SCG673

Leading Electronics Press Coverage of Speech Synthesis Products and Technology

from Texas Instruments

The bright promise of products that think came closer to fulfillment in the 70s. The trend for the 80s is clear: Products that talk.

Whether it's appliances, automotive consoles, transaction terminals, office systems, telecommunications, robotics or electronic toys and games, no other area of technology will have more impact than speech synthesis during the coming years.

Speech synthesis technology — pioneered by TI — promises to bring a whole new generation of voice-prompting products. Products that will change the way you live, and learn, and work.

Texas Instruments Solid State Speech™ technology gives designers a revolutionary new approach to use in creating machines that reproduce human speech accurately.

This technology, first introduced in TI's remarkable Speak & Spell™ electronic learning aid, provides designers with highly intelligible, reliable, low-cost speech production.

Further advances in Solid State Speech technology have produced lifelike, natural speech, and have very low memory requirements. This enables designers to consider and design talking products where they couldn't before.

In recent months, Texas Instruments has introduced several new Voice Synthesis products. Many of these products have been featured in technical articles in leading electronics publications. The following seven articles have been reprinted in this brochure, by courtesy of the magazines in which they first appeared.

- Systems approach puts the right words — and enough of them — into speech-synthesizers (*Electronic Design*, May 28, 1981)
- Synthesizer chip translates LPC to speech economically (*Electronic Design*, June 11, 1981)
- Phonemes, allophones, and LPC team to synthesize speech (*Electronic Design*, June 25, 1981)
- Speech recognition spurred by speech-synthesis success (*Electronic Design*, July 9, 1981)
- LPC speech-synthesis chips mate easily with micros (*Electronic Design*, July 23, 1981)
- Speech-synthesizer software generated from text or speech (*Electronic Design*, August 20, 1981)
- Software rules give personal computer real word power (*Electronics*, February 10, 1981)

We think you will find each article interesting and useful.

Systems approach puts the right words—and enough of them—into speech-synthesizers

Integrating high-quality speech responses into industrial process controls, aircraft instrumentation, banking systems, or consumer-electronic products involves much more than selecting off-the-shelf components. The designer must very carefully assess how speech as a whole, and each of the different kinds of speech, can best serve his particular application. For this reason, Bernard H. List, vice president for MOS functions at Texas Instruments' Speech Technology Center (Midland, TX), believes that speech decisions call for a systems perspective, with close interaction between the manufacturer of speech systems and the user.

List expects the speech business to follow a pattern that is similar, in many respects, to the one established by microcomputer chips, such as TI's TMS1000. Before that chip became a low-cost, widely used component, TI worked extensively with manufacturers of appliances and electronic games to develop the appropriate software for each application.

In the case of speech products, the process is much more difficult to implement, since the equipment manufacturer must first decide what sort of job he expects a speech synthesizer to do for him. "There are a number of applications in which speech will be doomed to failure," predicts List, "because speech or voice feedback will not be particularly useful." But speech is vital, he adds, in products that educate the user.

Speech should be discretionary in some, and perhaps in the majority, of applications. For example, in driver-warning systems in automobiles, speech can be an option, based on the consumer's decision. But in diagnostic systems for automobiles, especially computer-based service systems, speech will prove to be more and more essential.

The toys and games market was the first place in which speech chips made their appearance, with products like Speak and Spell (1977). However, the general downturn in the consumer area has forced manufacturers of speech chips to look for new market areas.

In each and every application, the following questions must be answered in detail before the speech-response system is built: How will speech be used in the system? What will the product say? What quality of speech is required? List points out that TI is prepared to help manufacturers with the answers to these questions through its network of Regional Technology Centers. But the answers will not be simple, and in all cases must be carefully thought out if the product embodying speech is to be successful.

Bernard H. List, vice president and manager of the MOS Functions Division at Texas Instruments (Midland, TX), joined TI in 1957. He has held several positions within the company since then, including manager of the Corporate Engineering Center, manager of U.S. MOS Operations, and manager of the Systems Information Sciences Laboratory. He received a BS and a PhD in electrical engineering from Johns Hopkins University.

Tradeoffs with memory space

Invariably, the first tradeoff the user faces is in memory-space allocations: a large vocabulary vs high-quality speech. The designer must decide how many words and which particular words will do the job.

Normal speech conveys approximately 100,000 bits per second. This amount is within the capability of available speech-synthesis systems, but makes extremely inefficient use of memory space. The three major speech-synthesis systems—linear predictive coding, waveform modulation, and phoneme stringing—all differ in how they use memory. Currently, implementing a speech system costs less than $50 per word, although the cost will likely fall in the future.

One step toward lower prices will be the creation of a standard library of preloaded ROMs, according to List. Each 128-kbit ROM will store 150 to 200 words, all geared for specific applications. Each ROM is likely to sell for $10. The complete library will contain 1500 words. Eventually, EPROMs will allow users to exchange words, and microprocessor-based systems will use expanded memory to call out words or phrases from floppy disks, tapes, or other storage systems.

While speech inevitably demands a large memory space, it does not demand inordinate amounts of computing power. One of the virtues of linear predictive coding (LPC), in fact, is that it simplifies the elements of speech, making the resultant digital code acceptable for manipulation by a 4-bit processor, like the TMS1000. In List's view, the new generation of 16-bit microproces-

sors and the coming generation of 32-bit processors will not simplify the job of speech synthesis (save for allophone-stringing), although they will be an asset to speech recognition.

Texas Instruments is working with all three types of speech-synthesis systems; but LPC is the one most typically employed by TI, because it maximizes the use of memory space.

LPC is modeled on the human vocal tract. It extracts the essential coded elements from a spoken word, so fewer bits are required to produce each word. The compression of data is approximately 100 to 1, List explains. The LPC algorithm is very flexible, since words or phrases can be concatenated to add emphasis and inflection after the LPC algorithm is applied. Thus, LPC will be the likely standard for speech synthesis – the model that other speech systems will attempt to match.

An alternative scheme is waveform modulation, in which the analog waveform is sampled, digitized, and stored for reconstruction, word by word, by the synthesis module. Because the original analog waveform is heavily sampled, very-high-quality speech can be obtained, but at the expense of memory. Waveform modulation consumes a great deal of memory space for each word or phrase. It is less expensive than LPC, but only in systems requiring a very small vocabulary. If the system calls for an expanded vocabulary, waveform modulation can take as much as three times the memory space required by LPC, as well as more computing power.

The third form of speech synthesis is a phoneme-stringing system. It is very similar to the allophone-stringing system built by TI for text-to-speech conversion. One commercial phoneme-stringing system has a library of 64 recorded phonemes, which are the basic sound elements in all words (TI's text-to-speech system has 128 allophones). The synthesizer strings together phonemes or allophones according to a preprogrammed set of rules for each language.

The advantage of this sound-stringing system is that it offers an almost infinite vocabulary within a very small memory space (6 kbits of ROM). The disadvantage is the poor speech quality. The voice that is produced sounds like a computer. In the future, says List, the phoneme or allophone-stringing systems will probably contain rules for emphasis on certain words or syllables, a procedure that will improve the quality of speech and make the spoken word pattern sound more human. But for the time being, the phoneme-stringing system does not have the quality of voice that many speech applications are likely to require.

"The mid-1980s are the time to expect a universal speech-recognition chip. LPC will be the dominant format."

The need for speech quality, List suggests, also generates a need for systems that measure the quality of speech. Currently, all means of judging speech quality are subjective. A dozen engineers will declare a synthesized voice acceptable, only to have a single untrained ear fail to recognize the synthesized message.

With this need for an objective measure of the quality of speech will come an acceleration of developmental efforts in speech-recognition systems. The current state of the art in voice-pattern recognition is represented by custom-made board-level and minicomputer products capable of recognizing a very simple vocabulary consisting of frequently repeated phrases. In the forefront of speech recognition are small entrepreneurial companies.

The transition from board-level products to chips will take place over the next two or three years. Even now, board-level systems are programmed to recognize particular voices and particular word sets; in fact, they are barely able to recognize words or speech apart from a particular speaker. What is needed, List suggests, is a universal speech-recognition device, one that is able to recognize spoken words regardless of the speaker's voice pattern. The very first of these will appear in systems such as consumer appliances, which require very few spoken words, such as "on," "off," "stop," and "go." The next generations will have much larger vocabularies and much smaller error rates, and will be found in such applications as automated bank tellers. The demands on memory will be much greater for voice-recognition systems than for speech synthesis.

The mid-1980s are the most likely time to expect a universal speech-recognition chip. List considers LPC the dominant format for speech or vocal-pattern recognition, since no easier technology is likely to emerge for recognizing voices, words, or speech patterns against precoded and prestored patterns in memory.

As with speech synthesis, speech recognition will call for careful thinking and planning as to the role of speech in the entire system. "This is no easy job," says List. "At TI, we've already had to hire linguists just to give us an idea of how to break speech into its essential elements, to code it and make it acceptable for manipulation by microprocessors. It is likely," he adds, with a smile, "that VLSI in the 1990s will force us to add philosophers to the staff."

Stephan Ohr

With standard LPC, allophonic stringing, or a combination of the two, a new single-chip speech synthesizer delivers good speech quality at a low cost, with little equipment overhead.

Synthesizer chip translates LPC to speech economically

Recent speech-synthesis advances based on the basic elements of linguistics are incorporated in Texas Instruments' new single-chip, the TMS5220 voice-synthesis processor (VSP), a p-channel MOS device packaged in a 28-pin DIP. With its vocal-track simulation, the chip can deliver exactly the degree of speech quality that the user needs in a male, female, or child's voice. The chip can operate in a high-quality linear-predictive-coding (LPC) system, in a medium-quality allophonic system, or in a combination system, with a minimum of external-controller supervision.

No longer need a low-cost synthesized-speech system sound like a metallic, mechanical monster. Because of recent improvements in speech-synthesis, ASCII-keyboard-entered or stored test material (based on a virtually unlimited vocabulary) can be converted to relatively high-quality speech at modest cost, by stringing together speech sound components called allophones. The speech generated with an allophone system is more accurate than that generated with a low-cost phoneme system, which can neither mesh neighboring sounds smoothly, nor provide all the needed sound variations. Nevertheless, the results with allophones still represent a compromise between the greater naturalness of a complex and costly standard linear-predictive-coding (LPC) system and the cruder, more mechanical speech of a low-cost phoneme system (see "Allophones—from Basic Sounds to Complete Words").

The 5220 was developed primarily to operate on standard LPC, which is a form of compressed-speech data. Stringing allophones together produces less natural speech than that provided by the extensive library required by standard LPC. Since the 5220 chip can operate in a combined allophonic-LPC mode, almost any degree of speech quality can be obtained

Douglas O. Wickey, Product Marketing Engineer
Texas Instruments, Inc.
Box 6448, MS 3036, Midland, TX 79701

Table 1. VSP commands and command format

Data-bus command code (D_0-D_7)*	Operation
X000XXXX	NOP
X001XXXX	Read Byte
X010XXXX	NOP
X110XXXX	Speak External
X011XXXX	Read & Branch
X100AAAA	Load Address
X101XXXX	Speak
X111XXXX	Reset

*A = Address
X = Don't care

Table 2. Parameter coding

Parameter	Levels	Code bits
Energy	15*	4
Pitch	64	6
K_1	32	5
K_2	32	5
K_3	16	4
K_4	16	4
K_5	16	4
K_6	16	4
K_7	16	4
K_8	8	3
K_9	8	3
K_{10}	8	3
12	247	49 + repeat = 50 bits

*Energy = 1111 is the stop code

by adjusting the mix.

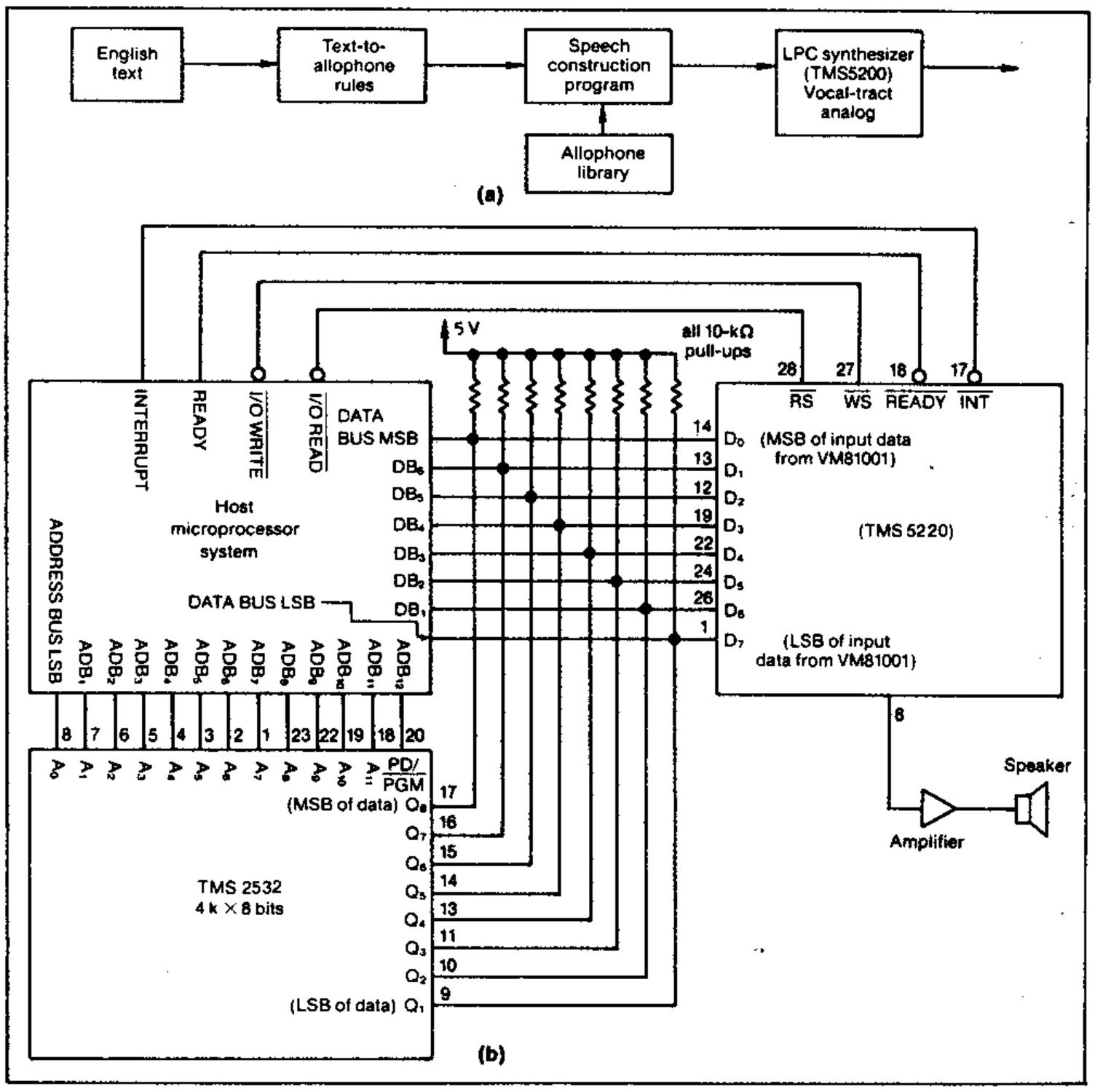
For very high-quality speech—to duplicate the voice of a particular speaker or to ensure absolutely that essential messages will be understood—LPC is the superior technique. But for reasonably good communications at relatively low cost, stringing allophones together is usually sufficient.

The cost savings of the allophone approach over an all-LPC system reflects the following modest system requirements:

- A small, 3-kbyte memory to hold the total 128-allophone library
- Just 7-kbytes of memory to hold TI's Text-20 Speech, a 650-rule set for translating American-language text into allophonic equivalents, and for contouring inflections with the help of pitch modifiers to make the speech sound more natural
- A home computer with a BASIC algorithm (needing just a modest amount of memory) to string together words from the rules and library, and also "naturalize" the resulting speech with smoothing parameters and prosody adjustments.

The limited library of allophones and 650 Text-20-Speech rules stand-in for the almost unlimited LPC library that might otherwise be required. The level of translation correctness currently attained is impressive: 92%. Mispronounciations in the remaining 8% of speech can be remedied by manually intervening in the program.

In an allophonic system, after the input text is converted to its equivalent allophonic strings, the system's speech-construction program changes the strings into a stream of LPC data. The data activate the TMS5220 speech chip, which contains the vocal-tract analog (Fig. 1a). In addition to the small amount of memory required to store the system's library and rules (about 10-kbytes), allophonic voice reproduction does not demand a sophisticated controller



1. A simple allophonic speech-synthesis system (a) can be implemented with a generic-host microcomputer and an EPROM (b) to control the voice-synthesis processor.

capability (Fig. 1b). On the other hand, an LPC system might need a 128-kbit ROM memory (like the TMS6100) for a typical 200-word vocabulary, and a more comprehensive 8-bit host-microcomputer system. For a large comprehensive vocabulary of about 3000 English words, or 30 minutes of speech, 16 such ROMs with a total storage capacity of 256-kbytes would be needed (Fig. 2).

Moreover, in an LPC system, the 5220 must operate at a higher bit rate (1200 bits/s) than in an allophonic system (400 to 600 bits/s) to attain the best speech quality inherent in each system. The speech quality is closely related to the synthesizer chip's bit rate —the closer the rate is to the mode's optimum, the more closely the reproduced speech resembles the natural human voice, and the less it resembles a machine's creation.

But no matter which speech-synthesizer mode is selected, the 5220 itself works the same way: LPC data, which are compressed-speech data, are supplied serially either via a host CPU or directly from an external ROM. The 5220 decodes this data to control a time-varying digital filter that emulates the human vocal tract. As in human speech, digital representations of periodic air impulses (vocalized sounds) or the rushing of air (sibilants) pass through this digital filter to be formed into words. The digital output from this filter then goes to an 8-bit d-a converter that produces an analog speech waveform. These synthesizing functions are directed by very few control lines (Fig. 3).

Synthesizer needs minimum control

The 5220 needs minimal control from a host processor. The CPU passes commands to the 5220 only to initiate specific activities; the CPU does not involve itself in carrying out the activity. The command structure is simple—just six operational commands make up the total list (Table 1).

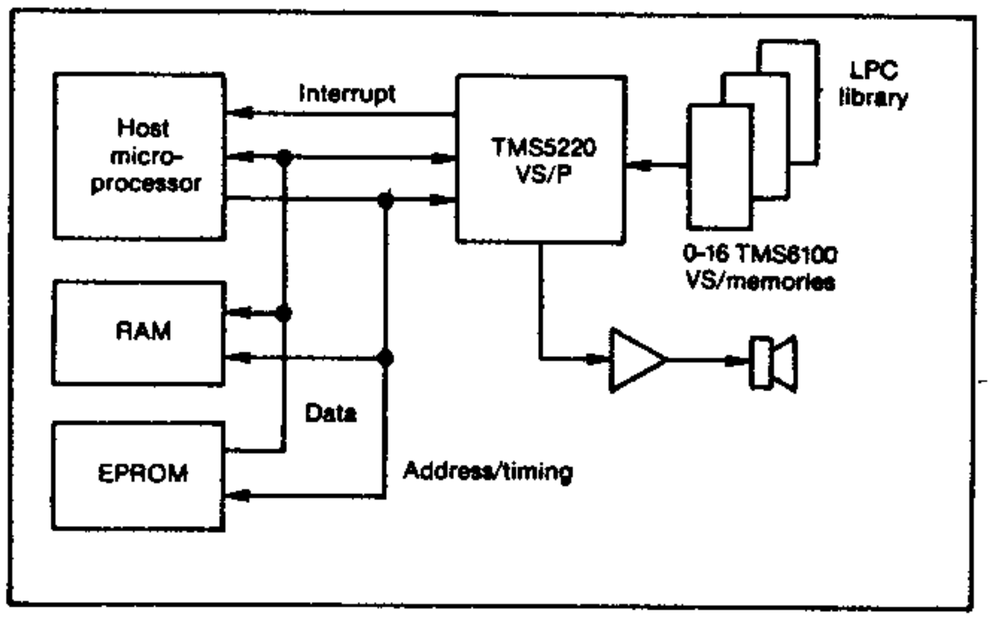
The host processor selects the desired word or phrase; locates the starting address of the data for that word; passes the command (such as Speak External) to the 5220; and finally, sends the required data from memory to the 5220 under program control.

A TTL-compatible 8-bit bidirectional bus (D_0 to D_7) in the 5220 (Fig. 3) makes the chip simple to interface. (The 5220 operates on ±5 V.) Moreover, the host processor can service the 5220 by a polling operation, by monitoring the status of the 5220, or by responding to interrupt-service requests from the 5220.

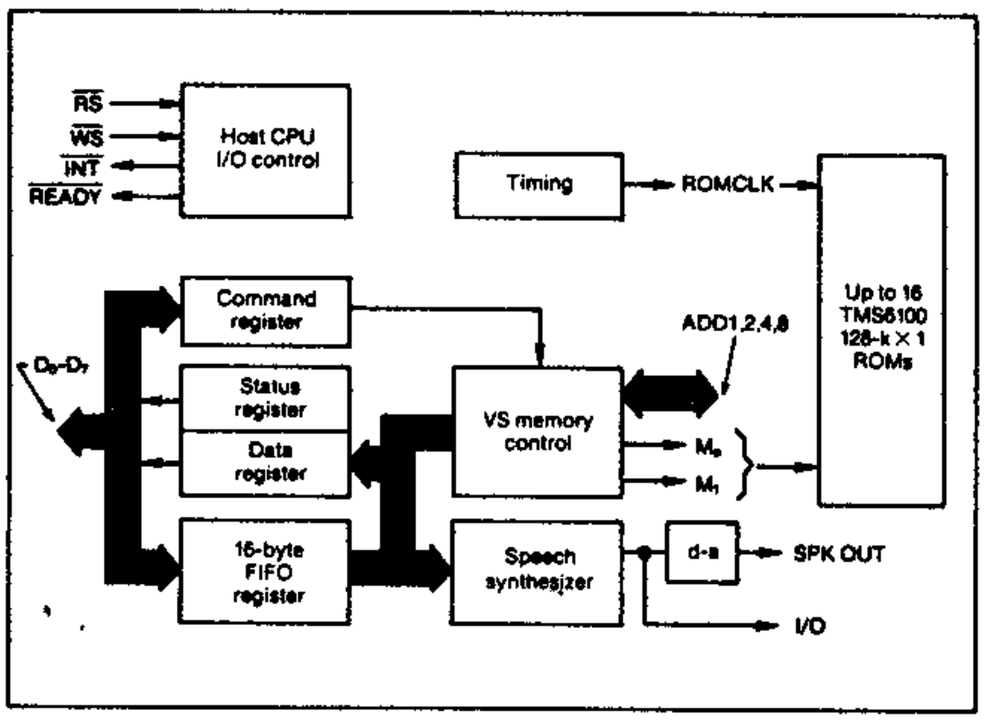
Four 5220 on-chip registers handle all the input/output data: A 128-bit FIFO buffer register and a command register receive inputs; a data register and a status register hold outputs. When the (Write Select) WS line goes low, inputs are directed either to the FIFO buffer for a Speak External command or to the command register (in all other cases). Once data are latched in a register, the 5220 lowers its READY line (active low) to signal the host processor that the data transfer is complete, releasing the processor for other work.

A new command that enters the 5220 while a previous command is still being executed will not be accepted; a READY-high flag forces the host microcomputer to execute wait states, unless the processor can be engaged in other tasks.

The 5220's on-chip FIFO buffer holds 16 bytes of speech data, which is about 50 ms (minimum), or two full frames, of speech sound (when operated at a nominal system-clock rate of 160 kHz). If the FIFO buffer's contents fall to less than 8 bytes, the chip's



2. The TMS5220 can support an extensive LPC library containing as many as sixteen TMS6100 mask-programmable 128-k × 1 ROMs.



3. The speech-synthesis processor (TM5220) includes a 16-byte FIFO input-data register, an input-command register, and output-data and output-status registers. Timing, host-μP I/O control, and memory-input control complete the chip's logic system.

control logic can generate an interrupt to the host processor indicating that more data are needed. Or, a bit-low status register is activated, which can signal the host processor about the need for data in a polling operation. Should speech output cease, the talk-status register clears, informing the host process that the next command is expected.

Since data enter the FIFO buffer via the data bus (D_0 to D_7), the address lines (ADD_1 to ADD_8) are available in this mode for stacking up several addresses. This design eliminates the need for stop codes; thereby, allophonic speech segments can proceed unhindered through the synthesizer and provide clearer voice signals.

The typical processor time for supplying 8 bytes to the 5220 FIFO (after an interrupt) is less than 500 μs. At 20 interrupts/s, typically less than 1% of the host processor's available time is needed to

Allophones—from basic sounds to complete words

Among human-speech components, allophones are more fundamental than any of the other linguistic components, including phonemes, diphones, demisyllables, and morphs. In modern speech-synthesis, each phoneme (generally about 64 total) encompasses a family of closely related sound characteristics called allophones, which modify the basic phonemic sounds.

A phonemic analysis of English speech shows that about 40 allophonic sound characteristics can provide the needed variations for all phonemes. For example, the phoneme for the letter "p" in English is rounded and aspirated in the word *poke;* rounded and unaspirated in *spoke;* aspirated in *pie;* unaspirated in *spy;* slightly aspirated in *taper;* released in *appetite;* and unreleased in *apt.* These accoustically different "p's"—so called voiceless bilabial stops—are allophonic variations of the phoneme "p."

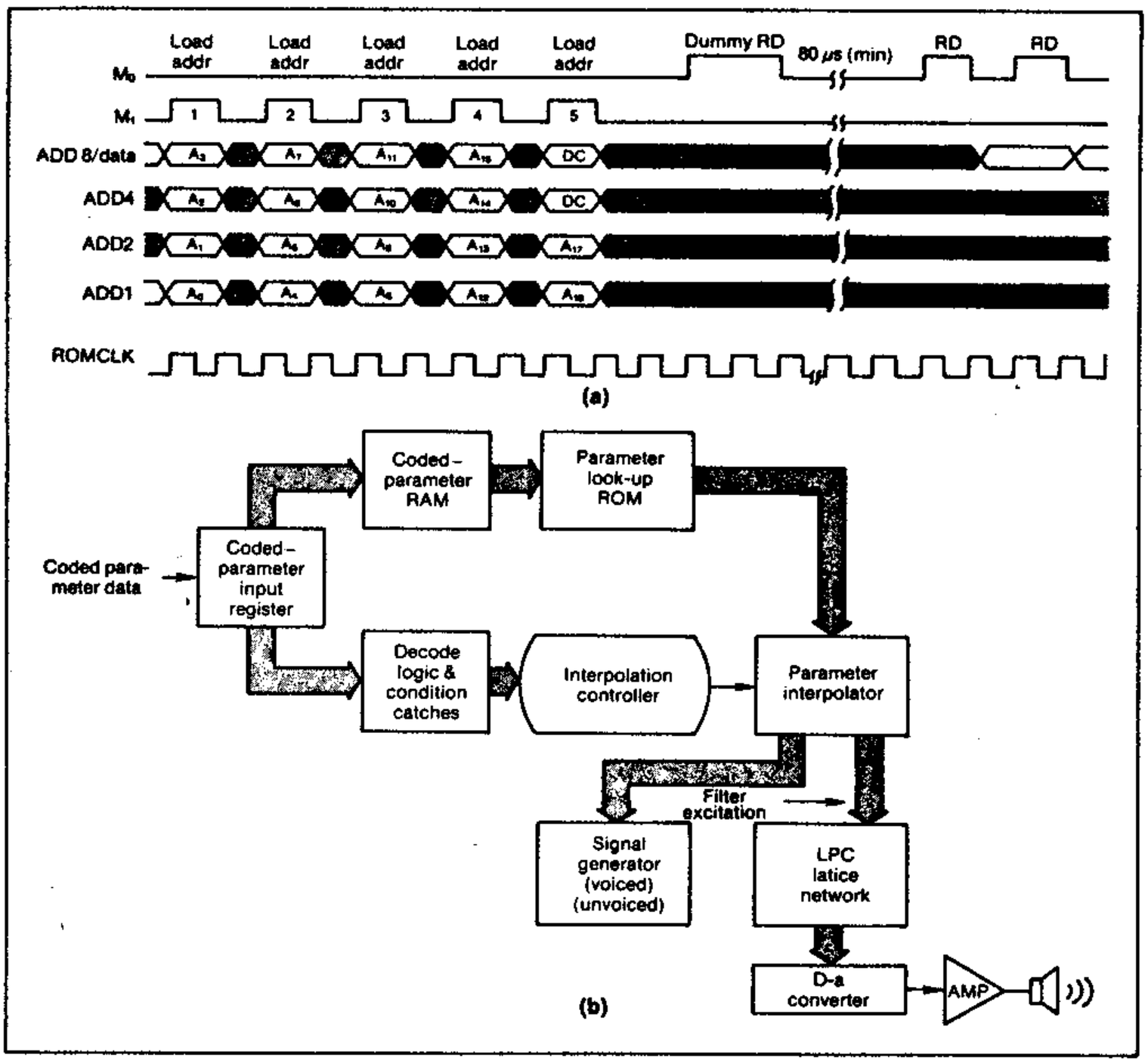
Thus, allophonic speech synthesis produces better quality than phonemics, because the allophones provide most of the subtle variations each English phoneme can encompass, and use each variation in the appropriate relationship. Although better than the phoneme approach, allophonic voice quality still tends to sound somewhat mechanical. Some speech systems, like Lingua and Mitalk, employ more complex diphone, demisyllable, or morph approaches for higher quality speech, but at a considerable increase in memory size. A morph system, for example, requires the storage of at least 12,000 individual units of speech—root words and morphemic affixes—which translate into 600 kbytes of data.

However, the allophonic-speech system requires less memory storage. It also needs less storage than other more widely-used encoding systems, such as the direct use of compressed-speech LPC, pulse-code-modulation (PCM), delta-modulation (DM), and formant-synthesis methods.

One way of implementing the allophonic system is with a table listing all the allophones and their phonetic symbols (the International Phonetic Alphabet), from which a programmer makes selections for stringing the allophones together to form words—a rather tedious and difficult task. TI, however, will provide a list of formulae and software for the English-language rules for generating most common English words with the help of a personal computer.

But the number and variety of English-language rules makes any simplified technique less than perfect. Again, TI has a solution: a selection of software modules that are graded according to accuracy. The least expensive module will provide 90% accuracy; then, the user can correct the erroneous 10%. For greater initial accuracy—entailing a greater cost and more memory—the software would support more rules and a larger vocabulary.

But vocabulary and rules are not all that an allophonics system must contend with. A key element in the allophonics speech-construction program is "smoothing" between allophones and adding prosody (tone or syllable accents) before transmitting the resulting LPC data to the voice-synthesis chip. An important technique for obtaining the perception of good speech quality is to provide a smooth energy contour for the joined phrases. So-called interpolation frames, created at both ends of a string, gradually taper the sound energy towards zero. Tapering, or interpolation, reduces abrupt sound changes that otherwise would be perceived as pops, squeaks, or squeals. Another technique is the TI-supplied speech-construction program. It includes stress and intonation patterns, which improve understandability and give a more natural speech output. Interpolation, and the stress and intonation patterns, go a long way towards making the quality of speech less mechanical sounding than with the phenome approach.



4. Coded parameter data from an external PROM (or the FIFO) are supplied serially to the speech-synthesizer block input (coded-parameter RAM) by pulsing line M_0 (a). Most important are the details of the speech-synthesis block, which emulates the human vocal system (b). The synthesizer block contains a d-a converter that provides an analog output.

provide speech control, following a Speak External command.

The Speak External command is particularly important, because it initiates the allophone-stringing mode. (The LSP mode is initiated by the Speak command.) After first clearing the FIFO buffer, the 5220 waits for the host microprocessor to load a minimum of 9 bytes of data into the buffer, before setting the Talk Status register and starting speech-synthesis calculations with the data. Calculations continue until either a stop code is encountered, or a buffer-empty termination occurs.

Speak Initiates LPC

The Speak command for the LPC mode initiates speech from phrase data stored in an external ROM. Upon this command, an internal signal sets the Talk Status immediately and initiates speech-synthesis calculations with the next available data from the ROM. Audio output begins on the next frame boundary (26 ms/frame) and continues until a stop code (energy = 1111) is received, at which point the audio output starts interpolating to a zero-energy level. Then, speech ceases on the next frame boundary; the Talk Status is cleared; and execution of the Speak command is complete.

For a read-out cycle, ($\overline{\text{Read Select}}$) $\overline{\text{RS}}$ goes low and the 5220 transfers its data-register contents to the data bus, as long as the preceding input was a $\overline{\text{WS}}$ cycle. The 5220 transfers its status-register contents to the data bus in all other cases.

The 5200 can access up to 16 mask-programmable, parallel-connected 128-kbits × 1, TMS6100 ROMs, without additional hardware, with its 4-bit, parallel bus (ADD_1, ADD_2, ADD_4, ADD_8), its M_0 and M_1 control lines, and a synchronized clock (ROMCLK).

The 6100 memory requires 20-bits of address: 14 bits to select a byte in memory; four chip selects; and 2 bits that are ignored. Addresses enter via ADD_1 through ADD_8 in five load-address sequences (Fig. 4a). Data are read out in a serial operation, actuated by toggling control line M_0. An on-chip address counter for the ROM increments every eight toggles of M_0. Four internal chip-selects are mask-programmable options for selecting among the parallel-connected ROMs.

Speech synthesis

In the speech-synthesizer block of the chip (shown in Fig. 3 and detailed in Fig. 4b), the LPC parameters of speech feed serially from either the external ROM or the FIFO buffer into an input register. Here, the data are "unpacked" and several tests performed to determine whether the repeat bit is set, the pitch is zero, or the energy is zero. The unpacked data then are stored in the Coded-Parameter RAM and serve as index values for selecting appropriate values from the Look-up ROM.

Outputs from the Look-up ROM are target values that the interpolation logic must reach in one 26-ms frame period, which is composed of eight 3.25-ms interpolation intervals. During each of these interpolation intervals, the interpolation logic generates pitch and energy parameters for the tone-signal generator, as well as the filter-excitation sequence and reflection-parameter values for the LPC Lattice Network. In each sample period, a different sound component of the digitized, synthetic speech is created, according to the words, pitch, and speech pattern desired. An 8-bit, 2%-linearity, d-a converter inside the chip provides an analog output.

In most LPC speech synthesizers (including the 5220), 12 parameters—the pitch and energy levels and ten so-called reflection coefficients (K_1 through K_{10})—establish the quality and range of the synthesis (Table 2). Taken altogether, the 247 levels of the parameters are coded with 49 bits, plus an extra bit for a repeat operation, to total 50 code bits that must be specified to the voice synthesizer.

For the 5220, however, TI provides firmware that makes the ten reflection coefficients pitch-dependent at the high end of the audio-frequency range. In addition, the first 16 tones, from 262 Hz to about 4 kHz, constitute a tempered C-major scale; the second 16 tones, which overlap the first, from about 700 Hz to 4 kHz, are monotonic—all new features that improve the speech quality.

The most-significant ten bits (of the 14 bits) of the lattice-filter-network output are sampled every 125 μs. The seven low-order bits determine the analog-output level of the converter; the sign bit (most-significant) and the next two bits are combined logically and used to clip the driver to either full-on or full-off (with appropriate polarity).

The converter output delivers up to 1.5 mA (with a resolution of 5.9 μA), which is enough to drive an audio amplifier, such as TI's SN76489AN that can power an ac-coupled speaker.□

Synthetic speech can be generated with a constructive synthesis or an analysis/synthesis approach. The technology of each now allows practical tradeoffs of quality and cost against simplicity.

Phonemes, allophones, and LPC team to synthesize speech

This article is part two in a series of articles on speech synthesis and recognition. Part one appeared in the June 11 issue (p. 213).

Merely enabling some devices like computers, games, and appliances to respond intelligibly with human-like speech can greatly enhance their appeal and even their value. But after the novelty wears off, what the device can say (vocabulary), how well it speaks (quality), and how much it costs (hardware and software complexity) will separate the truly value-adding synthetic-voice design from the flash-in-the-pan gimmick. The technology of speech synthesis has now advanced sufficiently to enable system designers to make practical tradeoffs.

Synthetic speech can be generated by two basic methods:

- Constructive synthesis (by rule)
- Analysis/synthesis.

Each approach offers different benefits and drawbacks; accordingly, special forms of each method, or combinations of both, must be selected to provide the best solution for particular applications.

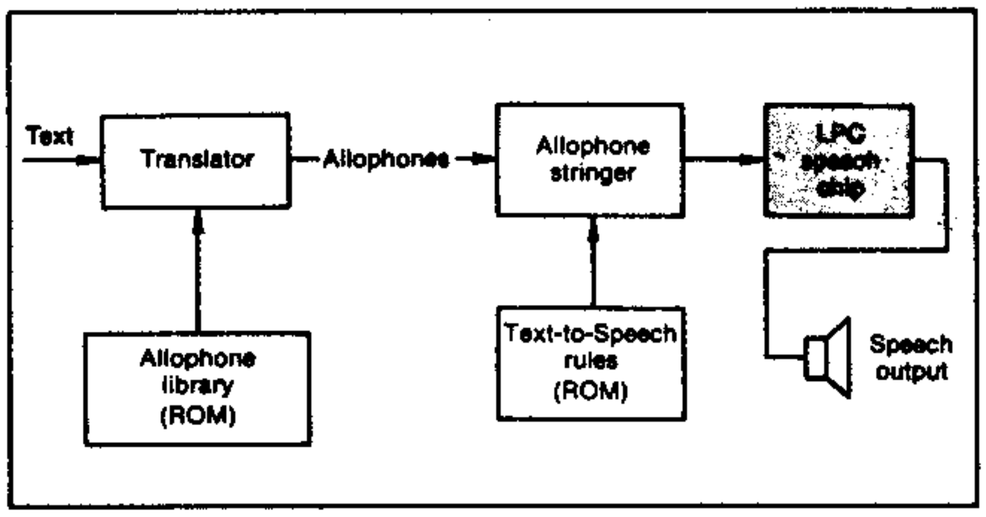
Constructive synthesis builds words from a prescribed set of linguistic or phonetic sound segments, such as phonemes. Each language has its own set of such sound segments. Phonemes include speech characteristics, such as voicing and manner. Vibrating vocal cords produce voiced phonemes, which in English include all the vowels and 11 consonants. Eight other consonants, produced without the vocal chords vibrating, are called unvoiced phonemes.

Manner is determined by a complete or partial physical closure at some point in the speech mechanism. Full closure results in a stop, or a plosive, phoneme; partial closures produce a fricative.

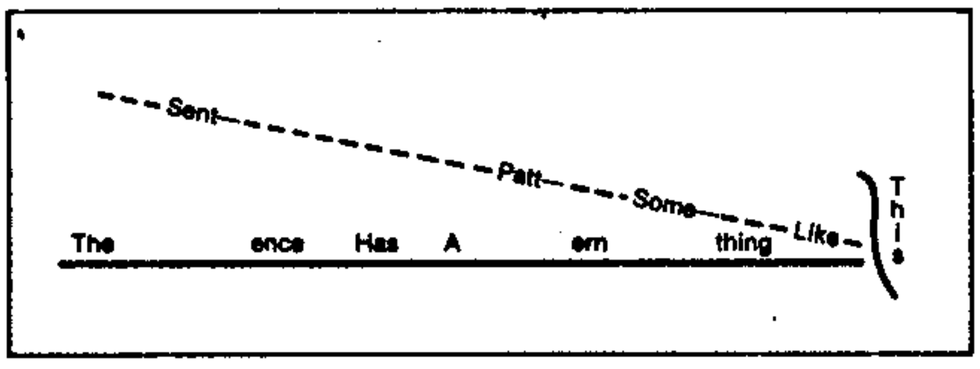
Win Smith, Manager of Advanced Product Planning
Sharon B. Crook, Manager of Speech Strategic Marketing
Texas Instruments, Inc.
P.O. Box 6448, Midland, TX 79701

Phonemes emphasize "alphabet" simplicity over speech quality. The American language can be reproduced from just 40 to 50 phonemes and an appropriate set of rules, which consume a minimal amount of memory. Thus, systems with extensive vocabularies are inexpensive. However, sound variations that depend on where in the word a given sound appears and what sounds precede and follow it, are ignored. Accordingly, the speech sounds mechanical.

Because analysis/synthesis systems derive their vocabularies from actual human speech, rather than



1. A ROM holds the Text-to-Speech rules for the contents of the allophone-library ROM, and software properly strings and interfaces the allophones and stresses the phrases. A personal computer then synthesizes speech with the aid of an LPC speech chip.



2. The naturalness of a sentence can be enhanced by stressing key parts of words. A statement generally has a descending-mode pitch, as shown; a question has a rising-mode pitch.

from piecemeal approximations and rules (as with phonemes), the speech derived with analysis/synthesis more closely resembles real speech in inflection, emotion, quality, and intelligibility. The "vocabulary" can be in the form of words, phrases, or complete sentences—but in all cases, the speech is derived from human speakers. Then, from the real speech waveforms, the analysis/synthesis system extracts data—and encodes it, as in the now popular linear predictive code (LPC)—with which the waveforms can be reconstructed at another time and place. Consequently, memory-storage requirements generally are large for extensive vocabularies. To keep memory within somewhat modest sizes, vocabularies in analysis/synthesis systems must be very limited.

However, TI has done extensive work on LPC, which significantly reduces the amount of needed memory storage for a "standard" LPC system. Nevertheless, phonome (and allophone) based systems still need considerably less memory than LPC systems.

Phoneme speech can be improved

Mechanical-sounding phoneme-based speech can be enriched with allophones—subdivisions of phonemes—given a modestly sized memory. Allophones can modify phonemes to represent the sound of natural speech more closely. A basic sound —for example, the phoneme /p/—can be modified by allophones in a rounded and aspirated manner, as in "poke"; rounded and unaspirated, as in "spoke"; aspirated only, as in "pie"; unaspirated, as in "spy"; slightly aspirated, as in "tapir"; released, as in "appetite"; or unreleased, as in "apt." Each of these English words has in common a voiceless bilabial stop called "p;" yet acoustically, each is distinct.

The price for better speech quality is the greater memory space required by the rules for allophones when compared with simple phonemic systems. Nevertheless, allophones are efficient for large vocabularies.

Even though allophonic speech has better quality than simple phonemic speech, phonomes modified with allophones are still less than perfect. Transitions between allophones make the speech sound unnatural, and intonations are characteristically monotonic. Yet, an allophonic-speech synthesizer offers a good compromise among many factors, including size of vocabulary, memory storage, quality of speech, versatility and flexibility, hardware and software complexity, and cost.

Moreover, an allophonic system lends itself to translation from American-English text with a reasonably small set of rules. TI has developed a Text-to-Speech system that can operate on a 99/4 home-computer system. It takes advanced allophonic-based speech synthesis into new territory. Input text, say from an ASCII keyboard, is automatically converted into the appropriate allophones, which are then converted into linear-predictive-coded (LPC) data that can activate a TMS5220 speech-synthesis chip to generate speech immediately (Fig. 1). A library of 128 allophones needs a modest 3-kbyte memory; and a set of 650 rules to manage them involves another 7-kbyte memory.

TI's text-to-phoneme rules are an augmented version of work done at the U.S. Naval Research Laboratory.[1] Rules added to the basic Navy list ensure that word-ending allophones properly terminate appropriate words; that longer vowels stress the monosyllables before voiced consonants; and that unstressed vowels are properly dispatched before suffixes such as "er," "ely," and "es" (after t and d). The resulting system can derive phonemes, or rather allophone strings, that are 97% correct on only one pass for the 2000 most commonly used words in the American-English language. When the frequency of word usage is taken into account, the correctness drops to about 92%.

The high degree of accuracy is attained by defining each text character string to be translated as it relates to the character strings on either side of it, as follows:

$$A\ [B]\ C = /D/$$

where, if B is the English-text character-string to be translated, A and C are the character-string environment descriptions to B's left and right. The allophone string /D/, then, represents the allophone-string translation of character-string B.

For example, in the process of translating the word "space," the allophone-stringing algorithm looks first at the 's' and supplies an initial allophone for /s/. But for the 'p,' it finds a rule where the left environment is an 's.' Also, since the 'p' is not a final sound, the algorithm translates the 'p' accordingly. Next, the rule is invoked that applies to an 'a,' where the right-side environment consists of a single consonant and the word ends with a word-final silent 'e.' This rule selects the appropriate "long-a" allophone. Finally, another rule for 'ce' inserts an /s/ component in the allophone string to replace the 'c' in the text; the rule also says that the 'e' is silent.

Two types of problems occur

For most words, the rules supply the correct phonemic and allophonic outputs. However, because of the complexity of the English language (which is a composite of words from several different language), two types of problems arise for the remaining words.

First, the rules can deliver the correct allophone,

but representative of the wrong phoneme, because of irregularities in the correspondence of English spelling to pronounciation. The pronounciation of words that begin with "pro" exemplifies this kind of error. "Progression," "to program," and "to make progress" represent three different ways of pronouncing "o." But because the rules cannot differentiate among the different environments in these words (the 'o' is between an "r" and "g" in each case), the system pronounces each of them with a "prah," and thus is "correct" only in the word "progress."

Second, the rules can deliver the wrong allophone, although for the right phoneme, because of irregularities in English-language stress patterns. The error most often encountered is the delivery of a strong vowel in what should be an unstressed position—as in the word "minister"—which should be pronounced with the second "i" reduced.

Compound words also give the system problems. The system often fails to recognize a morph boundary and tries to pronounce the silent "e," as in the word "baseball." Hyphenating such words can correct the problem (base-ball). Similarly, deliberately misspelling a word can correct a rule-imposed mispronounciation. For instance, the homonym "led" may be used to obtain a correct pronounciation of the metal "lead," instead of the verb "to lead." Similarly, the name "David" may be spelled "Davvid," so that the system says the first syllable as "Dave" instead of "Dav." Almost any word mispronounced by the rules can be misspelled in some way to produce an acceptable pronounciation.

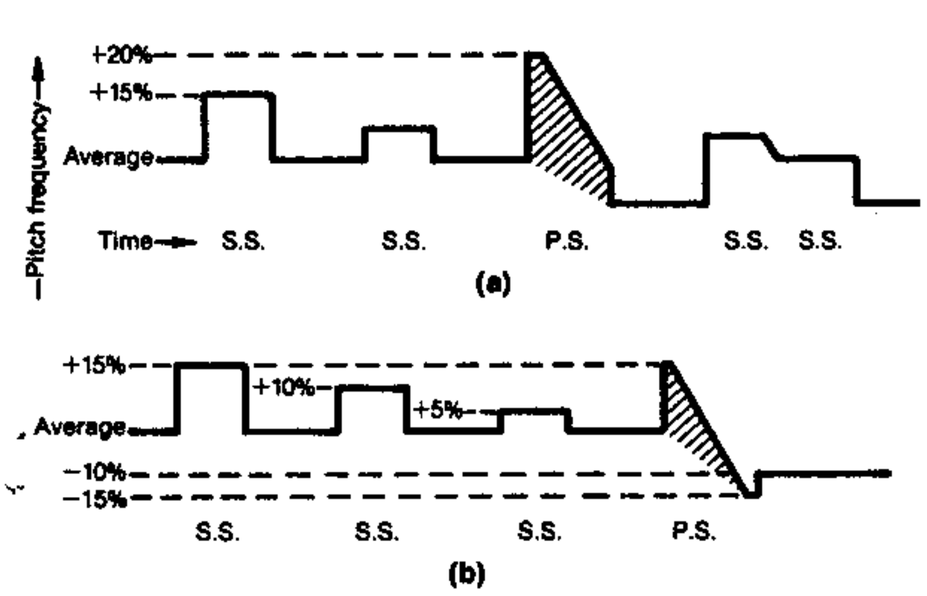
Contouring algorithms add stress

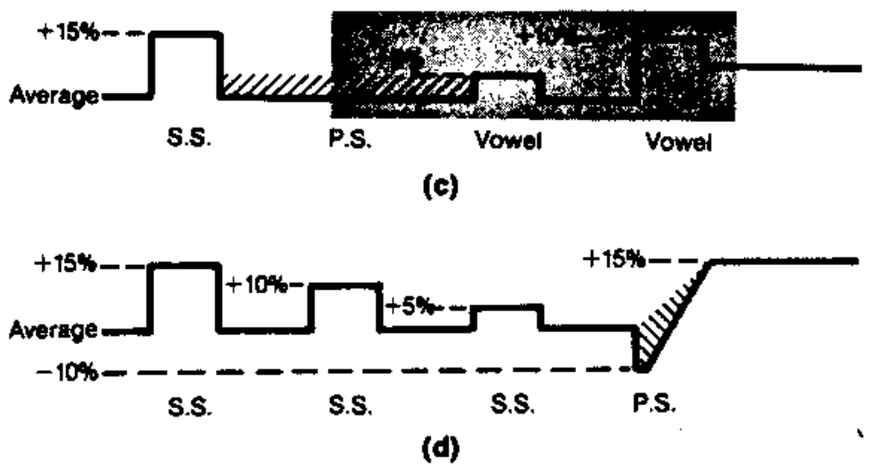
TI's Text-to-Speech (TTS) system translates secondary and primary speech-stress points into pitch variations, according to predefined rules. Contouring algorithms divide sentences into two major stress-profile types:

- Falling-phrase mode
- Rising-phrase mode.

Sentences in which phrase modes rise typically terminate in question marks. TTS currently stresses only vowel allophones. Future profiling algorithms probably will be broadened to stress additional allophones.

In falling phrases, the pitch level drops following a primary-stress (P.S.) point. When one or more secondary-stress (S.S.) points follow a falling primary point, the contouring algorithm drops the pitch from the primary-stress-point high of 20% above the average pitch level until it nearly matches the average pitch level (Fig. A). However, when the primary-stress point is in the last phrase, the algorithm drops the pitch from about 10% above to about 15% below the average level (Fig. B). Also, as in Fig. A and B, just after a primary stress, the algorithm tends to settle at a 10% below-average pitch level, eventually.





Secondary-stress points in the falling-phase mode are treated exactly the same, whether they occur before or after a primary stress. The algorithm always positions the first secondary-stress point 15% above the average level and spreads out additional secondary-stress points evenly—between the 15% rise and the average level.

Like the falling-phrase mode, the rising-phrase algorithm also centers around primary-stress points, except that the pitch contours rise. The rising-mode contouring algorithm also differs in that the rising contour spreads itself out over all the vowels following a primary-stress point. For example in Fig. C, a rise starts from the average level, which serves as the primary-stress point; and the contour swings up gradually—starting at 10% below and rising to 15% above the average (Fig. D).

Another category of problems is the smoothing of the transitions between allophone strings. Stringing together allophones (or even their appropriate phonemes) to make words generally produces unnatural, mechanical-sounding, or choppy forms of speech. To smooth out the transitions between adjacent phonemes, another speech-sound part, called the diphone, could be used.

Diphones could smooth transitions

Diphones are sounds that extend from the center of a phoneme to the center of the next phoneme. However, diphones add heavily to the storage requirements and complexity of a speech-synthesis system. Also, English is rife with unique coarticulation effects that require the allophones to be modified relative to both the preceding and the succeeding sounds. Hence, a diphone-synthesis system must provide storage for about 2650 diphones, with a formidable 236 kbytes of memory (one such machine was developed by Bolt Beranek and Newman, Inc. of Cambridge, MA). Moreover, the program to put the diphones together into high-quality speech will not fit into a simple microprocessor—a mainframe is needed.

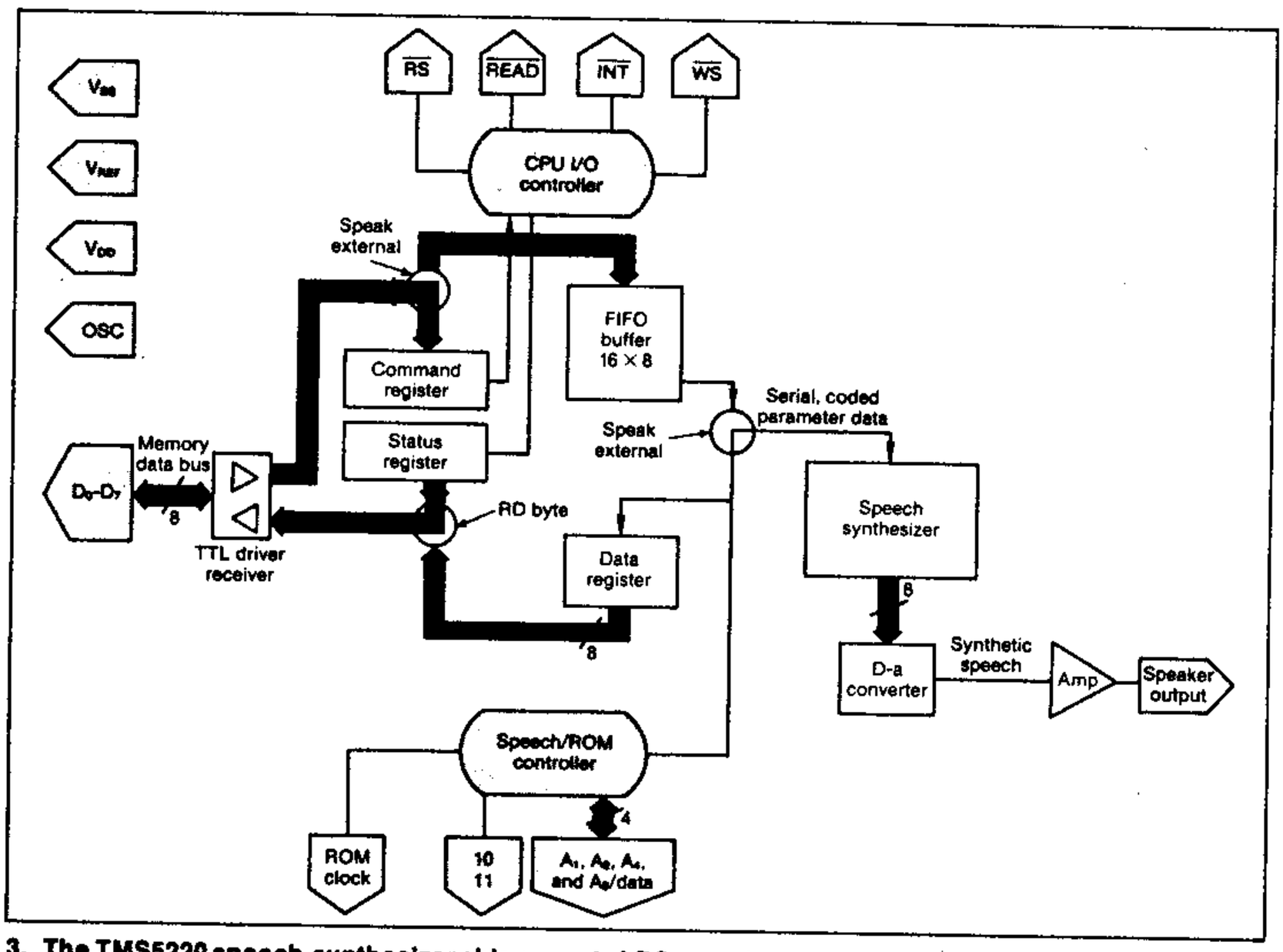
For even better speech quality, a system based on yet another set of speech parts—morphs—provides very natural sounding, very intelligible results. Morphs, the smallest units of sound that can convey meaning, consist of root words and phonetic affixes and suffixes. About 12,000 morphs are needed to synthesize the English language, and 600 kbytes of memory are needed to store them.

In another constructive-synthesis approach, involving demisyllables, initial and final half-syllables and phonetic affixes are stored. This method breaks English into about 800 units that require 50 kbytes of memory space.

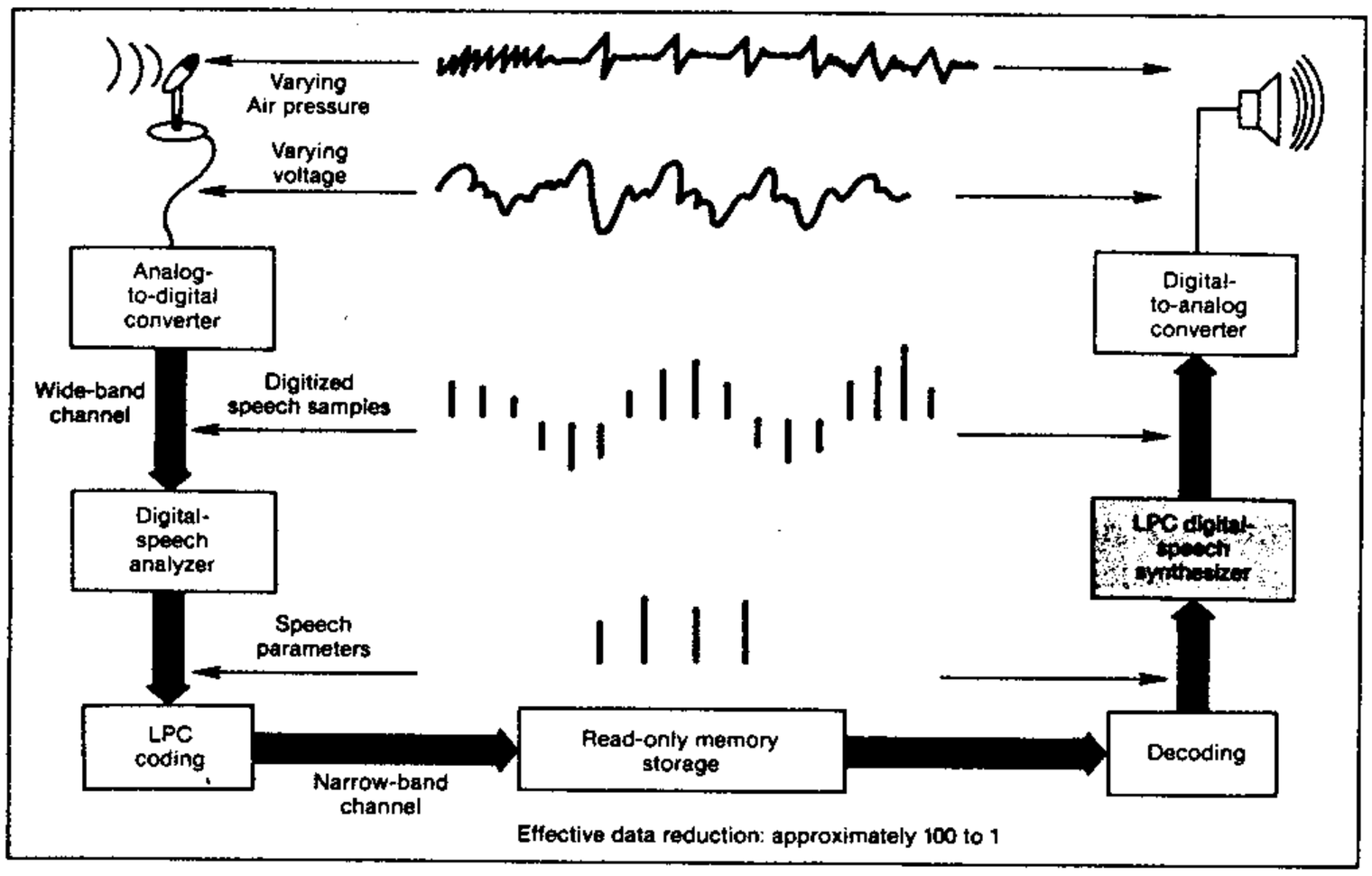
However, TI's Text-to-Speech system for the TM 99/4 home computer takes yet another approach to smooth the allophone transitions, so that otherwise mechanical-sounding allophone strings become more natural sounding intonation patterns, with a minimum of memory and complexity.

The TI approach is similar to the phoneme-based schemes found in such systems as the Votrax VS-6 and ML-1. English text, entered via an ASCII-coded keyboard, is immediately translated into allophones by a set of prestored text-to-allophone rules. A speech-construction program then operates on the concatenated prestored allophones, converting them into LPC parameters, from which the TMS5200 speech-synthesizer can generate speech with an almost unlimited vocabulary.

The speech-construction program is at the heart of the Text-to-Speech system. In addition to stringing together appropriate allophones, the program smooths the allophone transitions and adds prosody



3. The TMS5220 speech-synthesizer chip accepts LPC-type data at its bidirection input bus (D_0 to D_7) and delivers synthetic voice signals at its output.



4. LPC coding is very effective for compressing speech data. Because its compression ratio is approximately 100:1, LPC coding demands less memory storage and a narrower data-transmission bandwidth—requirements that are readily handled in inexpensive systems.

to the allophones before finally delivering LPC data to the synthesizer. Hence, much of the quality of the speech produced depends on the effectiveness of the program.

Another parameter—energy—also must be smoothed out. When required, the speech-construction program generates an interpolation frame with energies tapered toward zero at both ends of a string. This smoothing action helps reduce otherwise abrupt sound changes that would come through the system as pops, squeaks, squeals, or other annoying sounds.

But above all, in the struggle against mechanical sounds, rules for stress and intonation are included in the program, because they contribute heavily to both the naturalness and intelligibility of speech. Stress is emphasis placed on a certain syllable within a word; intonation covers a longer time frame, embracing the up-down pitch patterns within a long phrase or a sentence.

Randomly intoned (or nonintoned) English sounds very unnatural. Since most phoneme synthesizers draw on a very limited choice of pitch levels, the resulting speech sounds very mechanical. However, the stress and intonation patterns in the Text-to-Speech system are not limited to specific choices. They are based on a gradient-pitch control of the stressed syllables that follows the primary stress of the phrase. All the secondarily stressed syllables of a sentence can be thought of as lying along a descending line of pitch values relative to the unstressed syllables (Fig. 2).

To achieve proper intonation, the user marks only stressed syllables while entering text. The stressed syllables then become the anchor points of the pitch pattern; the program automatically assigns appropriate pitch values to the allophones the syllables are translated into.

In this way, the LPC parameters finally supplied to a TMS5220 speech-synthesizer chip (Fig. 3) have been smoothed between parameters and adjusted in pitch for stress and intonation (see "Contouring Algorithms Add Stress").

LPC data not the only way

But LPC is not the only way speech could have been digitized and encoded—although, because it compresses speech information about 100 to 1, it is highly efficient, requiring a minimum amount of memory when stored (as compared with other analysis/synthesis methods) and narrow-bandwidth channels for transmission (Fig. 4).

In basic pulse-code modulation (PCM), for instance, sampled amplitudes of an analog signal are converted to wideband binary numbers. Basic PCM requires more memory and wider channels; but encoding can be modified to compress data logarithmically (by the so-called μ or A laws), though not as efficiently as in LPC.

Basic delta modulation, which converts each sampled analog amplitude to a change relative to the value of the previous amplitude sample also requires a wide bandwidth. A variation, called variable-sloped-delta (VSD) modulation, can compress the data to some extent, but again LPC wins out easily. Some methods combine the noncompressing basic forms of PCM and delta modulation with data-

compression techniques that require prior knowledge of speech-signal characteristics. Still, none prove to be more efficient than LPC.

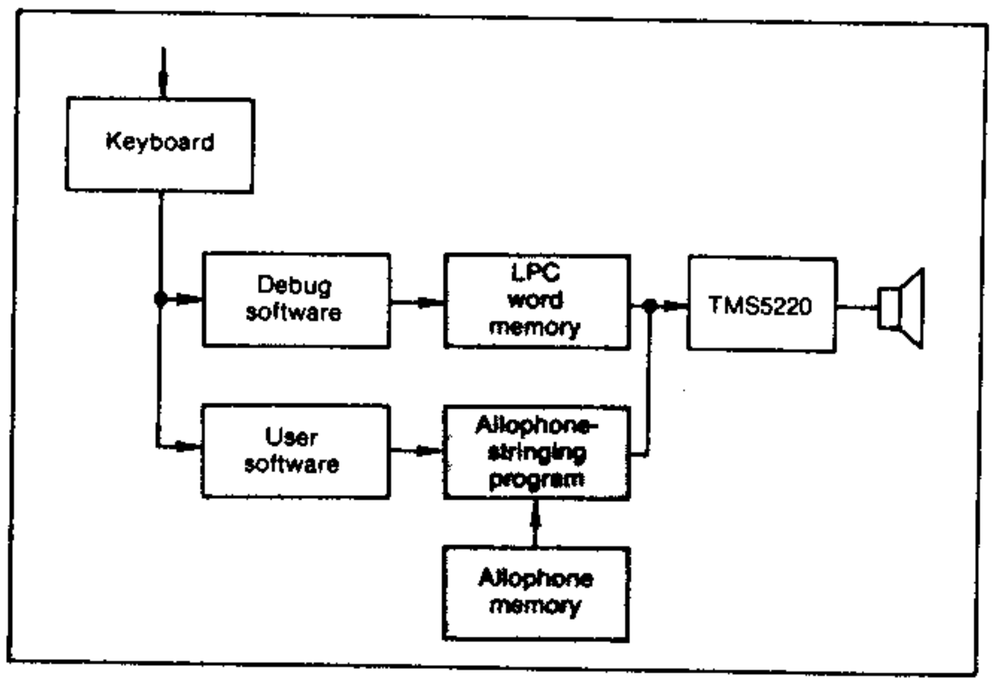
Basic PCM's data-rate requirement for good fidelity ranges from 64 to 96 kbits/s; the typical sampling rate required by basic delta modulation is not much better at 64 kbits/s; and VSD reduces this somewhat to beween 32 and 16 kbits/s. The best that even the most sophisticated, complex, and expensive waveform-encoding systems have achieved is in the range of 2.4 kbits/s.

However, all the parameter-encoding methods (like LPC) are less expensive and feature much lower data rates than waveform encoding, because fewer speech characteristics, or parameters, are sufficient to encode speech. The parameter codes synthesize speech with a digital model of the human vocal system, instead of merely reproducing the original waveform from the codes.

Of the parameter-encoding methods other than LPC, channel vocoding analyzes the spectral content of speech signals within a set of bandpass filters; and formant synthesis follows peaks in the speech signal's spectrum. Like LPC, both of these methods can compress speech data dramatically, but implementing them requires extensive processing power with algorithms that are far from perfected, so that considerable hand editing is needed. (Hand editing of the LPC type of results obtained from allophone stringing applies to just 8% of the least sophisticated allophone-stringing rule package. More advanced rule packages will provide even higher accuracies.)

Unlike these other parameter-encoding methods, LPC permits the implementation of an effective voice synthesizer (human vocal-tract model) with available LSI technology on a single chip (the TMS5220). The model can be described as a linear time-varying system excited by quasi-periodic pulses (for voiced speech) or random noise (for unvoiced speech), both of which are controlled by parameters derived from speech signals. In other words, speech parameters coded in LPC, based on the speech signal's energy, pitch, and voiced or unvoiced enunciation, control the bandpass of a variable filter, which emulates the transfer function of the human vocal tract. The filter is excited by tones or noise signals.

In TI's LPC-10 speech-synthesis chips (TMS5100, 5200, 5220), then, so-called K-parameter reflection coefficients control the shape of a ten-pole filter that models the vocal tract. Not only is all the speech-synthesis circuitry on a single chip, but the data rate is just 1.2 to 2.4 kbits/s—on par or better than that of more complex systems. The quality is high enough to accurately generate single words that are clearly understandable, even when out of context.



5. In a combination standard LPC and allophone-stringing approach, the LPC is dedicated to generating highly intelligible single words, and the allophone to generating phrases and sentences where the context improves intelligibility.

In addition, speech quality can be upgraded without raising the LPC data rate. The methods include:

- Updating LPC prediction parameters more often—commonly 30 to 100 times/s, but 200 times/s at most
- Increasing the number of quantization levels for each parameter—usually 2 to 6 bits per parameter
- Increasing the digital filter's order to 12 poles (10 poles in the 5100)—more than 12 is superfluous
- Extending hand editing of the input and output data.

Of course, those methods depending on changes in the LPC synthesizer chip cannot be implemented by the user.

LPC library available

One item the user need not worry about implementing is an LPC library, even for private vocabularies. TI has assembled codes for a large number of popular words into a "standard" library. Any special grouping of these words into phrases and sentences can be delivered on EPROMs (in small quantity) and on ROMs (in large runs). In addition, groups of widely applicable words, phrases, and sentences (such as those common in industrial-control applications) will soon be available on off-the-shelf ROMs. TI is able to develop custom vocabularies of as yet unrecorded words or phrases, even with special accents, dialects, or inflections. The speech of a celebrity (or historical person) can be synthesized.

In another application, an LPC library can endow a computer with the ability to deliver voice prompts and error messages that facilitate debugging and other interactive interfacing with the computer. Because such messages may be highly standardized and because intelligibility is important, especially

during the stress of a program's development stage, a high-quality LPC approach is a prime candidate for the speech-synthesizing system. But a pure LPC system would need a very large memory to be widely applicable and truly useful, regardless of the standardization of messages.

However, just a modest amount of memory suffices to store allophonic sound segments, the rules for concatenating and combining them with selected whole LPC words. From these elements, application programmers can readily piece together messages of their choice, selecting from among the allophonic or whole LPC words. Such a combination system even could include accents and dialects to appeal to people from particular areas of the country.

Figure 5 shows the block diagram of a typical speech-prompting subsystem implemented with a TMS5220 speech-synthesizer chip. When the software algorithm is keyed into the host computer, whole words in standard LPC can be selected for the most important messages, where intelligibility must not be compromised or where the operator would be made uncomfortable by a lack of clarity. In particular, short phrases or single-word messages should be generated with standard LPC words, because they are the most difficult to recognize without the clues inherent in context.

At less critical points in the program, especially for familiar phrases and sentences, the computer could access the allophone-stringing program. Allophone quality should suffice here, because the instructions are in context and relate to whatever the user may be involved in.

An aural interface vastly improves the interactive capability of an operator, who may be heavily involved in mechanical and visual processes, such as flying an airplane or operating a machine-tool. These applications will be the key to the spread of synthetic-speech interfacing.

A typical system could be put together in stages. First, an LPC memory would provide the essential words and phrases. Later, when user needs were clarified, an allophone memory and the required software could be added with a minimal amount of linking, almost without changing any of the previous programming. □

Reference

1. Elovitz, H.S.; Johnson, R.W.; McHugh, R.W.; and Shore, J.E., "Letter-to-Sound Rules for Automatic Translation of English Text to Phonetics," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-24, No. 6, December 1976.

Digital signal-processing techniques replace analog methods in the development of low-cost speech-recognition systems. But the ideal of a single-chip processor awaits future breakthroughs.

Speech recognition spurred by speech-synthesis success

The following article continues a series on speech synthesis and recognition. Part one appeared in the June 11 issue (p. 213), and part two in the June 25 issue (p. 121). A look at the future of speech technology was given by Bernard H. List, vice president for MOS functions at Texas Instruments' Speech Technology Center (May 28, p. 35).

Recent breakthroughs in synthetic speech, based on linear predictive coding (LPC), have breached the long-standing barriers to high-quality low-cost synthetic speech. Simple subsystems consisting of an LPC synthesizer, a vocabulary ROM, and an optional microcomputer now can add speech to such diverse products as learning aids, home appliances, machine controls, and talking elevators.

This success in speech synthesis is now spurring the search for low-cost speech-recognition systems. Predictably, digital-signal processing (DSP) techniques are replacing costly complex analog systems. In contrast to their role in speech synthesis, general-purpose DSP chips seem to be in the lead in speech recognition, because so far the economics of scale dictate the general-purpose approach.

No specialized high-volume applications have yet emerged in speech recognition, as in speech synthesis, to invoke the cost-cutting potentials of high production volume. Moreover, the complexity required for recognition tends to mitigate against the possibility of fitting such a processor on a single chip.

Figure 1 is a block diagram of the processing flow in a general speech-recognition system, which performs the following tasks: front-end filtering and amplifying, speech-feature extraction, time registration, stored-vocabulary lookup and comparison, and final-decision processing.

Analog speech signals enter the system's front end for appropriate impedance matching, amplification, and filtering. From this filtered, composite, speech-signal input a set of speech-identifying features, or parameters, is extracted, which (within limits) uniquely characterize each spoken word. These extracted parameters must, of course, duplicate the speech parameters in the system's stored vocabulary in the same way, so that valid comparisons can take place. The identifying features can be functions of frequency with time-smoothing factors or functions of time with implicit frequency-smoothing factors.

Following speech-parameter extraction, the speech-recognition system must determine the beginning and ending of each input word—a process called time registration. Once a word is completely defined (in time) and characterized (by its parameters) in digital form, it is compared with the system's lexicon. Finally, the decision-processing section identifies the information as indicating one of the system's valid words, or rejects it.

The front end of a speech-recognition system remains an analog stronghold. It usually consists of a microphone, an amplifier (which can be optional), and a low-pass filter (which is mandatory to prevent aliasing in the later digital sections of the system). Of these, the microphone often is a major source of signal corruption, especially when costs must be trimmed and the system's physical volume must be kept small. In addition, low-cost small microphones generally respond indiscriminately to both desired and extraneous inputs.

One way of reducing extraneous-sound pickup is to keep the microphone close to the sound source. An often-preferred solution places the microphone about 1 cm from the speaker's lips on a bracket that fits right over the speaker's head. Such an arrangement provides excellent signal-to-noise ratios; keeping the microphone any farther away from the sound

Richard H. Wiggins, Manager, Speech Systems Technology
George R. Doddington,
Manager of Speech Systems Research
Texas Instruments, Inc.
P.O. Box 225621, Dallas, TX 75265

source can cause problems.

When amplification is needed to boost the microphone's output, a simple amplifier can take care of the job of passing the required limited bandwidth of 4 kHz (the same bandwidth required for a good voice-grade telephone channel). The microphone signal must be raised only enough to drive an input a-d converter in a digital system or the speech-parameter extraction circuit followed by an a-d converter in an analog system (Fig. 1).

In digital systems, however, the front end also must include antialiasing filters that eliminate frequencies of more than half the input signal's sampling rate. Without antialiasing filters, such frequencies would "fold over" into the sampled-signal spectrum, with highly undesirable consequences.

Once past the front end with the speech-identifying parameters extracted from the speech signal in digital form, the data are ready for digital-signal processing (see "Processing Speech-Recognition Data").

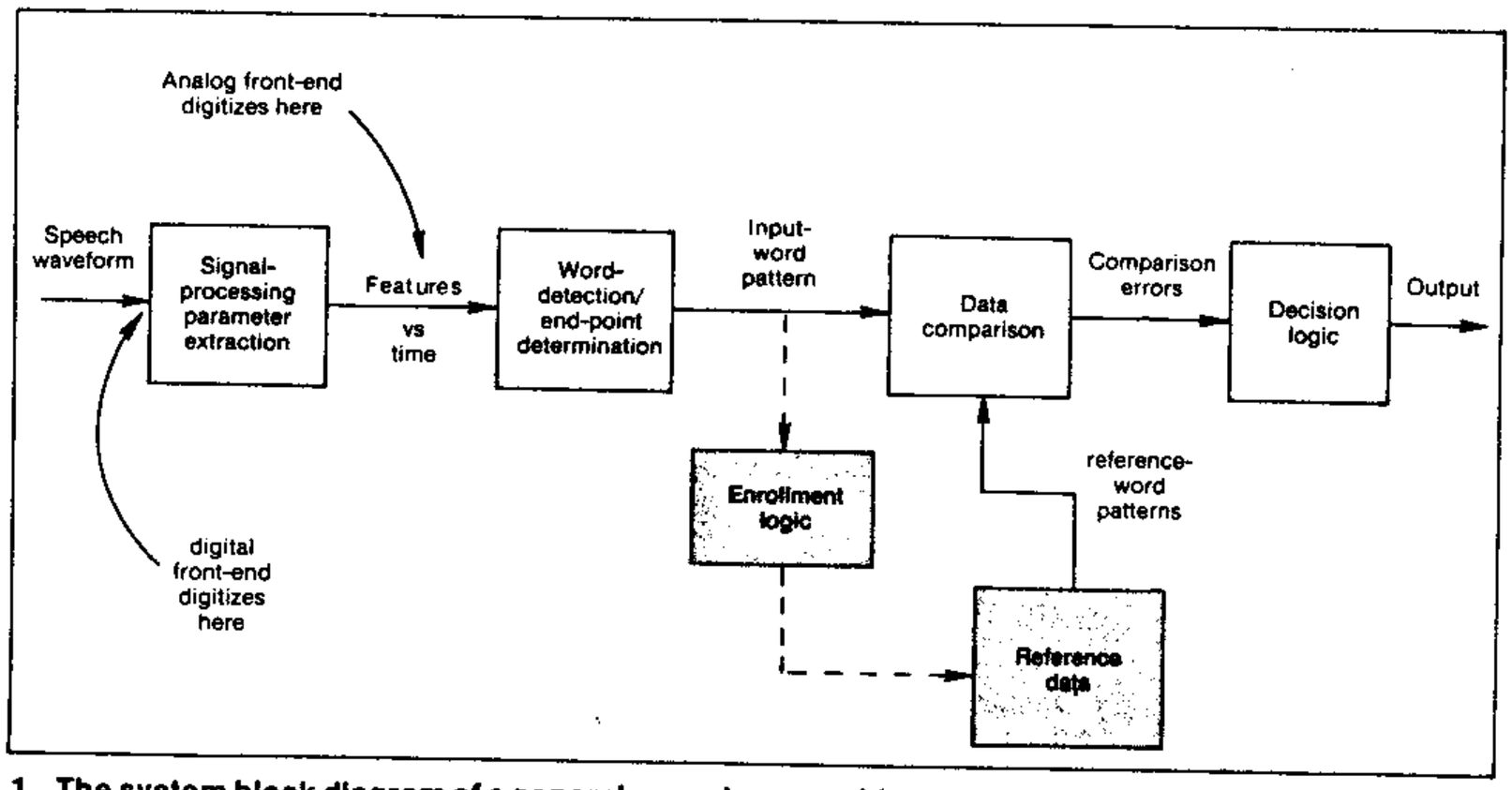
LPC: a natural for speech recognition

LPC can serve in speech recognition just as it serves in speech synthesis. With very few parameters, an LPC analysis efficiently describes the frequency/time-distribution content of a speech signal (Fig. 2). The number of parameters—or order—of the system ranges from 8 to 14 (10 is a reasonably accurate compromise).

Still, LPC does not supply enough data to re-create the original input waveform exactly, even with more than 14 coefficients. The extracted parameters merely identify specific words with minimal data. Thus, an important part of an efficient LPC speech-recognition design is to choose words with parameters that are not easily confused with those of other words.

LPC is an effective way to synthesizing speech with a time-variable digital-filter model of the human speech-production mechanism. Figure 3 shows a speech signal generated by an LPC tenth-order model of a time-variable digital filter. An LPC representation of an average spoken word is approximately 2560 bits long, when 16 bits are used to represent each of the ten autocorrelation coefficients of a tenth-order filter. These filter coefficients are determined at 16 points in time (or $160 \times 16 = 2560$ bits).

However, to extract enough speech parameters to fully activate a tenth-order LPC filter, the extracting circuit must perform roughly 10^5 arithmetic operations per second. The operations are primarily multiply-and-accumulate functions consisting of ten operations per sample (to determine the autocorrelation terms) at a rate of 10^4 samples per second. These numbers, however, are very approximate and depend on the specific algorithm implemented and the kinds of arithmetic operations required. Nevertheless, the numbers are adequate for shedding light on the



1. The system block diagram of a general speech-recognition system, whether processed by analog or digital techniques, usually looks the same. However, in an all-digital system the signal-processing for extracting the speech-identifying parameters is done with digital techniques; therefore, the speech analog signals from a microphone (and amplifier) must go through an a-d converter before entering the processor. In an analog system, the parameter extraction is by analog methods; accordingly, the analog-speech signals enter the analog processor directly. The analog processor's output is then converted into digital form, and an analog-to-digital converter completes the recognition task digitally.

performance and features that would be required from a candidate digital-signal-processing (DSP) chip (or set of chips).

After extracting the speech parameters, the DSP chip would then have to compare these data with the data in the system lexicon. The comparison measures the difference between the extracted parameters and the parameters of the word in the lexicon. If, for example, ten features per input word taken at 12 to 16 points in time are compared, then roughly 120 to 160 numbers must be scanned over the entire system vocabulary. For a maximum average input rate of about one word per second (which is reasonable when pauses are included), a reference vocabulary of 100 to 250 words would require between 10^4 and 10^5 comparisons. Accordingly, the DSP chip must perform some 10^5 to 10^6 operations per second to handle the task—a tough job to implement because of the high speed, but relatively simple, repetitive, and logical when compared with that of time registration.

Time registration, an important step in the comparison process, attempts to mark the points in time where an input utterance begins and ends. There, a simple linear relationship between the data bracketed by the timing-registration points and the stored words allows the two to be equated by a simple scaling method.

The most popular time-registration approach is based on a "rectangular" gross-energy profile of the input signals. A signal with an energy level below a specific threshold is interpreted as the absence of talking. Accordingly, the system works best with isolated, monosyllabic words.

More elegant, but highly complex, approaches for handling the time-registration task have recently tackled the problems posed by continuous speech in some large systems. The method usually is part of the data-comparison operation, but is currently too complex for a low-cost minimum-chip-count system.

Because low-cost energy-profile time-registration systems must be confined to monosyllabic isolated words, they have several disadvantages: Speech patterns are uncomfortably unnatural; input rates are lower because of the between-word pauses; and speakers need extensive training (Fig. 4).

Continuous speech is recognized

The commercial speech-recognition systems that can recognize continuous speech—for example, Exxon Enterprises' Verbex Model 1800 and NEC's DP 100—cost many thousands of dollars. They are a far cry from the low-cost solutions that eventually will encourage the spread of speech-recognition systems. Such systems must provide satisfactory recognition, yet be easy to add into other equipment, and cost

Processing speech-recognition data

Thanks to the blazing arithmetic speed of today's digital-signal-processing chips and their imminent improvement (as VLSI technology emerges), the analog realm is shrinking. Although most low-cost systems will eventually relinquish the speech-parameter-extraction function to digital techniques, the currently available speech-recognition systems still extract the speech-identifying features by analog means.

The analog approach is not without merit. It has long served as the most cost-effective means consistent with acceptable levels of performance. However, analog implementations tend to be less dependable and more costly to develop than digital ones. Also, analog circuitry is more susceptible to environment-induced drift, aging effects, and power-supply variations; above all, it is much more difficult to program than digital circuitry.

Meanwhile, rapid VLSI improvements are tolling the death knell of analog speech-parameter extraction. Low-cost monolithic a-d converters now routinely operate at high 8-kHz rates, 8-bit linearity, and 12-bit dynamic ranges, as required for a digital-processor input. LSI codec chips can handle both the a-d conversion and the required filtering. Inexpensive digital-signal-processing chip sets are becoming more abundant, with VLSI technology waiting in the wings to provide single-chip versions.

Such VLSI digital-signal-processing chips would be highly specialized microcomputers featuring fast forms of arithmetic processing. All of today's processing chips offer submicrosecond multiplication-and-accumulation operations. For example, NEC's μPD 7720 multiplies 16 × 16-bit operands into 32-bit products in 300 ns; a Bell Labs device (designed for Western Electric use only) multiplies 16 × 20-bit numbers to form 36-bit products in 800 ns.

By way of contrast, Intel's 2920 multiplies by shifting a series of variables simultaneously, called pipelining—a common microprocessor-software scheme. Because branching instructions do not exist, the 2920's 300-ns instruction-cycle-based operations are appropriate mostly for processing fixed-coefficient expressions.

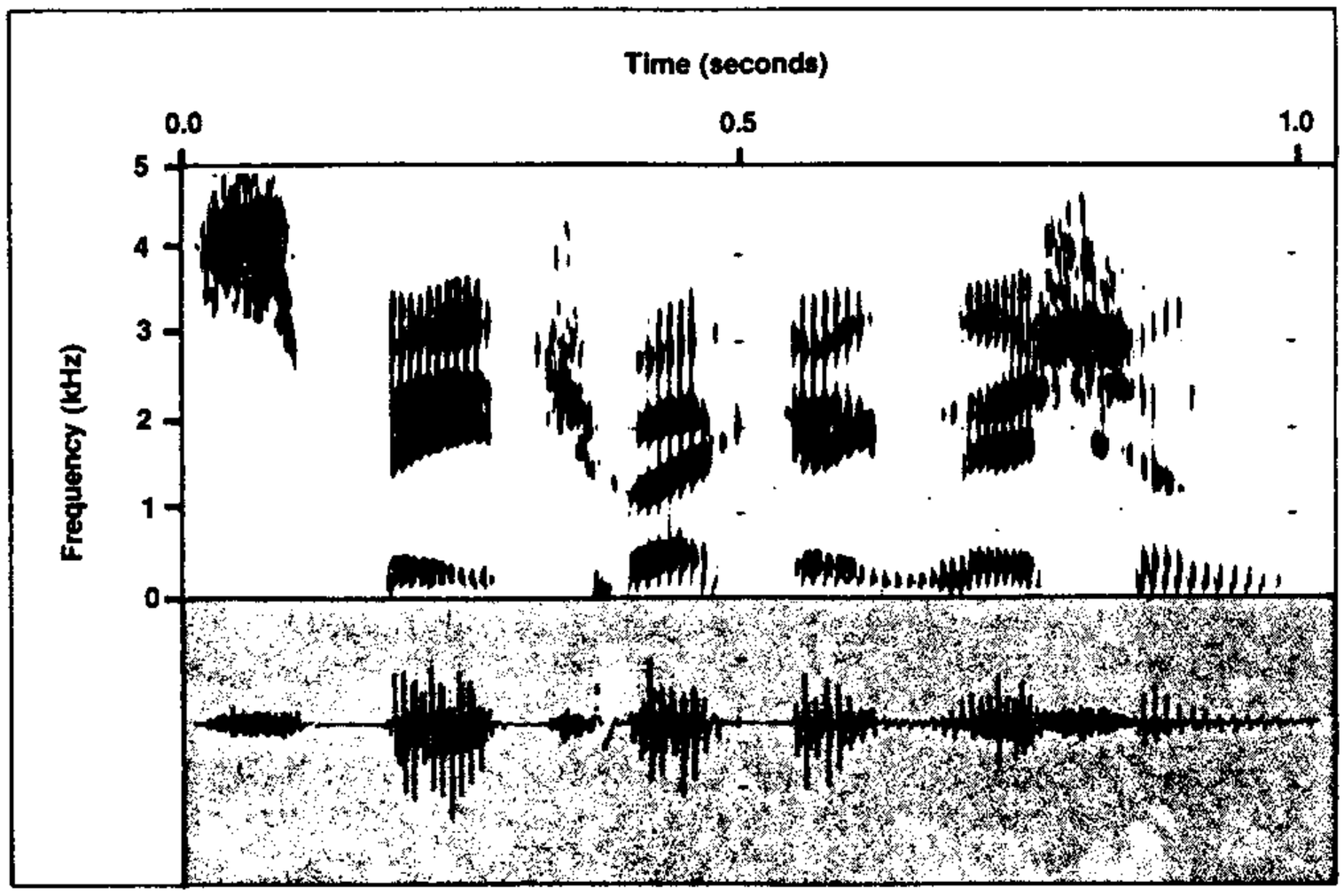
But fast arithmetic is only half the battle. The chips must also provide extensive comparison, evaluation, and decision-making capabilities. In addition to the arithmetic capability, the on-chip capability should include an instruction set that supports branching, the ability to address about 10^4 words of program storage, and the considerable RAM space needed to store the reference templates generated by speaker-dependent logic.

This RAM, for the near future, will probably reside off-chip. Also, for the present, speech-recognition systems can be built with a cluster of chips.

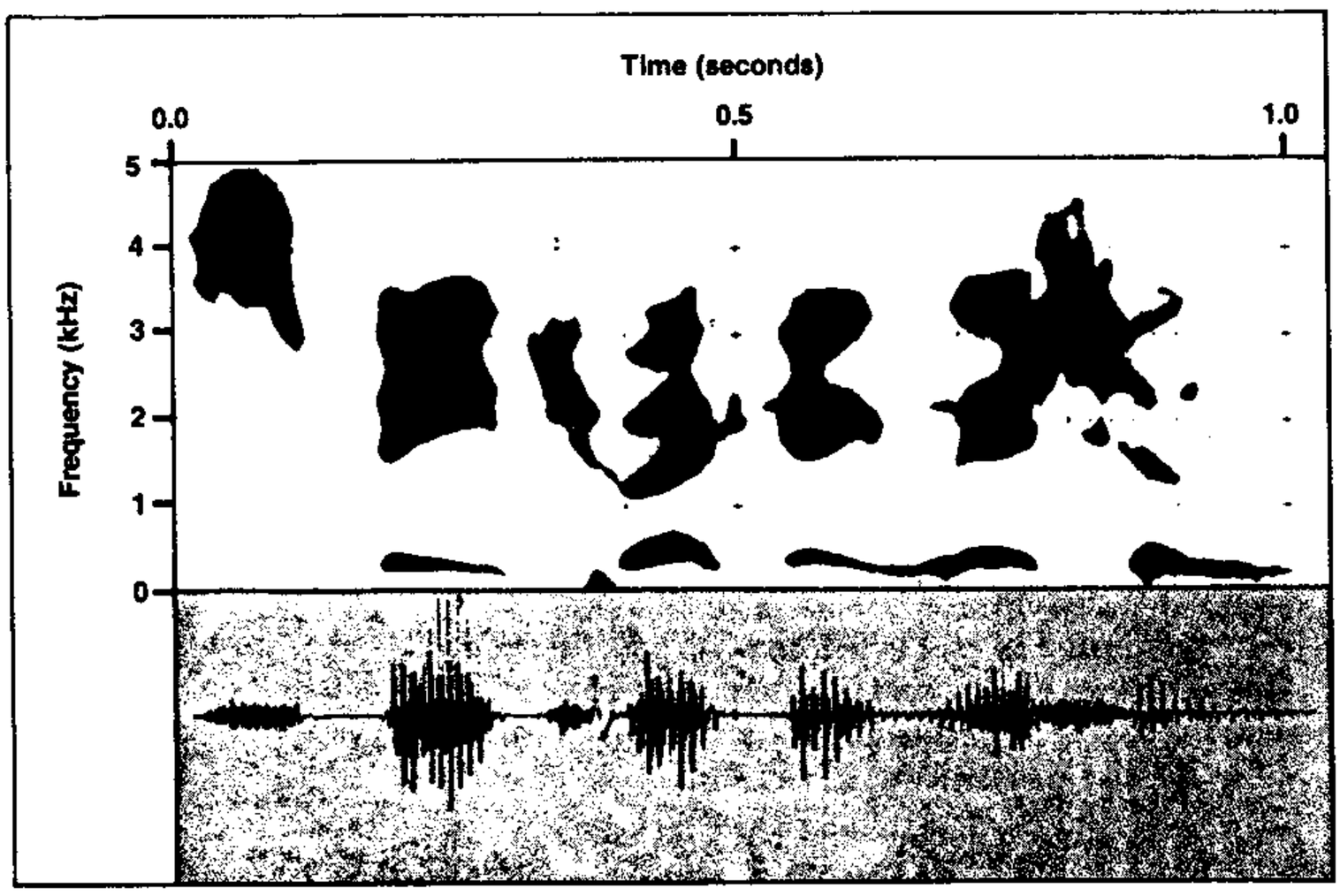
at most several hundred dollars.

To solve the time-registration problem, complex continuous-speech or connected-word-recognition systems continuously compare the extracted-input with the reference data; many points (in time) are examined to find the possible ends of words. The two tasks are interrelated so they demand a large memory, a high throughput rate, and a reasonably complete set of instructions. Therefore, the services of a sophisticated minicomputer, coupled with an arithmetic-array processor, are needed.

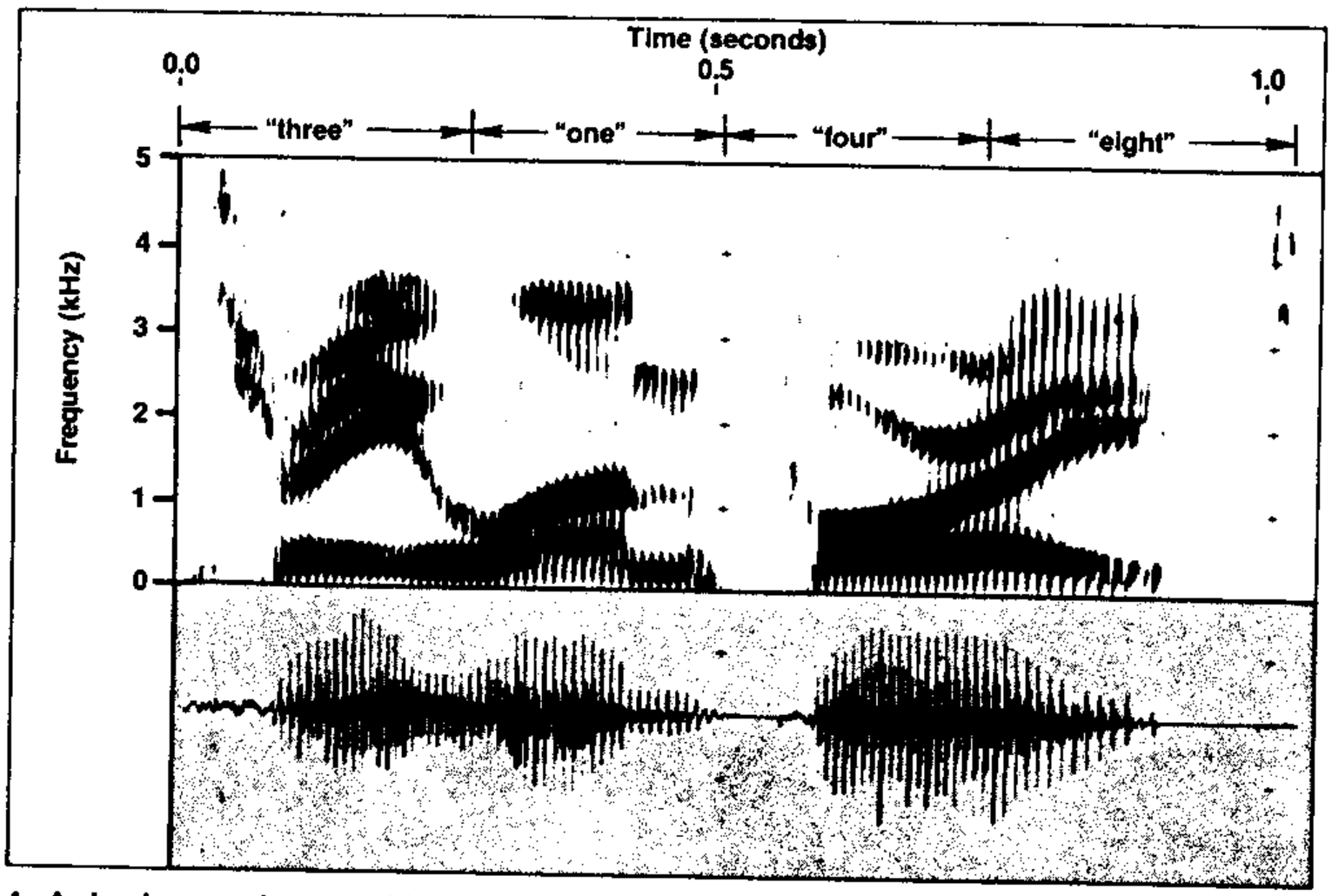
Simple or complex, the object of the system is to select one valid word or reject the utterance as illegal. (Supposedly, words not in a system's vocabulary will



2. The spectrogram (top) shows the relationship of frequency intensity vs time for the spoken utterance's "speech recognition." A more common oscilloscope display of the sound-pressure waveform (amplitude vs time) is shown also (bottom). Coarticulation between the two words almost obliterates the short pause between them.



3. When the utterance of Fig. 2 is synthesized with a tenth-order LPC model, both the frequency (spectrogram) and amplitude vs time of the synthesized speech correspond very closely with the original.



4. A simple speech-recognition system, which requires discrete monosyllabic words with definite pauses between them (about 0.25 s), would not be able to identify the four numbers in the spectrogram—3, 1, 4, 8—that were uttered in rapid-fire succession.

not be addressed to the system by the speaker; nevertheless, illegal words should be rejected rather than identified wrongly.) But word pairs, such as "seem" and "seen," which are difficult to discriminate, make the task thorny. Also, a talker's inexperience, extraneous sounds like coughing, and speech problems can produce illegal words.

Naturally, a system that adapts to the talker would be highly desirable, but then the costs would be prohibitive. Less costly is a set of predetermined reference features that represent an "average" of the prospective talkers. The takers then must adapt to the system.

In such a "taker-adapting-to-the-system" approach, word templates can be permanently stored in a low-cost ROM, rather than in a volatile RAM. The RAM storage requires sophisticated programming and complex logic circuitry at a high cost to process particular talkers.

But even with the talker adapting to the system, the logical operations for identifying or rejecting input utterances are complex. Extensive software-oriented architecture is usually required: The complete program can easily require 10^3 to 10^4 steps. Accordingly, even a simple continuous-speech-recognition system requires a program memory of considerable size—say, about 20 kbytes.

Single-chip speech recognition not feasible

No single LSI device, either available or imminent, can provide both the signal-processing and general-purpose capabilities needed for reasonably high-quality speech recognition-especially if an analog front-end function must be included on the chip.

However, current 16-bit general-purpose processors could operate economically with a specialized auxiliary processor, such as a digital "speech-recognition chip." The general-purpose processor could maintain control of the system bus, while the auxiliary processor concurrently performs its special arithmetic operations.

The speech-recognition chip must be able to carry out a wide variety of algorithms with a self-contained memory and logic to handle 10 to 100 program steps. For the present, the most practical chip would be organized for 16-bit words, and be capable of high-speed general-purpose arithmetic operations, such as multiplying two 16-bit numbers and accumulating 32-bit products. When enough volume develops for a specific application, the algorithm, word size, and speed (and cost) could be reduced to fit just the specific needs of that special use.

A unique product that could motivate a breakthrough in speech recognition—as TI's "Speak and Spell" did for speech synthesis—remains elusive; still, the speech-recognition field is young,. With machines already speaking, how long can it be before they also listen and understand?□

With linear-predictive-coded speech-synthesis chips easy to interface to most microprocessor and microcomputer systems, speech can be added as a primary or auxiliary capability at low cost.

LPC speech-synthesis chips mate easily with micros

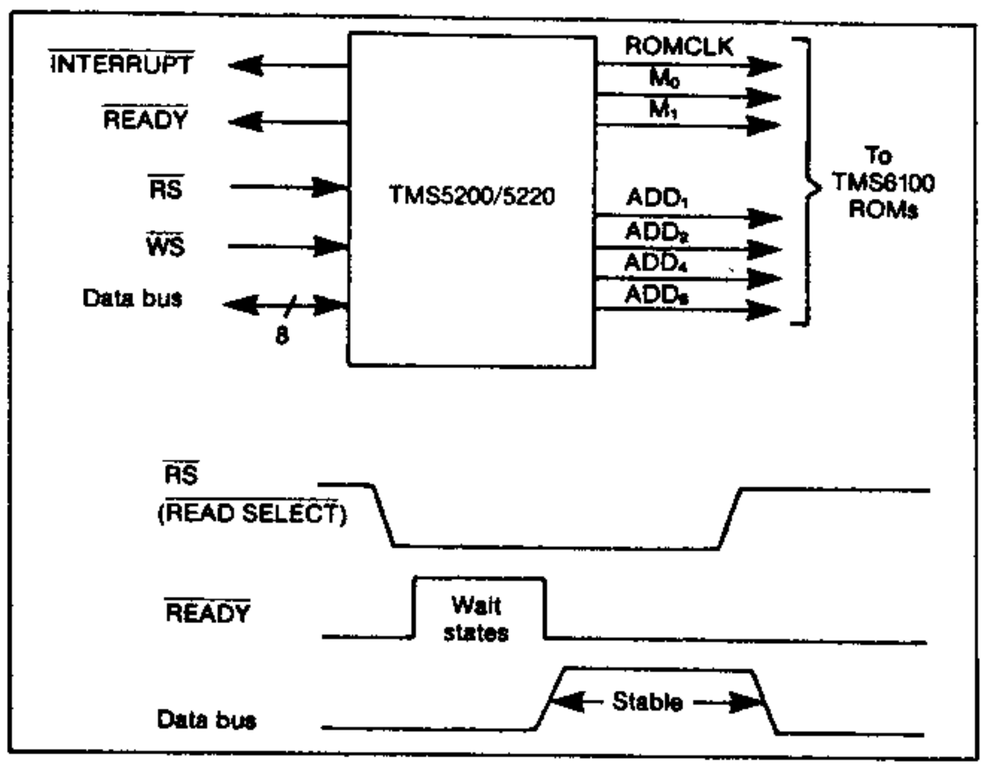
This article is the fourth in a series on synthetic speech, kicked off by an interview with Bernard H. List, vice president and manager of the MOS Functions Division at Texas Instruments (Midland, TX) in the May 28, 1981 issue, (p. 35). Subsequent articles in the June 11 (p. 213), June 25 (p. 121), and July 9 (p. 107) issues covered hardware and software aspects of both speech-synthesis and recognition equipment.

Chips for generating low-cost synthetic speech based on linear-predictive code (LPC) interface readily with most popular microprocessors and microcomputers. As a result, a wide variety of processor/computer-based designs could be substantially enhanced with a synthetic-speech capability at low cost and with minimal effect on both the design and production cycles—even when the speech must be tailored to special system requirements.

The TMS5100 4-bit synthesizer is suitable for low-end devices, in which cost is the prime consideration and speech is the main feature (as in the toy, "Speak and Spell"). For auxiliary speech function in systems with 8-bit microprocessor controllers, the TMS5200 fills the bill.

The interface between a 5200 (or the more advanced 5220) synthesizer chip and a host-controlling processor is provided by 12 control lines (Fig. 1): $\overline{\text{READ SELECT}}$ ($\overline{\text{RS}}$) and $\overline{\text{WRITE SELECT}}$ ($\overline{\text{WS}}$) input lines, $\overline{\text{READY}}$ and $\overline{\text{INTERRUPT}}$ output lines, and eight bidirectional data-bus lines. With 6-kΩ pull-up resistors, these lines are fully compatible with TTL circuitry; without pull-up resistors, the lines are compatible with CMOS circuitry.

Directing the 5200's speech-synthesis operations are six 8-bit macro commands: Reset, Load Address, Read and Branch, Speak, Speak External, and Read Byte. These are set up on the data bus by the host computer, and followed by a $\overline{\text{WS}}$ control signal. Although the 5200 operates under the host's control, the 5200 does not require full-time supervision. The host controller merely issues macro commands (and in some systems, also input data). Then the synthesizer takes over and carries out its synthesizer algorithm internally with the data provided from external ROMs, as directed by lines ADD_1 through



1. Interfacing the 5200 or 5220 speech-synthesizer chip with most popular microprocessors or microcomputers requires just four control lines and an eight-line bidirectional data bus to the host controller. Speech-defining data can enter the synthesizer from a separate ROM, such as the TMS6100, or from the controller's memory system via the data bus.

John Hayn, Manager of Design Engineering
Texas Instruments Corp.
Speech Technology Center
Midland, TX 79701

ADD_8, M_0 and M_1, and ROMCLK, much as any other dedicated, attached processor would.

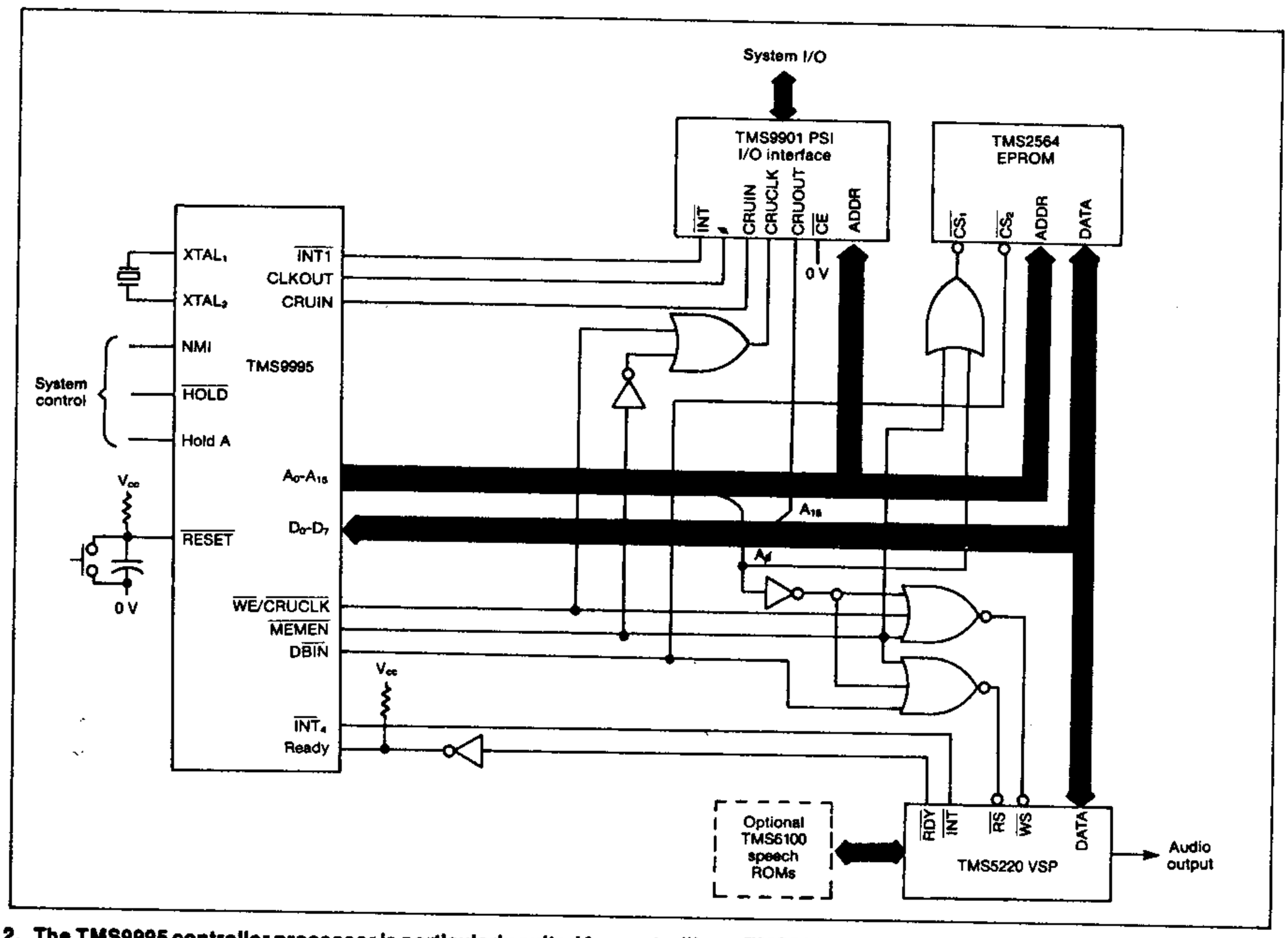
However, the host controller does not lose track of the synthesizer. The host can still stay informed of the 5200's condition by checking its status bits and the $\overline{\text{INTERRUPT}}$ output. Upon receipt of a $\overline{\text{READ SELECT}}$ (except immediately after receiving a Read Byte command), the 5200 delivers its status bits to the host controller over the data-bus lines. The status bits indicate whether the 5200 is speaking (talk status), or whether its FIFO buffer is less than half full, and therefore must be replenished before it can execute a Speak External command.

The status bits, in conjunction with an interrupt request, determine the specific reason for the $\overline{\text{INTERRUPT}}$. Such interrupts can be prioritized and assimilated into the system host's hierarchy. Or, they can be combined with the interrupts of other speech-related functions to form a composite speech subsystem that the system host can then treat as a semiautonomous module, which results in simpler software. In cost-sensitive applications, the speech-function $\overline{\text{INTERRUPT}}$ output may be completely disregarded, or polled.

But the $\overline{\text{READY}}$ signal, generally, cannot be disregarded. It matches relatively fast host processors to the basically slower synthesizer. The synthesizer behaves like a slow peripheral device with an access time of up to 7 μs, having between-access waiting periods (which the host must generate) of up to 300 μs. The host processor simply adds an appropriate number of wait states to its bus-cycle period until the 5200's $\overline{\text{READY}}$ indicates that data stability has been established at the 5200's data bus for reading or writing.

Operating the speech-synthesizer chip faster would be pointless because output speech is delivered in real time at the optimum rate for recognition by humans. Thus, the relatively slow operation of the synthesizer PMOS structures matches speech requirements. For example, with a nominal 640-kHz clock, the 5220 chip requires at most 50 bits of LPC data input even though it delivers (to its on-chip d-a converter) 8000 bits of synthetic speech each 25-ms period.

The speed difference between the PMOS of a



2. The TMS9995 controller processor is particularly suited for controlling a TMS5220 speech synthesizer, which then becomes just another auxiliary peripheral, while the 9995 carries on its other, generally more demanding functions of the rest of a system.

A memory specially organized for speech synthesis

Speech-synthesis processor chips like the TMS5100, 5200, and 5220 operate from linear-predictive-code (LPC) data, which represent a 100-to-1 data compression of the information in spoken words. While speech data could reach a synthesizer from a memory operating in a microprocessor (or microcomputer) controller chip's memory space (either on-chip ROM or auxiliary EPROM), there are decided advantages to having the data come from a voice-synthesis memory like the TMS6100 ROM (Fig. A).

The 6100, a low-cost, serial-output, densely packaged PMOS ROM, can be masked to provide the mixture of LPC data a particular system requires. The LPC speech data could be augmented by any other data; for example, thermocouple-linearization data for a talking thermometer.

For vocabularies of several thousand words, a single synthesizer chip can access up to 16 TMS6100 ROMs, each with 128 kbits, to provide approximately 30 minutes of continuous speech without repeating any of the stored words.

The 6100's storage array is specially organized to work efficiently with the 5200 or 5220. Accordingly, two control lines, M_0 and M_1, select one of four 6100 operating modes, or commands (Fig. B):

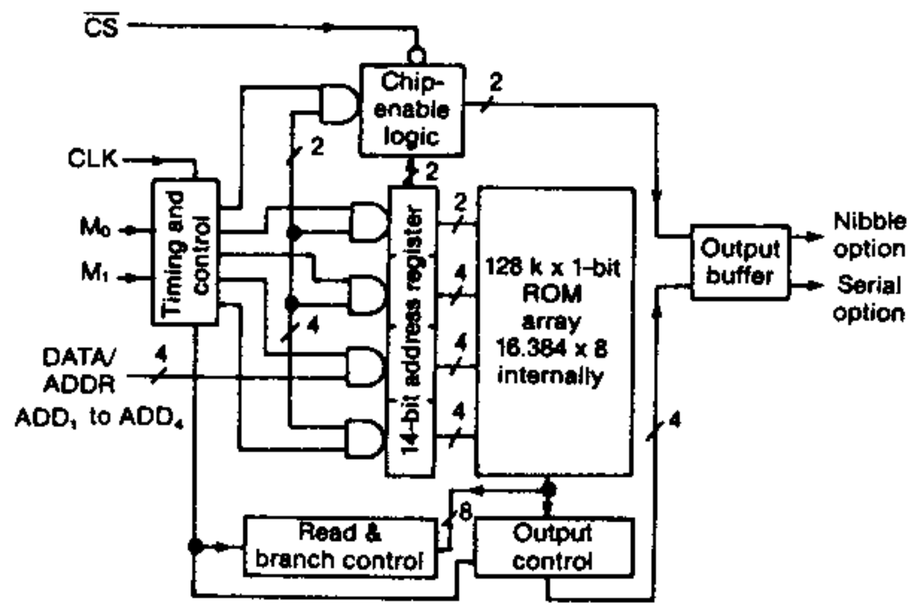
- Idle (no-op)
- Load Address
- Read
- Read and Branch.

The 6100 is addressed by five sequences of four nibbles each (on lines ADD_1 through ADD_4), or 20 bits total, clocked by five M_1 pulses (with M_0 quiescent). The ROM's chip-select logic (mask-programmable) responds to four of these address bits (to select one of 16 ROMs), and 14 of the bits form the actual byte addresses (the byte being read and the byte to be read). The remaining two bits are not used.

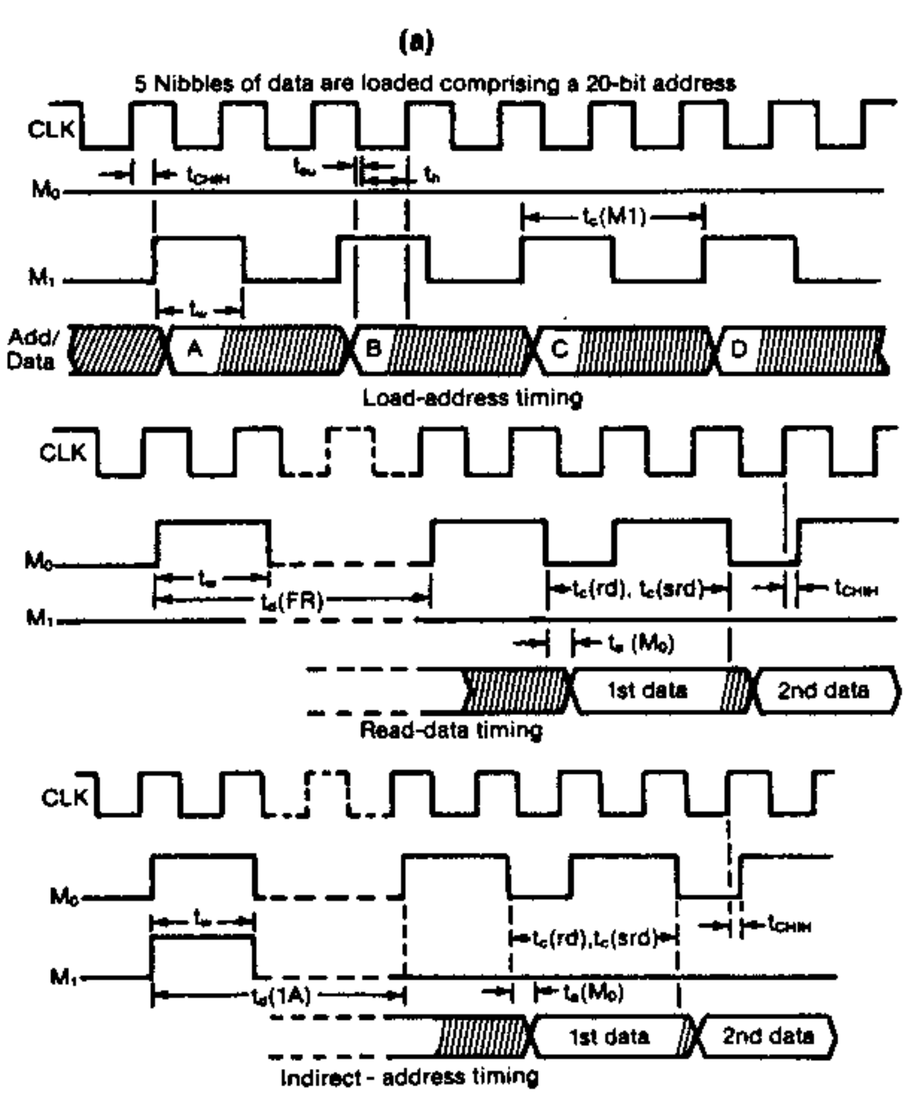
In the load-address mode, internal 4-bit address nibbles are loaded into the chip's address register, as directed by a load pointer. After each Load Address instruction, the load pointer advances four bits to the next address. Two Load Address commands (two nibbles, or one byte) are needed to establish the address of a consecutive data fetch.

The first Read instruction after a Load Address command (called dummy read), which is clocked by M_0 with M_1 quiescent, resets the 6100's internal load pointer and fetches the first address byte from the address bus. Data transfer then begins (after an M_1 pulse) with the second Read function following a subsequent Load Address function. While data are read out, the next data byte is fetched so that it can be immediately available for output.

The Read and Branch command (clocked by both M_0 and M_1) also resets the load pointer, but then loads the required address by fetching two consecutive bytes from memory (called indirect addressing), beginning at the current ROM address. The 14 low-order bits of the word made from joining these two bytes replace the contents of the address register, and the unit fetches its next data byte from this new address.



(a)



Note: M_0 and M_1 pulses are synchronized with the rising edge of the clock.

(b)

synthesizer and the NMOS of a typical processor chip is not unusual. In fact, current NMOS processor speeds are often faster (and getting faster still) than the memories and other peripherals they work with. As a result, many processors provide synchronization signals for interfacing the slower memory and peripheral devices. The same sync signals can serve a speech subsystem.

In addition to the synthesizer control signals that the system host processor supplies, LPC-coded speech data must be fed to the synthesizer from a storage device. Though the 5200 (and 5220) were originally tailored to process data from a TMS6100 voice-synthesis ROM (see "A Memory Specially Organized for Speech Synthesis"), the synthesizer can interface with a wide range of ROMs, EPROMS, or other memory devices, if they emulate (or are made to emulate) the TMS6100's operation.

Not only does the 6100 provide low-cost, dense storage, it also provides on-chip address-decoding and chip-selection logic. An on-chip memory-address register increments automatically under speech-synthesizer control. Also, an on-chip look-up-and-branch capability allows access to stored speech data that are independent of any preassigned vocabulary position. The synthesizer accesses the 6100 ROM with

Table 1. Write data to TMS5220 from TMS7040

```
82          MOVP   A,DDATA          OUTPUT DATA BYTE
0A
A3          ANDP   %RS0,BDATA       WRITE DATA STROBE LOW
FD
06
A6          BTJOP  %READY,ADATA,$   WAIT UNTIL READY TRUE
01
04
FC
A4          ORP    %RS1,BDATA       WRITE STROBE HIGH
02
06
```

Table 2. Read data from TMS5220 to TMS7040

```
A2          MOVP   %0,DDDR          DPORT TO INPUT MODE
00
0B
A3          ANDP   %RS0,BDATA       READ DATA STROBE LOW
FE
06
A6          BTJOP  %READY,ADATA,$   WAIT UNTIL READY TRUE
01
04
FC
80          MOVP   DDATA,A          INPUT DATA BYTE
0A
A4          ORP    %RS1,BDATA       READ DATA STROBE HIGH
01
06
A2          MOVP   %>FF,DDDR        DPORT TO INPUT MODE
FF
0B
```

a series of five 4-bit address words, or a total of 20 bits (of which just 18 bits are active).

Data from the host

Data also can flow from the system host to a 5200 or 5220 synthesizer via the synthesizer's 8-bit data bus. But in this case, the 5220's on-chip FIFO register file must be kept full by host-processor action, the result of an interrupt request from the synthesizer. In the most demanding situation, the host processor must provide 50 bits every 25 ms to the 5220 FIFO buffer. (The TMS5220 can accept a new byte of speech data every 23 μs).

Typically, a processor needs less than 500 μs to service such a FIFO interrupt request. Theoretically, the maximum interrupt rate could go as high as 40 interruptions per second. However, a fully loaded FIFO buffer usually contains more than a minimum of 2½ frames of speech data (11-bit repeat frames); accordingly, the average interrupt rate lies between 20 and 35 per second.

Even at 20 interrupts per second, the processor must dedicate as much as 1% of its time to the speech subsystem. Many applications cannot tolerate this level of host-processor overhead. In such cases, the 5220 could work directly with a 6100 ROM (or

Table 3. Applying trap subroutines for data transfers

```
                    Write data to TMS5220 from TMS7040
0004  WRITE  EQU   4                    TRAP 4 LABEL
  B2  SWRITE MOVP  A,@DATA              OUTPUT DATA BYTE
  0A
  A3         ANDP  %>F0,@DATA           WRITE DATA STROBE LOW
  FD
  06
  A6         BTJOP %>READY,@DATA,$      WAIT UNTIL READY TRUE
  01
  04
  FC
  A4         ORP   %>01,@DATA           WRITE STROBE HIGH
  02
  06
  0A         RETS                       RETURN
             *
             * Read data from TMS5220 to TMS7040
             *
0005  READ   EQU   5                    TRAP 5 LABEL
  A2 SREAD   MOVP  %0,@DDR              DPORT TO INPUT MODE
  00
  0B
  A3         ANDP  %>F0,@DATA           READ DATA STROBE LOW
  FE
  06
  A6         BTJOP %>READY,@DATA,$      WAIT UNTIL READY TRUE
  01
  04
  FC
  80         MOVP  @DATA,A              INPUT DATA BYTE
  0A
  A4         ORP   %>01,@DATA           READ DATA STROBE HIGH
  01
  06

  A2         MOVP  %>FF,@DDR            DPORT TO INPUT MODE
```

equivalent) and unburden the host.

Not surprisingly, the 5220 also works cleanly with Texas Instruments' TMS9995 microcontroller (ELECTRONIC DESIGN, Nov. 22, 1980, p. 219). And even though a microprocessor is controlling a speech subsystem, the 9995 can still be hooked up to a higher-level processor, with a programmable system-interface chip such as TI's TMS9902.

The 9995's 8-bit data bus is separate from the microcontroller's 16-bit address bus; hence, no demultiplexing is required to interface the 5220 (and its EPROM), which can simply plug into the 9995's 8-bit data bus. Moreover, the barest amount of logic completes the hardware interface—just two TTL gates and two inverters (Fig. 2). The gates develop the required $\overline{RS}$ and $\overline{WS}$ inputs to the 5220 from the 9995's $\overline{WRITE\ ENABLE}$ ($\overline{WE}$/$\overline{CRUCLK}$), $\overline{DATA\ BUS\ IN}$ ($\overline{DBIN}$), and $\overline{MEMORY\ ENABLE}$ ($\overline{MEMEN}$) outputs. One inverter inverts the 5220's $\overline{RDY}$ to the 9995's Memory Ready; the other inverter feeds the $\overline{WS}$ gate's input from DO on the data bus.

On top of that, the software required to add speech to a 9995-based programmable microsystem is as simple as the hardware. By treating the 5220 merely as a memory-mapped device, only one instruction—Move Byte—is necessary to handle all the required signal processing: The Memory Ready input to the 9995 implements the memory mapping of the 5220 and its associated EPROM.

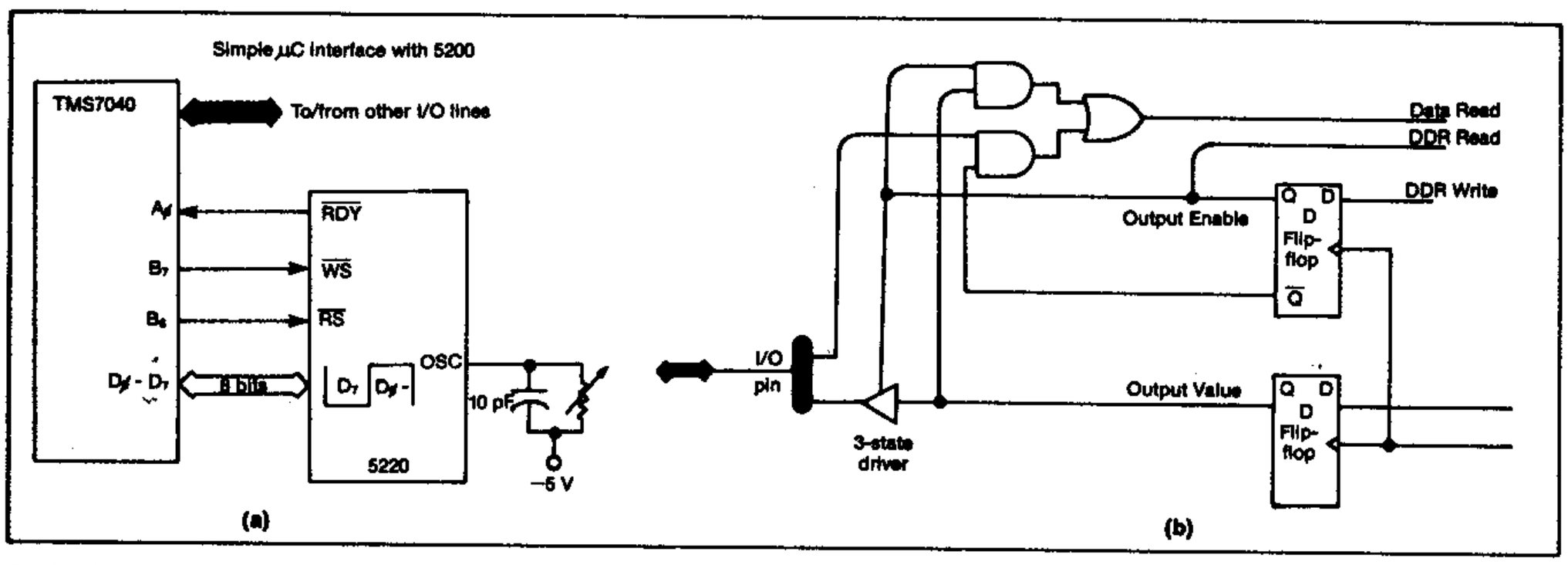
As it happens, almost all the more advanced 8 and 16-bit microprocessors—including the 9981, 9900, 8086, 68000, Z8000, 8085, 6809, and Z80—have roughly the same memory-ready capability, although some lack the 9995's separate data and address buses. Thus, the 5220 can readily interface just about every popular commodity-type 8 or 16-bit microprocessor, and virtually all can write to or read from the 5220 with only one instruction for each operation.

Interfacing microcomputers

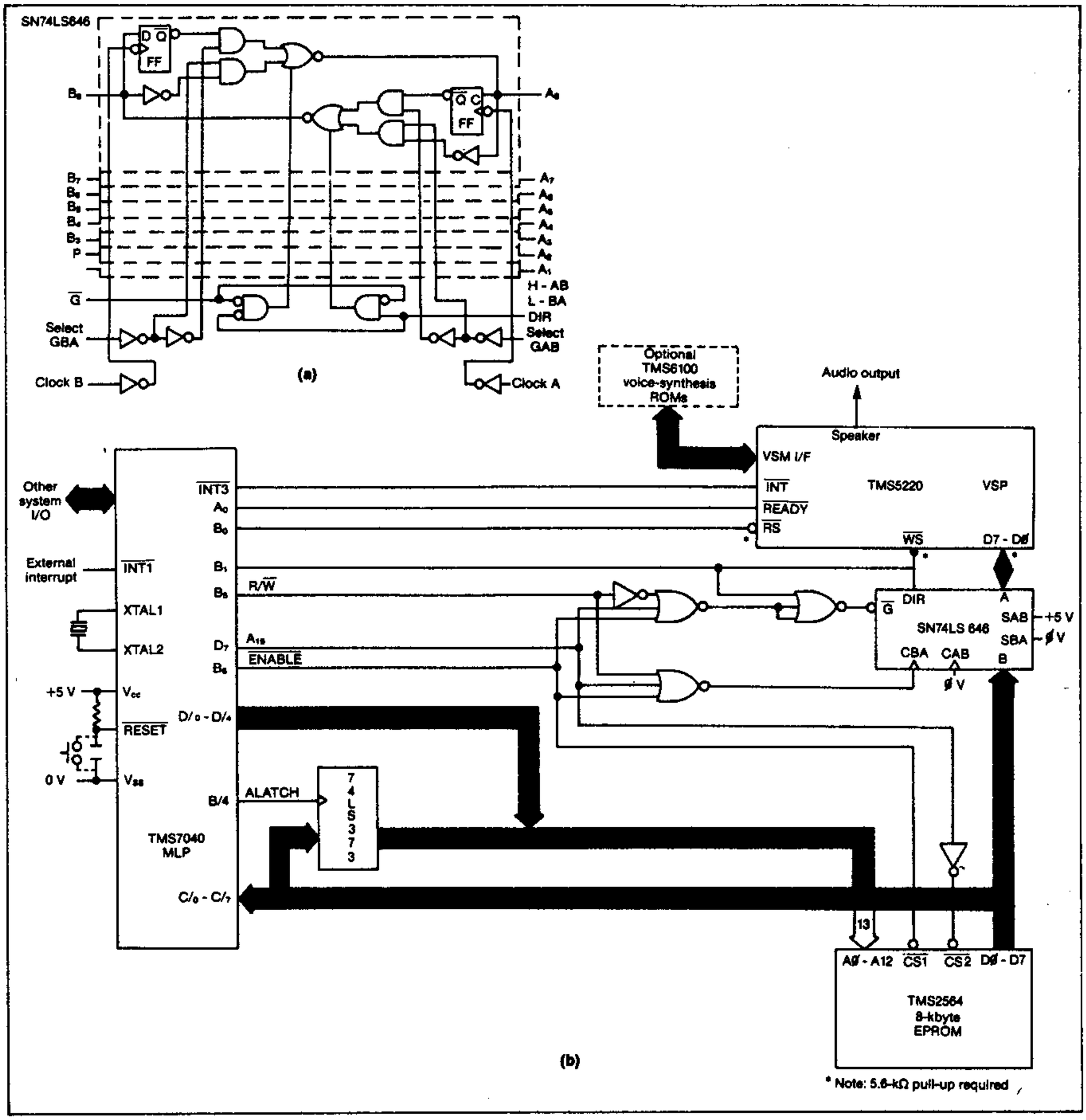
Interfacing to a 5220 chip is slightly more complex when the controller is a microcomputer rather than a microprocessor. Using a microcomputer is tempting because its on-chip features, such as RAM, ROM, timers, and I/O capability, make many low-cost applications economically feasible. Many 8-bit devices offer significant amounts of on-chip ROM and RAM. But microcomputer chips such as the TMS7000, 7020, and 7040 MLP family, the 8048, the 6801, and many others do not offer a Memory Ready input function like the 9995.

Still, the lack of a Memory Ready input can be worked around with software. When interfacing microcomputers, the 5220 should be mapped into the I/O space, which will simplify the hardware. Then, the microcomputer's parallel I/O interface communicates with the 5220 input buffer (FIFOs). In this way, the system emulates the memory mapping in software, but the simplified hardware then needs many more instructions for operating in the I/O space than in the memory space.

Substituting software for hardware is an excellent tradeoff for low-cost systems whose primary task is speech synthesis. An all-in-one microcomputer-controller/memory can cut an entire speech-synthesizer system down to just two chips: the micro and the synthesizer. An 8-bit microcomputer such as the TMS7040 (which carries a 4-kbyte ROM for the vocabulary), working with a TMS5220, can provide a speech-synthesis capability while still offering a powerful instruction set, on-chip RAM, the re-



3. A microcomputer, such as the TMS7040, together with a TMS5220, can be configured into a low-cost, minimal, two-chip speech synthesizer (a) that uses the 7040's on-chip ROM to store the speech data. Each bit of the 7040's bidirectional port (D_0 through D_7) can act as a receiver or transmitter under the control of data-direction flip-flops inside the 7040 (b).



4. An SN74LS646 octalbus-transceiver chip (a) can buffer/synchronize a fast processor with a slower speech synthesizer (b). The 646 contains eight register-buffered transceiver circuits with multiplexed three-state output drivers. When interposed between the 5220's and the 7040's bidirectional I/O ports, the 646 allows the 5220 and 7040 to exchange data asynchronously at their own rates.

mainder (after vocabulary storage) of its on-chip 32-kbit ROM, an on-chip interval timer, and extensive I/O capability.

Every peripheral file referenced

Because the 7040's 256-register peripheral file can directly reference every one of its on-chip control and data registers, a peripheral-file byte can serve as port or control data. The chip's 32 I/O pins are sectioned into four 8-bit ports. Since the C and D ports are bidirectional, they can directly interface the 5220. In Fig. 3a, the bidirectional port D is employed as a data port with each terminal programmable as an input or an output by means of data-direction flip-flops inside the 7040 (Fig. 3b). The C ports are similarly programmable.

Single pins on the 7040's other ports (A and B) take care of the handshaking signals that must pass between the 7040 and the 5220— $\overline{RDY}$, $\overline{WS}$, and $\overline{RS}$.

Table 1 shows the four 7040 instructions that make up a write operation (to the 5220) when a 5220 is controlled with a 7040 MLP. Table 2 shows the six-instruction listing for a read operation (from the 5220). Both the write and read operations share the same 7040 I/O port, D.

Because Write is used more often, its mode has been made the "normal" one for the port; thus, it is the shorter operation. Hence, the Read operation then needs the two additional instructions, "D port to input mode"—one at the beginning of the read operation, the other at the end to change from an output to an input port and back again.

To this end, the program-storage load on a 7040 (or other 8-bit microcomputer) can be reduced by using the device's subroutining capabilities—especially the Trap to Subroutine instruction in the 7040, where dedicated blocks of on-chip memory locations have been set aside for trap vectors. (The vectors point to the locations of user-defined subroutines. The 7040 accommodates up to 24 trap vectors, four of which are reserved for system functions.)

Thus, the write and read operations (Tables 1 and 2) can be programmed as trap-vector-located subroutine calls, at the expense of slightly longer-running times. Nevertheless, in low-cost applications, the processor is still almost always fast enough to run all the programming that can be stored in the available on-chip ROM space with time to spare.

Moreover, since the microcomputer needs to service a 5220 speech chip only once every 25 ms, the overall system-throughput impact is minimal. Then, the 20 or so bytes required for a write or read operation (Table 3) can be cut down to just two subroutines, and only one program byte need be stored for each—a Trap 4 label and a Trap 5 label—at a substantial saving in memory space.

The first instruction in the write operation outputs a byte (from the 7040 to the 5220). The next instruction in the sequence lowers the 5220's Write Strobe (Select) signal. Then, the following instruction makes the microcomputer wait until the 5220 signals that it is ready to receive data. The last instruction in the micros in the output routine raises the Write Strobe (Select) signal. The instructions for the read operation correspond one-to-one with those for the write operation, except that the first and last instructions are added to place the D port into an input mode and then to return it to the output mode. (The output mode for Write is "normal").

Unfortunately, such in-line programs for I/O-oriented operations may occupy significant amounts of memory space if used often, even though they are simple. Cost-conscious applications must keep memory size (and the number of instructions) to a minimum, especially when external memory devices must be added. However, a moderate amount of run time can usually be sacrificed to conserve expensive storage space by using subroutines.

Plenty of ports still left

Since just the first 11 ports of the 7040 peripheral file are hardware-defined for the 5220, the 7040 can still employ the other 21 ports normally—as it would, say, for an external keyboard. Thus, the 7040 can communicate with an 8-bit speech-synthesizer chip, such as the 5220, yet maintain substantial processing power.

More memory (or more peripheral devices) can be added to such a basic two-chip system by changing to a memory (or peripheral) expansion mode of operation. But in a memory-expansion (or peripheral-device) mode, microcomputers fall short. Microcomputers lack the logic needed for block or DMA-type data transfers. They also lack a Memory Ready input for synchronization; accordingly, a microcomputer cannot generate appropriate wait states for the 5220, which operates much slower than the microcomputer's memory cycle.

A TTL octalbus-transceiver chip, TI's 74LS646, provides a solution to this synchronization problem (Fig. 4a). With eight register-buffered transceiver circuits and multiplexed three-state drivers, the chip allows the 7040 microcomputer or the 5220 synthesizer to send or receive data asynchronously at each device's own rate (Fig. 4b).□

Serious users of synthesized-speech equipment will eventually need their own software development systems. For the present, the synthesizer-chip maker can supply a complete service.

Speech-synthesizer software generated from text or speech

This, the fifth article, ends a series on speech synthesis and recognition. Part one appeared in June 11 issue (p. 213), part two in the June 25 issue (p. 121), part three in the July 9 issue (p. 107), and part four in the July 23 issue (p. 161). The series was preceded by a look at the future of speech technology by Bernard H. List, vice president for MOS functions at Texas Instruments' Speech Technology Center (May 28, p. 35).

Equipment manufacturers serious about incorporating speech capabilities into their products must eventually acquire the tools for developing their own synthetic-speech software. An in-house capability offers clear advantages—short turnaround time, proprietary protection, originality and innovation, customization to special applications, and lower cost. For the present, however, Texas Instruments offers two related speech-synthesis software-development systems (or their services): a speech-composition and editing system, and a speech-collection and LPC-analysis system.

The first system is the simpler of the two. It operates from keyboard-entered text, and its integrated set of interactive utilities for text, phoneme, allophone, and LPC editing delivers EPROM-formatting data. A minimum speech-composition/editing system configuration includes a computer (like the DS990 Model 4 at about $29,500), software (costing about $5000), a PROM programmer (about $1350), and a speech-audition board (about $2000). In addition, the system includes a CRT editor terminal (like the 911 VDT) and software and hardware for carrying out standard text-editing commands, designated DX-10 (see Table 1), plus special speech-editing instructions.

The collection/LPC-analysis system starts at a "higher" level—collecting acoustic inputs of spoken words, phrases and sentences—and then performs a computationally intensive LPC analysis of the collected input. Thereafter, the equipment of the first system edits, auditions, and delivers formatted EPROM data for listening tests, followed by re-recording and re-editing (if necessary) until the desired results are attained.

Table 1. Composing/editing commands

ISS - Initiate speech system
TSS - Terminate speech system
EU - Edit utterance
***QE** - Quit edit

SL - Show line from file
***CL** - Copy lines
***ML** - Move lines

***DL** - Delete lines

***IU** - Insert utterance (file)
***IL** - Insert lines

ALP - Assign LPC parameter
Parameter (E, P, K1..K10)
Start frame
end frame
value

SLP - Scale LPC parameter
Parameter (E, P, K1..K10)
Start frame
end frame
scaling factor

SSS - Set single step control
Repetitions of previous frame
Repetitions of current frame
Repetitions of next frame

* DX-10: standard editing commands

Tom Brightman, Manager Speech-Technology Engineering
Texas Instruments Corp.
P.O. Box 6448, MS 3002
Midland, TX 79701

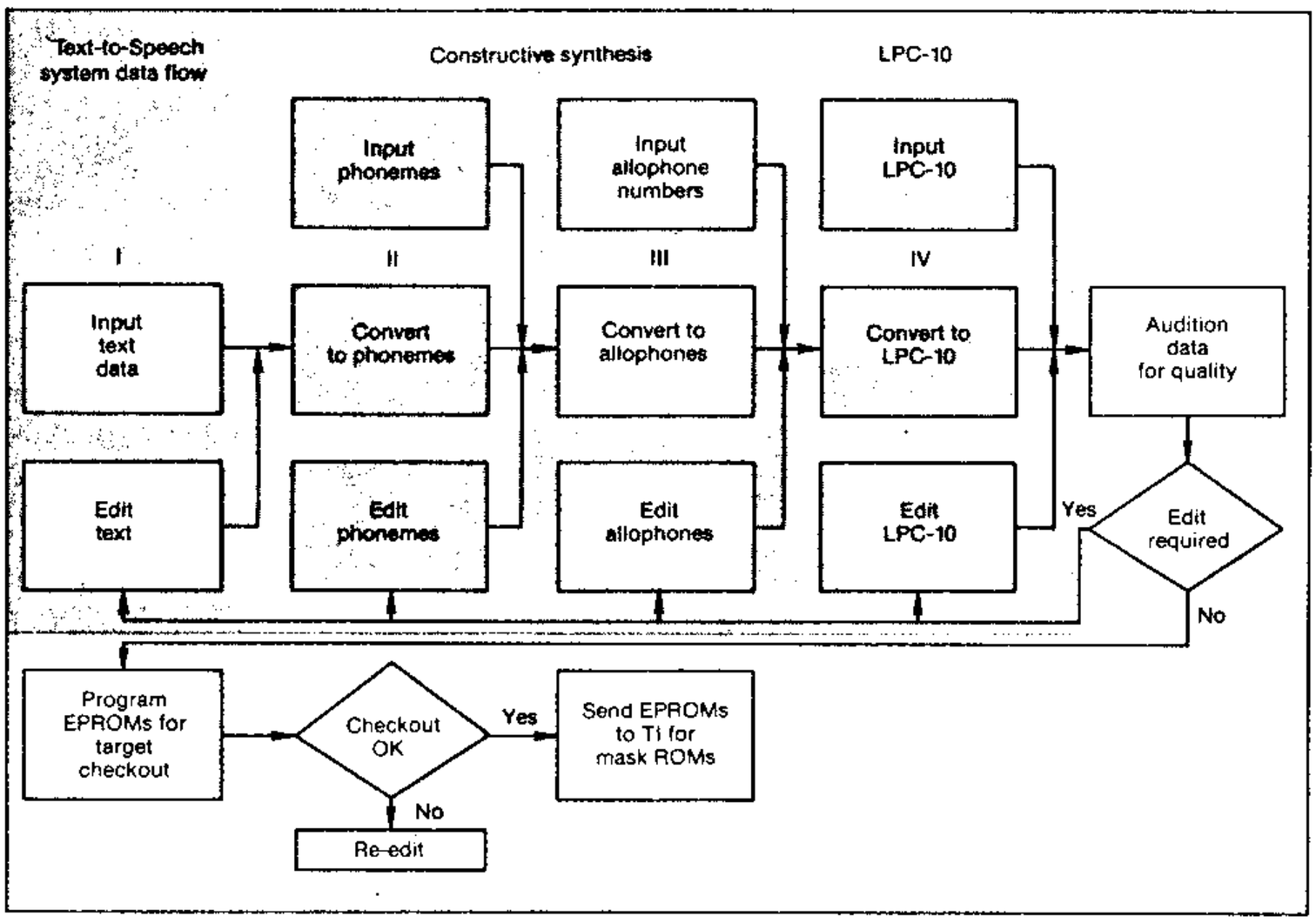
The additional equipment needed for the collection/LPC-analysis system includes an audio system for collecting and recording acoustic speech data (about $10,000) and software for the LPC analysis (about $5000). The cost of the facilities for managing and archiving the speech data-base is included in the composing/editing package.

Composing/editing works on four levels

Starting with a keyboard input, the speech-composing/editing (or Text-to-Speech) system allows editing the typed-in text on four levels—text, phonemes, allophones, and LPC-10 binary code (Fig. 1). First, the text can be edited as entered. Then it is converted to a phonemic-alphabet representation according to a comprehensive set of rules. A modest-sized 8-kbyte rule set generates phonemes with 90 to 92% accuracy. But to get somewhat higher accuracy, say, 96% or 97%, the rule set rises very rapidly—to about 50 kbytes—and much more steeply to get just another 1 or 2%.

The next conversion changes the phonemes to allophone notations—a more detailed variant of phonemes—which includes data based on word context such as pitch, timing, relative word placement, intonation, and other linguistic considerations. The final conversion is from allophone notations to TI's LPC-10 code, which in binary form can drive a speech-synthesizer chip like the TMS 5220 and produces audible speech output (see ELECTRONIC DESIGN, June 11, p. 213).

Trouble is, the development of nearly natural-sounding synthetic speech can involve a good amount of "art"—initial input and editing, listening, editing, listening again, and so on—until a desired effect is achieved. On a primitive level, separately "recorded" words could be strung together to form phrases or sentences. But words recorded by themselves, when put in the context of a sentence, usually sound artificial, and often are unintelligible. Similarly, words spoken in one context may be perfectly understandable, but when taken out of that context and spoken by themselves, or in another context, also can be unintelligible, or worse—ambiguous. For example, the number four can sound like fů, fůr, faw, foe, or the preposition "for"—which ranges from the unintelligible to the ambiguous, especially when spoken out of context.



1. The basic speech-composing/editing system provides equipment for editing keyboard-input data at four levels—text, phoneme, allophone, and LPC-10—as the data are converted from text to speech. After aural auditioning for quality, the data may be re-edited, or programmed onto an EPROM for further checks and possible additional editing. Finally, the data can be composed onto ROMs for production runs.

On top of the contextual problems, every conversion level from text to speech carries a probability of error with it. The ideal of 100% accuracy has not been attained by any system. The standard against which at least the phoneme level can be compared is the so-called Brown Corpus—a text-to-phoneme dictionary of the 20,000 most-common English words listed in the order of frequency of occurrence. Moreover, TI has developed the dictionary into a text-to-allophone form.

Then, words converted to phonemes (and allophones) by the speech-composition/edit development system could be looked up in the Brown Corpus and compared for accuracy. If the entire 20,000-word Brown Corpus could be incorporated into the development-rules system, 100% accuracy could be achieved, but then a huge memory would be needed, the system would operate very slowly, and it would be very expensive.

Instead, the TI Text-to-Speech approach is a more generalized system with somewhat less than perfect accuracy, but almost unlimited vocabulary. Moreover, manual editing procedures at any of the four levels—text, phoneme, allophone, or LPC—can improve the initial results substantially without the cost of all that enormous hardware and software.

Still, the hardware and software for even 90 to 92% first-cut accuracy with the TI composing/editing system is substantial and therefore requires an adequate minicomputer with hard-disk memories

```
1 THIS IS A TEST OF CONSTRUCTIVE SYNTHESIS.
  *EOF
```

Typical text edit screen (a)

```
1 DH '0 IH S - '0 IH Z - '0 AX - T EH S T - '0 AX V - K '0 AX N = S T
2 R AX K T IH V - S IH N TH EH Z IH S < >
  *EOF
```

Typical phoneme edit screen (b)

```
1 DH IH1 S# - IH1 Z# - AX1 - T+H EH3 S T# - AX1 V# - K + HEY AX1 AXIN
2 N# S T R AX3 K-HEY T IH2 V# - S IH2 IHIN N TH EH2 I IH2 IH1 IH1 S#
3 ---
  *EOF
```

Typical allophone edit screen (c)

```
 1  *  DH
 2          313  43  14  14   4   9   4   4   7   4   6   2
 3          341  43  16  14   5   9   4   3   3   6   4   4
 4          308  43  18  14   7   7   4   2   1   3   6   3
 5          385  43  19  15   7   7   6   2   2   3   4   5
 6  *  IH1
 7          819  44  17  20   4   7   4   7  12   6   4   4
 8         1146  45
 9          819  44
10  *  S#
11          983   0  38   8   2   6
12          901   0  31  14   3   5
13          737   0  31  20   6   7
14          253   0  31  17   3   4
15  *  -
16            0
17  *  IH1
18          819  44  17  20   4   7   4   7  12   6   4   4
19         1146  45
           ENERGY    K1  K2  K3  K4  K5  K6  K7  K8  K9  K0
           PITCH
```

Typical LPC edit screen (d)

2. Displays of an utterance in English text (a), phoneme alphabet (b), allophone notation (c), or LPC form are individually available for editing. The history of all changes after conversion from the initial text entry is automatically logged for each utterance, no matter at which level the changes are made, or from which terminal the changes originate.

(10-Mbyte) plus a professional-quality video terminal like the Model 911 VDT. However, the speech editor need not "care" about the system's complexity: Its performance parallels very closely that of a standard text editor with the exception of a few special-function keys that invoke the translation and speech-output features (Table 1).

The video terminal can call up and display any of the four levels. The text, of course, appears as keyed into the system: as in Fig. 2a, the sentence, "This is a test of constructive synthesis." At this level, selected words may be deliberately spelled incorrectly to improve the final synthesized output.

The sentence appears in phonemic alphabet form in the phoneme-edit display (Fig. 2b). For instance, the first word "THIS" is spelled in phonemic form as DH 'O IHS - . Again, editing can take place at this level, but then the editor requires a knowledge of phonemes and how changes will affect the final results.

The next level displays the translation (by the rules) to allophones (Fig. 2c). Since allophones essentially are detailed variations of phonemes, the phonemic alphabet is expanded to allophonics with subscripts. Accordingly, the phoneme for the word "THIS" becomes DH IH_1 S # - in allophone form. The vowel has acquired the subscript 1, which is the "short," or clipped, form of the IH phoneme, and the S has acquired a # symbol to indicate that it is a final S. The editor easily learns the few extra symbols.

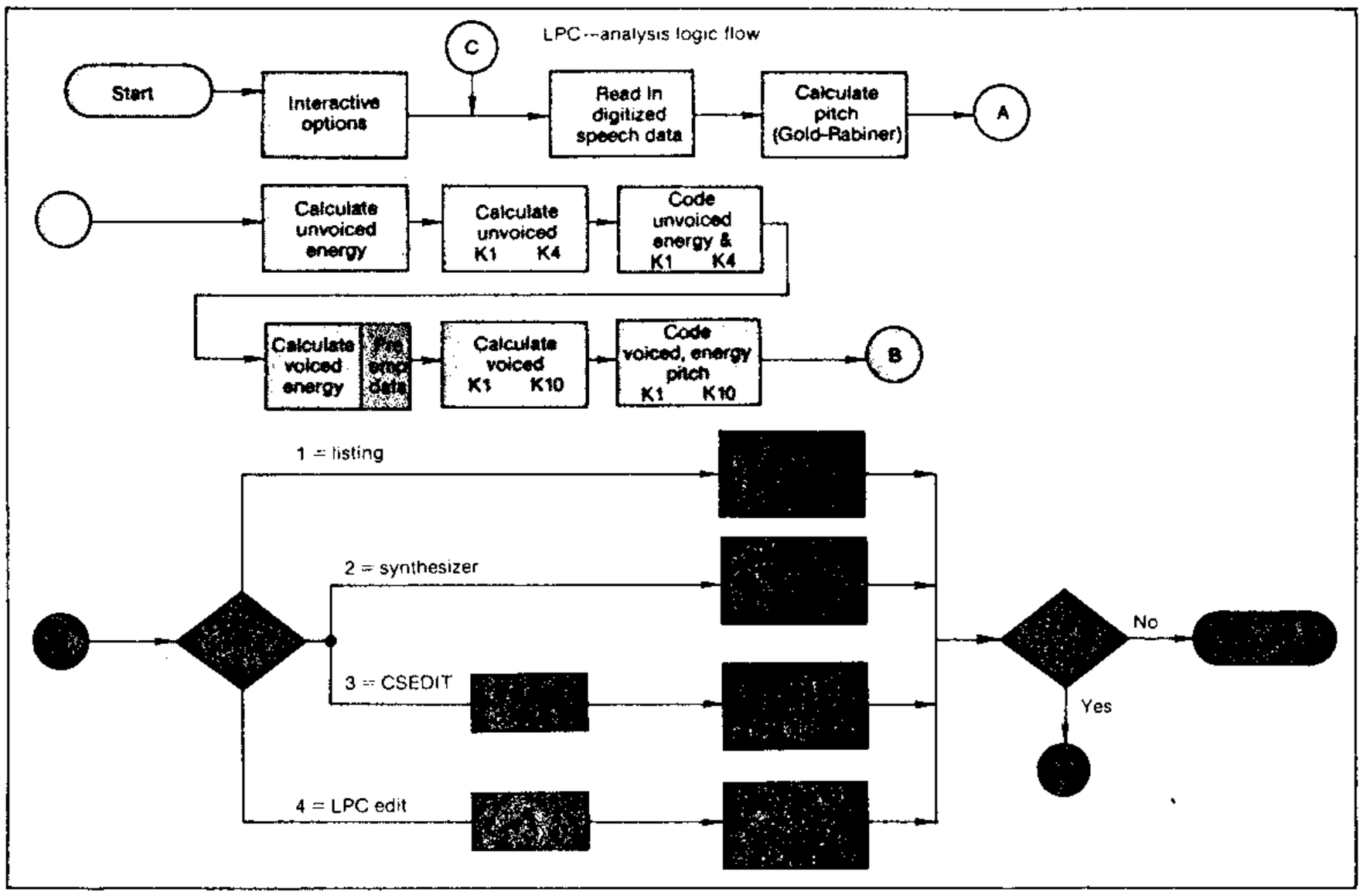
Interactive LPC-analysis interface

```
1. ENTER TOTAL NUMBER OF SECONDS OF SPEECH DATA TO BE PROCESSED          :      ***
   PLEASE REVIEW THE FOLLOWING DEFAULT VALUES. IF ** ALL ** OF
   THEM ARE DESIRED AS SHOWN, TYPE "Y" FOR YES, ELSE HIT RETURN
   AND CONTINUE WITH THE NORMAL INPUT SEQUENCE: *
2. ENTER ENERGY ANALYSIS WINDOW TYPE :      {DEFAULT=1, HAMMING WINDOW :      1
   1-HAMMING WINDOW  2-RECTANGULAR WINDOW  3-PITCH DEPENDENT WINDOW
3. ENTER ENERGY ANALYSIS WINDOW LENGTH IN MILLISECONDS       {DEF=25} :     25
4. ENTER ALPHA - THE PRE-EMPHASIS CONSTANT :          {DEFAULT=0.9375} :  .937500
5. ENTER THE ORDER OF THE LPC ANALYSIS      {DEFAULT=10, USE 2<=N<15} :     10
   {THE NUMBER OF AUTOCORRELATION TERMS IS BETWEEN 2 & 15}
6. ENTER THE FRAME UPDATE PERIOD IN MILLISECONDS          {DEFAULT=25} :     25
7. ENTER SPEECH DATA SAMPLE RATE {7000-12000} IN SECONDS   {DEF=8000} :     8000
8. DO YOU WANT TO CODE THE LPC PARAMETERS?          Y OR N {DEFAULT=N} :     N
```

(Screen 1)

```
THE INTERACTIVE INPUT IS COMPLETED. YOU HAVE CHOSEN THE FOLLOWING VALUES FOR THIS
RUN OF THE LINEAR PREDICTIVE CODING TECHNIQUE. PLEASE REVIEW THEM CAREFULLY:

     1. THERE IS XXX SECONDS OF SPEECH DATA TO BE PROCESSED.
     2. THE WINDOW TYPE IS A HAMMING WINDOW. CK ON WINDOW LENGTH CONTROL.
     3. THE ENERGY ANALYSIS WINDOW LENGTH IS XXX MILLISECONDS.
     4. THE PRE-EMPHASIS CONSTANT IS 0.937500.
     5. THE ORDER OF LPC ANALYSIS IS XX.
     6. THE FRAME UPDATE PERIOD IS XXX MILLISECONDS.
     7. THE SAMPLE RATE FOR THE SPEECH DATA IS XXXXX SAMPLES/SECOND.
     8. CODING OF THE LPC PARAMETERS IS DESIRED.

CHOOSE OUTPUT FILE FORMAT:  1, 2, 3, 4    {DEFAULT=3} : 3
     1 = TEXT FILE/VOICED & UNVOICED PARAMETERS PER FRAME/UNCODED RMS
     2 = TEXT FILE/SYNTHESIZER OUTPUT/CODED RMS/V & U PARAMETERS
     3 = CSEDIT OUTPUT
     4 = LPC-10 OUTPUT {FARMER}

CHOOSE THE NEXT STEP:  E, Q, C          {DEFAULT=E} : E
     TO EXECUTE THE PROGRAM,              TYPE "E"
     TO QUIT THE PROGRAM,                 TYPE "Q"
     TO CORRECT THE INPUT VALUES,         TYPE "C"

                    LPC ANALYSIS IS EXECUTING
```

(Screen 2)



3. LPC analysis requires a great deal of computing; thus, a professional-level minicomputer is needed to carry out all the steps within a reasonable time frame.

The LPC display is the most complex of the four (Fig. 2d). A code in the LPC-10 is characterized by 12 parameters called a frame—consisting of energy, pitch and ten reflection coefficients (K_1 to K_{10})—which are represented by 50 bits. The DH allophone needs four such frames to represent it, lines 2 to 5 in Fig. 2d; the IH needs three frames; the S# needs four frames; and so on.

The editor can interact at whatever level is most comfortable. Those without a linguistic background may choose to work solely at the text level, and do the editing by "creative" misspelling of words. Others with linguistic or phonetic knowledge may work with the phonemes or allophones. The engineer familiar with the synthesizer chip's operation might feel most at home with the binary LPC form (analogous to the machine code of a computer).

(When high-level languages were first introduced, engineers felt decoupled from the computer mechanism. The relationship between the program text and items such as memory location, register loading or unloading, and the actual logical or arithmetic operation in progress were too disconnected. The TI development system solves this problem by allowing the engineer to work at the "ones-and-zeros" level.)

In addition to the four representations of each utterance—or rather because of them—the development system also provides a history file to record transactions that change the initial contents of the four speech-data files. Not much imagination is required to appreciate the mess, should the different levels be tweaked (by different persons) to tune an utterance, without thoroughly documenting the changes; especially, since the development system can have several terminals at remote locations with access to the data files. Moreover, the history file is updated automatically—keeping all the editors involved "honest."

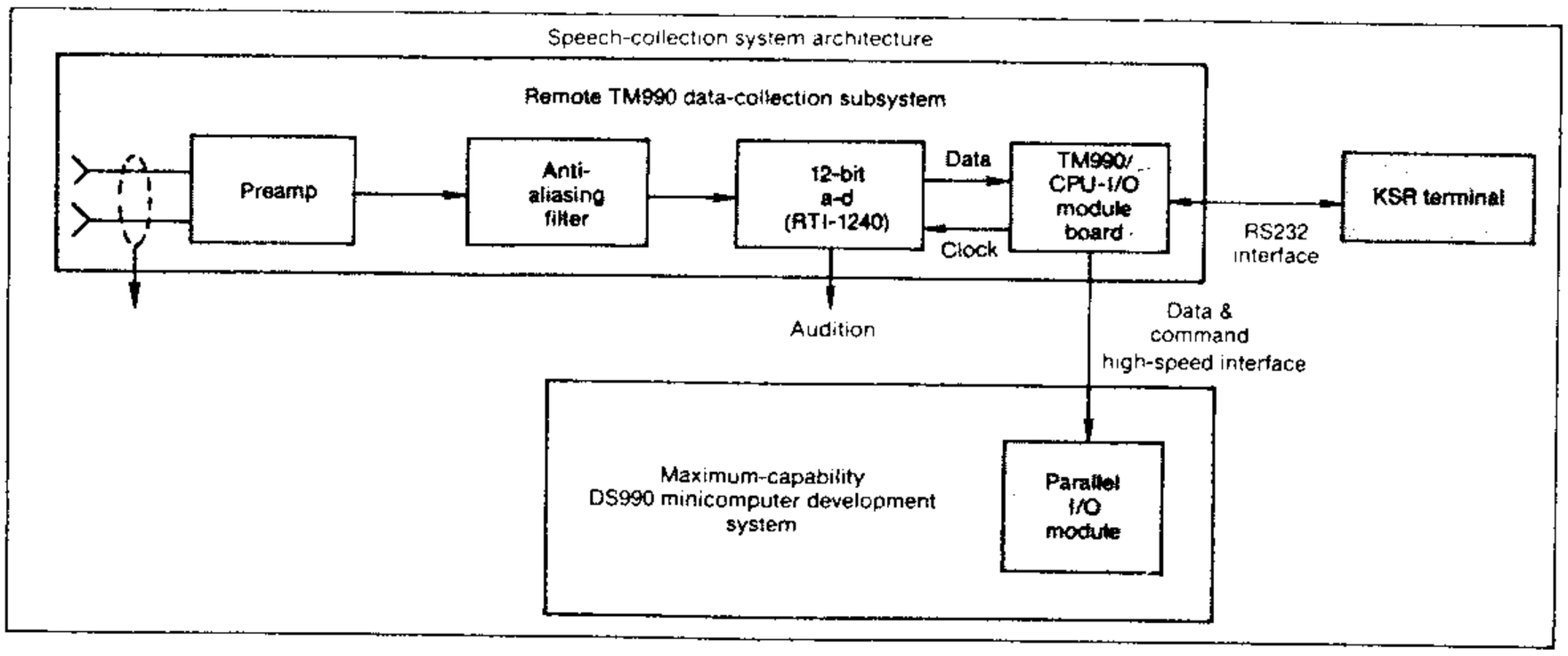
The combination of the five files—the four data files plus the history file—form a grouping called an "utterance," which then combine to make up the "library" for the group utterances in a speech.

From files to EPROMs

After the editing process has been completed, and the speech sounds the way it should, the next step is to format the file data for EPROMs (or mask-programmable PROMs for large quantities) for distribution to systems in the field.

A modest 500 or 1000-word speech, each word an average of 4 or 5 allophones long, would require a large memory to store the data sequentially. However, if one PROM carries all the allophones used in the speech, say, about 128 of them, and a second PROM selects the sequence in which they are used, the combined memory capacity required is far less—providing a very large vocabulary with minimum storage.

Of course, if just a few words are to be synthesized, this two-step allophone/sequence-PROM approach is



4. A portable acoustic-speech-data collecting system can be placed remotely at the customer's premises and interface with the data-processing center of the speech-synthesizer development system via a slow, standard RS232 serial interface or a much faster interface specially designed for the system.

not necessary, since the needed allophones can then be serially arranged in one small PROM and read out sequentially exactly as they are concatenated in the development system.

Interestingly, the allophone/sequence approach opens an important low-cost avenue for companies that do not want to invest in a full development system. Texas Instruments or other speech-service specialists can collect acoustic (or keyboard) speech data, analyze and edit them until satisfactory, and prepare an allophone vocabulary on a PROM, which the user then can combine with his own sequencer PROM to drive a synthesizer chip and generate speech.

Working from natural speech

However, to collect and analyze acoustic-sourced speech requires the second, more complex part of the development system—the speech-collection and LPC-analysis system—which presumes that the first text-to-speech system also is available for subsequent editing and PROM formatting. Although the text-to-speech part of the system can be carried out with a modest minicomputer arrangement (but substantial memory), the LPC analysis requires a great deal of computing (Fig. 3), and thus also a professional-level minicomputer like the DS990-Model 20, which contains a 990/12 CPU. Even so, the computer will lag behind real-time speech—taking about 12 s to process 1 s of speech—resulting in an overall throughput of about 100 analyzed words per hour, including recording-session time.

To save time, utterances are collected in a recording session, as previously collected utterances are processed by the development system. In this way, by the time a recording session is completed, the results of a previous one are ready for possible reiterated acoustic rework in a closed loop, until the desired product is attained.

Clearly, the speech-collection part of the system should be separate from the LPC-analysis and editing features. Indeed, a small and portable "box" contains the acoustic-collecting and an interfacing arrangement, which can be placed remotely in the customer's premises (Fig. 4) via a standard RS232 serial interface (which is relatively slow) for most computers, or a much faster interface specially designed for the systems.

This remote acoustic-recording setup not only collects speech data, but it also can play back the LPC synthetic-speech results. In addition, two video terminals—one for the speaker being recorded and the other at a "director's" location—control and prompt the speaker under the director's location—control and prompt the speaker under the director's supervision with displayed instruction, for instance, "speak the word" or "speak the phrase."

The director can select the utterances to be recorded or played back or re-recorded. When results sound good enough, the director can send the data to the computer for LPC analysis. After that, the director can listen to the results and decide whether editing or re-recording is needed (see Table 2).

Although the 5220 speech-synthesizer chip operates on an LPC-10 input, the development-system's software can analyze the acoustic speech data (sampled at rates of 7000 to 12,000 samples/second) into other LPC orders—from LPC-2 to LPC-15—so that it is not tied to any particular synthesizer chip.□

How useful?	Circle
Immediate design application	544
Within the next year	545
Not applicable	546

Software rules give personal computer real word power

Speech IC and algorithm for pronunciation let computer 'read' aloud words on its own display

by Kun-Shan Lin, Gene A. Frantz, and Kathy Goudie
Texas Instruments Inc., Dallas, Texas

☐ The field of electronic speech synthesis is widening. A new text-to-speech translator from Texas Instruments can "read" aloud messages typed into a computer terminal. The system's software analyzes the text into elements that are pronounced by a TMS 5200 speech synthesizer chip.

The translator will be first used in the TI 99/4 personal computer, where it will read aloud information displayed on the computer screen. Such data might include news or weather information received from a videotex or electronic mail service.

Unlike the company's Speak & Spell learning aid, the text-to-speech system does not use "canned" words and phrases stored in a read-only memory. Instead, it generates words from a stored library of 128 sounds, called allophones, which it concatenates, or connects to form speech. Text-to-allophone software chooses the allophones and determines their stressing and intonation. It then transfers this information to the speech-generating hardware, which translates it into speech sounds using a linear predictive coding technique.

At the core of this hardware is the TMS 5200 speech synthesizer and a central processing unit that delivers the data to the synthesizer in an orderly way—keeping track, for example, of times when the synthesizer buffer is almost empty. The CPU then takes fresh data and loads the synthesizer's internal buffer. The synthesizer, in turn, takes the data and produces the time-varying speech signals that are amplified to drive a speaker.

In addition, a TMS 6100 128-K read-only memory chip can store common words or sounds the designer may wish to include in the system.

The speech synthesizer can work with any standard 8-bit microprocessor, since it has been designed with a first-in, first-out buffer on board to store chunks of data, freeing the CPU for other tasks.

A typical text-to-speech system (Fig. 1) generates speech in two major steps—speech construction and speech synthesis. Speech construction is done in two stages. The first is the translation of the letters of the text into a digital representation of component sounds. The second is the concatenation, or stringing together, of these digitally coded sounds. These component sounds can be complete words or phrases or simply the elements of words (see "Speech construction tradeoffs", p. 123). Speech synthesis then converts the digital data into audible, synthetic speech. TI's approach constructs speech for later synthesis out of elemental component sounds.

A little linguistics

Deciding which components to use requires a closer look at speech sounds. When speech is pronounced, there may be hundreds of minor variations between sounds that are roughly categorized as the same. For example, the /P/ sound in pin is aspirated (followed by a puff of air), whereas the /P/ in spin is not. Sounds that are slightly different but generally perceived as the same in a language are called phonemes of that language. Subsets of phonemes that change slightly depending on the context or environment in which they appear are called allophones. Thus the unaspirated /P/ sound of spin and

Some speech-construction tradeoffs

In speech construction, a written sentence or phrase is analyzed to identify logically connected groups of sounds. Such groups may be speech components that are smaller than words, or they may be words themselves or groups of words such as phrases.

Speech construction is a subtle and complex procedure, since similar words have different stressing and intonation in different contexts. How well the task is accomplished depends not only on how well the rules of pronunciation can be modeled in software, but also on such factors as the amounts of memory available in the system and the size of a vocabulary it needs.

The three speech construction methods are phrase, word, and component-sound concatenation. Each technique has tradeoffs between speech quality, vocabulary size, and memory required. The larger the vocabulary allowed by a given construction technique, for example, the less the memory cost but the less natural-sounding the speech. Better-quality sound demands more memory and cuts into the vocabulary size.

In phrase concatenation, complete phrases are stored in memory and played back through a synthesizer. This method is sometimes referred to as analysis synthesis, since entire spoken phrases are analyzed to produce synthetic speech. The speech sounds natural since whole phrases at a time are recorded and the prosody—or rhythm—and subtle variations in pitch and loudness are preserved throughout the passage. However, storage and flexibility are problems because the phrases must be stored together and kept intact. Since only a finite number of phrases can be stored in a reasonable amount of memory, the vocabulary is limited.

Word concatenation offers more flexibility than phrase concatenation but at the expense of prosody and other qualities. As these are lost, the words tend to sound artificial when strung together as phrases. But the range of phrases that may be formed is greater for a given memory space, since words may be accessed and connected in any order.

TI chose a third technique called component-sound concatenation because it is the most versatile of all in generating words and sentences. Here a library of fundamental speech sounds, giving the virtually unlimited vocabulary needed for translating any typed passage into speech, is used. Because the sounds stored are basic to speech, almost any English word or phrase can be generated by concatenating the appropriate sound units from the library. The memory cost of creating a library and accessing it to create a word or phrase is almost insignificant. But a major difficulty lies in developing a method for connecting the components of speech sounds without sacrificing variations in rhythm, pitch, and loudness.

the aspirated /P/ sound of pin are different allophones of the same phoneme, /P/, and represent the sound more accurately than the phoneme. For this reason, the text-to-speech system here uses allophone stringing to form words and phrases.

In this system, an allophone varies from 50 to 250 milliseconds in duration and is coded according to the parameters needed for a speech synthesizer that uses linear predictive coding. There are 128 allophones in the library, including long and short pauses, which are coded for energy and filter coefficients, the parameters for setting the filter characteristics in the LPC synthesizer. The entire library takes up 3 kilobytes of storage.

Once the allophone library is established, a set of rules is needed for translating the ASCII text into an allophone string. TI used a set of rules based on one developed by the Naval Research Laboratory in Washington, D. C.

However, the NRL rules deal with phonemes only and have been altered in several ways to give the allophonic version of a phoneme in a particular environment. For instance, rules have been added specifying that certain allophones be used only at the ends of words.

In addition, rules have been included for better pronunciation of words often mispronounced with the NRL rules, like "create," "increase," "lost," and "human." The resulting set of about 650 rules chooses 97% of the phonemes correctly and 92% of the allophones correctly for a typical benchmark test. These rules use 7 kilobytes of storage.

The speech construction program strings the allophones and smooths the transitions between them. Energy levels between allophones are matched to obtain a smooth contour, and filter coefficients are smoothed to make transitions from sound to sound less abrupt.

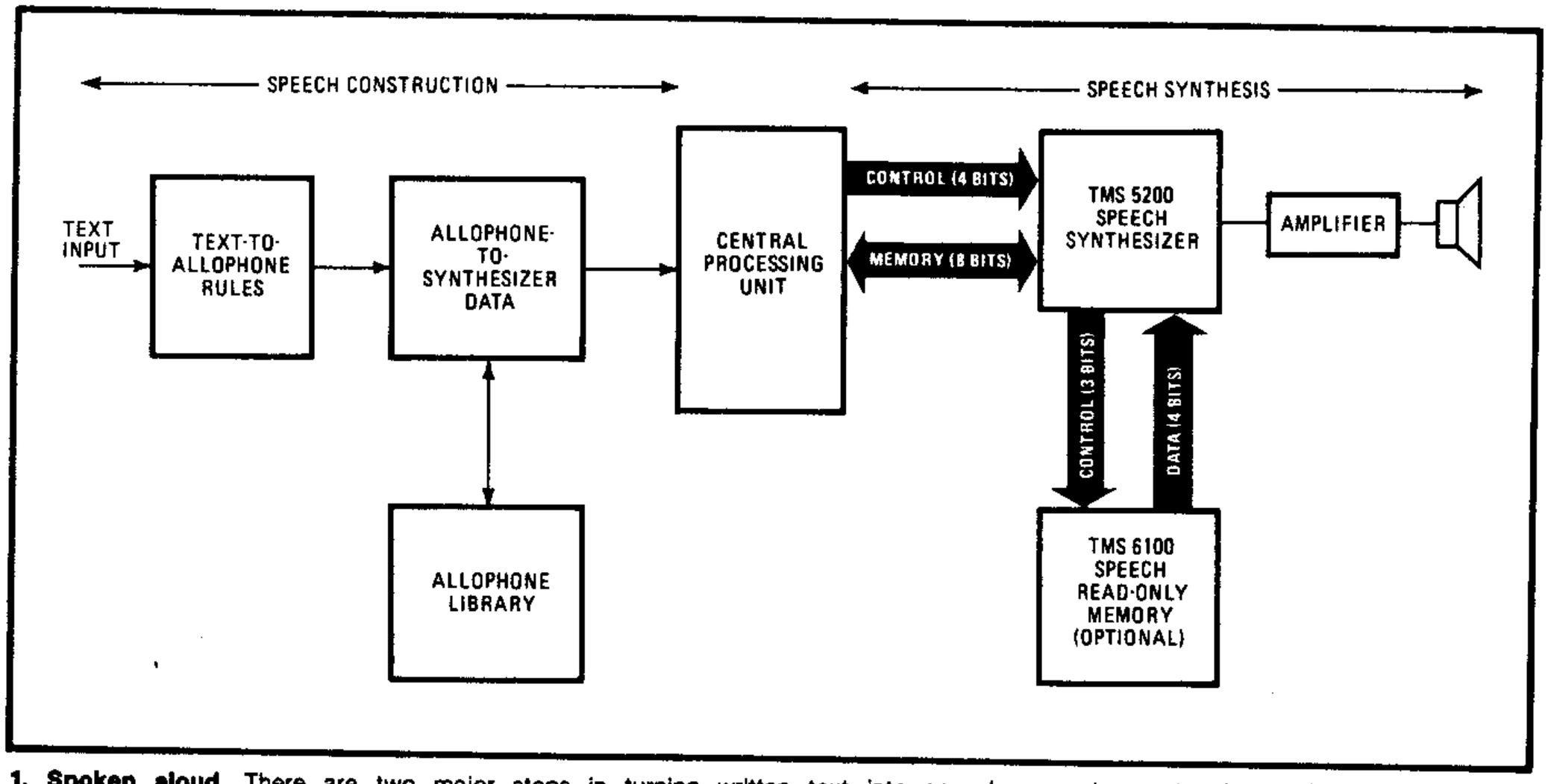
Once the allophones are concatenated, the quality of the speech depends on the stress and intonation patterns applied to the string. Since randomly stressed English sounds unnatural, both stress and intonation must be precisely applied. The pitch assignment is made by a speech-construction algorithm with only the stressed syllables being indicated by the user, who adds the needed pitch pattern to particular points in the sentence.

The control of inflection is based on gradient pitch control of the stressed syllables—that is, stressed syllables of a sentence can be thought of as lying along a line of pitch values tangent to the line of pitch values of the unstressed syllables. In a neutral intonation, the unstressed syllables would lie on a mid-level line of pitch while the stressed syllables would lie on a downward slanting line somewhat higher in pitch than the base line. The slope is constructed in software and the user need only mark the stressed syllables.

Delivering the speech

The speech construction data is used with one of several speech synthesis techniques that generate actual voice sounds. There are two speech synthesis approaches, waveform encoding and parameter encoding (See "Synthesis: another way to go," p. 124)

One of the parameter-encoding—or frequency-based—speech synthesis techniques is the channel vocoder method, which divides a speech signal into narrow frequency bands using a bank of bandpass filters and then stores the amplitude at each center frequency. These amplitudes, along with a variable-frequency source, control a bank of narrowband frequency resona-



1. Spoken aloud. There are two major steps in turning written text into speech—speech construction and speech synthesis. Construction, done in software, resolves the text input into electrical signals that the hardware-based synthesizer converts into "speech."

Synthesis: another way to go

Waveform encoding attempts to reproduce the amplitude-varying signal of natural speech by generating a similar waveform, in contrast to parameter encoding, which represents a speech signal in terms of frequency, or the spectral components of natural speech rather than its amplitude characteristics. Several such techniques are pulse-code modulation, delta modulation, and an amalgam of techniques called Forest Mozer's technique, after its inventor at the University of California at Berkeley.

The simplest waveform-encoding technique is uncompressed digital data recording, referred to as pulse-code modulation. Here the analog speech waveform is sampled and converted into digital information by an analog-to-digital converter. Once in a digital format, the speech signal is stored in memory and played back through a digital-to-analog converter and a low-pass filter. The problem with using PCM alone is that memory requirements quickly become excessive. The average data rate is 96,000 bits for 1 second of speech.

An alternative to PCM, delta modulation, compresses the amount of data needed to record speech digitally. As in PCM, the analog speech waveform is sampled, but this time only the changes in amplitude between samples are stored in memory. Since these changes are usually smaller than the absolute values of amplitude, the overall data rate is less than that of PCM. Thus delta modulation reduces the amount of memory required to store a library of words or phrases.

A simple delta modulation system uses a fixed step of change, or delta, in tracking the original speech waveform—the delta from the previous analog value is always some multiple of a fixed step size. The drawback is that for small changes in amplitude, errors in the form of quantizing noise are introduced.

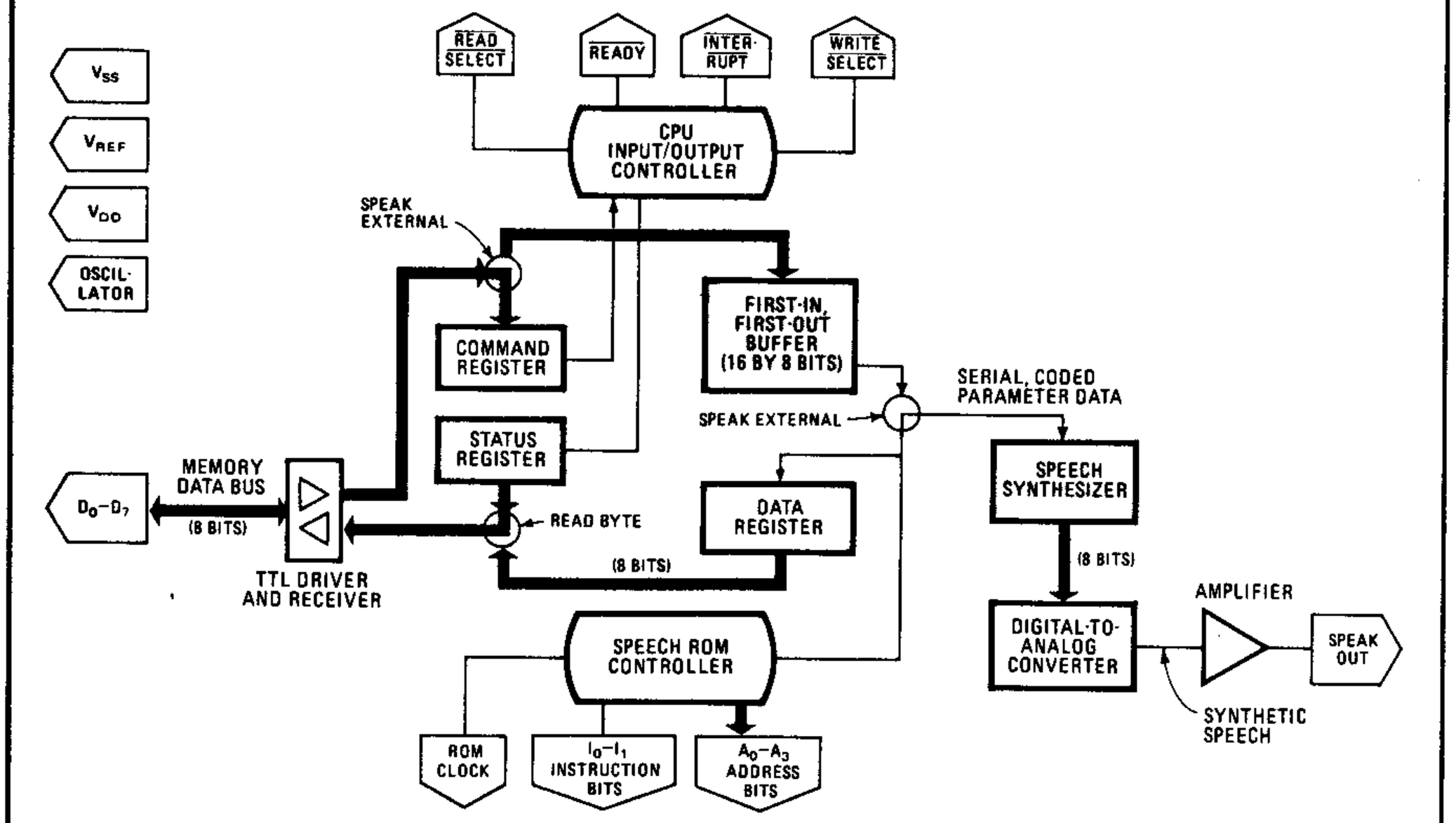
Delta modulation synthesis is improved by making the step size variable and proportional to the difference between the successive samples. Thus, smaller successive changes in amplitude are accurately tracked, using smaller quantizing steps when necessary and so reducing quantizing error or noise. This technique, dubbed continuously variable slope-delta modulation, produces data rates of 16,000 to 32,000 bits per second.

Finally, Forest Mozer's technique uses waveform compression extensively, and its data rate is about 2,400 b/s. The approach combines several techniques, taking advantage of two speech-perception characteristics. First, voiced speech is periodic, containing redundant information, and, secondly, listeners are insensitive to both phase and low-amplitude information. Thus the designer can strip all redundant information from the speech waveform [*Electronics*, April 10, 1980, p. 113], leaving only the most fundamental data for storage in memory.

tors corresponding to the vocoder's own bandpass filters. The data rate of a channel vocoder is typically 2,400 b/s.

Formant synthesis, the second parameter-encoding scheme, emulates the frequency response of voiced speech by generating sharp energy peaks at specific frequencies called formants. The amplitudes and bandwidths of these formants are recorded and used as inputs to excite a formant-based synthesizer. Formant synthesis typically has a data rate of approximately 500 b/s.

A third technique, linear predictive coding [*Electronics*, Aug. 31, 1978, p. 109], was chosen for this text-to-speech system. LPC is essentially a mathematical model of the human vocal tract, implemented as a filter network. The coefficients of the linear equations of the filter used in the model are calculated in the analysis of the original speech and used in the model to control the



2. Enhanced. Control circuits around the synthesizer simplify its interface with an 8-bit microprocessor. The synthesizer exchanges data and commands with the microprocessor along an 8-bit bus and four control lines. External 128-K ROM can also store word data.

"shape" of the vocal tract in speech reproduction. The parameters stored are the filter coefficients, the filter gain, and the frequency of the excitation source used to drive the filter. Good-quality speech is achieved using LPC at data rates from 1,200 to 2,400 b/s.

The 28-pin p-channel MOS TMS 5200, the main hardware component of the text-to-speech system, is a second-generation version of the TMS 5100 speech synthesizer. Several features (Fig. 2) have been added, including a 16-by-8-bit first-in, first-out buffer, to increase its flexibility when it is used with the system central processing unit.

The CPU interface shown in the figure consists of an 8-bit bidirectional data bus (D_0–D_7), read and write select line ($\overline{RS}$ and $\overline{WS}$), a ready line for synchronization (READY), and an interrupt line ($\overline{INT}$).

Activity on the memory data bus is controlled by the read and write select lines. When data is stable on the memory data bus, the ready line will go low to indicate that the CPU may complete a data transfer to or from the synthesizer.

What causes interruptions

The interrupt line indicates a change in the TMS 5200 status that may require the attention of the CPU. For example, the interrupt line, which is normally high, goes low to indicate either the end of speech or that the FIFO buffer is low.

The CPU interface consists of two input-holding registers (the command register and FIFO buffer) and two output-holding registers (the data and status registers).

The command register receives an instruction from the CPU and holds it for the TMS 5200 to interpret and execute. The 128-bit FIFO buffer, organized in a 16-byte parallel-in, serial-out format, holds the speech data used when executing the speak-external command. The buffer is low (BL status) when the FIFO is more than half empty, and the BE status means it is completely empty. When the BE is set, speech is terminated to prevent the synthesizer from processing invalid data.

The data register is an 8-bit serial-in, parallel-out holding register. It is used when speech data is transferred from an external ROM, such as TI's TMS 6100, to the CPU.

The 3 bits of the status register send information about the synthesizer to the CPU to indicate talk, BL, and BE conditions. They can be queried any time except during a memory read command. The CPU passes commands to the TMS 5200 by way of the memory data bus.

For the text-to-speech application, the only significant command is SPEAK EXTERNAL, which lets the CPU, rather than an external ROM, supply speech data to the synthesizer. Upon receipt of this command, the TMS 5200 loads the FIFO buffer with data from the CPU. The synthesizer remains idle until BL becomes false, at which time speech begins and the talk status is set. Data will continue to be taken from the FIFO until a stop code is encountered or the buffer becomes empty. During the execution of a SPEAK EXTERNAL command, no other commands are recognized by the TMS 5200.

The audio output of the TMS 5200 8-bit digital-to-analog converter delivers up to 1.5 milliamperes with a 1.8-kilohm resistor to ground. This signal can then be filtered and amplified to drive a speaker. □