│ 1│                       │1                    │
│ 2│                       │2                    │
│ 3│                       │3                    │
│ 4│                       │4                    │
│ 5│                       │5                    │
│ 6│                       │6                    │
│ 7│                       │7                    │
│ 8│                       │8                    │
└──────────────────────────────────────────────┘
```

Now here is where the training starts. Training should be done with a minimum of background noise present. These samples are the basis of all recognition comparisons. Any additional sounds to the words being spoken will be permanently saved as legal speech. This template will not match the same word without the additional background sound.

Prepare to speak each word after it has been highlighted by the EDITOR/TRAINER. The word list will be repeated twice. Selection of a word for training is random to help you avoid 'putting a period' (falling inflection) on the last word in a sequential list. Speak each highlighted word until the screen shows:

```
 _____
|  _____ |
| |                                      | |
| | ENTER COMMAND:                       | |
| |  (A,D,E,T,R,L,S,Q)                   | |
| |                                      | |
| |                                      | |
```

In case you are wondering what (A,D,E,T,R,L,S,Q) stands for, type '<return>

The EDITOR screen will show:

```
 _____
|  _____ |
| |                                      | |
| | ENTER COMMAND:                       | |
| | A)DD, D)ELETE, E)DIT, T)EST,         | |
| | R)ETRAIN, L)OAD, S)AVE, Q)UIT        | |
| |                                      | |
```

These are the functions you might select while in the EDITOR. At this point, you might wish to test the recognition ability before saving the templates you have created. Type 'T<return>' to enter the TEST mode. The screen will show:

```
 _____
|  _____ |
| |                                      | |
| | ENTER THE NUMBERS OF THE GROUPS YOU  | |
| |   WISH TO TEST: 1234                 | |
| |                                      | |
```

You type '<return>' the default = 1234 (all groups)
Again, it is OK to select all four groups even though you only have one to test. The EDITOR "knows" the other three groups are blank.

The EDITOR screen shows:

```
┌─────────────────────────────────────────────┐ ┐
│ ┌─────────────────────────────────────────┐ │ │
│ │                                         │ │ │
│ │  DO YOU WANT ACCURACY STATISTICS? NO    │ │ │
│ │                                         │ │ │
│ └─────────────────────────────────────────┘ │ │
└─────────────────────────────────────────────┘ ┘
```

You type '<return>' the default = NO
The EDITOR screen shows:

```
┌─────────────────────────────────────────────┐ ┐
│ ┌─────────────────────────────────────────┐ │ │
│ │                                         │ │ │
│ │  SAY THE WORD(S) YOU WISH TO TEST       │ │ │
│ │    HIT ANY KEY TO EXIT                  │ │ │
│ │                                         │ │ │
│ │    A:ACCEPT          R:REJECT           │ │ │
│ │                                         │ │ │
│ └─────────────────────────────────────────┘ │ │
└─────────────────────────────────────────────┘ ┘
```

Now speak any of the six words on the list. The LIS'NER
1000 will match the spoken word to one of the templates you
just trained. An 'A' or an 'R' will be displayed next to
the selected word indicating whether the recognition is
considered **accepted** or **rejected**. (If any word is frequently
rejected, you may want to retrain that word. Refer to the
section on the EDITOR).

Let's presume recognition accuracy is acceptable. Hit any
key to get out of TEST mode. Type 'S<return>' to save the
template. The EDITOR screen shows:

```
┌─────────────────────────────────────────────┐ ┐
│ ┌─────────────────────────────────────────┐ │ │
│ │                                         │ │ │
│ │  ENTER A NAME FOR THE                   │ │ │
│ │  TEMPLATES:                             │ │ │
│ │                                         │ │ │
│ └─────────────────────────────────────────┘ │ │
└─────────────────────────────────────────────┘ ┘
```

You type 'YOURNAME<return>' The template is now being saved
with your name as the template name. You type 'Q<return>' to
quit the editor. The EDITOR screen shows:

```
┌─────────────────────────────────────────────┐ ┐
│ ┌─────────────────────────────────────────┐ │ │
│ │                                         │ │ │
│ │  ENTER COMMAND: Q                       │ │ │
│ │  DO YOU REALLY WANT TO QUIT? NO         │ │ │
│ │                                         │ │ │
│ └─────────────────────────────────────────┘ │ │
└─────────────────────────────────────────────┘ ┘
```

You type 'Y<return>' the default = NO. We are back to the
MAIN MENU again. Now it is time for your opponent to train
his/her own template. Enter the editor and repeat the same
process of training the same words. Remember to save the
game templates under your opponent's name.

<div align="center">LINK THE TWO TEMPLATES TOGETHER</div>

Now that each player has created templates, we will use the
LINKER to combine them into one. From the MAIN MENU, select
TEMPLATE LINKER.   The LINKER screen shows:

```
                  NAME         #  REJ   SEARCH
                  ----         -  ---   ------
GROUP #1:
GROUP #2:
GROUP #3:
GROUP #4:
GROUP #5:
GROUP #6:
GROUP #7:
GROUP #8:
-----------------------------------------------

ENTER THE NAME OF THE TEMPLATE GROUP
 TO BE PLACED IN GROUP 1:
 ('C ' FOR CATALOG)
```

You type 'C<return>' to see the catalog. The bottom half of
the screen will scroll up displaying the catalog.   Hit any
key until the LINKER screen shows:

```
USER TEMPLATES

B 002 YOURNAME.HEADER
B 006 YOURNAME1
T 002 YOURNAME.WORDS
B 002 OPPONENT'S NAME.HEADER
B 006 OPPONENT'S NAME1
T 002 OPPONENT'S NAME.WORDS

ENTER THE NAME OF THE TEMPLATE GROUP
 TO BE PLACED IN GROUP 1:
 ('C ' FOR CATALOG)
```

You type 'YOURNAME<return>'
The LINKER screen shows:

```
 _____
|                                   |
|                                   |
|   PLEASE INCLUDE THE GROUP NUMBER (1-4)
|    WITH THE NAME.                 |
|                                   |
|                                   |
```

If you look at the catalog illustration on the preceding
page, you will notice a '1' after 'YOURNAME. The '1'
indicates the group the words were in when you trained them.
NOTE - If you had trained words in more than 1 group, there
would be a template saved with a name and a group number for
each group of words that you trained (i.e. DOSWORDS1,
DOSWORDS2, DOSWORDS3, and DOSWORDS4.)

You type '**YOURNAME1<return>**'
The LINKER screen shows:

```
 _____
|                                   |
|                                   |
|   ENTER A SEARCH BYTE FOR THIS    |
|    GROUP: 12345678                |
|                                   |
|                                   |
```

You type '1<return>' the default = 12345678 (all word groups
                                            may follow)
The LINKER screen shows:

```
 _____
|                                   |
|                                   |
|   ENTER THE NAME OF THE TEMPLATE GROUP
|    TO BE PLACED IN GROUP 2:       |
|    ( 'C ' FOR CATALOG)            |
|                                   |
```

You type '**OPPONENT'S NAME1<return>**'
The LINKER screen shows:

```
 _____
|                                   |
|                                   |
|   ENTER THE SEARCH BYTE FOR THIS  |
|    GROUP: 12345678                |
|                                   |
```

You type '**2<return>**' the default = 12345678
The LINKER screen shows:

```
┌─────────────────────────────────────────────┐
│                                             │
│                                             │
│   ENTER THE NAME OF THE TEMPLATE GROUP      │
│    TO BE PLACED IN GROUP 3:                 │
│                                             │
│                                             │
└─────────────────────────────────────────────┘
```

You type '**<return>** the default = NONE (end LINKER)
The LINKER screen shows:

```
┌─────────────────────────────────────────────┐
│                                             │
│                                             │
│   ENTER A NAME FOR THE LINKED GROUPS OF     │
│    TEMPLATES:                               │
│                                             │
│                                             │
└─────────────────────────────────────────────┘
```

You type a name such as '**GAME ONE<return>**' and the combined
templates are saved on diskette. NOTE - Remember which name
was linked first (GROUP 1) and second (GROUP 2). The first
name will be PLAYER 1, the second, PLAYER 2. When you begin
THE GAME, the starting player will be picked at random.

LOADING THE TEMPLATES AND RUNNING "THE GAME".

We should now be back to the MAIN MENU. Select TEMPLATE
LOADER and the screen scrolls through the catalog as you hit
any key. At the end of the catalog, the LOADER screen shows:

```
┌─────────────────────────────────────────────┐
│                                             │
│                                             │
│   ENTER THE NAME OF THE                     │
│    TEMPLATES:                               │
│                                             │
│                                             │
└─────────────────────────────────────────────┘
```

You type '**GAME ONE(return>**'
The LOADER screen shows:

```
┌─────────────────────────────────────────────┐
│                                             │
│                                             │
│   ENTER THE NAME OF THE APPLICATION         │
│    PROGRAM:                                 │
│                                             │
│                                             │
└─────────────────────────────────────────────┘
```

You type '**THE GAME<return>**' and you're off...

USING THE DEMONSTRATION GAME

Included on your MASTER DISKETTE is a simple game which
demonstrates the features and power of the LIS'NER 1000. The
game can be played with or without voice input. This is the
first feature of the LIS'NER 1000. During installation, the
software links itself to the keyboard so that it can
interface to many programs without modification. In the case
of the demonstration game, all inputs to the program are
made with BASIC 'GET' or 'INPUT' statements. Application
programs can be written and debugged without worrying about
the voice recognition inputs. When you want to operate with
voice, merely install the software and activate the voice
recognition function.

In order to play the game with voice, you should have
created the recognition templates for the words 'one',
'two', 'three', 'four', 'five', and 'cancel' (see
'Training THE GAME' for instructions). In order to use the
headset microphone with more than one person, simply lay it
down so that each person can speak to it from about 4 to 12
inches away. In a quiet room, the recognition results will
still be quite good.

The object of the game is to match the pairs of symbols
hidden on the screen. When the game starts, 25 white
squares will be shown on the screen. Behind each square is
a symbol. There are 12 possible symbols, with each symbol
located behind 2 squares (one symbol is behind three squares
to total 25). The starting player is selected at random.
PLAYER 1 or PLAYER 2 will be highlighted at the bottom of
the screen indicating "WHO'S TURN". A prompt for ROW or
COLUMN will also be highlighted.

NOTE - Remember that the first template linked (GROUP 1)
will become PLAYER 1, the second will be PLAYER 2. Each
player, in turn, selects first a ROW and then a COLUMN
(CANCEL leaves the COLUMN prompt and backs up for a new ROW
selection). After one square's symbol is revealed, try to
guess which square has a matching symbol hidden behind it. A
match gives the same player another try. A mismatch gives
the other player a turn. Play continues back and forth until
all matches have been found. The player with the largest
number of matches is declared the winner.

The Demonstration Game is included to suggest many great
possibilities for voice input. It is deliberately written
in listable BASIC to facilitate your experimentation with
voice input. The educational potential is particularly
impressive.

The obvious advantage to voice input is that there is no need to remove your eyes from the visual display in order to enter data into a program. Also, voice input is the most natural form of communication for a human being.

The Demonstration Game can be used for educational purposes at almost any level. The essential ingredient needed for success in the game is a sharp memory. Thus, it is competitive for any age group. The concept of using ROW/COLUMN addressing helps familiarize a child with a coordinate system. You can modify the pattern generation so that each row may be a particular color. This would require a child to identify a color with its name in order to specify a row. This concept of row/column naming can be extended to teach numbers, letters or even spelled words. The educational advantage is contained in the fact that communication with the computer is the spoken word. This is the first communication skill a child develops, and the same means of communication your child will use with other human beings. Perhaps your first programming exercise will be to adapt THE GAME for one person.

The hidden symbols used in the game can also be modified. Simply edit the "DATA" statements in the program which define the shapes. Each shape is defined by 25 low resolution pixels whose colors are defined by the numbers appearing in each symbol's data statements. A little experimentation will quickly reveal how to use them.

Consider the possiblities with this and other instructional programs to teach young learners a vocabulary of colors, numbers, letters, and even geometric shapes. Even relationships such as "left/right, larger/smaller, up/down, over/under", etc. can be taught. The simplest activities take on a new value with voice input!

Use the game program as a prototype for more sophisticated games. You could save game templates for family and friends (up to seven), and link them with an eighth template made of each person saying their own name. This eighth template could be used as a setup to THE GAME, selecting the players and pointing to player #1 and #2's word template.

One of the LIS'NER 1000 features you'll need to know about is search control, It is used to switch different templates in and out. Search provides a "MENU" type structure to the recognition words. This improves both recognition accuracy and speed. Here is how it is used:

In the demonstration game, there are two players and six inputs from each player. These are "1,2,3,4, 5,and CANCEL". This means that each player will train each of those six words and save them on disk. When the LINKER is used to combine the two individual templates into one, the linked template would look as such.

```
      GROUP 1  (player one)        GROUP 2    (player two)
1  1                         1  1
2  2                         2  2
3  3                         3  3
4  4                         4  4
5  5                         5  5
6  CANCEL                    6  CANCEL
7  (none)                    7  (none)
8  (none)                    8  (none)
```

Now the application program "knows" which player is expected to speak. At the beginning of a player's turn, the program can load the SEARCH CONTROL with the value which points to that player's group. This means that when the recognizer "hears" a word, it will attempt to match it with one of the words from the group indicated by the SEARCH CONTROL. Thus, it will not take the time to compare it to words from the other groups which are NOT valid inputs, thereby avoiding confusion.

Since the game structure demands a ROW name followed by a COLUMN name from the same person, the SEARCH BYTE for the Group should point to the same group. In other words, the SEARCH BYTE in Group 1 is 1 (Group 1) and the SEARCH BYTE for Group 2 is 2 (Group 2).

This demonstrates the two ways in which search control affects word template group selection. In the case of "WHO GOES?", only the application program knows and it must alter the SEARCH BYTE to point the recognition software to the correct Group (player). Once a ROW name has been entered, however, the Recognition Software "knows" that a COLUMN must follow from the same player. It uses the SEARCH BYTE associated with that group which indicates which group can follow,in this case the same player's Group names.

*********** EXPERIMENTATION IS THE BEST TEACHER ************

Most speech recognition systems are discrete utterance, speaker dependent systems. They may be in the form of expansion boards for existing computers or as stand-alone black boxes. Ultimately however, their function is common. When the user speaks, the computer analyzes the acoustical signal and compares it to stored templates, and decides which of them most closely resembles the spoken word. Once a candidate is chosen, the computer can respond to the user's utterance or output a control signal to another device.

Each stage of the analysis and pattern matching procedure can be carried out by a variety of techniques. Advances in technology which reduce the complexity of converting an analog input, reduce needed memory space, and speed computation, make the task easier.

The earliest techniques used a simple zero crossing detector to produce a pattern somewhat related to frequency. It soon became evident that speech, which is a complex combination of freqencies, could not be so easily represented. The next refinement was to break the voice frequencies out through a series of filters and separately record energy levels. While economically attractive since it used readily available components, the massive quantities of data gathered proved ponderous and slow to compute. Many systems on the market still use this technique.

One significant advance in estimating the amplitude spectrum of speech is called Linear Predictive Coding (LPC). This method predicts the amplitude of a speech waveform at any instant by combining the amplitudes at a number of earlier instants. The LPC coefficients that best approximate the speech waveform can be mathematically converted to approximate the amplitude spectrum. In speech applications the LPC analyzer is basically a lattice of filters which approximate a series of resonant cavities simulating the vocal tract.

The MICROMINT Voice Recognition System

The MICROMINT speech recognition system, called the LIS'NER 1000,is both a voice recognition and voice synthesizer board using the SP1000. The schematics presented are specifically for the APPLE II but are applicable to other 6502 based systems such as the Commodore 64.

The LIS'NER 1000 is designed to fit future as well as
present needs. The LIS'NER 1000 hardware forms the front end
of the recognition system by performing feature extraction
of the incoming audio signal. The board is supported by a
host microcomputer which compares these features with those
of the templates stored in memory and makes the recognition
decisions. Such a separation of system tasks leaves control
of system performance to the system designer. The LIS'NER
board can be used in speaker dependent or independent
systems with connected or unconnected speech. The designer
is not locked into a specific recognition algorithm that may
not suit his or her needs. Instead, the recognition
algorithm is contained in software resident in the host
microcomputer and can be easily upgraded to take advantage
of advances in recognition techniques, without requiring
hardware redesign.

A detailed description of the functions of the LIS'NER 1000
follows, but keep in mind a few basic concepts:

The GI SP1000 audio processor is programmed by the LIS'NER
1000 software to sample the analog audio data from the
microphone amplifier at a rate of approximately 6.2 KHZ. The
SP1000 analyzes the audio data, in a manner similar to
"averaging", though this is an over-simplification. The
results of the analysis is eight byte values corresponding
to a function or waveform definition and a ninth byte
corresponding to the average energy (in decibels) of the
waveform. The nine bytes are called a frame. The frame
collection rate used by the LIS'NER 1000 algorithm is 50
frames per second. Additional processing of this highly
compressed data is under control of the algorithm operating
in the APPLE, and will be used in the actual recognition
process.

The General Instrument SP1000

The GI SP1000 is a 5 volt 28 pin N-MOS microprocessor
peripheral chip that can be used for both speech recognition
and LPC speech synthesis. Using a bidirectional data bus
and control lines, it interfaces to most 8 bit processors as
a memory mapped peripheral device.

The unique aspect of the SP1000 is its ability to do Linear
Predictive Coding (LPC) analysis in real time. LPC analysis
solves for the coefficients A(I) in an equation of the form:

$$X'(K) = A(1) * X(K-1) + A(2) * X(K-2) + ...A(N) * X(K-N)$$

where

$X'(K)$ is an approximation of $X(K)$

Typical techniques used to solve for the coefficients involve matrix calculations and manipulations. Such techniques, which require large amounts of memory and extensive calculations, preclude their use on an inexpensive device at this time.

The SP1000 uses a modified form of a "Correlation Cancellation Loop" (CCL) in a reconfigurable lattice structure. The CCL approach can be used to operate directly on the incoming data stream without extensive buffering of data or exorbitant processing power. The predictor coefficients $A(I)$ are taken from the integrator output of each stage. The stages can be cascaded for higher order analysis and multiplexed in low bandwidth applications.

In simpler terms, by modifying the feedback control scheme within the filter itself, the ultimate number of computations is reduced. The CCL approach requires 300 bits of working storage versus 3000 bits for a standard covariance or autocorrelation analysis. It also has the interesting property of being able to "run backwards" with a minimum of reconfiguration. This property allows the SP1000 to be used as a speech synthesizer as well as an analyzer for recognition.

The SP1000 can perform useful speech analysis with a relatively inexpensive 8 bit A/D converter. The major reason for this is the use of an on-board Automatic Gain Control (AGC) algorithm. The three gain outputs from the SP1000 are used to control a variable gain amplifier. The SP1000 tests the two most significant bits of each incoming sample and lowers the gain if they are too high. The net effect is to keep the amplitude of the analog signal within the dynamic range of the A/D converter, preventing distortion and stabilizing the signal level entering the lattice filter.

When used as a synthesizer, the filter is presented with LPC coefficients of the speech frame to be synthesized. Typically, these coefficients are computed on a minicomputer and stored as files to be loaded into the microprocessor's memory. Eventually, MICROMINT hopes to supply an allophone set which will allow the user to synthesize any word using a text to speech algorithm or dictionary table. The functional use of the LIS'NER 1000 in recognition applications is not dependent on this software and it can be added when it is available.

The desire for user programmable voice output capability immediately did not go unnoticed however. The LIS'NER 1000 is designed to be a functional recognition/synthesizer board. The APPLE II printed circuit board for this project is etched to accommodate an SSI-263 phonetic speech synthesizer chip as well (see Sweet Talker II, Ciarcia's Circuit Cellar, BYTE magazine March '84). Adding the 263 with supporting text to speech algorithm facilitates true voice I/O so the board supports both phonetic and allophone (LPC) generated speech.

LIS'NER 1000 Speech Recognition Board

Figure 1 is a block diagram of the recognition portion of the LIS'NER 1000. It interfaces to the APPLE II through an 8 bit bidirectional bus, and a few control lines. The SP1000 occupies 4 address locations and the SP1000 is written to or read from as any other peripheral at that address. Data is transferred through the data lines whenever the chip select line is active. The read/write line determines the direction of the transfer and the 2 address lines specify the particular register within the chip. Of the 4 registers, 3 are read/write and one is write only.

The recognition system consists simply of the SP1000 and an assortment of analog components. The analog interface consists of filters, amplifiers, switches, and an A/D converter. The purpose of the circuitry is to convert the utterances spoken by the user into a form that the chip can understand. The entire circuit is designed to run on +5 Volts and, except for the SP1000 connection to the host computer, is virtually the same for all applications.
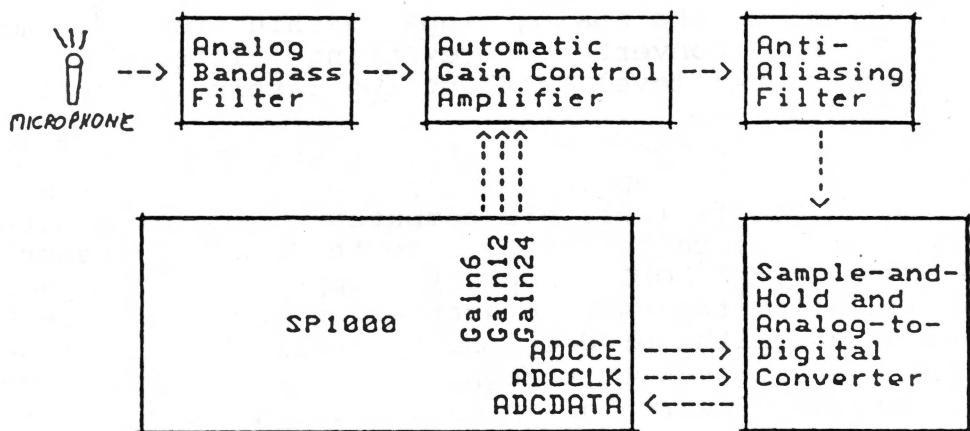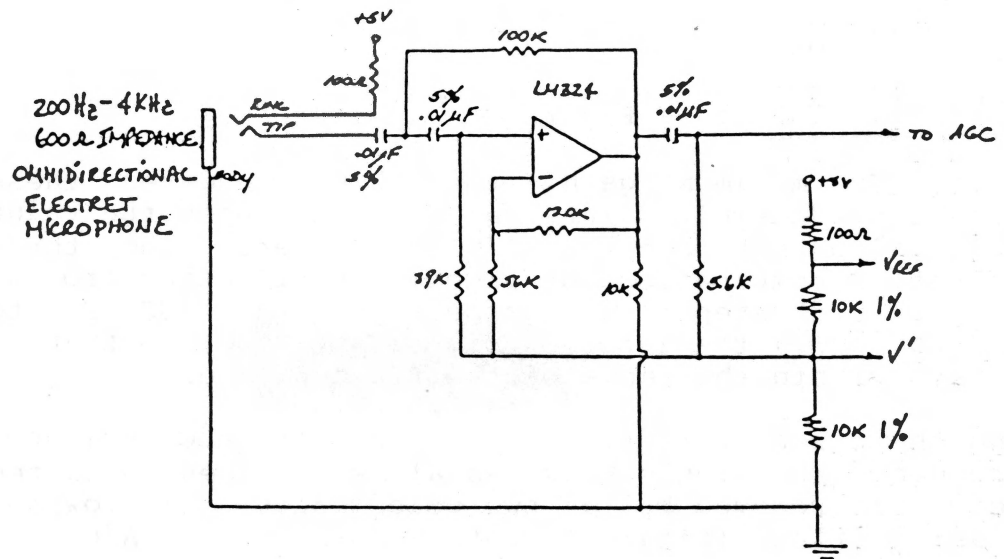

Figure 1

The first section (Figure 2) is the microphone input and high pass filter. For best performance, a 600 Ohm impedance, condenser-type electret microphone should be used. To avoid background noise pickup, a microphone headset combination is suggested. This keeps the mike close to the mouth and limits interference.

The output from the highpass filter is connected to an automatic gain-controlled amplifier (Figure 3). The SP1000 provides three output lines which control switches to vary the resistor values within a circuit consisting of two non-inverting operational amplifiers connected in series. These signals are GAIN 6, GAIN 12, and GAIN 24 corresponding to 6, 12, and 24 db signal levels (this is a voltage gain of 2, 4, and 15.8, respectively).
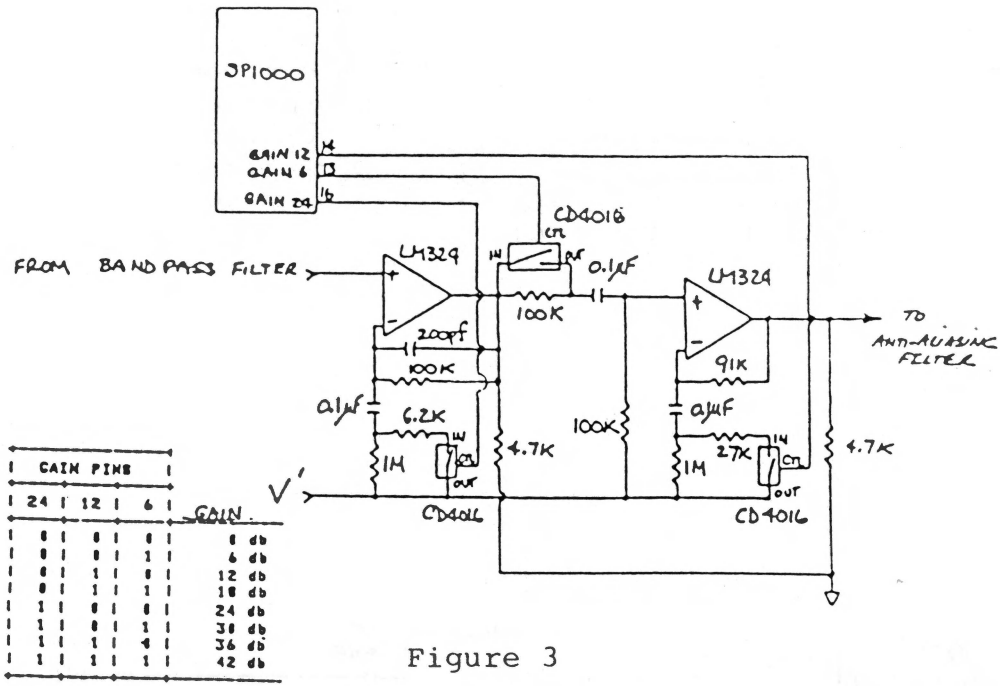
Figure 3

Intially set to maximum gain, the SP1000 updates these
signals at a predetermined interval depending upon the value
of the digital output from the A/D converter. The three
lines create 8 combinations of signal amplification from 42
dB to 0 dB in 6 dB steps. The purpose of the AGC is to
monitor and modify the incoming signal amplitude so that it
always stays within the range of the A/D converter.

Switching these resistors in and out in the AGC produces
high frequency transients known as aliases. These unwanted
frequencies are removed by a two-pole 3200 Hz lowpass
anti-aliasing filter (Figure 4) before going to the A/D.

Figure 4

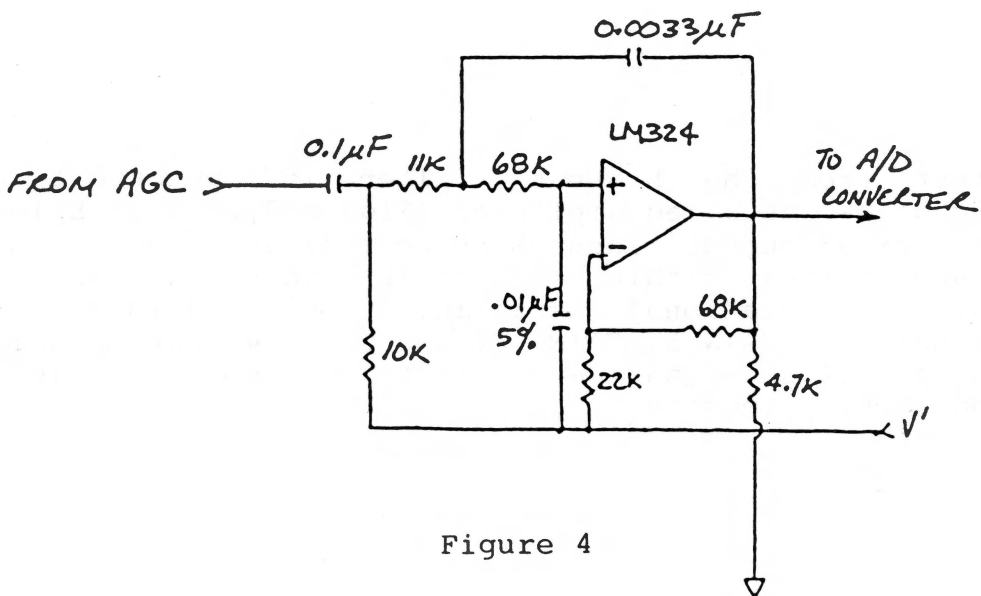Once we have a conditioned signal with all the extraneous
noise removed, it is directed to the sample and hold A/D
converter to be read by the SP1000 (Figure 5).
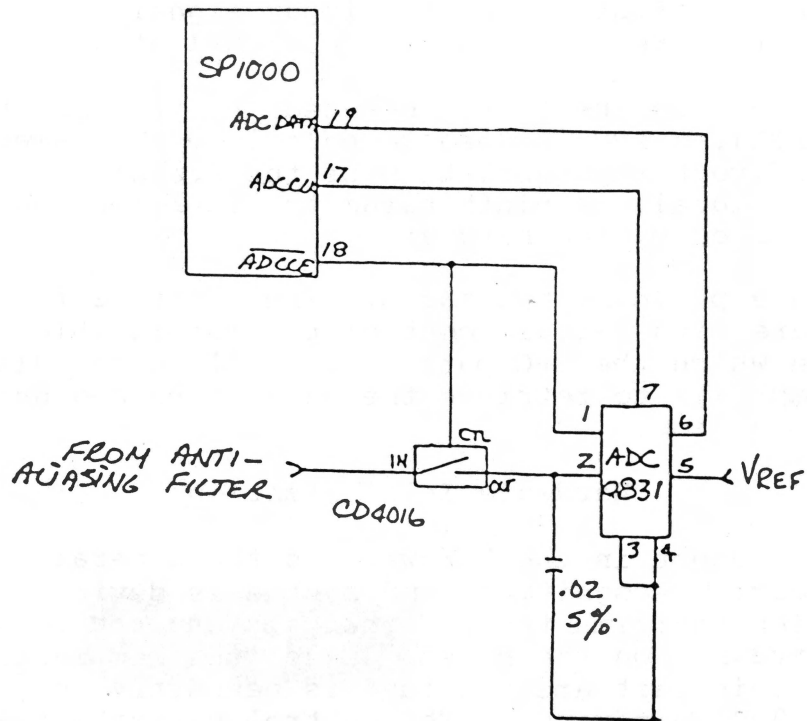
Figure 5

The SP1000 provides two signals for controlling the A/D
converter and the sample and hold circuit. These are ADCCLK
and ADCCE. The ADCCE provides an active low chip select
that turns off the sample switch and enables the A/D (the
switch is normally closed, and opens when the A/D reads the
voltage level stored on the capacitor). The A/D is a
National Semiconductor ADC0831 8 bit serial output converter
clocked at 150 kHz provided through ADCCLK. The serial
output data is read through the ADCDATA line. The SP1000 can
be programmed to read the input data at 5K to 16K samples
per second. As configured by the algorithm, the sample rate
is 6.25 K samples per second. The bit rate of the ADCCLK is
fixed at slightly less than 500 kHz.

A complete schematic showing the recognition and LPC
synthesis, as well as the SSI-263 synthesis portion of the
APPLE II LIS'NER 1000 is shown, in the back of the manual.

HOW THE SOFTWARE USES THE HARDWARE

The hardware portion of the LIS'NER 1000 system is designed
to perform the essential task of generating a parametric
representation of an audio waveform. The purpose of
creating the parametric representation is to capture the
unique characteristics of the input signal in a form which
will facilitate recognition of the signal at a later time.

The representation used is generated by Linear Predictive
Coding (LPC). LPC is used to calculate 8 parameters which
form an indirect representation of the frequency spectrum of
the input signal. A ninth parameter representing energy is
also calculated by the hardware.

The software performs two tasks. The first is to initialize
the hardware with certain control parameters which determine
the way in which the LPC parameters will be calculated. The
second task is to retrieve the parameters and use them for
voice recognition.

Hardware Initialization

The key component in the hardware is the General Instrument
SP1000 voice recognition and synthesis device. It is this
device which performs the LPC analysis and controls the rest
of the hardware on the board. Thus, the few initialization
steps for this part are all that is necessary to start the
LIS'NER 1000 hardware. The control parameters are listed
below with a brief explanation and the reader can refer to
the SP1000 data sheet for complete information.

SAMPLE RATE: The sample rate is the number of times per
second the analog to digital convertor will measure the
voltage of the input waveform. This is set at about 6.5 kHz
for analysis of frequencies up to about 3.2 kHz.

FRAME PERIOD: Each set of parameters (eight LPC and one
Energy parameter) is referred to as a 'frame' of data. The
frame period is the amount of time (number of samples) which
the SP1000 will use to calculate its parameters. The frame
period used is 20 milliseconds.

GAIN PERIOD: The SP1000 provides automatic gain control of
the input amplifier. The purpose is to maintain a useful
signal amplitude without letting the signal become so large
as to saturate the amplifier or analog to digital convertor.
The SP1000 monitors the data coming in and switches the gain
up or down for best results. The gain period is the amount
of time the SP1000 will wait between these gain adjustments.
The gain period used in the LIS'NER 1000 software is one
half the frame period.

GAIN SET POINT: This parameter controls the level at which the SP1000 will vary the gain of the amplifier to provide an input voltage to the A/D convertor over a range of about 2 volts.
The final controls select Recognition mode for the SP1000 and determine interrupt driven or polled operation. The LIS'NER 1000 uses polled operation.

Recognition Software

The second function of the LIS'NER 1000 software is to perform voice recognition using the parameters provided by the hardware. Since the LIS'NER 1000 is a speaker dependent recognizer, it must be trained to recognize specific words spoken by a specific voice. Thus, the recognition software itself can be thought of as two pieces of software, one for training and one for recognizing.

Training

The training routines (TRAIN.OBJ0 on the diskette) are used only by the EDITOR. The functions performed in training are outlined below. The outline assumes that the EDITOR has prompted the user to speak a word and has told the recognition software that a training word is about to be spoken.

TRAINING

    CALCULATE BACKGROUND NOISE LEVEL

    LISTEN FOR WORD

        WAIT FOR SIGNIFICANT INCREASE IN ENERGY WHICH INDICATES THE START OF A WORD

        SAVE ALL FRAMES OF DATA IN A BUFFER

        WAIT FOR RETURN TO SILENCE LEVEL LASTING LONG ENOUGH TO INDICATE END OF WORD

    CREATE TEMPLATE

        APPLY COMPRESSION ALGORITHM TO REDUCE THE WORD TO NORMALIZED LENGTH OF 12 FRAMES

        IF MULTIPLE TRAINING PASSES ARE BEING USED, THEN AVERAGE THIS TEMPLATE WITH THE TEMPLATES FROM OTHER REPETITIONS OF THIS WORD

        SAVE TEMPLATE IN MEMORY

    EXIT TRAINING

Once training has been completed for all words and the user interface software has saved the templates, the recognition mode may be used. The outline below shows the flow of recognition. It assumes that all housekeeping functions like loading templates and indicating their locations have been performed. The reader will notice that all functions are identical to the training functions until the point of MATCHING is reached.

RECOGNITION

CALCULATE BACKGROUND NOISE LEVEL

    LISTEN FOR WORD

        WAIT FOR SIGNIFICANT INCREASE IN ENERGY WHICH
            INDICATES THE START OF A WORD

        SAVE ALL FRAMES OF DATA IN A BUFFER

        WAIT FOR RETURN TO SILENCE LEVEL LASTING LONG
            ENOUGH TO INDICATE END OF WORD

    CREATE TEMPLATE

        APPLY COMPRESSION ALGORITHM TO REDUCE THE
            WORD TO NORMALIZED LENGTH OF 12 FRAMES

    MATCH TEMPLATE

        COMPARE TEMPLATE JUST CREATED TO TEMPLATES
            CREATED IN TRAINING SESSION TO FIND THE
            ONE WHICH IS CLOSEST

    APPLY REJECTION CRITERIA

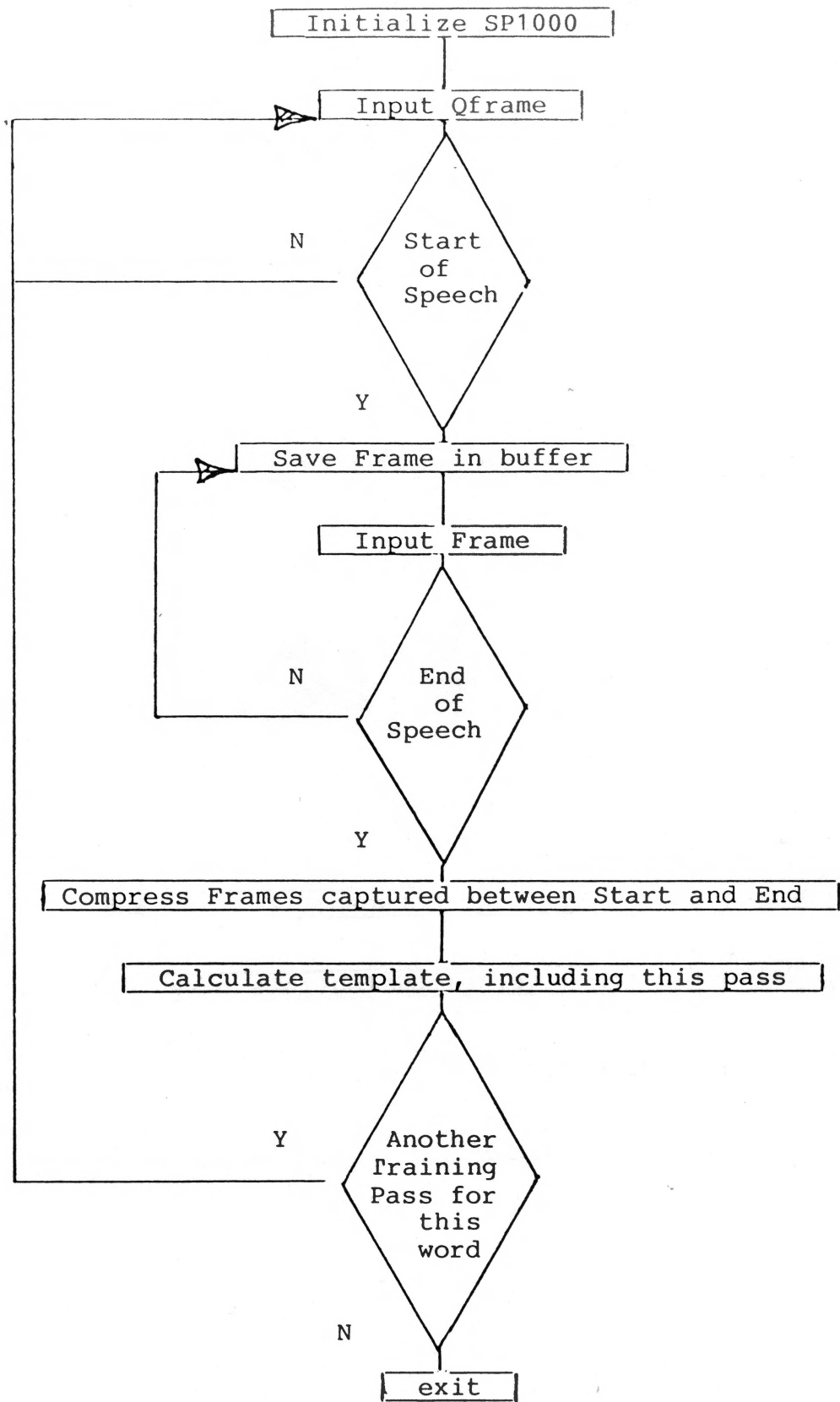   DOES THE TRAINING TEMPLATE WHICH IS CLOSEST
            COME CLOSE ENOUGH TO BE CONSIDERED
            A CORRECT MATCH?

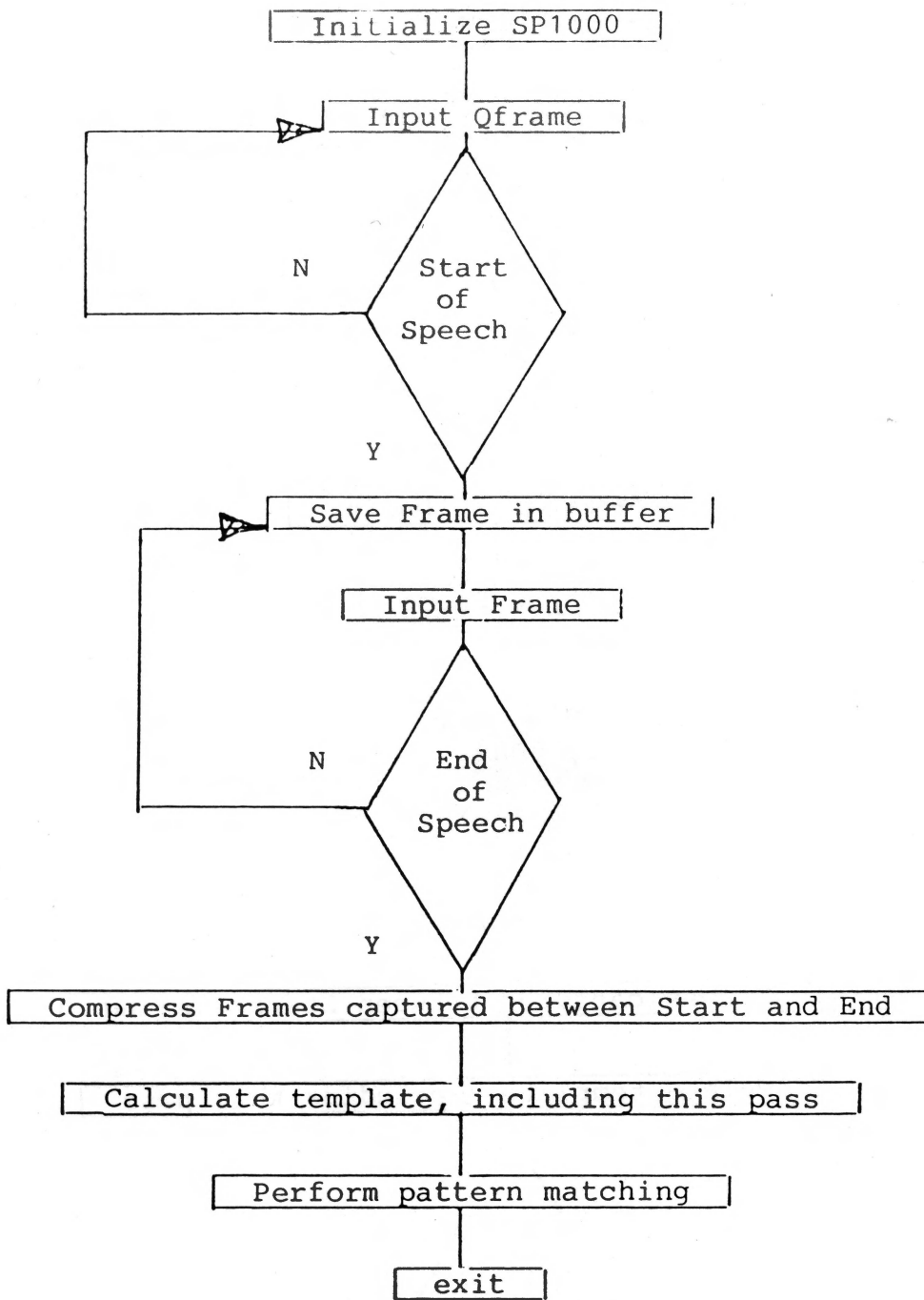   IF YES, THEN SAVE MATCH AND RETURN
                    TO USER INTERFACE SOFTWARE

                IF MATCH IS NOT CLOSE ENOUGH,
                    INDICATE REJECTION AND RETURN
                    TO "LISTEN" AT TOP OF
                    "RECOGNITION"

TRAINING flow chart functional blocks:

```
                          ┌─────────────────────┐
                          │  Initialize SP1000  │
                          └─────────────────────┘
                                     │
                          ┌─────────────────────┐
              ▷───────────│    Input Qframe      │
              │           └─────────────────────┘
              │                      │
              │                     ╱ ╲
              │                    ╱   ╲
              │         N         ╱ Start╲
              │    ┌─────────────╱   of    ╲
              │    │             ╲  Speech ╱
              │    │              ╲       ╱
              │    │               ╲     ╱
              │    │                ╲   ╱
              │    │                 ╲ ╱
              │    │                  │ Y
              │    │      ┌─────────────────────┐
              │    │   ▷──│  Save Frame in buffer │
              │    │   │  └─────────────────────┘
              │    │   │             │
              │    │   │  ┌─────────────────────┐
              │    │   │  │    Input Frame       │
              │    │   │  └─────────────────────┘
              │    │   │             │
              │    │   │            ╱ ╲
              │    │   │           ╱   ╲
              │    │   │    N     ╱ End ╲
              │    │   └─────────╱   of   ╲
              │    │             ╲ Speech ╱
              │    │              ╲      ╱
              │    │               ╲    ╱
              │    │                ╲  ╱
              │    │                 ╲╱
              │    │                  │ Y
              │    │  ┌───────────────────────────────────────────┐
              │    │  │ Compress Frames captured between Start and End │
              │    │  └───────────────────────────────────────────┘
              │    │                  │
              │    │  ┌───────────────────────────────────────────┐
              │    │  │ Calculate template, including this pass    │
              │    │  └───────────────────────────────────────────┘
              │    │                  │
              │    │                 ╱ ╲
              │    │                ╱   ╲
              │    │      Y        ╱Another╲
              │    └──────────────╱Training  ╲
              │                   ╲Pass for   ╱
              │                   ╲ this     ╱
              │                    ╲ word   ╱
              │                     ╲      ╱
              │                      ╲    ╱
              │                       ╲  ╱
              │                        ╲╱
              │                         │ N
              │                ┌─────────────┐
              │                │    exit     │
              │                └─────────────┘
```

RECOGNITION flow chart functional blocks:

```
                    ┌─────────────────────┐
                    │  Initialize SP1000  │
                    └─────────────────────┘
                               │
              ┌───────────────────────────────┐
          ▷▷──│        Input Qframe           │
         │    └───────────────────────────────┘
         │                     │
         │                  ╱─────╲
         │                ╱         ╲
         │              ╱    Start    ╲
         │       N    ╱        of        ╲
         │  ◁────────          Speech     
         │              ╲               ╱
         │                ╲           ╱
         │                  ╲───────╱
         │                     │
         │                     │ Y
         │          ┌─────────────────────────┐
         │      ▷▷──│   Save Frame in buffer   │
         │     │    └─────────────────────────┘
         │     │               │
         │     │      ┌──────────────────┐
         │     │      │   Input  Frame   │
         │     │      └──────────────────┘
         │     │               │
         │     │            ╱─────╲
         │     │          ╱         ╲
         │     │        ╱    End      ╲
         │     │  N   ╱       of        ╲
         │     │ ◁────        Speech
         │     │        ╲               ╱
         │     │          ╲           ╱
         │     │            ╲───────╱
         │     │               │
         │     │               │ Y
         │     │  ┌──────────────────────────────────────────────┐
         │     │  │ Compress Frames captured between Start and End │
         │     │  └──────────────────────────────────────────────┘
         │     │               │
         │     │  ┌──────────────────────────────────────────┐
         │     │  │  Calculate template, including this pass   │
         │     │  └──────────────────────────────────────────┘
         │     │               │
         │     │      ┌──────────────────────────┐
         │     │      │  Perform pattern matching  │
         │     │      └──────────────────────────┘
         │     │               │
         │     │         ┌──────────┐
         │     │         │   exit   │
         │     │         └──────────┘
```

The following list of locations may be useful to the
advanced programmer who wishes to integrate recognition into
his programs.

base - Software revision 1.0 is assembled to operate at
      address $8000 (32768).   Less than 100 bytes
      remain between the end of recognize and the
      standard DOS buffers. (MAXFILES = 3)

base+$2C,$2D:INITREC - This is the address of the
      initialization routines. NOTE: that this is
      an ADDRESS and not an instruction. An example
      of how to use this from BASIC might be:

  10 INITREC = PEEK(base+44) + PEEK(base+45) * 256
    : CALL INITREC

Some of the programs on your disk use a variable
      name of SETUP rather than INITREC.

base+$2E,$2F:RECOG - This is the address of the entry
      point to the recognize routines. This is not
      normally used in buffer mode, but used
      extensively in index mode. An example of how
      to use this from BASIC might be:

  20 RECOG = PEEK(base+46) + PEEK(base+47) * 256
    : CALL RECOG

base+$30,$31:TESTWRD - This is the address of the entry
      point to the test routines. Not recommended
      for general use due to entry parameter setup
      requirements and normal result variations.

base+$32:SLOT - The slot number of the board, multiplied
      by 16 is placed here. For example, if the
      slot number was 4 then 64 would be the value
      held in this location.

base+$33:RETFLG - This location controls the output mode
            of the speech recognition routines. If 0 then
            buffer mode is used and the output will be the
            command sequence associated with the selected
            template. If 1, then index mode is used.

        To select buffer mode : POKE(base+51),0
        To select index mode : POKE(base+51),1

base+$34:SRCHBYTE - Contained here is the search control
            byte of the previous word (set to $0F during
            initialization).   This allows the application
            program to have control over the search
            process used by the recognize routines. For
            example, if a program has two menus with five
            commands each, it is desirable to be able to
            search one list or the other, but not both. So
            before prompting for a command from menu one,
            the user can place a $01 in SRCHBYTE to
            restrict the search to group 1. Since each of
            the eight possible groups (1-8) has   a
            corresponding SEARCH CONTROL BIT in this BYTE
            (bits 0-7) the user can place a $02 in
            SRCHBYTE ($02 turns bit 1 on = group 2) before
            prompting for a command from menu two and will
            resrict the search to group 2.

base+$35:BESTGRP - While in the index mode, the number
            of the group (0-7) containing the word with
            the best fit may be found here upon return
            from the recognition routines.

base+$36:BESTWRD - While in the index mode, the number
            of the word (1-8) with the best fit may be
            found here upon return from the recognition
            routines.

base+$37:INDEXKEY - While in the index mode, if a key is
            pressed while the recognition routines are
            waiting for a word to be spoken, the keystroke
            is placed here. The locations BESTGRP and
            BESTWRD will contain zero. To intercept the
            keystroke, simply monitor the previous two
            locations for zero.

base+$38:ACCPTVAL   -   The   result   of   the   routine
                is placed in this location. Normally used by
                an application in index mode to determine if
                the last word recognized should be accepted or
                rejected.   Contains 0 for accept and 255 for
                reject.

base+$39:CONFID - This location holds the confidence
                level (1-4) of the word match while in test
                mode and is not altered by the recognition
                routines.   If it contains a 0, a key was
                pressed before a word was recognized.   Not
                normally used.

base+$3A:TESTTYP - The value in this location determines
                the mode the test routines will enter. If a 0
                is placed here, the test routines will return
                a confidence level. If a 1 is placed here,
                the test routines will just test for
                accept/reject. Not normally used.

base+$3B:ACTIVE - This byte is both status and control.
                It indicates whether the user has toggled
                voice input using the keyboard 'ctrl V'.

    30 ASTATUS = PEEK (base+$2B) : REM STATUS OUTPUT

        ASTATUS = 0 = voice input disabled
        ASTATUS = 1 = voice input disabled

    30 POKE base+$2B,ASTATUS : REM USER CONTROL INPUT

base+$3C:OVFLW - Raw template data is stored in a two
                second buffer during "listen" mode. This two
                second limit equates to five to eight words
                spoken as a single utterance. In the event of
                a buffer overflow this location is flagged
                with a value of 255. If you use this status
                in an application it should be reset to 0
                prior to calling for input.

    1000 POKE OVFLW,0
    1010 INPUT G$
    1020 IF OVFLW = 255 THEN PRINT CHR$(7) : GOTO 1010
    1030 RETURN

```
MSB        LSB
|7|6|5|4|3|2|1|0|  6000H              SEARCH FLAG BYTE    6000H        |TEMPLATE 1|  6000H
  | | | | | | | |                     REJECTION PARAMETER  6001H       |TEMPLATE 2|  6080H
  | | | | | | | -group 1                                   6002H       |TEMPLATE 3|  6100H
  | | | | | | -group 2                                                  |TEMPLATE 4|  6180H
  | | | | | -group 3                                                    |TEMPLATE 5|  6200H
  | | | | -group 4              TEMPLATE                                |TEMPLATE 6|  6280H
  | | | -group 5                                                        |TEMPLATE 7|  6300H
  | | -group 6                                                          |TEMPLATE 8|  6380H
  | -group 7
  -group 8
MSB          LSB
|X|X|X|4|3|X|X|X|  6001H                                    606EH
                               | COMMAND SEQUENCE |         607FH

  |0|0| REJECTION 0 |
  |0|1| REJECTION 1 |
  |1|0| REJECTION 2 |
  |1|1| REJECTION 3 |
  | REJECTION PARAMETER |
```

```
                    MSB        LSB              LOAD ADDR LSB    8000H      |GROUP 1|  8000H
                    |7|X|X|4|3|2|1|0|           LOAD ADDR MSB    8001H      |GROUP 2|  8004H
|Individual|0|                                  COMMON FLAG      8002H      |GROUP 3|  8008H
|Common    |1|                                  SEARCH FLAG BYTE 8003H     |GROUP 4|  800CH
|COMMON FLAG |                                                              |GROUP 5|  8010H
                                                                           |GROUP 6|  8014H
|Rejection 0|  |0|0|                                                       |GROUP 7|  8018H
|Rejection 1|  |0|1|                                                       |GROUP 8|  801CH
|Rejection 2|  |1|0|                            LOW REJECTION VALUE   8020H  |REJ SET 0|  8020H
|Rejection 3|  |1|1|                            HIGH REJECTION VALUE  8021H  |REJ SET 1|  8023H
|REJECTION PARAMETER|                           THRESHOLD VALUE      8022H  |REJ SET 2|  8026H
                                                                           |REJ SET 3|  8029H
           |1| |0|0|0|
           |2| |0|0|1|
           |3| |0|1|0|
           |4| |0|1|1|
           |5| |1|0|0|
           |6| |1|0|1|
           |7| |1|1|0|
           |8| |1|1|1|
           | TEMPLATE |
```

```
                          0000H
                          0800H

           BASIC
           (USER)

           GROUP   8     6000H
           GROUP   7     6400H
           GROUP   6     6800H
           GROUP   5     6C00H
           GROUP   4     7000H
           GROUP   3     7400H
           GROUP   2     7800H
           GROUP   1     7C00H

           GROUP HEADER  8000H
                         802CH
           RECOGNITION

                         9600H

           DOS

                         D000H

           ROM

                         FFFFH
```

Memory Map of Voice Recognition & User Program Space

```
MSB           LSB
|7|6|5|4|3|2|1|0| 5F00H
 | | | | | | | |-group 1
 | | | | | | |-group 2
 | | | | | |-group 3
 | | | | |-group 4
 | | | |-group 5
 | | |-group 6
 | |-group 7
 |-group 8

MSB           LSB
|X|X|X|4|3|X|X|X| 5F01H
         |0|0| REJECTION 0
         |0|1| REJECTION 1
         |1|0| REJECTION 2
         |1|1| REJECTION 3
         | REJECTION PARAMETER |
```

SEARCH FLAG BYTE	5F00H
REJECTION PARAMETER	5F01H
	5F02H
TEMPLATE	
	5F6EH
COMMAND SEQUENCE	5F7FH

TEMPLATE 1	5F00H
TEMPLATE 2	5F80H
TEMPLATE 3	6000H
TEMPLATE 4	6080H
TEMPLATE 5	6100H
TEMPLATE 6	6180H
TEMPLATE 7	6200H
TEMPLATE 8	6280H

	0000H
	0800H
BASIC (EDITOR)	
WORK4	5F00H
WORK3	6300H
WORK2	6700H
WORK1	6B00H
TRAIN.OBJ0	6F00H
DOSWORDS4	7000H
DOSWORDS3	7400H
DOSWORDS2	7800H
DOSWORDS1	7C00H
GROUP HEADER	8000H
RECOGNITION	802CH
	9600H
DOS	
	D000H
ROM	
	FFFFH

MACHINE CODE TRAINING SUBROUTINES	6F00H

```
                MSB           LSB
               |7|X|X|4|3|2|1|0|
|Individual |0|  | | | | | |
|Common     |1|  | | | | | |
|COMMON FLAG|   | | | | | |

|Rejection 0 |0|0| | | |
|Rejection 1 |0|1| | | |
|Rejection 2 |1|0| | | |
|Rejection 3 |1|1| | | |
|REJECTION PARAMETER| | |

           |#1|0|0|0|
           |#2|0|0|1|
           |#3|0|1|0|
           |#4|0|1|1|
           |#5|1|0|0|
           |#6|1|0|1|
           |#7|1|1|0|
           |#8|1|1|1|
           |TEMPLATE #|
```

LOAD ADDR LSB	8000H
LOAD ADDR MSB	8001H
COMMON FLAG	8002H
SEARCH FLAG BYTE	8003H
LOW REJECTION VALUE	8020H
HIGH REJECTION VALUE	8021H
THRESHOLD VALUE	8022H

GROUP 1	8000H
GROUP 2	8004H
GROUP 3	8008H
GROUP 4	800CH
GROUP 5	8010H
GROUP 6	8014H
GROUP 7	8018H
GROUP 8	801CH
REJ SET 0	8020H
REJ SET 1	8023H
REJ SET 2	8026H
REJ SET 3	8029H

USING RECOGNITION WITH PACKAGED SOFTWARE

Recognition may be used in buffer mode with any program, BASIC or machine language, user written or purchased, without modification to the program so long as it uses the standard monitor vector to access the keyboard. This includes INPUT and GET statements in BASIC and a call to RDKEY ($FD0C) from machine language. Recognition will not work with programs which access the keyboard directly. This includes scanning the address $C000 until a keypress is found.

Most machine language programs and many purchased BASIC programs use the latter method, so recognition will not work unless those programs are modified. Standard BASIC programs, including those written by the user, will work.

Using Recognition With Your Programs

INTEGER BASIC PROGRAMMERS NOTE: The MICROMINT does not presently support the LIS'NER 1000 with any INTEGER BASIC programs. However, because the LIS'NER machine language routines interface to BASIC through DOS rather than by any "tricky" programming techniques, there is no theoretical reason why the techniques described throughout the manual for APPLESOFT cannot be used for INTEGER BASIC.

The best pattern to use to initialize the recognition routines and load your templates is found in the template loader program. The template loader normally prompts you for the name of your templates but, if you know in advance what they are, that information can be contained in your program.

The first step in the loading process is to load the recognition routines. These must go at $8000. Next, load the header file, which contains the number of templates, the rejection parameter set, and the search byte for each group. For example, if we were loading the DOS templates, this file would be DOSWORDS.HEADER. The next step is to check the search byte of each group in the header. If it is zero, there are no templates in that group and the loading may stop. If it is non-zero, load that group from disk anywhere in memory and place the load address in the header so the recognition routines know where to find the templates. Continue in this manner until all the groups have been loaded. In this example, these would be the files DOSWORDS1-DOSWORDS4.

The next step is optional, and depends on your application. You may load the spoken words used during training. These are contained in the .WORDS text file (e.g. DOSWORDS.WORDS). There is one word per input line and eight words per group. If a group has fewer than eight words, blank lines are left. They might be used in a program to remind the user which words may be spoken. Next, the initialization routine is called to patch the recognition routines into DOS. The address of this call may be found in the list of special memory locations. Finally, the application program is run.

Programming in Buffer Mode

This section will disect the **TEMPLATE LOADER** program and suggest some minor changes to help explain how you may write programs containing voice recognition. It is suggested that you bring your APPLE up in BASIC, load in the **TEMPLATE LOADER** program, and follow this section carefully.

PROGRAM LINE #'S

1 - 13   Program title, author, date, and copyright notices.

20   Sets the high memory pointer just under the template storage area.

25   Sets the screen to TEXT mode (40 column X 24 lines) and HOMEs the cursor (0,0).

30   Sets D$ = CTRL D (execute following string as DOS command).

35   Turns off the display of any disk activity.

40 - 45   Sets AD (base) = 32768 (H8000) and BLOADs the **RECOGNIZE.OBJ0** file to address AD. This file is always loaded here. (NOTE 1)(NOTE 2)

50 - 55   Sets SETUP = address of the INITREC routine and SL = address of the SLOT location. (NOTE 1)

60 - 65   Sets up a DOS error fix routine.

70	OPENs and READs the SLOT text file, sets S = the slot information, and CLOSEs the file. This can be eliminated if you know what slot the board will be in.
CHANGE	70 S = 4 (if the LIS'NER is in slot 4)
75	POKEs the S (slot number) * 16 into address SL.
80	Lists the CATALOG. Not neccessary.
CHANGE	DELETE 80
85	Asks for you to INPUT the name of the templates to load for your program and sets N$ = to this name. This can be eliminated if you know the name of the templates.
CHANGE	85 N$="ON/OFF"
90	BLOADs N$ with the file extention .HEADER at AD. (see line 40)
95 - 115	Sets FOR/NEXT loop Z = 1 to 8. Checks the group header for a valid #Z template. BLOADs N$;Z (in our example it will be ON/OFF1 - ON/OFF8) at AD - (Z*1024) if the template is valid, otherwise exits FOR/NEXT loop. (NOTE 2)
120	Readjusts the high memory pointer upward if you are using less than eight templates. This gives you the maximum amount of user memory space.
125	Call SETUP (see line 50).
130 - 145	Asks you for the name of your application program and RUNs it. This is unneccessary if your code can be contained here.

```
CHANGE          130 PRINT SPC(240) SPC(240) SPC(240) SPC(120):
                VTAB(1) : PRINT "TURN THE SCREEN (ON/OFF)";

                135 INPUT A$

                140 IF A$ = "OFF" THEN NORMAL : GOTO 130

                145 IF A$ = "ON" THEN INVERSE : GOTO 130

ADD             150 GOTO 135
```

This short program will accept voice input as well as keyboard input on line 135. Lines 140 - 145 take action on the input. In this case the screen gets painted white or black, but with the proper interface it could have turned the lights in your room off and on.

198 -215   GOSUB routine to poke DOS Error Fix Routine to location 768 - 783.

298 - 355   ON ERROR GOTO routine that checks and handles errors to prevent program termination.

If you have made these changes, save the program as 'EXAMPLE'. Now run the EDITOR to a create a template with 2 words in the first group, ON and OFF. When asked for the control sequence use ON<esc><return><return> and OFF<esc><return><return>. SAVE the template as 'ON/OFF'. When you RUN the BASIC program 'EXAMPLE', the following recognition events occur:

1.   RECOGNIZE.OBJ0 is BLOADed to H8000.

2.   INITREC address is calculated.

3.   SLOT address is calculated.

4.   LIS'NER 1000 slot number * 16 is stored at SLOT.

5.   (template name).HEADER is BLOADed to H8000.

6.   (template name)1 is BLOADed to H7C00.
        (template name)2 is BLOADed to H7800 if any exist.
        (template name)3 is BLOADed to H7400 if any exist.
        (template name)4 is BLOADed to H7000 if any exist.
        (template name)5 is BLOADed to H6C00 if any exist.
        (template name)6 is BLOADed to H6800 if any exist.
        (template name)7 is BLOADed to H6400 if any exist.
        (template name)8 is BLOADed to H6000 if any exist.

7.   INITREC is CALLed to initialize recognition.

8.   User program can now receive input from the
        LIS'NER 1000 voice recogniton board.

NOTE 1 See the section on SPECIAL MEMORY LOCATIONS.

NOTE 2 See the section MEMORY MAP OF VOICE RECOGNITION &
USER PROGRAM SPACE.

Programming in Index Mode

As an alternative to using buffer mode, the user may opt to
use index mode in his programs. This allows the user to
control screen activity and recognition results from within
his program. Take, for example, a case where the user wants
to use voice input to control the movement of the cursor on
the screen. Rather than having the recognized command
sequence appear automatically on the screen, as would happen
in buffer mode, he may only want to know the group and word
numbers of the recognized word. In this way, he may move
the cursor without disrupting the rest of the screen.

Also under user control in index mode is the search byte and
rejection results. Just as in buffer mode, the user may
place a value in SRCHBYTE (described in another section) to
control which groups the recognize routines will scan while
trying to match the next spoken word. When the recognize
routines return to the user, the results of the tests are
also available in ACCPTVAL (also described in the Special
Memory Locations section). In the event of a rejection,
this allows the user to display a friendly message
requesting that the word be repeated. While in buffer mode,
the only indication of rejection is the appearance of the
"v" in the solid block cursor on the screen, indicating
activation by a word or sound. If rejection occurs the "v"
simply disappears, leaving the solid block cursor again.

The following is a sample APPLESOFT program using index mode. It may be used as a guide in writing or modifying programs to use this recognition mode.

```
10 REM SAMPLE PROGRAM USING INDEX MODE
20 REM THIS ASSUMES THE RECOGNITION ROUTINES
30 REM AND TEMPLATES ARE ALREADY LOADED
40 REM INTO MEMORY
50 BASE = 32768 : REM BASE ADDRESS
60 INIT = PEEK (BASE + 44) + PEEK (BASE + 45) * 256
70 RECOG = PEEK (BASE + 46) + PEEK (BASE + 47) * 256
80 SLOT = BASE = 50 : MODE = BASE + 51
90 BG = BASE + 53 : BW = BASE +54
100 KEY = BASE + 55 : REJ = BASE + 56
110 POKE SLOT, 4 * 16 : REM SLOT #
120 POKE MODE, 1 : REM SET INDEX MODE
130 CALL INIT : REM INIT ROUTINES
140 REM
150 REM INSERT PROGRAM HERE
160 REM
500 REM NOW WE NEED A WORD...
510 CALL RECOG : REM WAIT FOR A WORD
520 IF (PEEK (BG) = 0) AND (PEEK (BW) = 0) THEN 800 :
          REM KEYSTROKE FOUND
530 IF PEEK (REJ) = 0 THEN 560: REM WORD ACCEPTED
540 PRINT "PLEASE REPEAT YOUR LAST COMMAND"
550 GOTO 510
560 G = PEEK (BG) + 1 : W = PEEK (BW)
570 REM 'G' CONTAINS RECOGNIZED WORD'S GROUP NUMBER
580 REM 'W' CONTAINS RECOGNIZED WORD'S NUMBER
590 REM
600 REM USE RESULTS IN PROGRAM
610 REM
800 K$ = CHR$ (PEEK (KEY)) : REM GET KEYSTROKE FROM
          MEMORY
810 REM
820 REM USE RESULTS IN PROGRAM
830 REM
840 REM CONTINUE PROGRAM
850 REM
```

RESISTORS

( ) R1	91 K		WHT-BRN-ORG
( ) R2	4.7 K		YEL-VIO-RED
( ) R3	68 K		BLU-GRY-ORG
( ) R4	4.7 K		YEL-VIO-RED
( ) R5	120 K		BRN-RED-YEL
( ) R6	4.7 K		BLU-RED-RED
( ) R7	10 K		BRN-BLK-ORG
( ) R8	22 K		RED-RED-ORG
( ) R9	33 OHM 1/2W		ORG-ORG-BLK
( ) R10	100 K		BRN-BLK-YEL
( ) R11	56 K		GRN-BLU-ORG
( ) R12	5.6 K		GRN-BLU-RED
( ) R13	4.7 K		YEL-VIO-RED
( ) R14	100 K		BRN-BLK-YEL
( ) R15	1 M		BRN-BLK-GRN
( ) R16	27 K		RED-VIO-ORG
( ) R17	6.8 K		BLU-GRY-RED
( ) R18	11 K		BRN-BRN-ORG
( ) R19	10 K		BRN-BLK-ORG
( ) R20	10 K 1%		BRN-BLK-BLK-RED
( ) R21	330 OHM		BRN-BLK-YEL
( ) R22	1 M		BRN-BLK-GRN
( ) R23	4.7 K		YEL-VIO-RED
( ) R24	4.7 K		YEL-VIO-RED
( ) R25	39 K		ORG-WHT-ORG
( ) R26	100 K		BRN-BLK-YEL
( ) R27	100 K		BRN-BLK-YEL
( ) R28	1 M		BRN-BLK-GRN
( ) R29	10 K 1%		BRN-BLK-BLK-RED
( ) R30	1 K		BRN-BLK-RED
( ) R31	10 OHM		BRN-BLK-BLK
( ) R32	10 K POT		
( ) R33	2.2 K		RED-RED-RED
( ) R34	4.7 K		YEL-VIO-RED
( ) R35	1 K		BRN-BLK-RED
( ) R36	2.2 K		RED-RED-RED

SOCKETS

( ) IC1	14 PIN DIP SOCKET
( ) IC2	14 PIN DIP SOCKET
( ) IC3	8 PIN DIP SOCKET
( ) IC4	8 PIN DIP SOCKET
( ) IC5	28 PIN DIP SOCKET
( ) IC6	24 PIN DIP SOCKET
( ) IC7	14 PIN DIP SOCKET
( ) IC8	16 PIN DIP SOCKET

CONNECTORS

( ) J1	RCA AUDIO, PC MOUNT
( ) J2	MINI PHONE, STEREO, PC MOUNT

MISCELLANEOUS

( ) PCB1	LIS'NER 1000 PRINTED CIRCUIT BOARD
( ) XR1	7.16 MHZ RESONATOR
( ) JP1	1X3 PIN HEADER
( ) SJ	SHORTING JUMPER
( ) D1	1N4732 ZENER DIODE

I.C.'S

( ) IC1	LM324 QUAD OP AMP	
( ) IC2	CD4066 QUAD CMOS ANALOG SWITCH	
( ) IC3	LM386 AUDIO AMP	
( ) IC4	ADC0831 ANALOG TO DIGITAL CONVERTER	
( ) IC5	SP1000 LPC SPEECH PROCESSOR	
( ) IC6	SSI263 PHONEME VOICE SYNTHESIZER	(Optional)
( ) IC7	74LS00 QUAD NAND GATE	(Optional)
( ) IC8	74LS156 DUAL DECODER	(Optional)

CAPACITORS

( ) C1	.0033 MFD MONOLITHIC 5%(.S. 332)
( ) C2	.1 MFD MONOLITHIC 5%(.S. 104)
( ) C3	.1 MFD MONOLITHIC (104)
( ) C4	.1 MFD MONOLITHIC (104M)
( ) C5	10 MFD ELECTROLYTIC
( ) C6	.1 MFD MONOLITHIC (104M)
( ) C7	.1 MFD MONOLITHIC 5%(.S. 104)
( ) C8	.01 MFD MONOLITHIC 5%(.S. 103)
( ) C9	.1 MFD MONOLITHIC 5%(.S. 104)
( ) C10	.01 MFD MONOLITHIC 5%(.S. 103)
( ) C11	.02 MFD MONOLITHIC 5%(.S. 223)
( ) C12	.1 MFD MONOLITHIC (104M)
( ) C13	39 PFD MONOLITHIC (39K)
( ) C14	22 PFD MONOLITHIC (22K)
( ) C15	.1 MFD MONOLITHIC (104M)
( ) C16	.1 MFD MONOLITHIC (104M)
( ) C17	.01 MFD MONOLITHIC 5%(.S. 103)
( ) C18	.01 MFD MONOLITHIC 5%(.S. 103)
( ) C19	220 PFD MONOLITHIC 5%(221 J)
( ) C20	.1 MFD MONOLITHIC 5%(.S. 104)
( ) C21	100 MFD ELECTROLYTIC
( ) C22	100 MFD ELECTROLYTIC
( ) C23	.1 MFD MONOLITHIC (104)
( ) C24	.1 MFD MONOLITHIC (104M)
( ) C25	10 MFD ELECTROLYTIC
( ) C26	.1 MFD MONOLITHIC (104M)
( ) C27	100 MFD ELECTROLYTIC
( ) C28	100 MFD ELECTROLYITC
( ) C29	.1 MFD MONOLITHIC (104M)
( ) C30	.1 MFD MONOLITHIC (104M)
( ) C31	.02 MFD MONOLITHIC 5%(.S. 223)
( ) C32	10 MFD ELECTROLYTIC
( ) C33	.1 MFD MONOLITHIC (104M)
( ) C34	.02 MFD MONOLITHIC 5%(.S. 223)
( ) C35	10 MFD ELECTROLYTIC
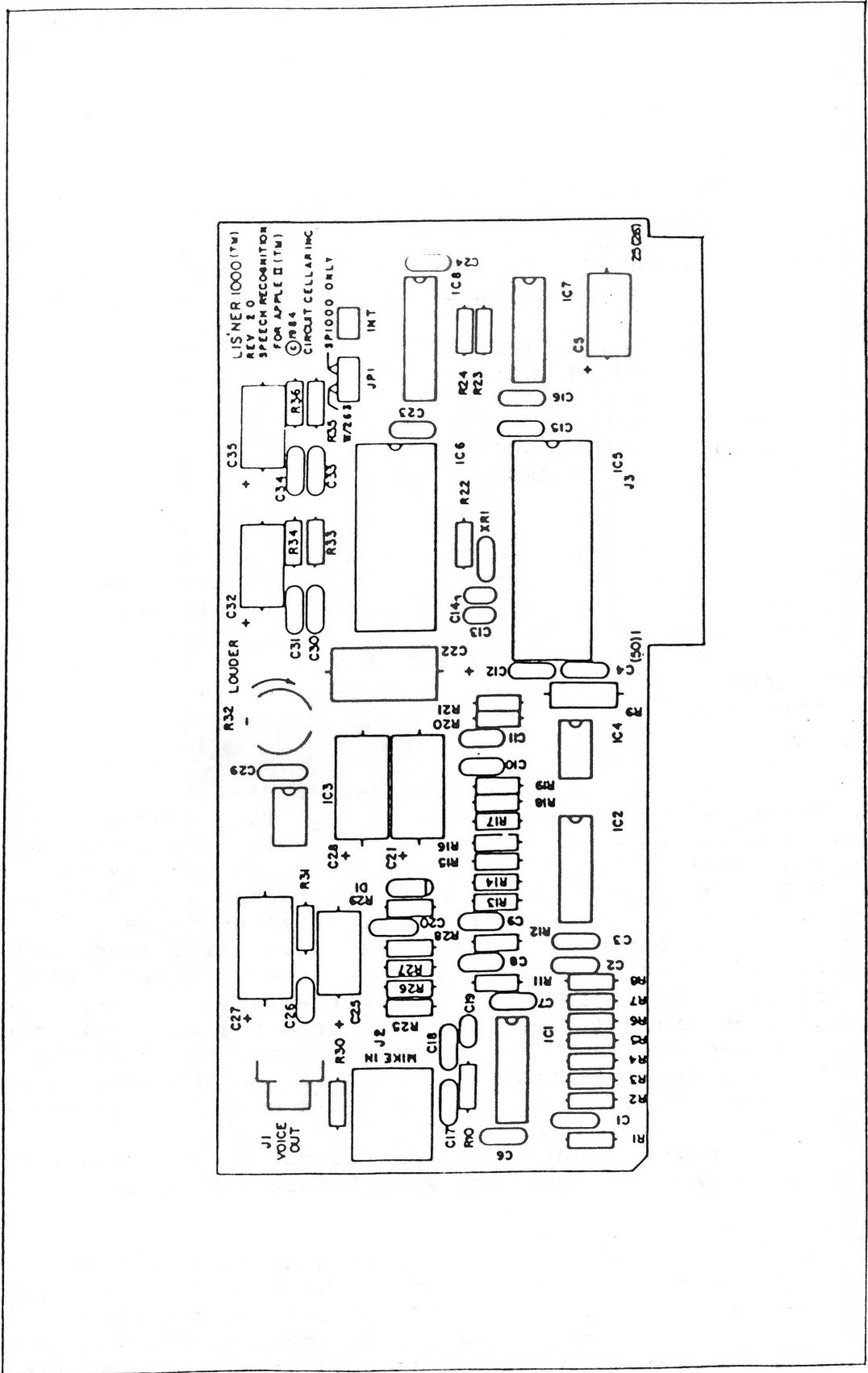
LIS'NER 1000 Silkscreen

Assembly Instructions for the LIS'NER 1000

1. ( )
        The LIS'NER 1000 kit produced by MICROMINT has been
given special attention to insure that all parts were
included before packaging. Use the check-off parts list to
become acquainted with each part. Once the parts have been
verified, you may want to gather the following tools:

                LOW WATTAGE SOLDERING IRON
                ROSIN CORE SOLDER
                SMALL WIRE CUTTERS
(optional)   NEEDLE NOSE PLIERS
(optional)   LEAD BENDING JIG
(optional)   I C INSERTION TOOL

2. ( )
        All resistor leads are spaced on .4 inch centers.
Notice on the silkscreen overlay that the resistors are
laid out in rows and the part numbers run in order. Insert
the resistors into the circuit board, bending each lead over
on the circuit side to prevent them from falling out. Clip
the leads off as you go and solder each on the circuit side
of the board.

3. ( )
        Monolithic capacitors have radial leads and can be
inserted without forming the leads. Take special care when
matching the monolithic caps with their placement on the
board, many values look identical. Do not mix the .1 ufd
with the 5% .1 ufd, the 5% caps are marked .S. followed by
the value in number number number of zeros format.
        Electrolytic capacitors have axial leads and must
be formed. Bend the leads so that the printed value and
polarity arrow can be seen from the top. The arrow points
toward the minus (-) end and away from the plus (+) end of
the capacitor. This polarity must be followed when
installing each electrolytic capacitor on the circuit board.
The plus (+) end of the electrolytic capacitor is marked on
the circuit board.
        Insert, bend, clip, and solder a few capacitors at
a time. Always recheck your work before moving on.

4. ( )
        All the IC sockets have a notch in one end. This
notch indicates the pin #1 end of the socket. Refer to the
silkscreen layout, all the IC outlines on the circuit board
have a notch to signify the pin #1 end. Match the notches
on the socket and the silkscreen outline when inserting the
sockets. Before soldering a socket, verify that every lead
protrudes through the circuit board.

5. ( )
        Insert the 1X3 pin header at JP1 and solder it.

6. ( )

Insert the 7.16 MHZ resonator at XR1, clip, and solder the leads. Insert the zener diode D1 making sure to match the striped end (cathode) with the silkscreen. Bend clip and solder the leads.

7. ( )

Insert J1 and J2 and solder all the leads.

8. ( )

Put down the soldering iron and check each component for; proper value, placement, polarity (if any), and soldering technique. Adequate inspection is a necessary part of building any kit.

9. ( )

Cleaning the soldering flux off of the board is a good idea, though not completely necessary. Rosin fluxes have a very low activity level. Some solvents (i.e. TRI-CLOR) may affect some plastics (i.e. mini phone jack) so keep them away from the top of the board.

10. ( )

Install SJ on JP1 toward the SP1000 ONLY marking.

11. ( )

Before inserting chips in your board, install the board in your APPLE without chips and run a program. If the computer runs OK then proceed with step 12, if not inspect your board for a shorted trace.

12. ( )

Remove the board from the computer. All IC's have a notch and/or a dot at one end of their body. This is the pin #1 end of the device. Remember pin #1 on the IC sockets? Just match pin #1 of the IC to pin #1 on the appropriate socket. Insert IC1-5.

13. ( )

(OPTIONAL SPEECH OUTPUT CHIP KIT) Insert IC6-8.
(Move SJ on JP1 toward the W/263 marking.)

14. ( )

Recheck all the IC's. Is each IC in the correct socket? Are all the legs seated in the socket? Now install the board back in your APPLE and follow the installation section of this manual. It is suggested that you run the LPC SYNTHESIS DEMONSTRATION first to give an indication that the SP 1000 is working and then run the COEFFICIENT DISPLAY to indicate that the amplifier section is working. A whistle in the microphone should increase the byte value of the 9th byte (energy byte) if all is OK. NOTE: This value never exceeds '00111111'.

APPENDIX A - DISKETTE PROGRAM LISTINGS

APPENDIX B - GENERAL INSTRUMENT SP1000 DATA

APPENDIX C - NATIONAL SEMICONDUCTOR ADC0831 DATA

APPENDIX D - LIS'NER 1000 SCHEMATIC DIAGRAM

PROGRAMS FOR THE OPTIONAL SSI-263 SYSTHESIZER

MICROMINT'S SSI-263 ALGORITHM

```
A 004 RULE EDITOR LOADER
A 003 SAMPLE 1
A 004 SAMPLE 2
A 003 SAMPLE 3 - REV 1.0
A 003 SAMPLE 3 - REV 1.2
A 004 TEXT TO SPEECH DEMO
B 003 COMPOSITE DRIVER
B 024 RULE EDITOR
B 009 TEXT TO SPEECH
B 002 MB$ GETTEXT
B 002 SSI-263 TTS DRIVER
B 006 SSI-263 TEXT TO SPEECH
B 005 SSI-263 INFLECTION
B 002 MKB:RULE.INDEX
B 002 MKB:RULE.LENGTH
B 039 MKB:RULE.TABLE
B 002 MKB:EMPT.INDEX
B 002 MKB:EMPT.LENGTH
B 003 MKB:EMPT.TABLE
A 010 DEMO
B 003 LADIES
B 003 INTRSONG
B 003 WARMUP
B 002 MRY
B 005 MICE
B 003 ROWROW
B 002 THANKS
B 002 CONT1
B 003 HELL1
B 002 ENDIT
B 002 CLOSING
B 004 AGENTMAN
B 003 AGENTTWO
B 005 SUZANNAH
```

NOTE: This disk is provided to those users who have
purchased the VR03,VR04, or VR05 upgrade kit.

10/29/84

LIS'NER 1000 PROGRAM DISK #1

LIS'NER DEVELOPMENT AND SUPPORT PROGRAMS

SIDE 1

A 005 LIS'NER 1000 MENU

A 079 LIS'NER 1000 TRAINER & EDITOR
A 016 THE GAME
A 009 LIS'NER LPC SPEECH DEMO
A 009 TEMPLATE LOADER
A 015 TEMPLATE LINKER
A 002 DISPLAY FRAME
T 002 SLOT

RECOGNITION PROGRAM FILES

B 022 RECOGNIZE.OBJ0
B 002 TRAIN.OBJ0
B 003 KDISPLAY.OBJ0
B 002 DOSWORDS.HEADER
B 006 DOSWORDS1
B 006 DOSWORDS2
B 006 DOSWORDS3
B 006 DOSWORDS4
T 002 DOSWORDS.WORDS

SP1000 SYNTHESIS PROGRAM FILES

T 019 SYNTHESIS.S
B 002 SYNTHESIS.OBJ
B 030 GI SPEECH PROMO
B 012 NUMBERS
B 007 READY
B 010 TIME1
B 008 TIME2
B 007 WASHER
A 007 SYNTH.B
B 002 SYNTH.OBJ

USER TEMPLATES

T 001 GAME.WORDS
B 001 MYGAME1
B 001 OPPONENT1

SIDE 2

A 002 HELLO

A 003 RECORD
A 004 DISPLAY
A 003 CONVERT
A 002 RECOG
T 017 DEMO
T 009 TRAIN
T 016 MATCH
T 005 AVG
T 004 TPCOPY
T 007 LISTEN
T 005 INITWORD
T 005 READFRM
T 037 HEAR
T 006 COMPNOIS
T 004 POINTENERGY
T 007 SHIFTDOWN
T 027 STMACH
T 010 HALF
T 022 PROCRUST
T 045 DTW
T 014 UTILS
T 016 VARS
B 022 DEMO.OBJ0

G I SYNTHESIS PROGRAM FILES

A 007 SYNTH
T 019 SYNTH.S
B 002 SYNTH.OBJ
T 010 PARKING
T 007 PLEASEREPEAT
T 005 ZERO
T 004 ONE
T 003 TWO
T 005 THREE
T 006 FOUR
T 004 FIVE
T 005 SIX
T 006 SEVEN
T 008 EIGHT
T 005 NINE
T 004 TEN
T 005 TESTREC

10/29/84

GENERAL INSTRUMENT	SP 1000

VOICE RECOGNITION/SYNTHESIS DEVICE

FEATURES

- N-channel silicon gate construction
- Single 5V supply
- On chip oscillator
- 0°C to +70°C operation
- User programable sample rate
- TTL compatible
- Standard microprocessor interface including 6809

RECOGNITION FEATURES

- Software controlled sampling frequency from 5.0 kHz – 15.9 kHz
- Automatic gain control outputs for control of external input amplifier
- 8 stage Linear Predictive Coding (LPC) lattice analyzer

SYNTHESIZER FEATURES

- 10 stage lattice synthesizer
- Software controlled sampling frequency from 4.0 kHz – 12.7 kHz
- Excitation look up ROM for voiced/unvoiced source
- 8 bit on chip digital to analog conversion with PWM digital output

PIN CONFIGURATION
28 LEAD DUAL IN LINE

Top View

Pin	Signal		Pin	Signal
1	V_{SS}		28	$\overline{IRQ}$
2	$\overline{WAIT}$		27	D7
3	A0		26	D6
4	A1		25	D5
5	R/$\overline{W}$		24	D4
6	$\overline{CS1}$		23	D3
7	STROBE		22	D2
8	XTAL IN		21	D1
9	XTAL OUT		20	D0
10	CLOCK IN/OUT		19	ADCDATA
11	DIGITAL OUT		18	$\overline{ADCCE}$
12	$\overline{RESET}$		17	ADCCLK
13	GAIN 6		16	GAIN 24
14	GAIN 12		15	V_{DD}

DESCRIPTION

In voice recognition applications, the SP 1000 forms the system front end by performing LPC feature extraction on an incoming audio signal. It must be supported by an 8 bit microcomputer which would compare the audio signal features with those of templates stored in memory, in order to make a recognition decision.

This results in a system design which leaves definition of key system features in the control of the system designer. Thus, this device can be used in speaker-dependent or speaker-independent systems, recognizing isolated words or connected speech.

The advantage of this approach is that the designer is not locked-in to a recognition algorithm which may or may not suit the design requirements. In addition, since the recognition algorithm is contained in software or firmware it can be easily upgraded to take advantage of advances in recognition techniques, without requiring hardware redesign.

ARCHITECTURAL DESCRIPTION

Processor Interface:
The SP 1000 Voice Recognition/Synthesis Device is designed to interface with a standard microprocessor bus, with data lines, address lines, chip select line, and read/write line. Eight bits of data may be read from, or written to the chip by

SP 1000 BLOCK DIAGRAM

AO A1 R/W   DO   D7

$\overline{CS1}$ →
STROBE →
INPUT CONTROL LOGIC

DATA INPUT/OUTPUT

SYNTHESIS EXCITATION

$\overline{IRQ}$ ←
$\overline{WAIT}$ ←
$\overline{RESET}$ →

XTAL IN →
TIMING
XTAL OUT ←
3.58 MHz ←

ENERGY

DATA STORAGE REGISTERS

RECONFIGURABLE LATTICE FILTER

$\overline{ADCCE}$ ←
ADCCLK ←
ADCDATA →
RECOGNITION INPUT CONVERTER

PULSE WIDTH MODULATOR → DIGITAL OUT

GAIN GAIN GAIN
6   12   24
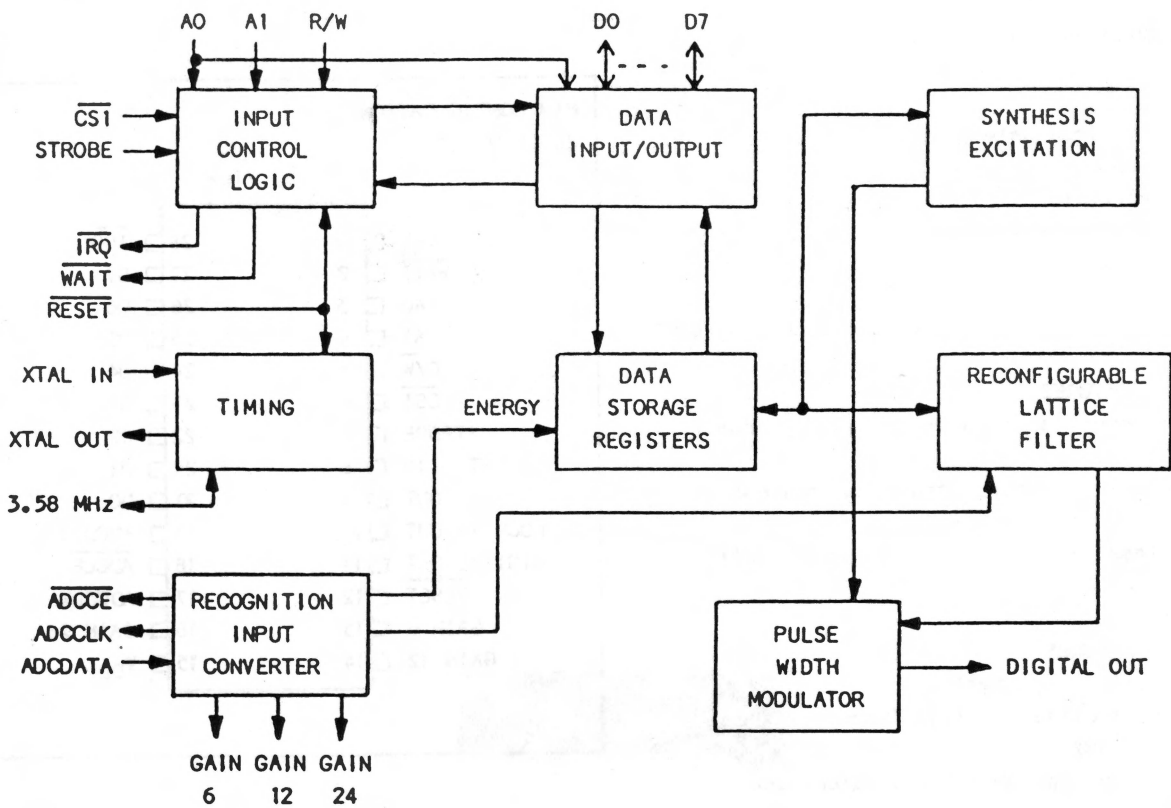
VOICE RECOGNITION/SYNTHESIS SYSTEM BLOCK DIAGRAM

RECOGNITION DATA STORAGE

7.16 MHz
22 pF   22 pF

RAM
ADDR
DATA

MICROPROCESSOR

XTAL OUT   XTAL IN   +5V
V_DD
$\overline{CS1}$ DIGITAL OUT
$R/\overline{W}$
$\overline{IRQ}$
$\overline{WAIT}$
$\overline{ADCCE}$
ADCCLK

LOW PASS FILTER → SPEAKER

SYNTHESIS DATA STORAGE

ROM
ADDR
DATA

STROBE
AO, A1   /2
DO–D7   /8
I/O

ADC DATA

A/D

ANALOG GAIN CONTROL

ANALOG INPUT (VOICE)

GAIN 24
GAIN 12
GAIN 6

$\overline{RESET}$

+5V
100K

V_SS

.1uF

DS5000 1B-2
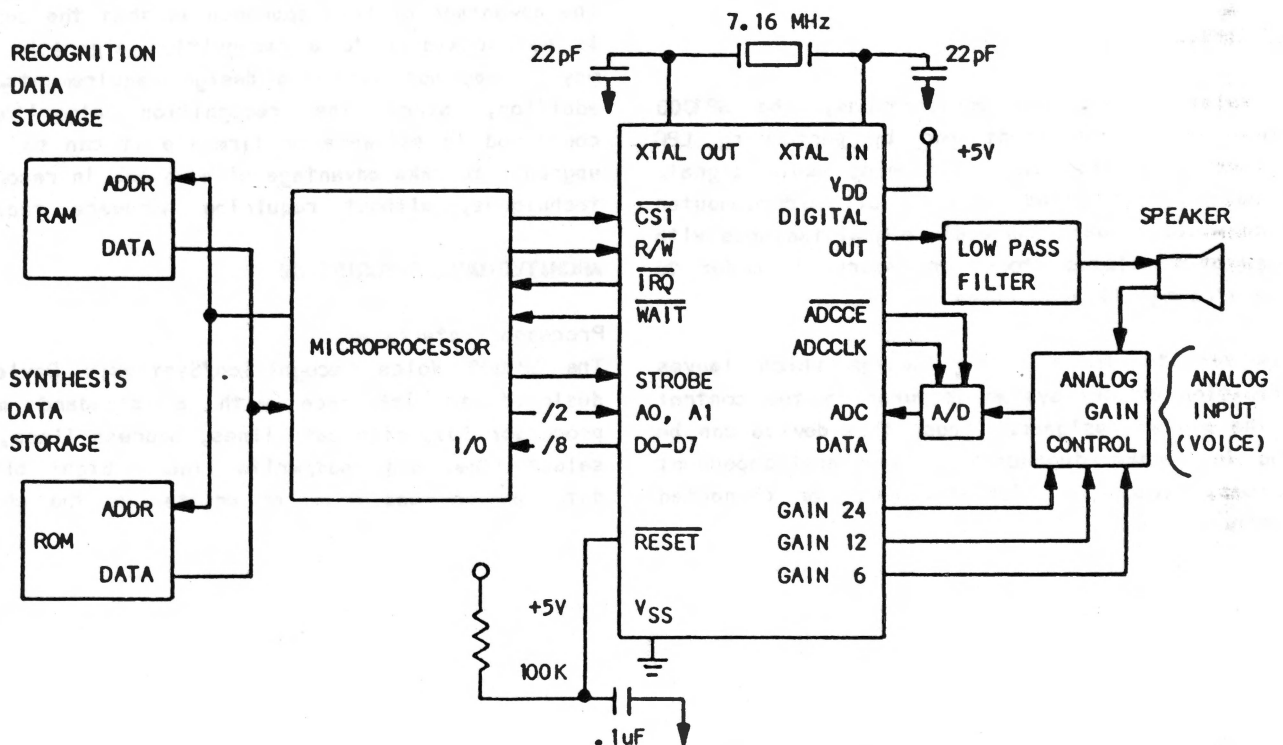
Table 1: OPERATION CODES

STROBE	$\overline{CS1}$	R/$\overline{W}$	A1	A0	Code	Operation
L	X	X	X	X		(Chip not selected)
X	H	X	X	X		
H	L	L	L	L	0	Write to Control Register
H	L	L	L	H	1	Write to Parameter Address Register
H	L	L	H	L	2	Write to Parameter Data Input Register, LSB = 0
H	L	L	H	H	3	Write to Parameter Data Input Register, LSB = 1
H	L	H	L	L	4	Read Status Register
H	L	H	L	H	5	(Not Implemented)
H	L	H	H	L	6	Read Parameter Data Output Register
H	L	H	H	H	7	Same as Code 6, plus Initiate Internal Fetch

the processor, following standard peripheral protocol. A nine-bit word may be written to the chip during a coefficient write. Data is transferred via the data lines whenever the chip select line is active. The read/write line determines the direction of the transfer, and the address lines specify a particular register within the chip as source or destination. Another way of looking at the address lines is that they define, together with the read/write line (R/$\overline{W}$), a particular operation to be performed. Table 1 presents the available defined operations. The code column will be used for reference, e.g. the operation of writing to the Parameter Address Register will be referred to as a Code 1 operation.

Before describing the operations in detail, the data organization inside the device will be reviewed. There is a sharp distinction between control/status information and parameter data. Parameter data inside the chip resides in two large recirculating shift registers. The recirculation period is the same as the sample period. Access to a particular parameter can occur only once each recirculation period, when that parameter is passing by a fixed point of access. The access is bit serial, and the processor cannot access parameter data directly. One of the functional blocks inside the chip is a special interface to overcome this problem.

The interface bears some analogy to a serial communication channel for simplex mode operation. Parameter data written by the processor is first stored in the Parameter Data Input Register. A BUSY flag is set to indicate that the register is full. The contents of the Parameter Data Input Register will be written to the destination specified by the Parameter Address Register as soon as that destination becomes accessible. Upon completion, the BUSY flag is reset to signal the processor that the Parameter Data Input Register is empty and ready for another data transfer.

For the processor to read a parameter value, the interface must first perform a fetch operation. The fetch operation is initiated by the processor. The BUSY flag is set to indicate fetch in progress. As soon as the parameter (specified by the Parameter Address Register) becomes accessible, the value (truncated to eight bits if necessary) is transferred to the Parameter Data Output Register, and the BUSY flag is reset. The processor can now read the parameter value from the Parameter Data Output Register.

The reading and writing of parameter data requires a finite amount of time. During that time the interface cannot be disturbed. The processor, therefore, must monitor the state of the BUSY flag in all dealings with the device. The BUSY flag is reflected in the STATUS Register.

GENERAL INSTRUMENT	SP 1000

PIN FUNCTIONS

Pin Number	Name	Function
1	V_{SS}	Ground
2	$\overline{WAIT}$	This output pin is normally high, but is taken low if the chip is selected for code 1, 2, 3, or 7 operation while the BUSY flag is set. It remains low until the BUSY flag is reset, or the chip is de-selected. This is an open drain output.
3	A_0	Least significant address line/ninth data line. (Least Significant Bit, input only.)
4	A_1	Most significant address line.
5	R/W	Read/Write. The state of input Pins 3, 4, and 5 determines the operation to be performed, according to Table 1.
6, 7	$\overline{CS1}$, STROBE	Chip select inputs: $\overline{CS1}$ active low, STROBE active high. Both must be active to select the chip.
8	XTAL IN	Crystal In. Input connection for a 7.16 MHz crystal.
9	XTAL OUT	Crystal Out. Output connection for a 7.16 MHz crystal.
10	CLOCK IN/OUT	3.58 MHz test Clock Input/buffered Clock Output. As an input, it allows the chip to be driven by an external clock (one-half the crystal frequency), in which case the crystal oscillator is disabled. As an output, it provides the buffered system clock which is one half of the crystal frequency.
11	DIGITAL OUT	Synthesized speech Digital Output. This is the output of the Pulse Width Modulator (PWM). The PWM, in conjunction with an external low pass filter, forms a digital to analog converter with eight bits of resolution.
12	$\overline{RESET}$	A logic 0 resets the chip. It must be returned to a logic 1 for normal operation.
13, 14, 16	GAIN 6, GAIN 12, GAIN 24	These 3 outputs can control 3 analog switches. The switches determine the gain of the external amplifier which precedes the analog to digital converter.
15	V_{DD}	Power supply pin.

PIN FUNCTIONS (continued)

Pin Number	Name	Function
17	ADCCLK	ADC Clock. This output is the 150 kHz gated clock. It is active only while $\overline{ADCCE}$ is active, otherwise it is held low.
18	$\overline{ADCCE}$	ADC Chip Enable. This output provides chip select for the analog to digital converter. It also controls the sample and hold circuit.
19	ADCDATA	ADC Data in. This input pin receives the serial data from the analog to digital converter.
20-27	D0-D7	Data lines. These 8 pins are the bi-directional data bus for connection to the processor. D7 carries the most significant bit.
28	$\overline{IRQ}$	Interrupt Request. This active-low output provides interrupt of the processor. This pin is an open drain output.

PARAMETER DESCRIPTION

Following is a short description of the parameters as listed in Table 2.

Synthesis

K_{10}-K_1:

These are the ten lattice filter coefficients for synthesis mode. Using the least significant address line as the ninth data line allows these to be specified with nine bits of resolution. K_{10} is the coefficient associated with the first filter section (closest to the excitation).

EXCTYP:

This eight-bit wide field is split in half to contain two parameters; Post Filter Gain and Excitation Type.

a) Post Filter Gain

The three least significant bits determine the post filter gain by determining which consecutive eight bits from the nominal sixteen-bit output of the filter will be sent to the digital to analog converter. A code of zero will choose the eight most significant bits (with proper sign extend) of the filter output. An increase of one will push the choice one bit to the right, such that a code of seven will choose the eight bits following the three most significant bits of the filter output.

Table 2: PARAMETER ADDRESS REGISTER

MSB							LSB
7	6	5	4	3	2	1	0
NOT USED		COEF BUF	NOT USED	INDEX			

Bits 0-3   Specify Parameter Index

Bit 5   Specifies Buffer (=1 for SR in recognition mode; otherwise = 0)

Index	Parameter	
	Synthesis	Recognition
0	K_{10}	K_{R1}
1	K_9	K_{R2}
2	K_8	K_{R3}
3	K_7	K_{R4}
4	K_6	K_{R5}
5	K_5	K_{R6}
6	K_4	K_{R7}
7	K_3	K_{R8}
8	K_2	ENRGY
9	K_1	T1
A	EXCTYP	T2
B	EXCAMP	SR
C	T1	-
D	T2	-
E	SR	-
F	-	-
B	-	SR

BIT 5 = 0 (applies to rows 0 through F)

BIT 5 = 1 (applies to row B: SR)

EXCAMP:

This field controls the amplitude of the selected excitation wave form. It is a gain factor specified as a base 2 logarithm, which is applied to each sample of the internally generated excitation. A special case exists when the excitation type is 1. The internally generated excitation is now a constant value. By writing a new value to the EXCAMP field every sample period, the constant value is modulated and externally generated excitation of the synthesizer is achieved. The gain factor is expressed in two parts: the most significant bit gives the sign of the gain factor (it is actually needed only for externally specified excitation); the seven least significant bits are the base 2 logarithm of its magnitude. These seven bits are interpreted as four exponent bits and three mantissa bits.

The reading of a particular, randomly selected parameter requires the following steps:

1. Load the Parameter Address Register by executing a write instruction, Operation Code 1. (See Table 1.)

2. Initiate a fetch operation (internal to the chip), by executing a read instruction with Operation Code 7. This will cause the BUSY flag (bit 7 of the STATUS Register) to become set. The fact that a read instruction is used will cause the current contents of the Parameter Data Output Register to be overwritten.

3. Wait for the BUSY flag to become reset, signaling the completion of the fetch sequence. The STATUS Register is read by a read instruction, Operation Code 4.

4. Read the desired parameter value by executing a read instruction, Operation Code 6.

Upon completion of a fetch (or store) operation, the Parameter Address Register is automatically updated to specify the next parameter in sequence. The sequence for synthesis is: K_{10}, K_9....K_1, EXCTYP, EXCAMP, T1, T2, SR. For recognition the sequence is K_{R1}, K_{R2}....K_{R8}, ENRGY, T1, T2, SR. This automatic update simplifies sequential reading of parameters. Sequential reading of parameters requires the following steps:

1. Load the address of the first parameter to be read, as in step 1, above.

2. Initiate the first fetch operation, as in step 2, above.

3. Wait for the BUSY flag to become reset, as in step 3, above.

4. Execute a read instruction with Operation Code 7. This will read the parameter value and also initiate the fetch of the next parameter in the sequence.

5. Go to step 3.

Synthesis and Recognition

T1 and T2:

These two identical fields control the two programable timers. The timers are actually sample time counters, and the fields contain the value to be counted down for each.

SR (Sample Rate):

The low order six bits of this field indirectly determine the Sample Rate by specifying how many clock cycles of dead time should be inserted between the processing of each filter stage. The formula for Sample Rate is:

$$f_s = f_c/[(28+n) * m] \quad \text{where}$$

f_s is Sample Rate.
f_c is system clock rate (3.579545 MHz).
n is the integer represented by the six low order bits of the SR field.
m is the number of filter stages, 10 for synthesis and 8 for recognition.

The two most significant bits of the field are used for a different parameter pertaining to recognition only.

Recognition

$K_{R1}-K_{R8}$:

These are eight bit coefficients produced by the adaptive lattice filter in recognition mode. Two different representations of these are available to the processor. Averaged values are read from the first circular buffer (BUF=0 in the Parameter Address Register) and instantaneous values may be read from the other buffer (BUF=1).

ENRGY:

This eight bit field contains information generated by the chip. It is a measure of the magnitude of incoming speech samples averaged over time, and is expressed on a logarithmic scale.

SR (two high order bits only)

This 2-bit field specifies the set point for gain control.

PROCESSOR BUS TIMING

	MIN	MAX
1. Address Set Up Time	70 ns	–
2. Address Hold Time	20 ns	–
3. Strobe Pulse Width	210 ns	–
4. Write Data Set Up	280 ns	–
5. Write Data Hold	30 ns	–
6. Read Data Access		220 ns
7. Read Data Hold	10 ns	
8. WAIT Access		140 ns
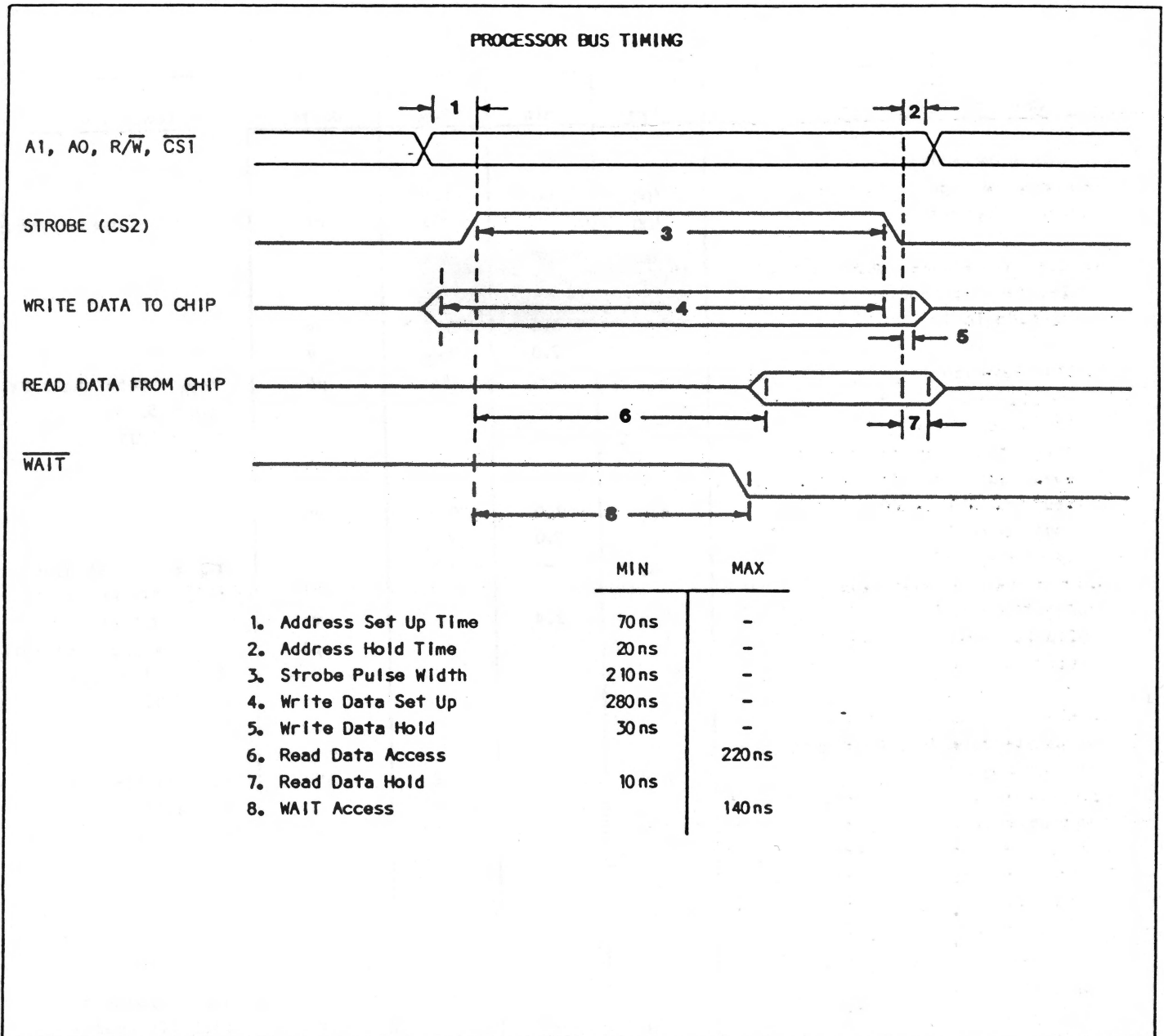
ELECTRICAL CHARACTERISTICS

Maximum Ratings*

V_{DD}·························· −0.3V to +8V
Storage Temperature·········· −25°C to +125°C

Clock

Crystal Frequency················ 7.15909 MHz
Standard Conditions (unless otherwise stated)
Clock In (Optional)············· 3.579545 MHz

DC CHARACTERISTICS

Operating Temperature T_A = 0°C to +70°C
Standard Conditions (unless otherwise stated)
V_{DD} = 5V ± 5%
V_{SS} = GND

*Exceeding these ratings could cause permanent damage to the device. This is a stress rating only and functional operation of this device at these conditions is not implied − operating ranges are specified in Standard Conditions. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Characteristics	Sym	Min	Max	Units	Conditions
DC Characteristics					
Supply Voltage	V_{DD}	4.5	5.5	V	
Supply Current	I_{DD}	−	100	mA	25°C, No output loads.
Inputs (ADCDATA, STROBE, $\overline{CS1}$, R/$\overline{W}$, A1, A0, $\overline{RESET}$)					
Input Logic 0	V_{IL}	0.0	0.8	V	
Input Logic 1	V_{IH}	2.0	V_{DD}	V	
Input Leakage	I_L	−	10	µA	V_{PIN} = 5.5V and 0.0V V_{DD} = 5.5V V_{SS} = 0.0V
Input/Outputs (D0−D7 CLOCK I/O)					
Input Logic 0	V_{IL}	0.0	0.8	V	
Input Logic 1	V_{IH}	2.0	V_{DD}	V	
Output Logic 0	V_{OL}	−	0.6	V	I_{OL} = 1.60 mA 50pf (4 LS TTL Loads)
Output Logic 1	V_{OH}	2.4	−	V	I_{OH} = 100 uA
Output Leakage	I_L	−	10	µA	V_{PIN} = 5.5V and 0.0V V_{DD}, CS1 = 5.5V V_{SS} = 0.0V
Open Drain Outputs (IRQ, MRDY)					
Output Logic 0	V_{OL}	−	0.6	V	I_{OL} = 1.60mA 50pf (4 LS TTL Loads)
Output Leakage	I_L	−	10	µA	V_{PIN} = 5.5V and 0.0V V_{DD} = 5.5V V_{SS} = 0.0V
Outputs (DIGITAL OUT, GAIN 6, 12, 24, ADCCLK, $\overline{ADCCE}$)					
Output Logic 0	V_{OL}	−	0.6	V	I_{OL} = 1.60mA 50pf (4LS TTL Loads)
Output Logic 1	V_{OH}	2.4	−	V	I_{OH} = −100 µA

SP 1000				GENERAL INSTRUMENT			

REGISTERS

CONTROL REGISTER:

MSB 7	6	5	4	3	2	1	LSB 0
NU	SYN/ REC	ENB T1	ENB T2	ADD/ REPL	TIMER 1 TIMER 2	TEST 2	TEST 1

Bit 7 - NU   Not Used

    6 - SYN/$\overline{REC}$ 1 = Synthesis, 0 = Recognition

    5 - ENB T1   Set to One to Enable Timer 1 Interrupt to IRQ pin 28

    4 - ENB T2   Set to One to Enable Timer 2 Interrupt to IRQ pin 28

    3 - ADD/$\overline{REPL}$ 1 = Add to, 0 = Replace Parameter Value in First Buffer ((BUF) = 0)

    2 - TIMER 1   1 = Timer 1, 0 = Timer 2, Causing Implementation of New Parameter Set

    1 - TEST INTERNAL DATA   1 = Test Mode. Internal data on Pin 11
                              0 = Normal Mode. D/A PWM on Pin 11

    0 - TEST EXTERNAL CLOCK   1 = Test Mode. Oscillator not used. Pin 10 is input. Use 3.58MHz
                              0 = Normal Mode. Pin 10 is output of 3.58MHz. lator is used.

STATUS REGISTER:

MSB 7	6	5	4	3	2	1	LSB 0
BUSY	S/R	T1	T2	COINC	NU	NU	NU

Bit 7 - BUSY   Processor Interface Busy

    6 - S/$\overline{R}$ High for Synthesis, Low for Recognition

    5 - T1 OUT   Timer 1: Time-Out Flag (Latched)* (See note 1)

    4 - T2 OUT   Timer 2: Time-Out Flag (Latched)* (See note 1)

    3 - COINC   Current Slot Number Equals Parameter Address Register Slot Number

    2 - NU   Not Used

    1 - NU   Not Used

    0 - NU   Not Used

Note 1:
*The timer flags are cleared by trailing edge of READ STATUS command.

PARAMETER ADDRESS REGISTER:

MSB 7	6	5	4	3	2	1	LSB 0
NOT USED		COEF BUF	NOT USED	INDEX			

Bits 0-3   Specify Parameter

Bit 5   Specifies Buffer

Bits 4, 6, 7   Not Used

![National Semiconductor logo]

June 1982

ADC0831, ADC0832, ADC0834 and ADC0838
(COP431, COP432, COP434 and COP438)
8-Bit Serial I/O A/D Converters with Multiplexer Options

General Description

The ADC0831 series are 8-bit successive approximation A/D converters with a serial I/O and configurable input multiplexers with up to 8 channels. The serial I/O is configured to comply with the NSC MICROWIRE™ serial data exchange standard for easy interface to the COPS™ family of processors, and can interface with standard shift registers or µPs.

The 2-, 4- or 8-channel multiplexers are software configured for single-ended or differential inputs as well as channel assignment.

The differential analog voltage input allows increasing the common-mode rejection and offsetting the analog zero input voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution.

Features

- NSC MICROWIRE compatible—direct interface to COPS family processors
- Easy interface to all microprocessors, or operates "stand-alone"

- Operates ratiometrically or with 5 V_{DC} voltage reference
- No zero or full-scale adjust required
- 2-, 4- or 8-channel multiplexer options with address logic
- Shunt regulator allows operation with high voltage supplies
- 0V to 5V input range with single 5V power supply
- Remote operation with serial digital data link
- T²L/MOS input/output compatible
- 0.3″ standard width 8-, 14- or 20-pin DIP package

Key Specifications

Resolution	8 Bits
Total Unadjusted Error	± 1/2 LSB and ± 1 LSB
Single Supply	5 V_{DC}
Low Power	15 mW
Conversion Time	32 µs

Typical Application

TRI-STATE® is a registered trademark of National Semiconductor Corp.
COPS™ and MICROWIRE™ are trademarks of National Semiconductor Corp.
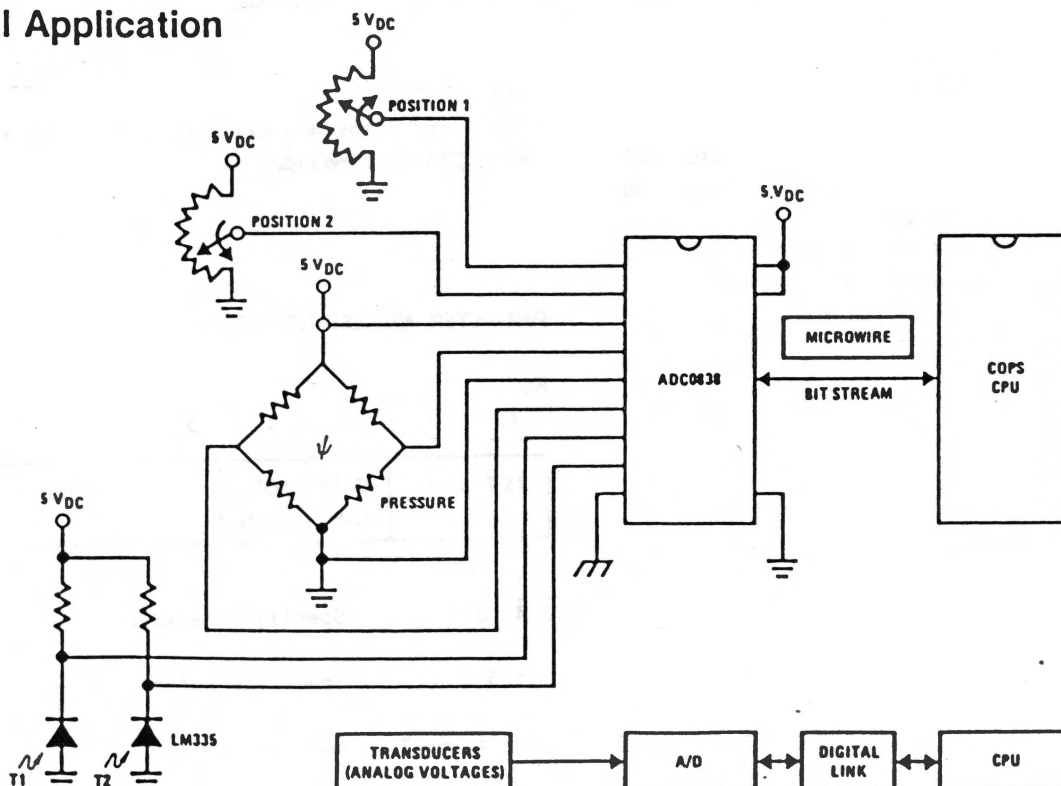
Absolute Maximum Ratings (Notes 1 and 2)

Current into V$^+$	10 mA
Supply Voltage, V_{CC} (Note 3)	6.5V
Voltage	
Logic Inputs	$-0.3V$ to $+15V$
Analog Inputs	$-0.3V$ to $V_{CC} + 0.3V$
Storage Temperature	$-65°C$ to $+150°C$
Package Dissipation at $T_A = 25°C$ (Board Mount)	0.8W
Lead Temperature (Soldering, 10 seconds)	300°C

Operating Ratings (Notes 1 and 2)

Supply Voltage, V_{CC}	4.5 V_{DC} to 6.3 V_{DC}
Temperature Range	$T_{MIN} \leq T_A \leq T_{MAX}$
ADC0831BD, ADC0832BD, ADC0834BD, ADC0838BD	$-55°C$ to $+125°C$
ADC0831BCD, ADC0831CCD, ADC0832BCD, ADC0832CCD, ADC0834BCD, ADC0834CCD, ADC0838BCD, ADC0838CCD	$-40°C$ to $+85°C$
ADC0831BCN, ADC0831CCN, ADC0832BCN, ADC0832CCN, ADC0834BCN, ADC0834CCN, ADC0838BCN, ADC0838CCN	0°C to $+70°C$

Converter and Multiplexer Electrical Characteristics

The following specifications apply for $V_{CC} = V^+ = 5V$, $T_{MIN} \leq T_A \leq T_{MAX}$ and $f_{CLK} = 250$ kHz unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
Total Unadjusted Error: (Note 4)					
ADC0831B, 32B, 34B, 38B	V_{REF} Forced to 5.000 V_{DC}			$\pm 1/2$	LSB
ADC0831C, 32C, 34C, 38C	V_{REF} Forced to 5.000 V_{DC}			± 1	LSB
Reference Input Resistance		1.9	2.4		kΩ
Common-Mode Input Range (Note 5)	All MUX Inputs and COM Input	GND -0.05		$V_{CC} + 0.05$	V
DC Common-Mode Error	Differential Mode		$\pm 1/16$		LSB
Power Supply Sensitivity	$V_{CC} = 5V \pm 5\%$		$\pm 1/16$		LSB
I_{OFF}, Off Channel Leakage Current (Note 6)	On Channel = 5V Off Channels = 0V $T_A = 25°C$	-1 -50			μA nA
	On Channel = 0V Off Channels = 5V $T_A = 25°C$			1 50	μA nA
I_{ON}, On Channel Leakage Current (Note 6)	On Channel = 0V Off Channels = 5V $T_A = 25°C$	-1 -200			μA nA
	On Channel = 5V Off Channels = 0V $T_A = 25°C$			1 200	μA nA

AC Electrical Characteristics

The following specifications apply for $V_{CC} = 5V$, $t_r = t_f = 20$ ns and $T_A = 25°C$ unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
f_{CLK}, Clock Frequency		10	250	400	kHz
Clock Duty Cycle		40		60	%
T_C, Conversion Time	Not Including MUX Addressing Time			8	$1/f_{CLK}$
t_{SETUP}, $\overline{SE}$ or $\overline{CS}$ Falling Edge or Data Input Valid to CLK Edge			100	250	ns

AC Electrical Characteristics (Continued)

The following specifications apply for $V_{CC} = 5V$, $t_r = t_f = 20$ ns and $T_A = 25°C$ unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
t_{HOLD}, Data Input Valid after CLK Rising Edge			35	90	ns
t_{CSPW}, Minimum $\overline{CS}$ High Interval			35	120	ns
t_{pd1}, t_{pd0}—CLK Falling Edge to Output Data Valid (Note 7)	$C_L = 100$ pF Data MSB First Data LSB First		650 250	1000 600	ns ns
t_{1H}, t_{0H}—Rising Edge of $\overline{CS}$ to Data Output and SARS Hi-Z	$C_L = 10$ pF, $R_L = 10k$ (See TRI-STATE® Test Circuits)		125	250	ns
C_{IN}, Capacitance of Logic Inputs			5		pF
C_{OUT}, Capacitance of Logic Outputs			5		pF

DC Electrical Characteristics

The following specifications apply for $V_{CC} = 5V$ and $T_{MIN} \leq T_A \leq T_{MAX}$ unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
$V_{IN(1)}$, Logical "1" Input Voltage	$V_{CC} = 5.25V$	2.0		15	V
$V_{IN(0)}$, Logical "0" Input Voltage	$V_{CC} = 4.75V$			0.8	V
$I_{IN(1)}$, Logical "1" Input Current	$V_{IN} = V_{CC}$		0.005	1	μA
$I_{IN(0)}$, Logical "0" Input Current	$V_{IN} = 0V$	−1	−0.005		μA
$V_{OUT(1)}$, Logical "1" Output Voltage	$I_{OUT} = -360$ μA, $V_{CC} = 4.75V$ $I_{OUT} = -10$ μA, $V_{CC} = 4.75V$	2.4 4.5			V V
$V_{OUT(0)}$, Logical "0" Output Voltage	$I_{OUT} = 1.6$ mA, $V_{CC} = 4.75V$			0.4	V
I_{OUT}, TRI-STATE Output Current (DO, SARS)	$V_{OUT} = 0.4V$, $T_A = 25°C$ $V_{OUT} = 5V$, $T_A = 25°C$		−0.1 0.1	−3 3	$μA_{DC}$ $μA_{DC}$
I_{SOURCE}	V_{OUT} Short to GND, $T_A = 25°C$		14		mA
I_{SINK}	V_{OUT} Short to V_{CC}, $T_A = 25°C$		16		mA
I_{CC}, Supply Current (Note 3)	ADC0831, ADC0834, ADC0838 ADC0832 (Includes Ladder Current)		1 3	2.5 5.2	mA mA

Note 1: Absolute Maximum Ratings are those values beyond which the life of the device may be impaired.

Note 2: All voltages are measured with respect to ground.

Note 3: An internal zener diode exists from V_{CC} to GND on the V^+ and V_{CC} inputs. The breakdown of these zeners is approximately 7V. The V^+ zener is intended to operate as a shunt regulator and connects to the V_{CC} via a diode. When using this regulator to power the A/D, this diode guarantees the V_{CC} input to be operating below the zener voltage (7V − 0.6V). It is recommended that a series resistor be used to limit the maximum current into the V^+ input.

Note 4: Total unadjusted error includes offset, full-scale, linearity, and multiplexer errors.

Note 5: For $V_{IN}(-) \geq V_{IN}(+)$ the digital output code will be 0000 0000. Two on-chip diodes are tied to each analog input (see Block Diagram) which will forward conduct for analog input voltages one diode drop below ground or one diode drop greater than the V_{CC} supply. Be careful, during testing at low V_{CC} levels (4.5V), as high level analog inputs (5V) can cause this input diode to conduct—especially at elevated temperatures, and cause errors for analog inputs near full-scale. The spec allows 50 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 50 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.950 V_{DC} over temperature variations, initial tolerance and loading.

Note 6: Leakage current is measured with the clock not switching.

Note 7: Since data, MSB first, is the output of the comparator used in the successive approximation loop, an additional delay is built in (see Block Diagram) to allow for comparator response time.

Timing Diagrams

ADC0831 Timing
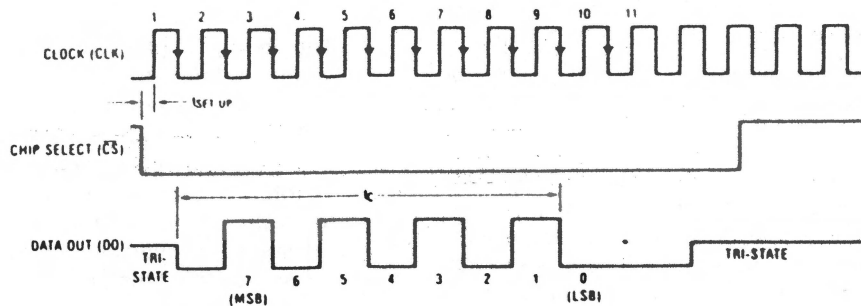
* LSB first output not available on ADC0831

ADC0831 Single Differential Input

Dual-In-Line Package

TOP VIEW
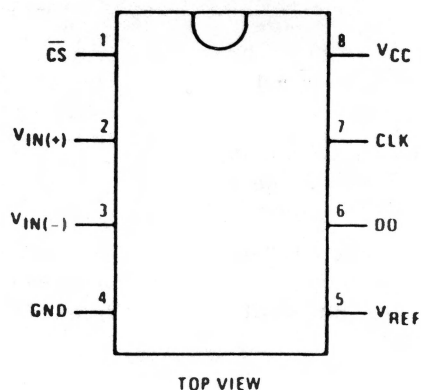
THE DIGITAL INTERFACE

A most important characteristic of these converters is their serial data link with the controlling processor. Using a serial communication format offers two very significant system improvements; it allows more function to be included in the converter package with no increase in package size and it can eliminate the transmission of low level analog signals by locating the converter right at the analog sensor; transmitting highly noise immune digital data back to the host processor.

To understand the operation of these converters it is best to refer to the Timing Diagrams and Functional Block Diagram and to follow a complete conversion sequence. For clarity a separate diagram is shown for each device.

1. A conversion is initiated by first pulling the $\overline{CS}$ (chip select) line low. This line must be held low for the entire conversion. The converter is now waiting for a start bit and its MUX assignment word.

2. A clock is then generated by the processor (if not provided continuously) and output to the A/D clock input.

3. On each rising edge of the clock the status of the data in (DI) line is clocked into the MUX address shift register. The start bit is the first logic "1" that appears on this line (all leading zeros are ignored). Following the start bit the converter expects the next 2 to 4 bits to be the MUX assignment word.

Applications (Continued)

4. When the start bit has been shifted into the start location of the MUX register, the input channel has been assigned and a conversion is about to begin. An interval of 1 full clock period (where nothing happens) is automatically inserted to allow the selected MUX channel to settle. The SAR status line goes high at this time to signal that a conversion is now in progress and the DI line is disabled (it no longer accepts data).

5. The data out (DO) line now comes out of TRI-STATE and provides a leading zero for this one clock period of MUX settling time.

6. When the conversion begins, the output of the SAR comparator, which indicates whether the analog input is greater than (high) or less than (low) each successive voltage from the internal resistor ladder, appears at the DO line on each falling edge of the clock. This data is the result of the conversion being shifted out (with the MSB coming first) and can be read by the processor immediately.

7. After 8 clock periods, the conversion is completed and the SAR status line returns low to indicate this.

8. If the programmer prefers, the data can be provided in an LSB first format [this makes use of the shift enable ($\overline{SE}$) control line]. All 8 bits of the result are stored in an output shift register. On devices which do not include the $\overline{SE}$ control line, the data, LSB first, is automatically shifted out the DO line, after the MSB first data stream. The DO line then goes low and stays low until $\overline{CS}$ is returned high. On the ADC0838 the $\overline{SE}$ line is brought out and if held high, the value of the LSB remains valid on the DO line. When $\overline{SE}$ is forced low, the data is then clocked out LSB first. The ADC0831 is an exception in that its data is only output in MSB first format.

9. All internal registers are cleared when the $\overline{CS}$ line is high. If another conversion is desired, $\overline{CS}$ must make a high to low transition followed by address information.

The DI and DO lines can be tied together and controlled through a bidirectional processor I/O bit with one wire. This is possible because the DI input is only "looked-at" during the MUX addressing interval while the DO line is still in a high impedance state.

All of the logic inputs can be taken to 15V independent the magnitude of the supply voltage, V_{CC}.

REFERENCE CONSIDERATIONS

The voltage applied to the reference input to these converters defines the voltage span of the analog input (the difference between $V_{IN(MAX)}$ and $V_{IN(MIN)}$) over which the 256 possible output codes apply. The devices can be used in either ratiometric applications or in systems requiring absolute accuracy. The reference pin must be connected to a voltage source capable of driving the reference input resistance of typically 2.4 kΩ. This pin is the top of a resistor divider string used for the successive approximation conversion.

In a ratiometric system, the analog input voltage is proportional to the voltage used for the A/D reference. This voltage is typically the system power supply, so the V_{REF} pin can be tied to V_{CC} (done internally on the ADC0832). This technique relaxes the stability requirements of the system reference as the analog input and A/D reference move together maintaining the same output code for a given input condition.

For absolute accuracy, where the analog input varies between very specific voltage limits, the reference pin can be biased with a time and temperature stable voltage source. The LM385 and LM336 reference diodes are good low current devices to use with these converters.

The maximum value of the reference is limited to the V_{CC} supply voltage. The minimum value, however, can be quite small (see Typical Performance Characteristics) to allow direct conversions of transducer outputs providing less than a 5V output span. Particular care must be taken with regard to noise pickup, circuit layout and system error voltage sources when operating with a reduced span due to the increased sensitivity of the converter (1 LSB equals $V_{REF}/256$).

Span Adjust: $0V \leq V_{IN} \leq 3V$
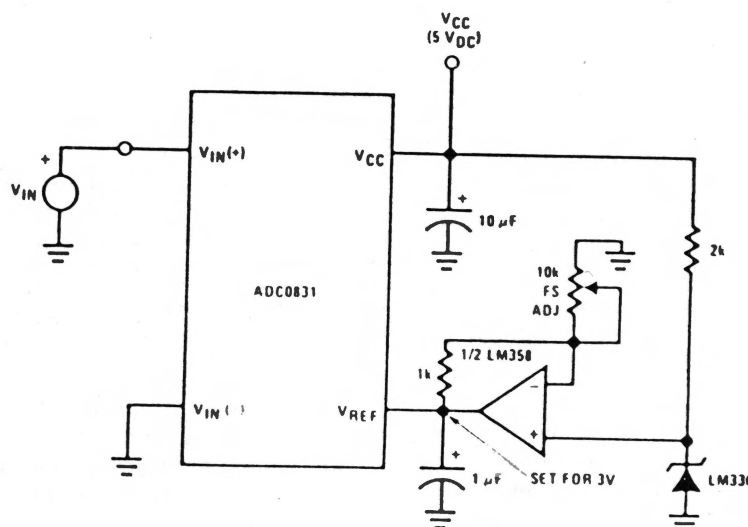
Applications (Continued)

THE ANALOG INPUTS

The most important feature of these converters is that they can be located right at the analog signal source and through just a few wires can communicate with a controlling processor with a highly noise immune serial bit stream. This in itself greatly minimizes circuitry to maintain analog signal accuracy which otherwise is most susceptible to noise pickup. However, a few words are in order with regard to the analog inputs should the input be noisy to begin with or possibly riding on a large common-mode voltage.

The differential input of these converters actually reduces the effects of common-mode input noise, a signal common to both selected "+" and "−" inputs for a conversion (60 Hz is most typical). The time interval between sampling the "+" input and then the "−" input is 1/2 of a clock period. The change in the common-mode voltage during this short time interval can cause conversion errors. For a sinusoidal common-mode signal this error is:

$$V_{error}(max) = V_{peak} (2\pi f_{CM}) \left(\frac{0.5}{f_{CLK}} \right)$$

where f_{CM} is the frequency of the common-mode signal,
V_{peak} is its peak voltage value
and f_{CLK} is the A/D clock frequency.

For a 60 Hz common-mode signal to generate a 1/4 LSB error (≈5 mV) with the converter running at 250 kHz, its peak value would have to be 6.63V which would be larger than allowed as it exceeds the maximum analog input limits.

Due to the sampling nature of the analog inputs short spikes of current enter the "+" input and exit the "−" input at the clock edges during the actual conversion. These currents decay rapidly and do not cause errors as the internal comparator is strobed at the end of a clock period. Bypass capacitors at the inputs will average these currents and cause an effective DC current to flow through the output resistance of the analog signal source. Bypass capacitors should not be used if the source resistance is greater than 1 kΩ.

This source resistance limitation is important with regard to the DC leakage currents of input multiplexer as well. The worst-case leakage current of ±1 μA over temperature will create a 1 mV input error with a 1 kΩ source resistance. An op amp RC active low pass filter can provide both impedance buffering and noise filtering should a high impedance signal source be required.

OPTIONAL ADJUSTMENTS

Zero Error

The zero of the A/D does not require adjustment. If the minimum analog input voltage value, $V_{IN(MIN)}$, is not ground, a zero offset can be done. The converter can be made to output 0000 0000 digital code for this minimum input voltage by biasing any $V_{IN}(−)$ input at this $V_{IN(MIN)}$ value. This utilizes the differential mode operation of the A/D.

The zero error of the A/D converter relates to the location of the first riser of the transfer function and can be measured by grounding the $V_{IN}(−)$ input and applying a small magnitude positive voltage to the $V_{IN}(+)$ input. Zero error is the difference between the actual DC input voltage which is necessary to just cause an output digital code transition from 0000 0000 to 0000 0001 and the ideal 1/2 LSB value (1/2 LSB = 9.8 mV for V_{REF} = 5.000 V_{DC}).

Full-Scale

The full-scale adjustment can be made by applying a differential input voltage which is 1 1/2 LSB down from the desired analog full-scale voltage range and then adjusting the magnitude of the V_{REF} input or V_{CC} for a digital output code which is just changing from 1111 1110 to 1111 1111.

Adjusting for an Arbitrary Analog Input Voltage Range

If the analog zero voltage of the A/D is shifted away from ground (for example, to accommodate an analog input signal which does not go to ground), this new zero reference should be properly adjusted first. A $V_{IN}(+)$ voltage which equals this desired zero reference plus 1/2 LSB (where the LSB is calculated for the desired analog span, 1 LSB = analog span/256) is applied to selected "+" input and the zero reference voltage at the corresponding "−" input should then be adjusted to just obtain the 00_{HEX} to 01_{HEX} code transition.

The full-scale adjustment should be made [with the proper $V_{IN}(−)$ voltage applied] by forcing a voltage to the $V_{IN}(+)$ input which is given by:

$$V_{IN}(+) \text{ fs adj} = V_{MAX} - 1.5 \left[\frac{(V_{MAX} - V_{MIN})}{256} \right]$$

where:

V_{MAX} = the high end of the analog input range

and

V_{MIN} = the low end (the offset zero) of the analog range. (Both are ground referenced.)

The V_{REF} (or V_{CC}) voltage is then adjusted to provide a code change from FE_{HEX} to FF_{HEX}. This completes the adjustment procedure.

POWER SUPPLY

A unique feature of the ADC0838 and ADC0834 is the inclusion of a 7V zener diode connected from the V$^+$ terminal to ground which also connects to the V_{CC} terminal (which is the actual converter supply) through a silicon diode, as shown in *Figure 3*.
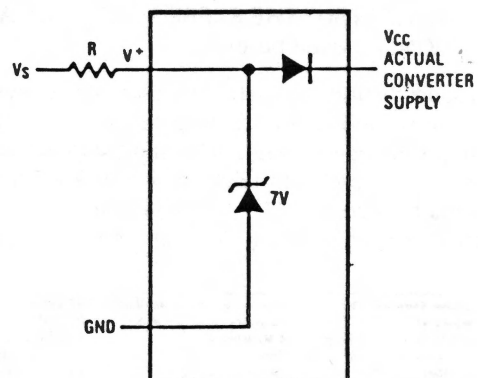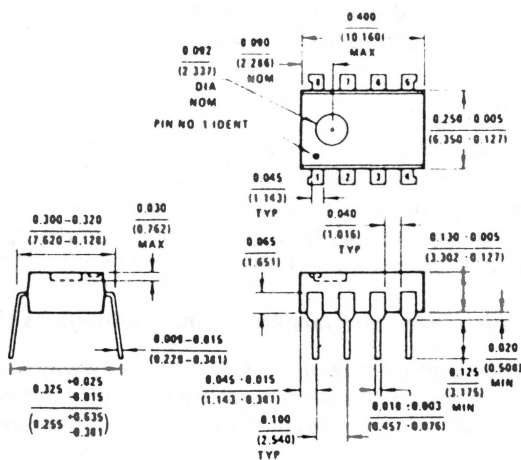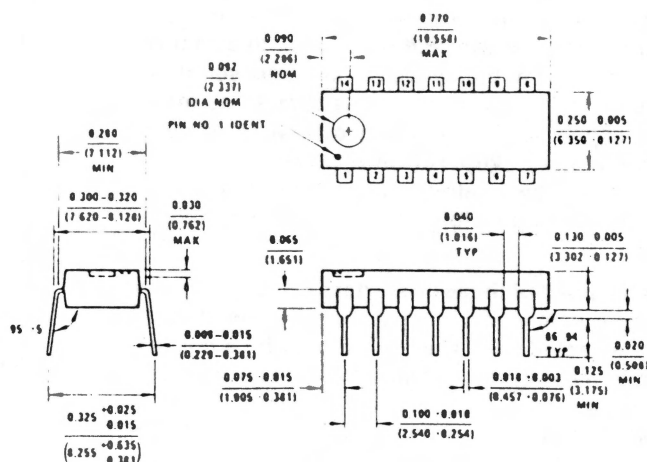

FIGURE 3. An On-Chip Shunt Regulator Diode

Physical Dimensions (Continued) inches (millimeters)

Molded Dual-In-Line Package (N)
NS Package Number N08A

Molded Dual-In-Line Package (N)
NS Package Number N14A

Applications (Continued)

This zener is intended for use as a shunt voltage regulator to eliminate the need for any additional regulating components. This is most desirable if the converter is to be remotely located from the system power source.

An important use of the interconnecting diode between V^+ and V_{CC} Here, this diode is used as a rectifier to allow the V_{CC} supply for the converter

to be derived from the clock. The low current requirements of the A/D (~3 mA) and the relatively high clock frequencies used (typically in the range of 10k–400 kHz) allows using the small value filter capacitor shown to keep the ripple on the V_{CC} line to well under 1/4 of an LSB. The shunt zener regulator can also be used in this mode. This requires a clock voltage swing which is in excess of 7V. A current limit for the zener is needed, either built into the clock generator or a resistor can be used from the CLK pin to the V^+ pin.

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right, at any time without notice, to change said circuitry
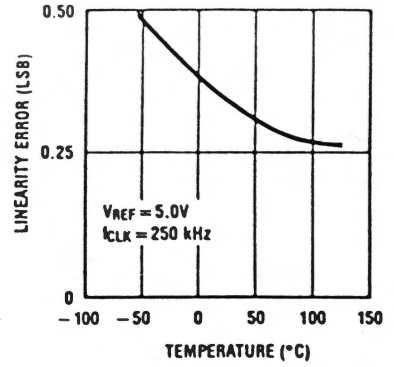
Typical Performance Characteristics

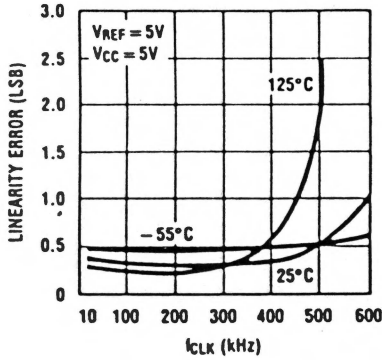### Unadjusted Offset Error vs V_REF Voltage
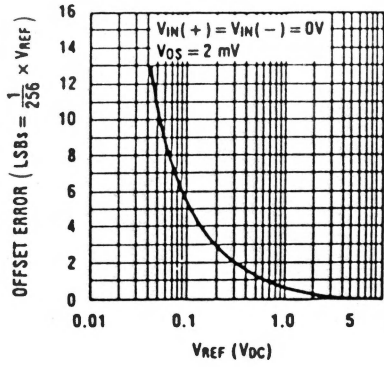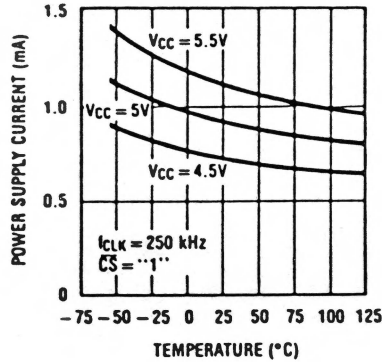

Linearity Error vs V_REF Voltage
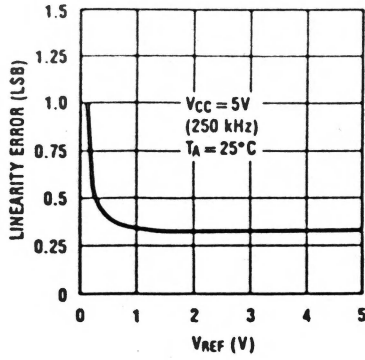
Linearity Error vs Temperature

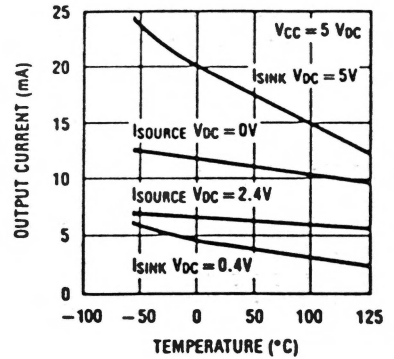### Linearity Error vs f_CLK

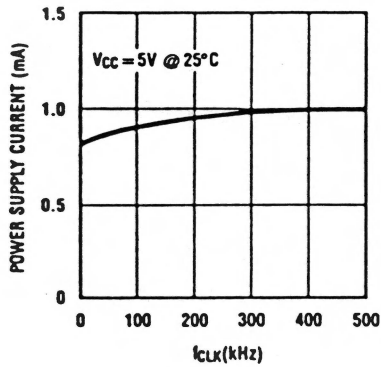### Power Supply Current vs Temperature (ADC0838, ADC0831, ADC0834)
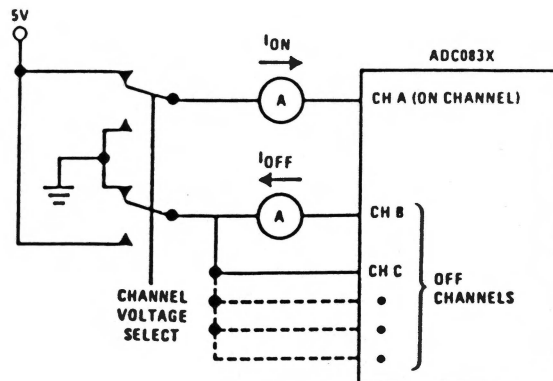

Note: For ADC0832 add I_REF

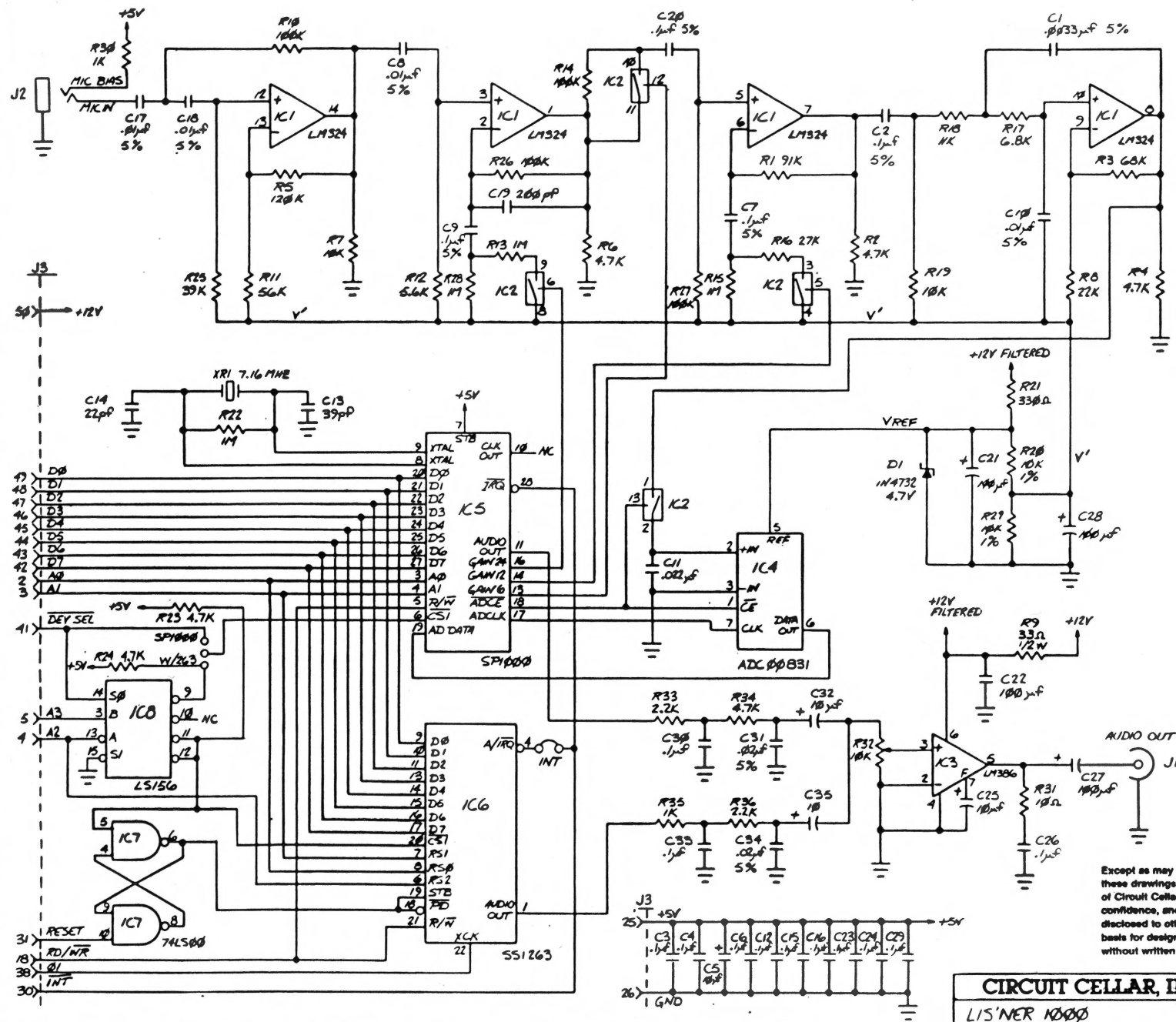### Output Current vs Temperature

Power Supply Current vs f_CLK

Leakage Current Test Circuit

LIS'NER 1000 CIRCUIT BOARD SCHEMATIC
REVISION 2.0   for the APPLE II tm

REF DES	DEVICE	+5V	GND
IC1	LM324	4	11
IC2	CD4016	14	7
IC3	LM386	—	2,4
IC4	ADC0831	8	4
IC5	SP1000	15	1
IC6	SSI263	13,24	2,12
IC7	74LS00	16	8
IC8	LS156	14	7

Except as may be otherwise provided by contract, these drawings and specifications are the property of Circuit Cellar, Inc., and are issued in strict confidence, and shall not be reproduced, copied, disclosed to others in whole or in part, or used as a basis for design, manufacture or sale of apparatus without written permission from Circuit Cellar, Inc.

CIRCUIT CELLAR, INC.

LIS'NER 1000

REV. 2.0