Lis´ner1000

THE LIS'NER 1000

BY STEVE CIARCIA

Build a low-cost, high-performance speech-recognition system

The concept of a computer understanding speech is not new. For years we have watched Capt. Kirk and Mr. Spock on the bridge of the *Enterprise* talking with the ship's computer or have remarked at the diabolical mind of HAL in *2001: A Space Odyssey*. These computers represent the ultimate in automatic speech recognition (ASR). Unfortunately, most of their capabilities are still science fiction.

The ultimate goal of all speech-recognition techniques is to characterize the spoken word into a recognizable pattern. Specifically, ASR is the ability that would let a computer recognize the spoken word. Exactly how the words are spoken, however, determines the hardware cost and analysis techniques employed.

SPEECH-RECOGNITION UNITS

The first type of unit is the *speaker-dependent* recognition system, which creates its recognition vocabulary by "listening" to the voice of a single speaker. It then concerns itself only with recognizing the same word as spoken by that speaker.

First, the user speaks into a microphone all the words the machine is to recognize. The acoustical characteristics of each word are analyzed and stored as templates, which are digital patterns used by a recognition algorithm to identify words. The procedure of creating templates is referred to as *training*. Depending upon the available memory and the recognition-algorithm speed, the total vocabulary can be from 4 to 100 words. Generally speaking, the more words in the vocabulary, the longer it takes to recognize a specific word and the more sophisticated the algorithm must be.

The second type of unit is the *speaker-independent* recognition system. Other than HAL or the *Enterprise* computer, few functioning speaker-independent systems exist that have more than a 10-word vocabulary. This system requires no template training by a single speaker. Its speech templates are preprogrammed, and the matching algorithm is supposed to be adaptable to the voices of a variety of speakers and accents.

The third type of unit is *unconnected speech*. Also called discrete-utterance recognition, unconnected speech is simply single words preceded and followed by pauses. This is

Steve Ciarcia (pronounced "see-ARE-see-ah") is an electronics engineer and computer consultant with experience in process control, digital design, nuclear instrumentation, and product development. He is the author of several books about electronics. You can write to him at POB 582, *Glastonbury,* CT 06033.

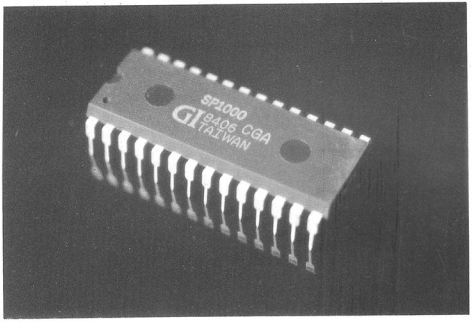


Photo 1: *The General Instrument SP1000 voice-recognition chip.*

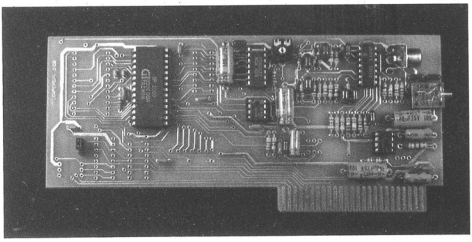


Photo 2: *A prototype printed-circuit board of the Apple II recognition-only Lis'ner 1000 circuit. The two connectors on the top right are for an external speaker (RCA) and the microphone (miniphono). An IBM PC version is in the works.*

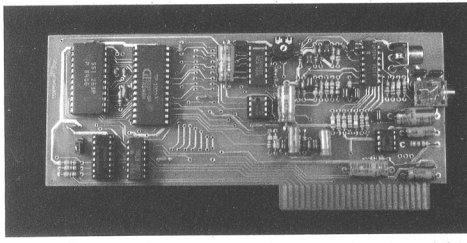


Photo 3: *A prototype printed-circuit board containing Lis'ner 1000 circuitry, which performs recognition, and speech-synthesis circuitry for the Apple II. It contains both the SP1000 and an SSI-263 speech synthesizer with a text-to-speech algorithm. Together they facilitate a functional hands-off computer with complete speech I/O.*

the easiest recognition approach and generally the technique used in most inexpensive systems. Each template is for a single utterance that must be spoken as a discrete word rather than as part of a longer word or phrase.

The final type of unit is *connected speech*. Also called continuous speech, this is the way we normally talk. Unfortunately, much of our understanding is dependent on recognizing the words in context. One major problem for the computer is coarticulation, where words are blended so that there are no distinct word boundaries for direct template matching. The result is costly computing overhead because every template must be aligned with every possible interval of the utterance. While significant advances have been made in this area, connected-speech ASR systems are expensive and generally require some monitoring of context as well.

Today, most speech-recognition systems are discrete-utterance speaker-dependent units. They may be in the form of expansion boards for existing computers or stand-alone black boxes. Ultimately, however, their purpose is singular. When the user speaks, the computer analyzes the acoustical signal, compares it to the stored templates, and decides which most closely resembles the spoken word. Once a candidate is chosen, the computer can itself 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. 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 frequencies, 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 linear predictive coding (LPC). Also known as autoregressive analysis, 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 that approximate a series of resonant cavities, thus simulating the vocal tract.

A Circuit Cellar System

Up until now, it hasn't seemed worthwhile to present a speech-recognition system that merely imitated others. My article in the March 1982 BYTE ("Use Voiceprints to Analyze Speech," page 50) demonstrated the separated-filter and energy-level recording technique in the hopes that I could learn enough to quickly present an ASR system based on that principle. While feasible in theory, I ultimately scrapped the idea as having too many components, even if they were readily available. Since then, I have been watching for any new components that might improve the situation.

Fortunately, the wait has not been in vain. The new SP1000 voice-recognition chip from General Instrument allows me to demonstrate the construction of a low-cost, high-performance voice-recognition system (see photo 1). To my knowledge, this project is one of the first recognition devices using the SP1000.

The Circuit Cellar speech-recognition system, which I've called the Lis'ner 1000, is both a voice-recognition and voice-synthesizer board using the SP1000. The schematics I present are specifically for the Apple II, but they are applicable to other 6502-based systems such as the Commodore 64. The Apple II version plugs into any of the computer's expansion slots, but slot 4 is preferred. The Commodore 64 version (shown in the opening photo) plugs into the rear expansion connector. The Commodore board is configured for recognition only; the Apple II board, shown in photos 2 and 3, supports the LPC speech output from the SP1000 and has optional provision for an SSI-263 phonetic speech synthesizer with a text-to-speech algorithm.

The Lis'ner 1000 hardware forms merely the front end of a recognition system by performing feature extraction of the incoming audio signal. The host microcomputer 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. You can use the Lis'ner 1000 board in speaker-dependent or speaker-independent systems with connected or unconnected speech. The designer is not locked into a specific recognition algorithm that may not be suitable for a particular application. 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.

This project is one of the first recognition devices using the SP1000.

In an effort to more fully support

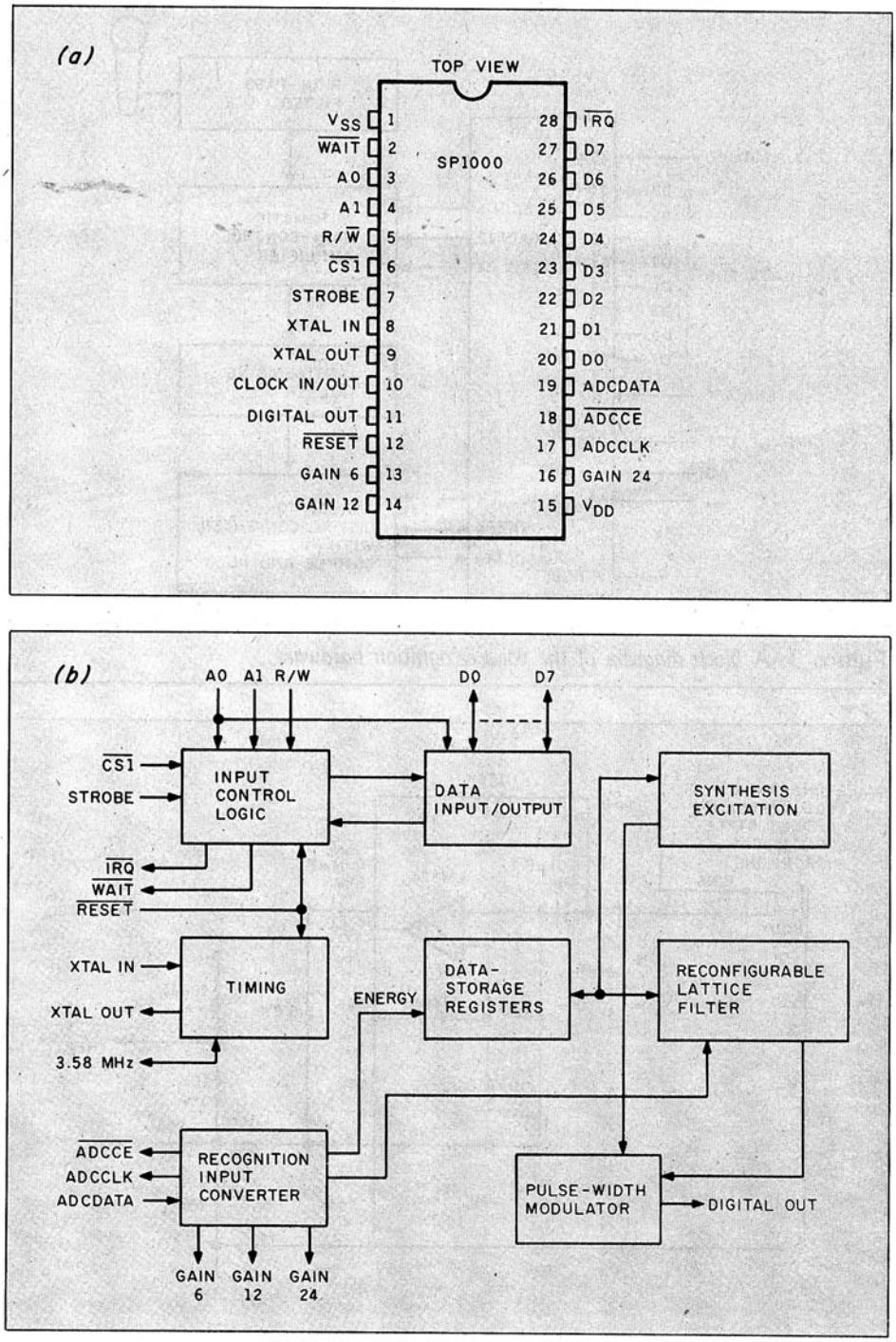


Figure 1: *The* SP1000 *pin configuration (a) and block diagram (b).*

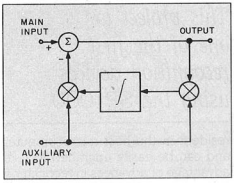


Figure 2: A CCL *loop*.

the Lis'ner 1000 and make it immediately usable. I've had developed a package of software routines that allows the board to function as a voice-operated keyboard on the Apple II and Commodore 64. Not to lose touch with true experimenters, however, the source code necessary to make the basic system function will be available to those who build the project and want to modify the software (see Experimenter Support on page 123).

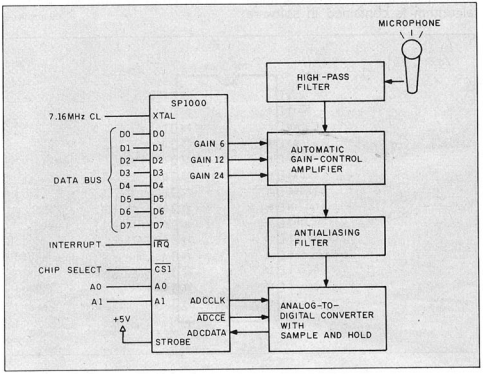


Figure 3: A *block diagram of the voice-recognition hardware.*

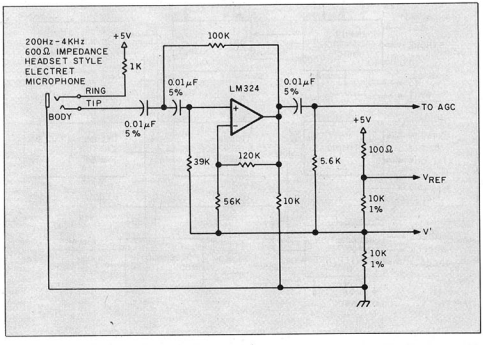


Figure 4: *The high-pass filter for the microphone.*

The software I provide makes the Lis'ner 1000 a speaker-dependent, discrete-utterance recognition system. The present software supports 64 words in two groups of 32. You'll need a disk drive for either unit to function with the present software.

Before I get too far ahead, however, let me describe the SP1000 chip itself and what's necessary to build the Lis'ner 1000.

General Instrument SP1000

The SP1000, block-diagramed in figure 1, is a 5-volt (V) 28-pin NMOS (negative-channel metal-oxide semiconductor) microprocessor peripheral chip that can be used for both speech recognition and LPC speech synthesis. Using a bidirectional data bus and control lines, the SP1000 interfaces to most 8-bit processors as a memory-mapped peripheral device.

The unique aspect of the SP1000 is its ability to do LPC analysis in real time. LPC analysis solves for the coefficients Ai in an equation of the form:

$$X'_K = A_1X_{K-1} + A_2X_{K-2} + \ldots + A_NX_{K-N}$$

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 vast amounts of memory and extensive calculations, preclude their use on an inexpensive device at this time.

The SP1000 uses a modified form of the correlation cancellation loop (CCL) shown in figure 2 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 (Ai) 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 3 kilobits for a standard covariance or autocorrelation analysis. It also has the interesting property of being able to run backward 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 (analog-to-digital) 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 2 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, General Instrument intends to supply an allophone set that will let the user 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, which can be added when it is available.

The desire for user-programmable voice-output capability immediately did not go unnoticed, however. I anticipated the interest in a functional recognition/synthesizer board and purposely designed the Lis'ner 1000 to perform as one. While the project described is for an SP1000-only device, the Apple II printed-circuit board for this project is also etched to accommodate an SSI-263 phonetic speech-synthesizer chip (see "Build a Third-Generation Phonetic Speech Synthesizer," March, page 28). Adding the SSI-263 and the text-to-speech algorithm facilitates true voice I/O (input/output) and supports both phonetic-generated and allophone-generated (LPC) speech.

Building the Lis'ner 1000

Figure 3 is a block diagram of the recognition portion of the Lis'ner 1000, which interfaces to the Apple II and Commodore 64 through an 8-bit bidirectional bus and a few control lines. The SP1000 occupies four address locations and is written to or read from as any other peripheral device 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 two address lines specify the particular register within the chip. Of the four registers, three are read/write and one is write only.

The source code that is necessary to make the basic system function will be available.

A typical system consists 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 V and, except for the SP1000 connection to the host computer, is virtually the same for all applications.

The first section (see figure 4) contains the microphone input and high-pass filter. For best performance, you should use a 600-ohm-impedance, condenser-type electret microphone.

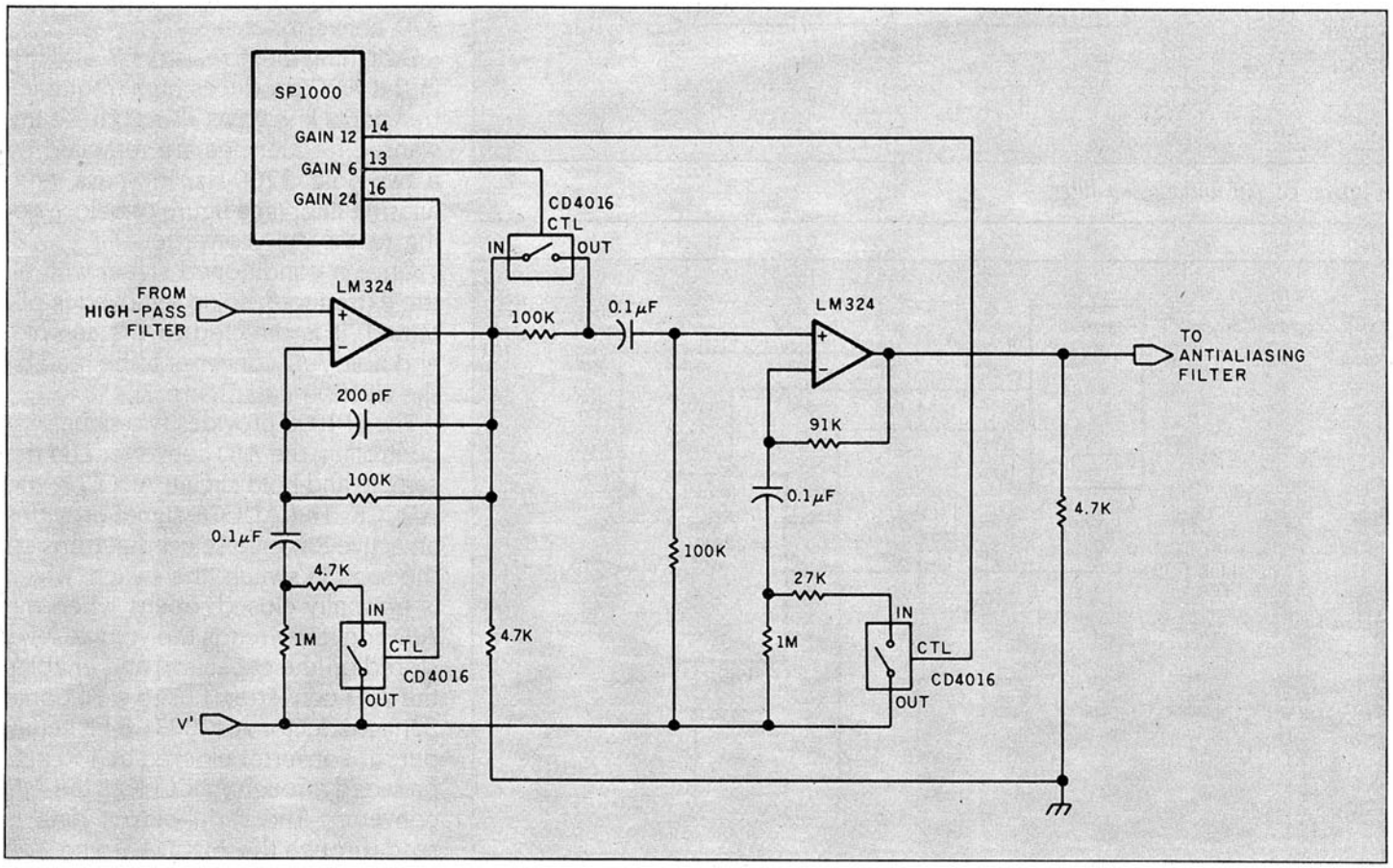


Figure 5: A *programmable automatic-gain-controlled amplifier.*

Table 1: *Signal amplifications possible with different combinations of the* SP1000 *gain pins, which are shown in figure* 5.

Gain Pins			
24	12	6	Gain
0	0	0	0 dB
0	0	1	6 dB
0	1	0	12 dB
0	1	1	18 dB
1	0	0	24 dB
1	0	1	30 dB
1	1	0	36 dB
1	1	1	42 dB

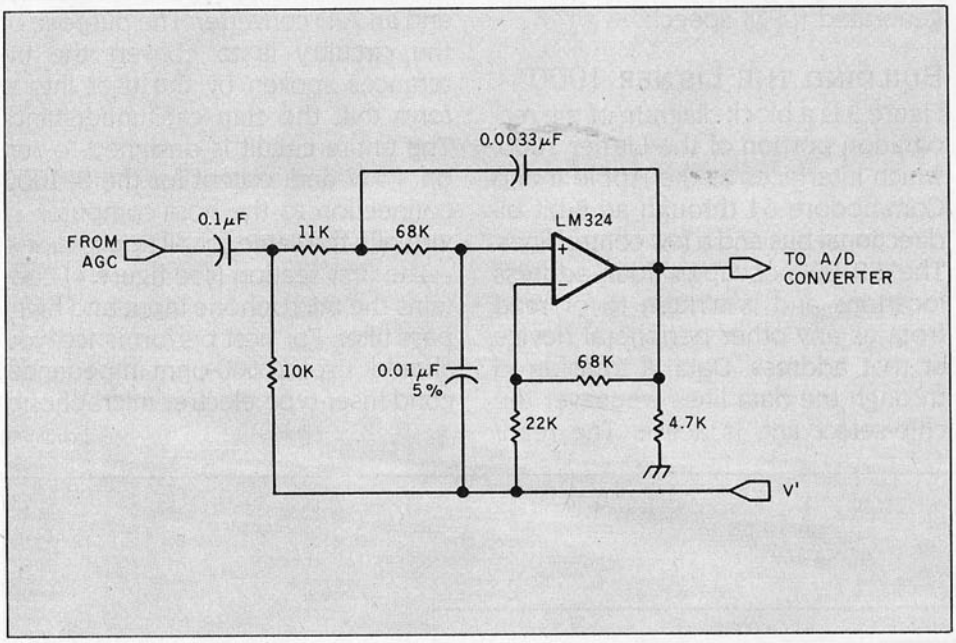


Figure 6: An *antialiasing filter.*

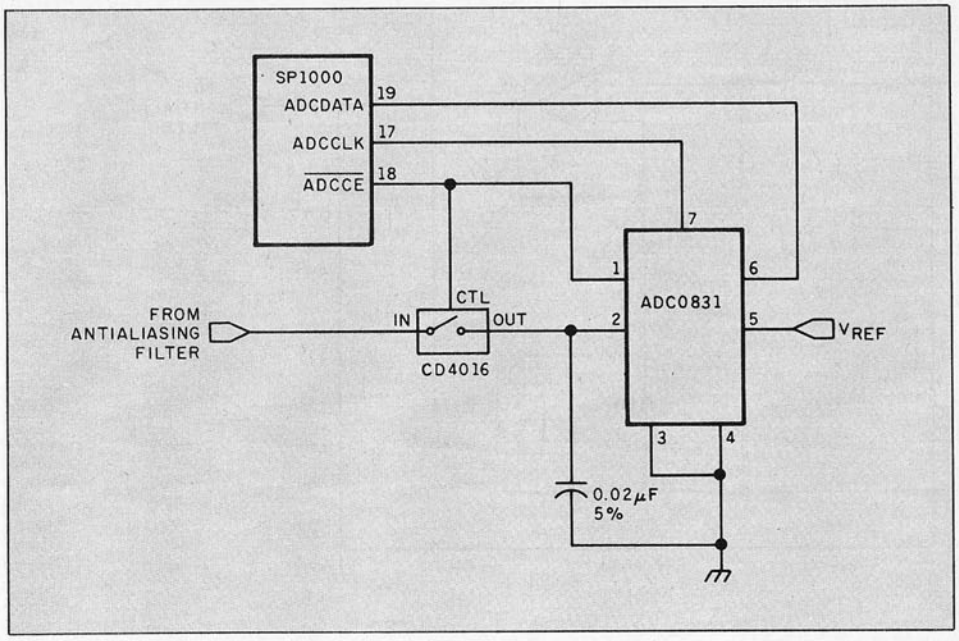


Figure 7: *The sample-and-hold 8-bit* A/D *converter.*

To avoid background noise pickup, I suggest the microphone headset combination shown in the opening photo. This keeps the microphone close to the mouth and limits interference.

The high-pass filter removes all sounds below 250 Hz.

The output from the high-pass filter is connected to an automatic-gain-controlled amplifier (see figure 5). The SP1000 provides three output lines that control switches to vary the resistor values within a circuit consisting of two noninverting operational amplifiers connected in series. These signals are GAIN 6, GAIN 12, and GAIN 24, corresponding to 6-, 12-, and 24-decibel (dB) signal levels (this is a voltage gain of 2, 4, and 15.8, if you are interested). See table 1 for the gain produced by combining these pins.

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 eight combinations of signal amplification from 0 dB to 42 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, low-pass, antialiasing filter (see figure 6) before going to the A/D converter.

Once a conditioned signal with all the extraneous noise removed is obtained, it is directed to the sample-and-hold A/D converter to be read by the SP1000 (see figure 7).

The SP1000 provides two signals for controlling the A/D converter and the sample-and-hold circuit: ADCCLK and ADCCE. The ADCCE signal provides an active-low chip select that turns off the sample switch (the switch, which is normally closed, opens when the A/D converter reads the voltage level stored on the capacitor) and enables the A/D converter. I used a National Semiconductor ADC0831 8-bit serial-output converter clocked at 150 kHz provided through ADCCLK as the A/D converter. The serial-output data is read through the ADCDATA line. You can program the SP1000 to read the input data at 5k to 16k samples per

second. As configured in this project, the sample rate is 6.25k samples per second.

A complete schematic showing the recognition part and LPC-synthesis portion of the Apple II Lis'ner 1000 is shown in figure 8. Figure 9 shows the circuit changes necessary to add the SSI-263 specifically for the Apple II. Figure 10 is the Commodore 64 version.

SP1000 Software

Figure 11 is a flowchart of the basic software control of the Lis'ner 1000. The routines described assume that the SP1000 is implemented as a discrete-utterance speaker-dependent unit. The software can be segmented into two major functions: the creation of training templates and actual recognition of utterances relative to the training templates previously created.

Training

The purpose of training is to create a set of patterns, each of which represents a specific utterance. (Note that an utterance may be a single word or a phrase.) When recognition is performed, these patterns are compared to a pattern created from the word to be recognized. The pattern or template from the training utterances that is closest to the word to be recognized is the one the system chooses as the recognized word.

A well-designed training process will create templates that capture the unique features of an utterance in a form simple enough to facilitate the matching process and the efficient use of a system's memory. With this in mind, let's examine the training process implemented here.

The first step is initialization of the hardware and software. The SP1000 is an extremely flexible device that allows the user to specify several parameters that govern its analysis calculations. The parameters include the sample rate (6.25 kHz), the analysis-frame duration (20 milliseconds [ms]), and the gain-update period (10 ms). Once these parameters have been specified and the software has enabled the interrupts, the SP1000 will provide the processor with a fresh analysis frame at the end of each frame period.

The software initialization consists of setting the counters for the number of templates and the number of training passes for each template. The number of templates (utterances) is variable. The system uses two training passes for each template.

After initialization, the program enters the endpoint-detection process. Since this is a discrete-utterance recognizer, it must identify the start and finish of each utterance it "hears." This applies to both training and recognition. The endpoint-detection algorithm is designed around a finite-state machine with four states: silence, rising, plateau, and falling.

The SP1000 continually analyzes the audio input and sends its analysis data to the host processor. Whenever no speech is reaching the microphone, the SP1000 will be analyzing the ambient room noise. This represents the silence state. While in this state, the processor is constantly calculating a noise level based on the average energy of the last 16 frames of silence. If an incoming frame has an energy 6 dB or more above the noise level, the machine enters the rising state. Similar energy measurements control the state transitions throughout the duration of the utterance until the machine exits back to silence, indicating that the end of the utterance has been reached.

Once the machine enters the rising state, it saves all the analysis frames generated by the SP1000 until the end of the utterance has been found. At that point, the data collected is tested with criteria pertaining to minimum duration and dynamic range to confirm its legitimacy as speech input and pinpoint the endpoints more closely. A normalization process is also performed on the energy coefficients to equalize weighting.

At the end of this process, we have captured a parametric representation of the utterance. The next step is to include that representation in a training template.

It is worth noting that the data collected for an utterance with one second of duration is calculated as follows: (8 bits/coefficient) × (9 coefficients/frame) × (50 frames/second) × (1 second) = 3600 bits of data (450 bytes). Utterances of 3 seconds in duration would generate 1350 bytes. If left in this form, a few dozen utterances would take a sizable quantity of memory just for storage.

Fortunately, the system need save only the unique characteristics of an utterance in order to perform good recognition. The unique sounds that constitute a particular utterance will usually be several frames in duration. Thus, the algorithm tests the utterance data one more time, essentially to perform a type of averaging in which adjacent frames with similar coefficient values are combined to form one new frame that replaces the two old ones. This process reduces the total number of frames in an utterance to 12. Theoretically, these 12 frames are representative of the unique speech sounds that occurred in the utterance. This process also provides a time normalization for all utterances. Since all utterances are reduced to 12 frames, they all have an identical length for comparison purposes.

The purpose of training is to create a set of patterns, each representing a specific utterance.

The resulting 12 frames constitute a template. Since different repetitions of an utterance are never exactly alike, even when spoken by the same person, the software averages two templates created from two repetitions of the utterance in order to form a more general template. This is stored as the training template for that utterance. The final size of the training template is 12 frames of 9 coefficients each (or 108 bytes of data).

Recognition

Recognition is performed with the same front-end software as the training. It uses the finite-state machine and post-processing functions to identify the endpoints of the utterance and performs time normalization to create a 12-frame unknown utterance template. It then tries to find the best match among the training templates previously stored. The two key elements of the matching process are the frame-to-frame distance measure and nonlinear time alignment.

Distance Measure

I have mentioned the closeness of templates, which is used to determine

the best match. But just how do you determine the closeness of two templates? The answer lies in a frame-to-frame distance measure, which is used to build a template-to-template distance. The smaller the template-to-template distance, the closer the two templates are to one another.

A Chebyshev distance measure is employed as the frame-to-frame distance measure. The equation ABS VAL $(A_i - B_i)$ is summed for all i, where i is the distance from frame A to frame B, A_i is the Ith element of frame A, and B_i is the Ith element of frame B.

Thus, frame-to-frame distance measurement consists of simply summing the magnitudes of the differences of corresponding elements in the frames being compared.

To find a template-to-template distance, the frame-to-frame distance measure is applied within the context

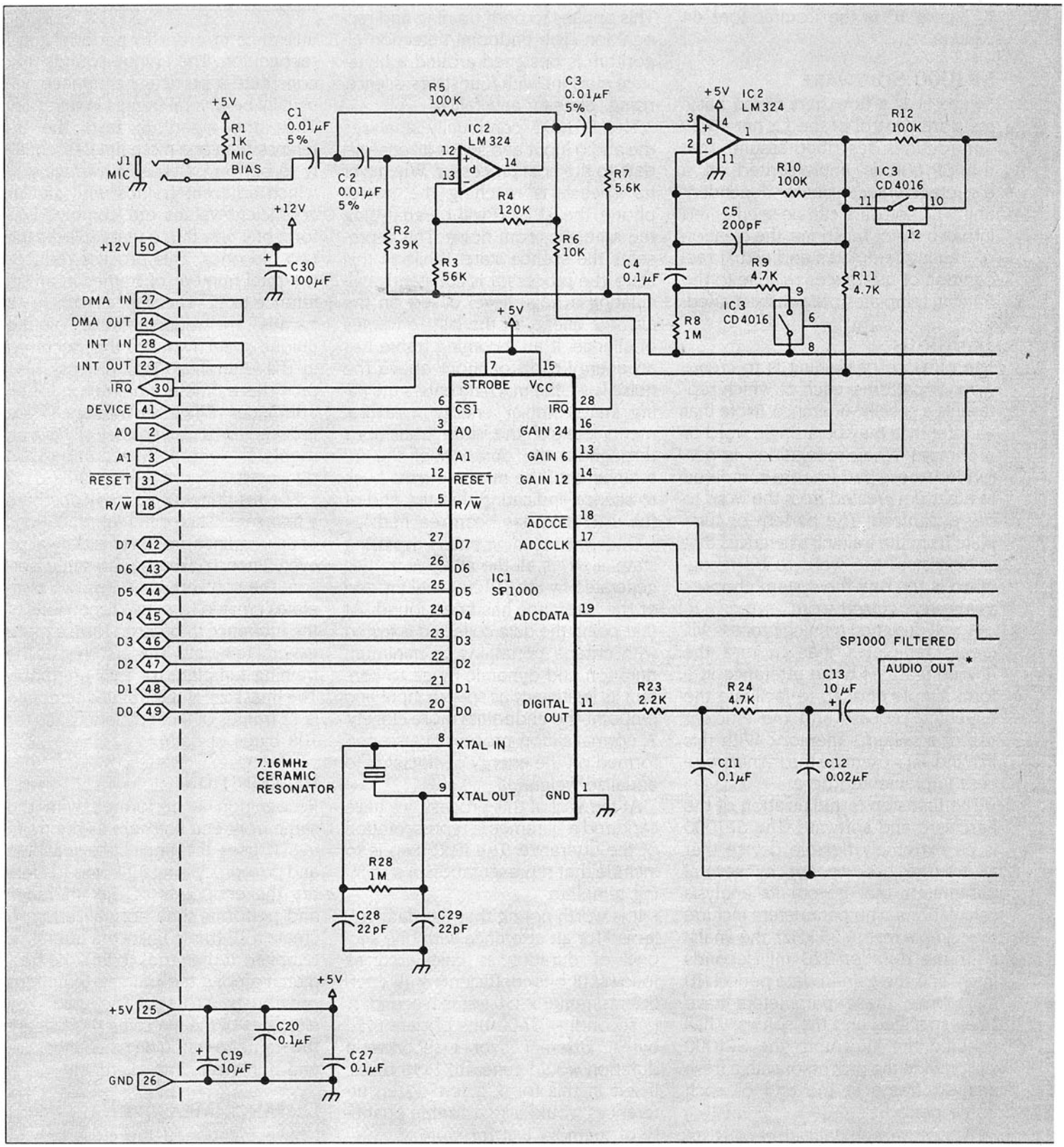


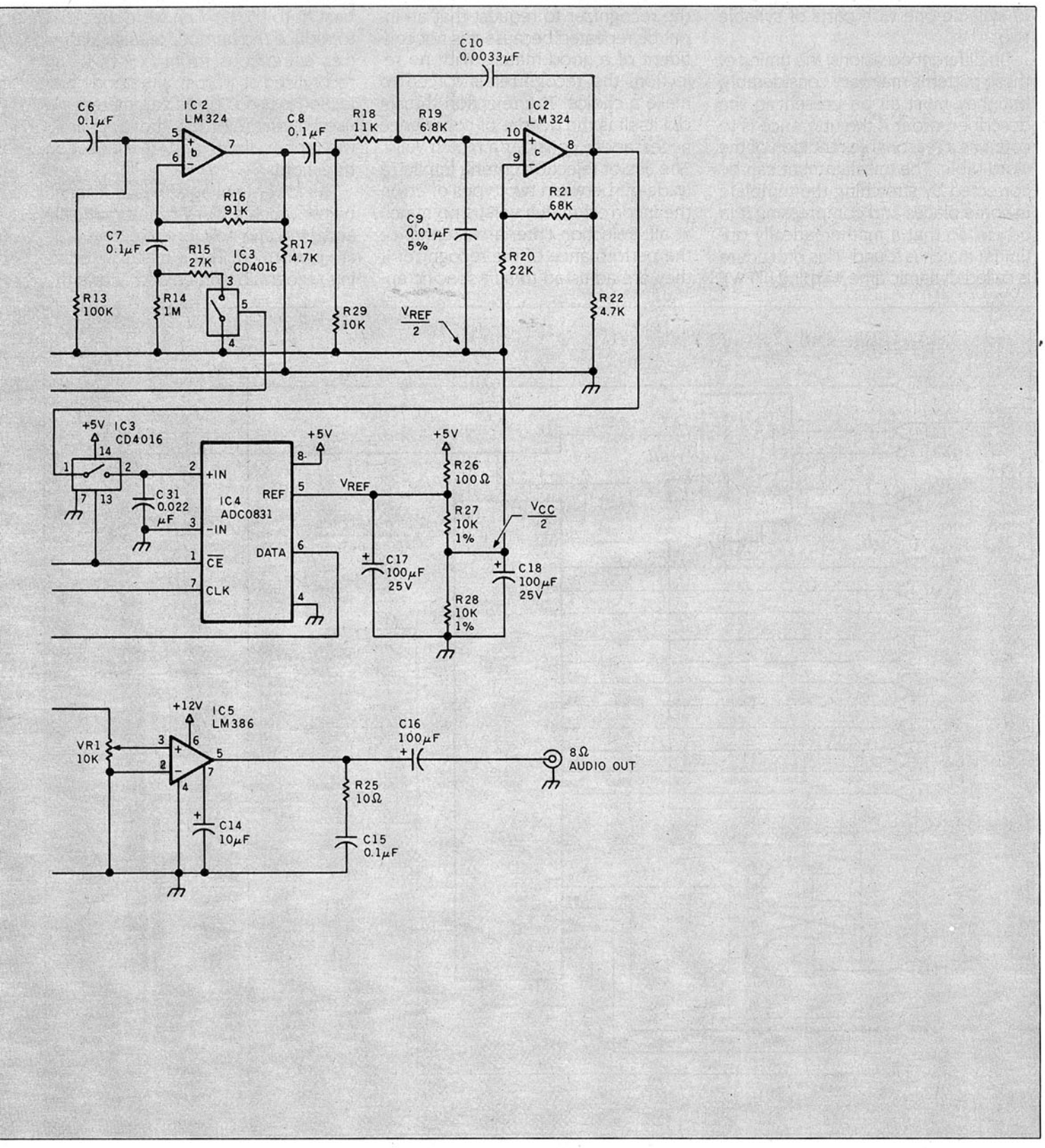
Figure 8: *The Lis'ner 1000 schematic without the SSI-263 for the Apple II.*

of the nonlinear time alignment of the frames.

Nonlinear Time Alignment

The need for nonlinear time alignment arises because human beings do not speak the same words exactly the same way each time. Volume and duration of words obviously vary, but a more subtle variation is very significant to a speech recognizer. The individual speech sounds comprising a word vary in duration relative to one another in different repetitions of the same word. This time distortion is nonlinear because simply stretching or compressing one entire repetition will not time-align the boundaries of the speech sounds with those of another repetition of the same utterance.

Consider a word with two syllables, such as table. Two repetitions of this



word may have the same total duration, but the first syllable may constitute 50 percent of repetition one and only 30 percent of repetition two. If we created templates for the two repetitions and compared them on a frame-by-frame basis, we would not get the best match because at some point we would be comparing parts of syllable one with parts of syllable two.

On different occasions, the timing of these patterns may vary considerably, but they must all be present in the described order if the utterance is to count as a reasonable rendition of the word table. The misalignment can be corrected by stretching the template in some places and compressing it in others, so that a mathematically optimum match is found. This procedure is called dynamic time warping (DTW). (See "Speech Recognition: An Idea Whose Time Is Coming," January, page 213.)

Rejection Threshold

Once we have a best match, we have to determine if it is usable or not. One method is to qualify the match by setting a rejection threshold. This allows the recognizer to request that an input be repeated because it is not confident of a good match. With no rejection, the recognizer is forced to make a choice. The rejection threshold itself is the degree of confidence necessary to consider a match valid. The use of rejection criteria implies a trade-off between two types of error, the incorrect match versus no match at all. Rejection criteria can enhance the performance of the recognizer if they are adjusted to suit specific applications. The problems caused by the two types of error are application dependent.

As part of the recognition process, a template is made of the word just spoken, and it is compared to the templates made during training. For each comparison, a distance is computed that is used to determine the best fit to the spoken word. In order to reduce the number of false alarms (i.e., extraneous room noises being recognized as words), a method of rejection is used. Three parameters are used during rejection: the lower limit, the upper limit, and the rejection threshold.

The lower limit specifies a distance below which a word is automatically accepted and no more rejection tests are performed. This is useful in reducing recognition times and allows the

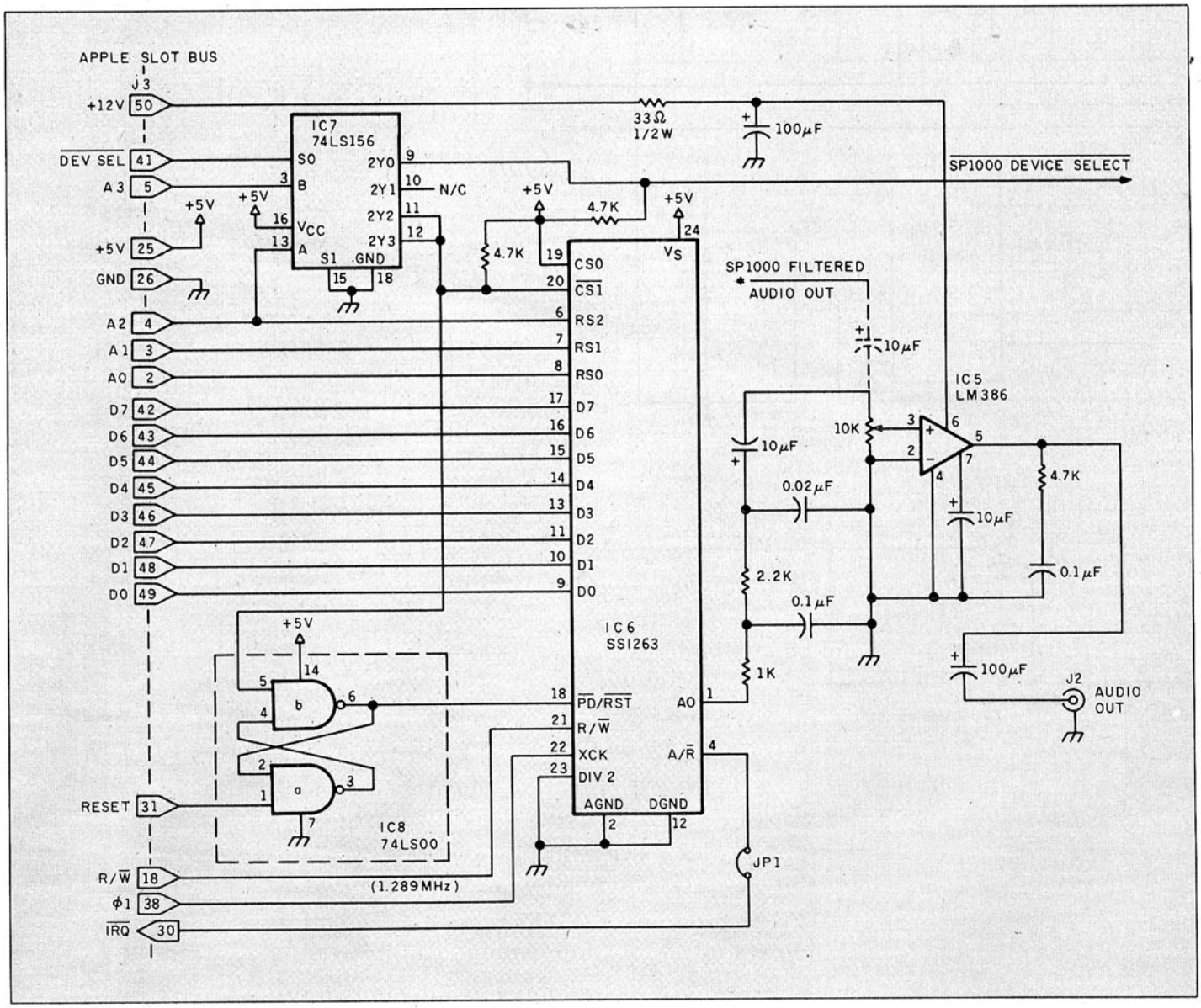


Figure 9: *The Lis'ner 1000 schematic with the SSI-263 for the Apple II.*

obvious correct matches to pass through. However, if it is set too high, many false alarms may occur.

The upper limit specifies just the opposite, the distance above which a word is automatically rejected. This too is helpful in speeding reaction times and discards obvious room noises such as clapping. If this number is set too low, a large incidence of rejecting good words will occur, resulting in a good deal of frustration for the user.

The last parameter is the rejection threshold, which is used to control just how close the spoken word may be to the two next closest reference templates. In short, a small rejection threshold results in a higher degree of rejection; a large rejection threshold is more forgiving and rejects less.

These three parameters are combined to tailor the system to the user's particular needs. If a highly speaker-dependent system is desired, a small lower limit, a small upper limit, and

(continued)

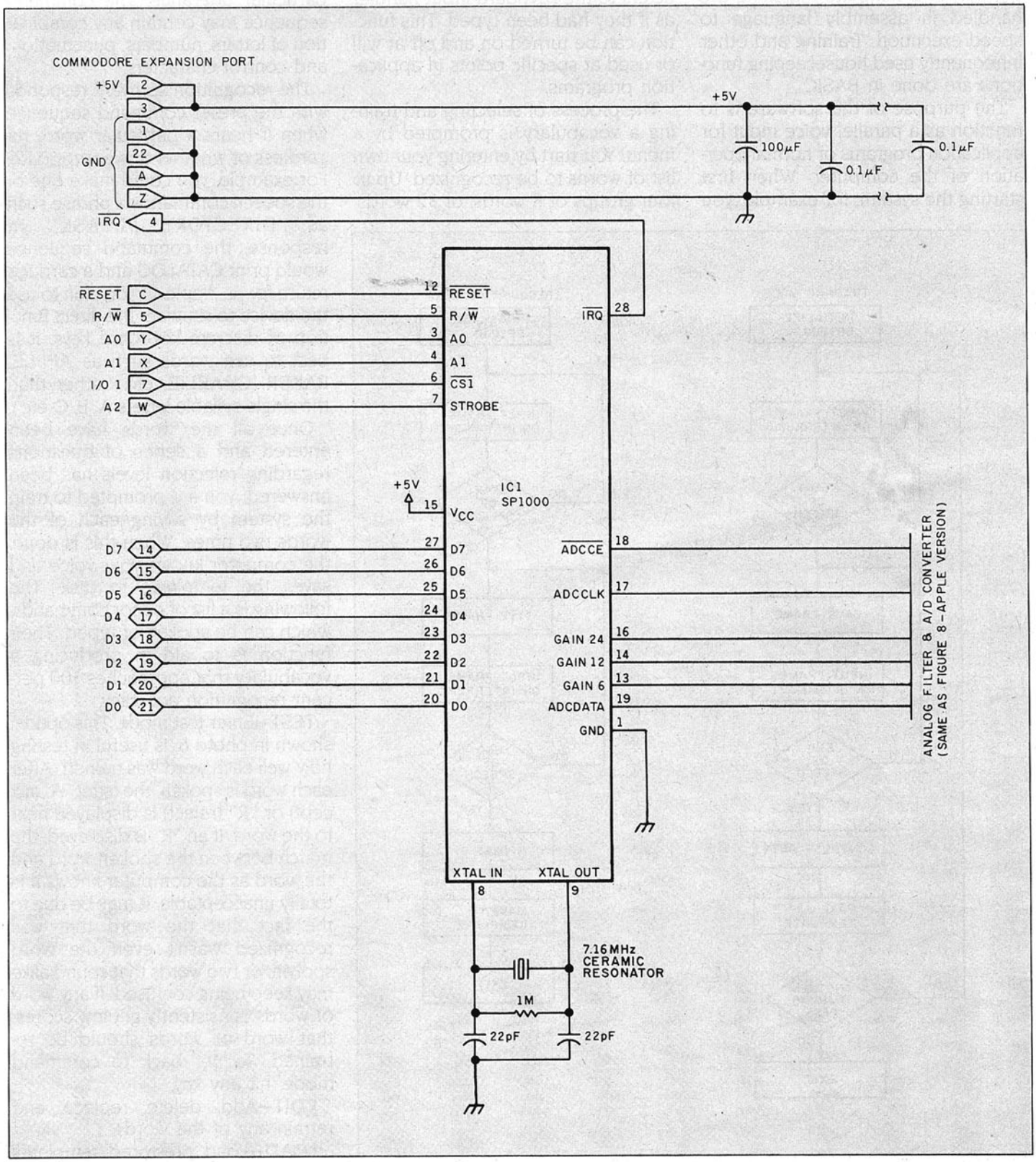


Figure 10: *The Lis'ner 1000 schematic for the Commodore 64.*

a small rejection threshold should be used. The result would be that only the person who trained the system would have good recognition results.

LIS'NER 1000 SOFTWARE

The Lis'ner software consists of a combination of BASIC and assembly-language routines. The recognition algorithm and template matching are handled in assembly language to speed execution. Training and other infrequently used housekeeping functions are done in BASIC.

The purpose of the software is to function as a parallel voice input for application programs or normal operation of the computer. When first starting the system, for example, you are prompted to train a preselected vocabulary of DOS (disk operating system) and system commands. Rather than typing CATALOG and a carriage return, you merely have to say "catalog" and "return" (you can still type any part of it if you wish). In effect, the Lis'ner 1000 can be programmed to send a sequence of characters to the keyboard input handler as if they had been typed. This function can be turned on and off at will or used at specific points in application programs.

The process of selecting and training a vocabulary is prompted by a menu. You start by entering your own list of words to be recognized. Up to four groups of 8 words, or 32 words, are entered at one time, as shown in photo 4. A total of 64 words may be entered into the system. Next, you are asked for each spoken word followed by its corresponding command sequence, as shown in photo 5. The command sequence is the group of characters that the recognizer routine will respond with when it hears this particular utterance. The command sequence may contain any combination of letters, numbers, punctuation, and control characters.

The recognition software responds with the preset command sequence when it hears a particular word, regardless of whether it is appropriate. For example, you could make one of the speech commands a phrase such as "DIRECTORY, PLEASE." In response, the command sequence would print CATALOG and a carriage return for an Apple. (If you plan to use the device to simulate the direct function of discrete keyboard keys, it is best to use words such as APPLE, BAKER, CHARLIE, etc., rather than the single-syllable letters A, B, C, etc.).

Once all the words have been entered and a series of questions regarding rejection levels has been answered, you are prompted to train the system by saying each of the words two times. When this is done, the computer knows your voice and saves the templates to disk. The following is a list of editor commands, which can be spoken or typed. Their function is to aid in producing a vocabulary that approaches 100 percent recognition accuracy.

TEST—Enter test mode. This option, shown in photo 6, is useful in testing how well each word was trained. After each word is spoken, the letter "A" (accept) or "R" (reject) is displayed next to the word. If an "R" is displayed, the match between the spoken word and the word as the computer knows it is totally unacceptable. It may be due to the fact that the word that was recognized wasn't even the word spoken, or two words that sound alike may keep being confused. If any word or words consistently get low scores, that word or words should be retrained. To get back to command mode, hit any key.

EDIT—Add, delete, replace, and retrain any of the words.

LOAD—Load prestored templates so that editing and training may be performed.

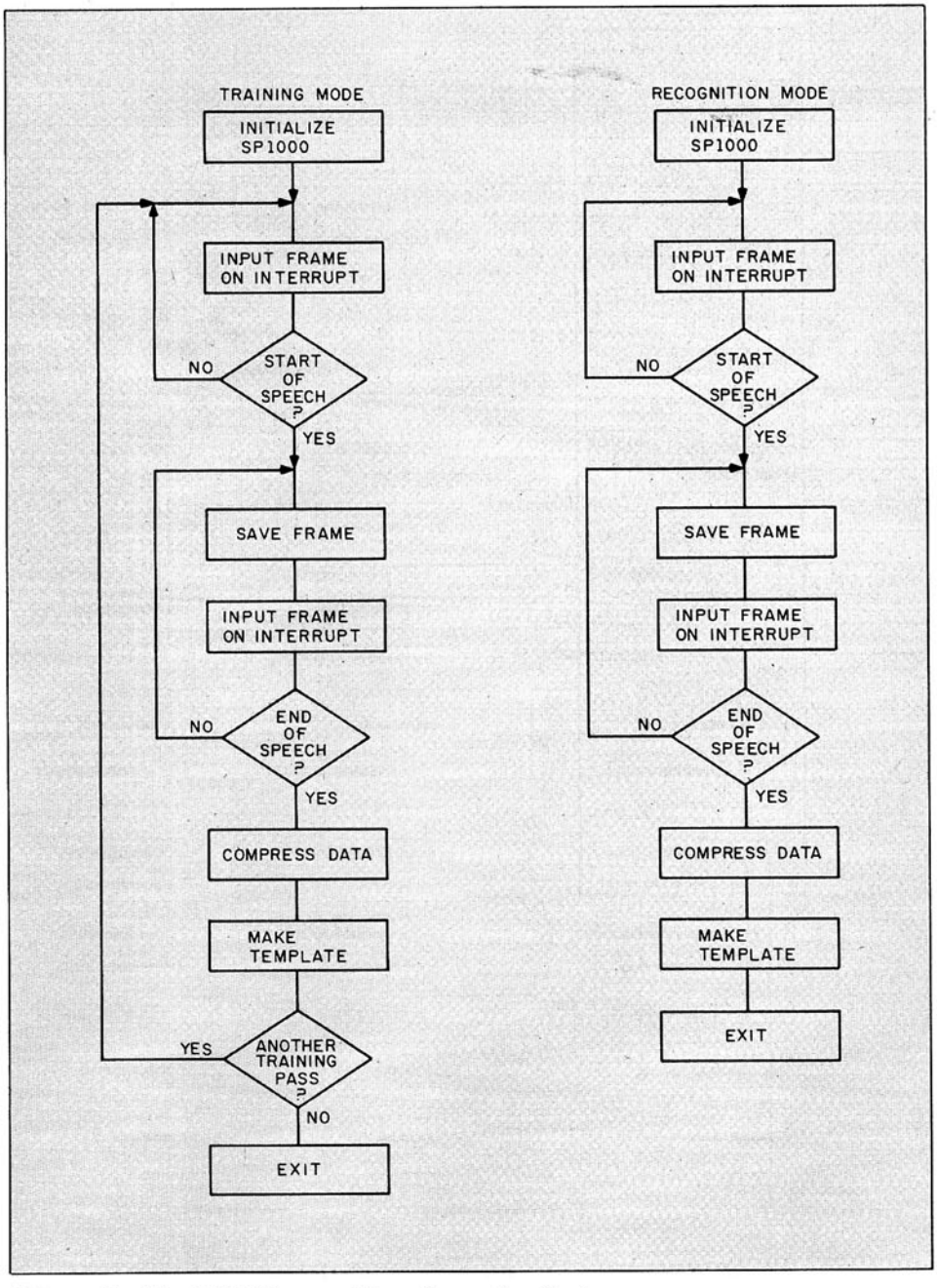


Figure 11: *The* SP1000 *recognition-software flowchart.*

SAVE—Save the templates being worked on for later use. This is used to save all the work you've done up to now. These templates may then be loaded by one of the Hello programs at some later date for use in your application programs.

QUIT—Leave the editor and return to BASIC. Once all your editing and saving are done, you may enter BASIC with the recognize routine and DOS templates still active.

Software design is of course dynamic. Some aspects of the Lis'ner software I've described here may have been modified by the time you read this.

Experimenter Support

I try to support the individual experimenter as much as I possibly can, and this project is no exception. To aid you in building the Lis'ner 1000 or an SP1000-based system, I have coordinated parts and software suppliers.

The Lis'ner software package consists of a combination of source-code and executable-only code files that are much too lengthy to print for distribution or to be published here. The Lis'ner software is supplied as BASIC source code with assembly-language executable code. Since I expect that many of you won't be happy until you've personally experimented with dynamic time warping and converted the routines to run on a different processor, I am making available demonstration source code for an SP1000 recognition algorithm for the Apple II. This code, which is less complicated than the Lis'ner software, was written by General Instrument. Also included are LPC coefficient files that will demonstrate the SP1000's synthesis capability.

Although this software is well annotated, it is unsupported and distributed for its educational value only. It contains all the necessary structure should you care to roll your own. (If you do convert these routines, I would be very interested in seeing your handiwork.)

The Experimenter Support package contains the General Instrument SP1000 demonstration software, Lis'ner software, and the *Lis'ner 1000 User's Manual*. It is available on disk for either the Apple II (except IIc) or Commodore 64 (please specify) directly from me for a $17 shipping-and-handling charge ($27 for overseas airmail). This offer is valid until March 1, 1985.

Finally, while it isn't a requirement that you include a picture this time when you write to me, I'd like to see your finished product so that I can add your picture to the many hundreds I've received on previous projects.

Conclusion

The toughest part about writing this article was deciding how much to say about recognition techniques. I have

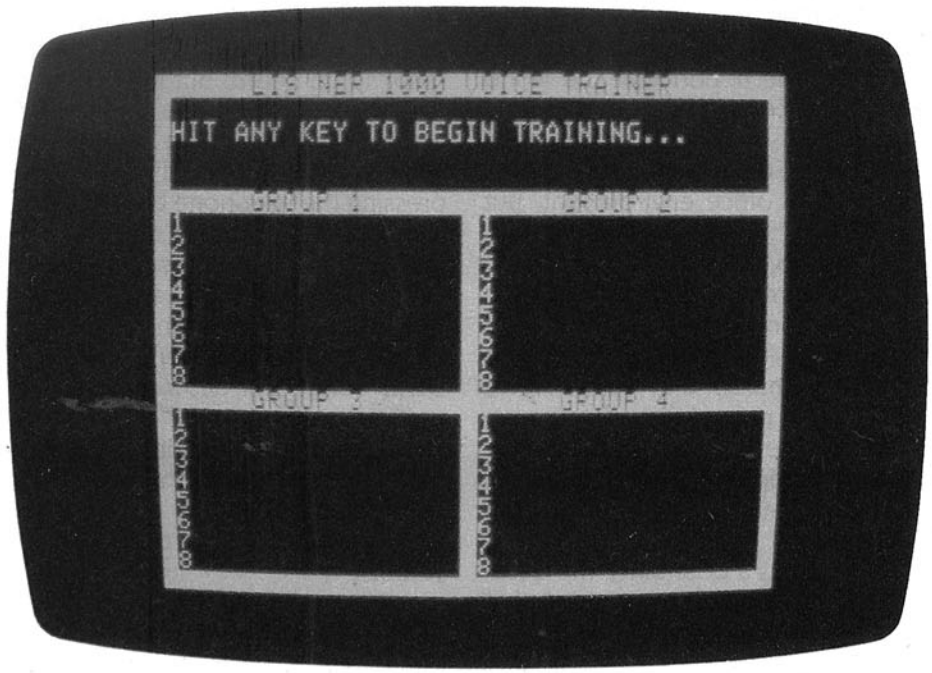


Photo 4: *In this training mode, you select and train up to 32 words at a time. Multiple overlays of these template dictionaries result in potential recognition vocabularies of thousands of words. In practice, 64 concurrent-available words is a reasonable search vocabulary that maintains a high response reaction time.*

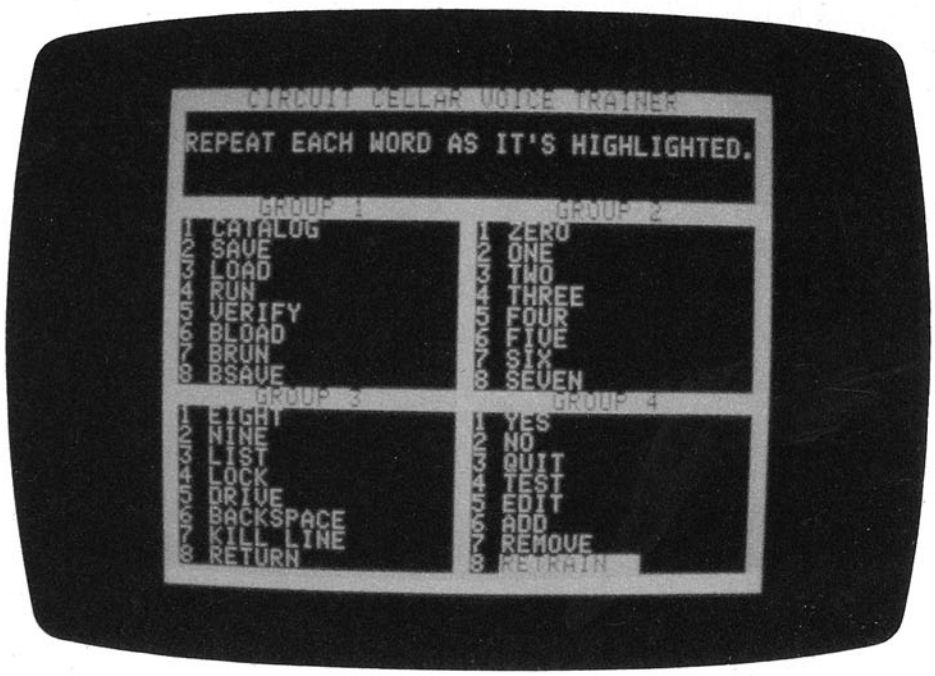


Photo 5: *These standard* DOS *commands comprise one of the vocabularies that the user is directed to train. Once trained, many keyboard entries can now be verbal.*

barely scratched the surface in my explanation.

It is equally difficult for me to list and describe the multitude of potential applications for computerized voice recognition. Besides the obvious aids for the disabled, the Lis'ner 1000 can be used in order-entry systems, voiceprint-security systems, video games, and telephone communications. Also, many people subscribe to the notion that the world needs a voice-operated typewriter. In my opinion, it will be a long time before voice entry becomes commonplace in an office environment, but there have been inroads.

I intend to apply the board to telephone communications so that I can call and correspond with my computer. Since the Apple II Lis'ner has both recognition and synthesis, it would seem natural that all conversation over the phone with the Apple should be spoken. "Hello, computer, how are you?" "Fine, Steve, your house is still here."

While this is a possibility, the quality of the telephone lines suggests that an alternate means of backup communication also be used. Some time ago I wrote an article about DTMF (dual-tone, multiple-frequency) decoders ("Build a Touch Tone Decoder for Remote Control," December 1981, page 42). In it, I suggested that one way to communicate with your computer was through an auto-answer device with a DTMF decoder. Once the computer answers, simply send your message by pressing the Touch-Tone keys on the telephone.

Your first thought might be to add a DTMF decoder in parallel with the recognition board, but it is quite unnecessary. DTMF tones and spoken words are all sounds as far as Lis'ner is concerned. It is simply a matter of pressing the telephone buttons while in the template training mode to program the Lis'ner to respond to the DTMF tones. Adding a few select words in addition will make it a truly unique answering system. Using just DTMF tones will allow invited subscribers a certain level of access to your system, but combining speaker-dependent voice recognition with DTMF tone recognition will allow you to reserve certain functions only for yourself.

Circuit Cellar Feedback

Circuit Cellar Feedback is a new feature I'm starting. Every month, I'll answer letters about past projects. This month's Circuit Cellar Feedback begins on page 430.

Next Month

I'll show you how to build an AC I/O controller. ■

Special thanks to Dennis Intravia for his work on the recognition software.

Diagrams and data specific to the SP1000 are reprinted courtesy of General Instrument.

Editor's Note: Steve often refers to previous Circuit Cellar articles. Most of these past articles are available in reprint books from BYTE Books, McGraw-Hill Book Co., POB 400, Hightstown, NJ 08250.

Ciarcia's Circuit Cellar, Volume I covers articles that appeared in BYTE from September 1977 through November 1978. *Volume* II covers December 1978 through June 1980. *Volume* III covers July 1980 through December 1981. *Volume* IV covers January 1982 through June 1983.

The following items are available from

The Micromint Inc.
561 Willow Ave.
Cedarhurst, NY 11516
(800) 645-3479 for orders
(203) 871-6170 for information

1. Apple II Lis'ner 1000 with SP1000 recognition/synthesis components only—includes headset-style microphone and software on disk.
VR01 assembled and tested $189
VR02 complete kit $149
2. Apple II Lis'ner 1000 with SP1000 recognition/synthesis components and SSI-263 phoneme synthesizer chip with text-to-speech algorithm—includes headset-style microphone and software on disk.
VR03 assembled and tested $259
VR04 complete kit $219
3. VR01/VR02 phoneme-synthesis upgrade to VR03. Includes SSI-263, miscellaneous components, and text-to-speech algorithm on disk.
VR05 VR01/VR02 upgrade kit $79
4. Commodore 64 Lis'ner 1000 with SP1000 recognition/synthesis components—includes headset-style microphone and software on disk.
VR10 assembled and tested $149
VR11 complete kit $119
5. Apple II speech experimenter's kit—includes SP1000, 7.16-MHz ceramic resonator, ADC 0831 A/D chip, Lis'ner manual, and Lis'ner software on disk.
VR20 complete kit $60

Please include $4 for shipping and handling in the continental United States, $10 elsewhere. New York residents please include 8 percent sales tax.

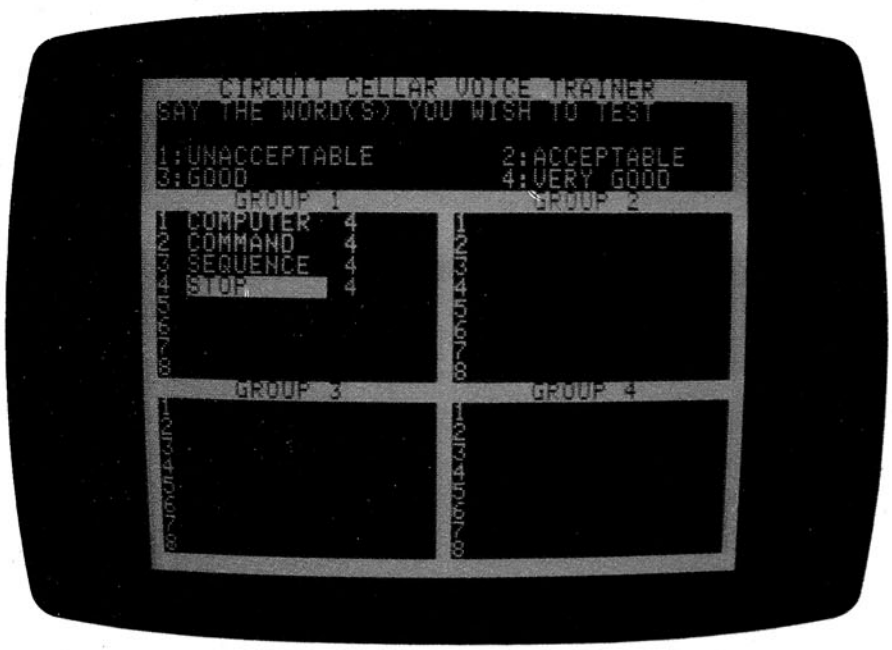


Photo 6: *One of the features of the Lis'ner software is the ability to make and test a recognition vocabulary. In the modified editor program shown here, as the words are spoken, the acceptance level is noted so the user can select words with less interference. Words like "computer" and "sequence" have few differentiation problems. "Nine" and "mine" would present difficulties, as would "next" and "text."*