Speech Synthesizer

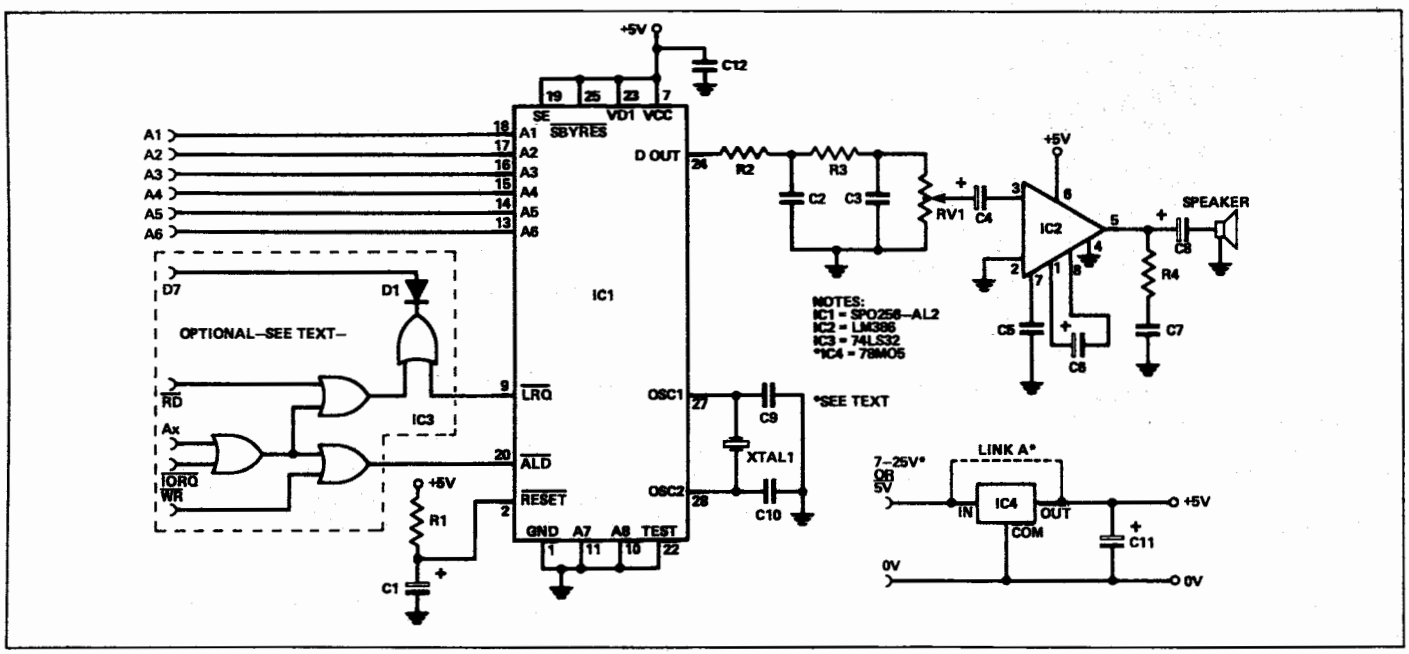
By A. Hubert

OUTPUT from the IC comes in the form of a pulse width modulated signal from the "DIGITAL OUT" pin of the IC. In the circuit (Figure 2) this signal is converted from a PWM signal to an analogue one, and filtered by the 5kH3 low pass filter formed by R2, R3, C2 and C3. RV1 is the volume control, the resulting signal passing through IC2 which, with surrounding components, amplifies the signal to drive a speaker.

R1 and C1 provide the power-on reset pulse for IC1, XTAL, C9 and C10 form a simple clock circuit for IC1. C12 is a decoupling capacitor for IC1.

The rest of the circuit will be determined by which computer you are interfacing to. For the ZX81, an additional IC (IC3) and a diode (D1) are required. The additional circuitry locates the unit in the Z80's I/O map, using the simple Sinclair method of assigning each I/O device a selecting address line. A7 is used in this case, mapping the unit at I/O address 127. When read, the circuit gates that LRQ signal onto bit 7 of the data bus via D1

Provision has also been made on the PCB, for an on-board 5V regulator. This is especially useful for Sinclair computers where the internal regulator tends to run hot in use. The regulating circuit simply consists of IC4, a 78MO5 voltage regulator that will accept input voltages from 7V to 25V, and supply up to 500mA. C11 decoupled the output of this IC. If you don't require this portion of the circuit, simply make a link to the two outside pins of IC4.



Designer's Notebook

The techniques and integrated circuits used to synthesise speech by computer.

By Ron Keeley

OF THE three speech synthesisers built to interface with personal computers, those from Texas Instruments and General Instruments use the method called Linear Predictive Coding. National Semiconductor's Digitalker is quite different.

It uses a digital version of the "synthesis by numbers" method together with very effective data compression techniques that reduce the amount of information needed to characterise a word by as much as 100:1.

The word is synthesised from this data by driving a digital-to-analogue converter with information that defines the shape of the waveform and its duration. The process begins by extracting the spectral information from a sample word, followed by "hands on" analysis and manipulation.

For a start, the fact that a speech waveform may not change very much over a period of 10mS or even 100mS means that much of this redundant information can be replaced by a few bits that cause the waveform to be repeated for the required time. In some cases even the number of bits that specify the spectral information can be reduced, and because very little information is carried by the higher frequencies, these can quite often be eliminated altogether, along with particularly low amplitude sections of the waveform (which correspond to quiet segments in speech).

Another technique is to shuffle the harmonics so that they are all in phase, as far as possible (this would bring most of the peaks and dips into line along the time axis). Phase alignment reduces the data by half, and although this results in considerable phase distoriton, the ear is not very sensitive to it.

The Digitalker system is very efficient and produces probably the best quality speech of the three systems available. The data processing is carried out by skilled operators who literally 'play around' with the speech sample, removing and compressing the data as much as possible while retaining good quality speech reproduction.

Speaking In Code

The speech synthesis from both Texas Instruments and General Instruments are based on a very powerful method called Linear Predictive Coding. LPC is an efficient mathematical technique for extracting information about speech waveforms that results in very compact data. It is based on a fairly simple idea from the theory of digital signal processing, but after that it gets very complicated, very quickly.

It starts with the idea of the predictability of waveforms; any periodic waveform such as a sine wave, or any quasi-periodic waveform like a short-term speech segment, is predictable. By looking at one representative segment you can make a pretty good guess about what the waveform will be doing later.

LPC is based on an idea that takes this one step further; if the sampling rate of an ADC is increased, the difference between successive samples becomes quite small (Figure 1), and for a periodic waveform you can make a good prediction of the value of a sample based on a knowledge of the values of the past few samples.

The basis of LPC, then, is that you can predict a present value for a characteristic of any waveform, based on a knowledge of previous values — in other words you can predict the behaviour, spectrum, or whatever, of any periodic waveform as long as you know something of its past performance. This can be done by means of a 'predictor equation' which combines past values with a prediction coefficient — but that's just the beginning of LPC. The problem, is to find the correct coefficients!

So how does this apply to speech synthesis? Well, suppose you had a synthesis model like that in Fig. 2. You can see that

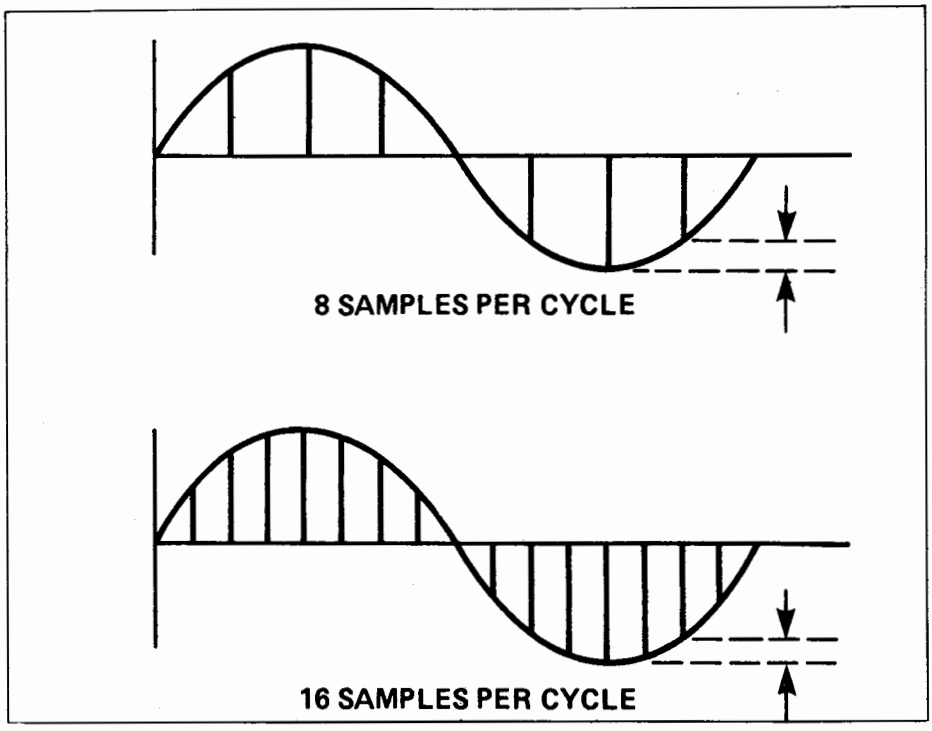


Fig. 1. The difference between successive samples in A-D conversion is smaller as the sampling rate increases.

it corresponds closely to a variable transmission line except that the electronic components have been replaced by a digital filter.

A digital filter is the computer equivalent of an analog filter; its inputs and outputs are data bytes representing analogue values, and 'filtering' is performed by carrying out a series of calculations with the data. Any analog filter — high pass, low pass, bandpass, etc. — can be described by a voltage transfer function which expresses the relationship between the output and input voltages. In a digital filter — low pass, bandpass and so on — the transfer function becomes a 'difference equation' which expresses the relationship between the present output and input in terms of past inputs or outputs (sound familiar?). The process of solving the difference equation corresponds to passing a signal through the filter; in practice this is done by performing a number of additions and multiplications which combine the present input with a number of past outputs, as shown in Figure. 3. Figure 4 shows the operations performed by a single stage of the filter in Figure 3; the important thing to note about this is that the value of the multiplication coefficient K varies the effect of the filter.

As you've probably guessed, the digital filter in Figure 2 is in fact a ten-stage digital lattice filter. Now, all filters can be described in terms of their frequency response, or in terms of their spectral response (if the input to a filter was a waveform containing an infinite number of harmonics all of the sale amplitude, its output would contain only a certain number of harmonics at different amplitude, and these would define the spectral response of the filter).

So, if we could match the spectral response of our filter to the spectrum of an actual short-term speech segment, we would have a near-perfect model of the spectral response of the vocal tract that produced our speech sample. The problem is to find the filter coefficients K, that will produce the right response. One way to do this would be to make a guess at the values, refining your guesses over a number of attempts (in a similar fashion to finding a square root by Newton's method) until the correct response was produced — but this would take a long, long time, even with the fastest, most powerful computers.

Fortunately all this effort is not needed, because the problem of finding the filter coefficient is equivalent to finding the predictor coefficient in LPC! As we hinted earlier, this is not exactly easy — the predictor equation has to be expanded to ten simultaneous equations in ten variable, which have to be solved for the predictor coefficients. However each

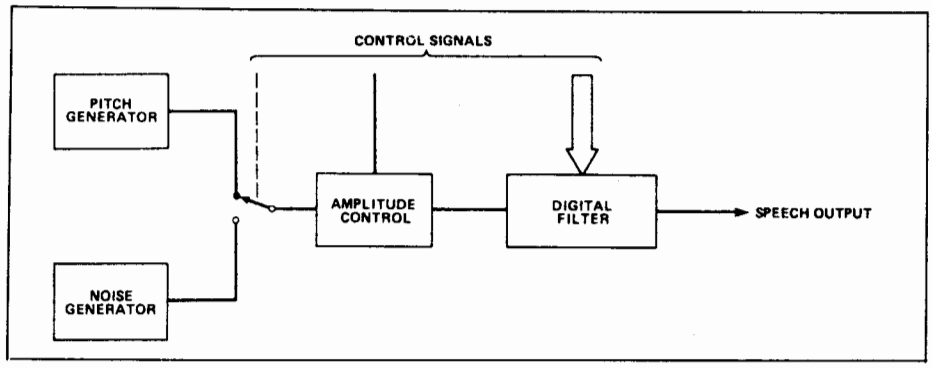


Fig. 2. The block diagram of a speech synthesiser similar to the articulatory model, but using a digital filter.

equation can be written in terms of known values, derived from a real speech segment, and the unkown coefficient, so each equation contains only one variable. Thus LPC provides a direct and relatively fast method for finding the filter coefficients, and it also provides a closer match to a smaple speech segment spectrum than any other method.

The only element missing from our near-perfect digital speech model (even LPC methods do not reveal the whole truth about the characteristics of speech!) is the excitation — the sound sources that correspond to fricative and voiced speech sounds (plosive sound sources are not used in any speech synthesis system; they are formed from the other two sources).

Ideally the excitation would be a true representation of actual speech sounds; in practice a match can be obtained through LPC methods. Effectively, the excitation is obtained by inverse filtering through the LPC-derived model. Imagine the peaks and dips of the filter as being inverted — rounded peaks becoming sharp notches and notches becoming rounded peaks. Then if the real speech segment is pushed back through this inverse filter, what comes out is a LPC representation of the required excitation signal!

Unfortunately the spectral characteristics of voiced sounds are as complicated as those of the vocal tract itself and would require just as much processing and data storage space if they were to be modelled exactly. In practice, therefore, only the frequency (pitch) and energy (loudness) data are encoded and stored.

Chips For Speech

Now that we have an idea of how LPC works, we can take a look at how it is used. Figure 5 shows a block diagram of Texas Instruments' TMS5100IV (now replaced by the TMS5110 and 5220), and the General Instruments SP-0256 chip is shown in Figure 6. At the heart of both ICs is a digital lattice filter that models the vocal tract, using LPC-derived speech parameters.

Texas Instruments speech synth chips are basically word synthesisers (that is, the program cannot call up less than a whole word), but they go about it in a cunning way. The process starts with breaking down a sample word into 20mS slices that Texas call frames. Each frame is then LPC encoded to extract the coefficients for the digital filter and for the excitation generator. A set of speech parameters are then stored inan external ROM, from where they can be called up by the synthesiser chip to control the synthesis functions. However the ROM speech parameters are not filter coefficients, but a set of pointers to the filter coefficients, which are actually stored in a ROM section on board the speech chip itself! this is because, over a large sample of everyday words, it turns out that most if not all speech can be synthesised from a limited set of filter coefficients.

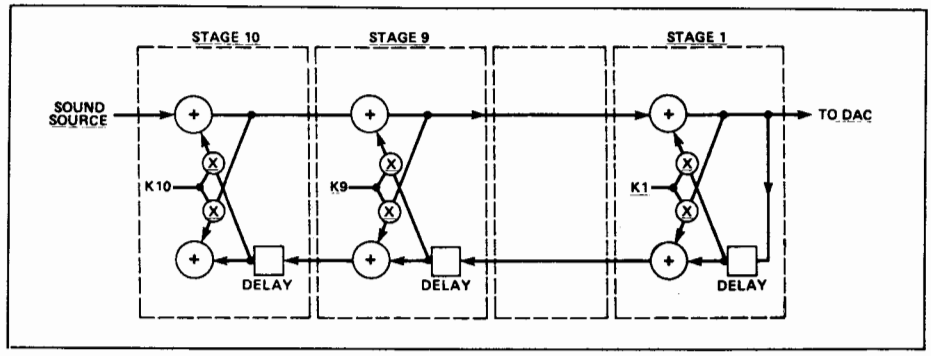


Fig. 3 A ten-stage digital lattice filter. The output is a function of the present input, and combined previous outputs.

A new set of speech parameters is called up every 20mS to model the time-variable filtering properites of vocal tract, but because of the repetitive quality of speech waveforms, a full set of 12 parameters (49 bits) is not always needed; when the speech waveform is essentially stable of changing only slowly for a period of more than 20mS, as few as four bits can be called up to maintain the synthesised waveform.

Incidentally, a set of coefficients corresponding to a 20mS frame does not necessarily correspond to a phoneme, an allophone, or any other conventional unit of speech; all such units are derived from phonetic analysis, and they may well extend for more or less than 20mS time period that Texas Instruments assing to their frames.

The parameters, when called up by the synthesiser, are first loaded into an in put register and then transferred to a block of RAM. These parameters point to LPC coefficients in the parameter look-up table, and these 10-bit values passed serially to the parameter interpolation section. The purpose of this block is to modify the coefficients so that one 20mS frame blends smoothly into the next. Without this function each frame would be distinct from its neighbours and the resultant speech would consist of a series of disjointed robot-like sounds.

The output of the interpolation block directly drives the excitation generator (this provides the digital equivalent of voiced and unvoiced speech sounds) and the 10-stage digital lattice filter. The output of the filter drives a digital-to-analogue converter to produce a representation of sound that, at least, we can all understand.

The chips also contain decoding and control logic (much of which is now shown in the diagram). The decode logic block works in conjunction with the paramter RAM to select the required coefficients from the look-up table, while the condition latches control the complex sequence of calculations that take place in the filter section. Outputs from this logic also control the interpolation function, via the interpolation controller.

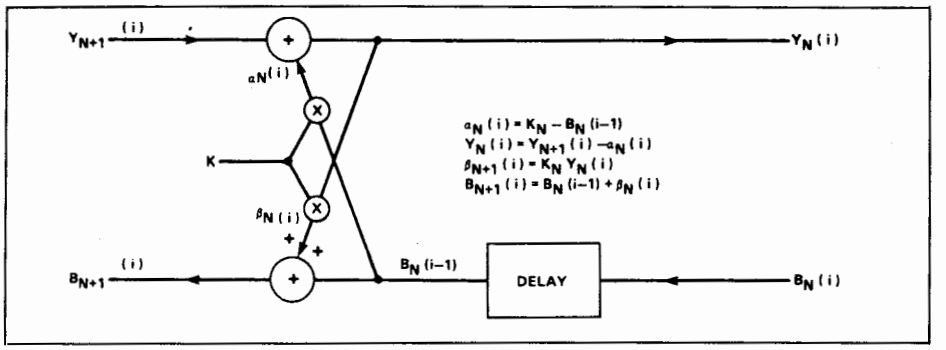


Fig. 4. One stage of the lattice filter in Figure 3, in detail.

Two further points are worth noting; the first is that this system provides high quality speech in a very efficient manner, and the second is that the coefficients which a responsible for this are unfortunately not accessible, in any meaningful way, to the programmer.

The General

General Instruments' SP-0256 chip is basically similar to the Texas Instruments products, though they differ in the way LPC data is sorted and manipulated.

For a start the SP-0256 stores LPC coefficients for 64 allophones in an on-board ROM that is directly accessible to the programmer. This provides a form of synthesis by rule — that is, the programmer enforces the rules by selecting the correct allophone! This system is more flexible than an word-oriented one, with a nearly unlimited vocabulary, but needs considerable programming skills to achieve the quality of speech produced by word-synths (details of the allopones provided, and methods for stringing them together, can be found elsewhere in this issue).

General Instruments also provide different versions of the SP-0256 with word vocabularies for speaking clocks and telephone message generation. Another version to be released shortly departs from allophone synthesis; instead the ROM will be coded with LPC data corresponding to demi-syllables (half syllables), which carry more information about the transitions between speech sounds and therefore provide smoother, more realistic speech synthesis.

Speaking Of The Future. . .

This has been a long and winding tour through the subject of speech synthesis — unfortunately its a long and very winding subject — and the story is far from over. The integrated circuits available now represent a considerable advance on the technology of ten years ago, of the order of cramming a roomful of equipment into a standard 28-pin IC package, and in the future we can reasonably expect at least the same scale of integration. ■

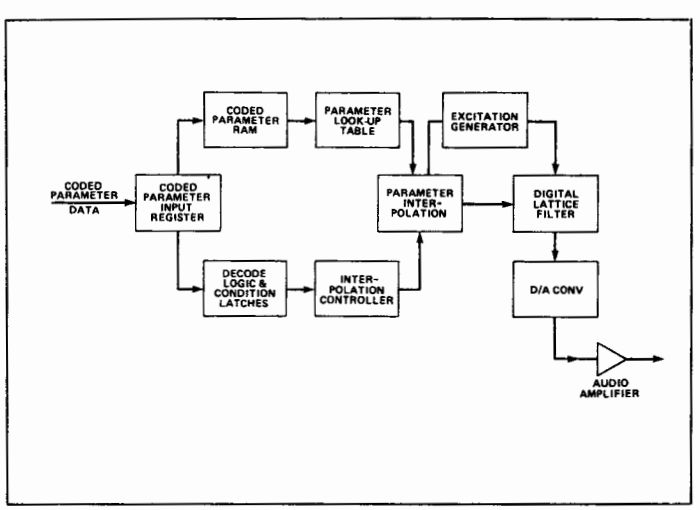


Fig. 5. A block diagram of Texas Instruments' TMS 5100 speech synthesiser IC. Not all the functional blocks are shown.

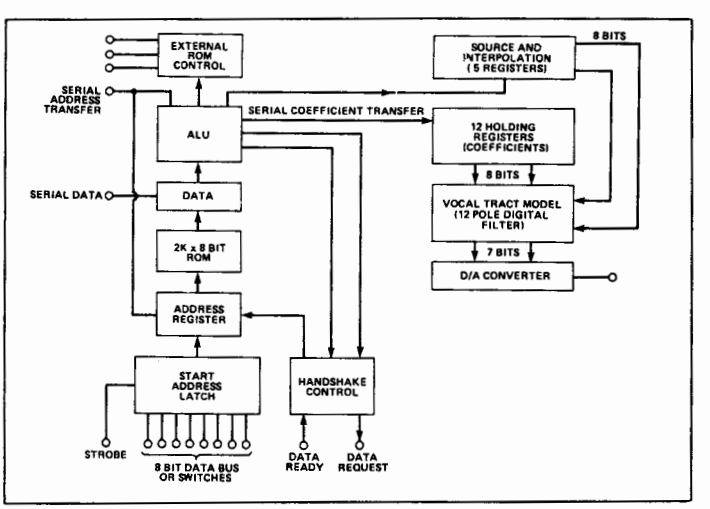


Fig. 6. A block diagram of General Instruments' SP0256 speech synthesiser IC.

COMPUTER SPEECH

Tim Orr takes a rest from his circuit mania this month to explain how speaking machines are moving off the TV screen and into the home computer market.

COMMUNICATION VIA SPEECH is a tremendously efficient way of transmitting information. A computer terminal with just a VDU or a hard copy printer compels the operator to be continually looking at the display. This limits the operator's freedom to do other jobs, such as controlling equipment, reading literature, typing, etc. If the computer had the option of being able to talk how much easier many operations would become. VDU's could also 'talk' their data and computer games could speak their instructions.

Computers have had this 'speech' option for many years, but as technology has improved, the size and cost of the equipment has been reduced to realistic proportions and the speech quality has got better. The microprocessor boom has helped this process and there are now several peripheral plug-ins that can be made to talk and even 'listen and understand'!

ROM For Improvement?

There are many methods by which a computer can generate speech. Some systems use a library of stored spoken text on a disc, just as the speaking clock does. Short phrases and individual words are sequentially selected by the computer programme and strung together to form the desired sentence.

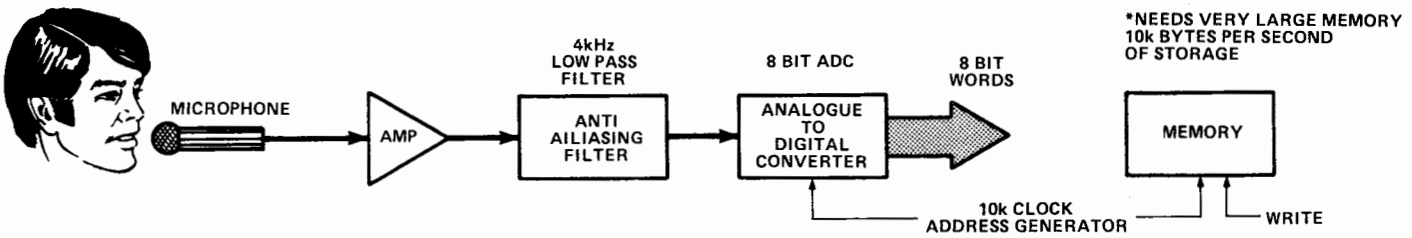
This technique is fine for some applications, where the set of phrases is small or where there will be no need to change them, because this means changing the disc. However, the unit is physically large and suffers from all the faults of any mechanical system.

An all electronic method of speech storage can be implemented using ROM's. Spoken words can be converted into a digital code (using an ADC), and programed into a ROM. Various words and phrases can then be selected by the computer and used to generate sentences by converting the reassembled data back into analogue information. This technique is the same in concept as the disc method, only the storage medium is electronic.

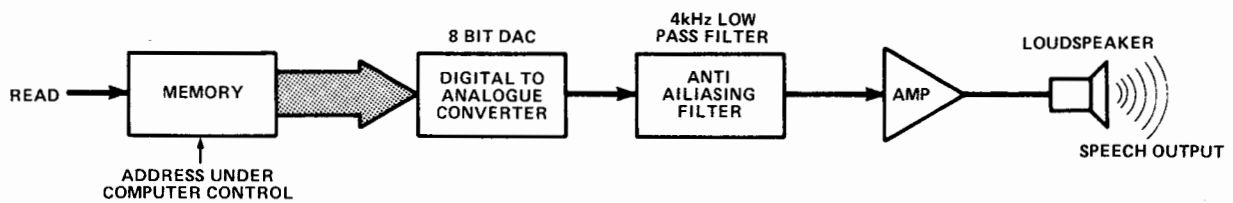
However, this type of storage would require enormous amounts of memory to generate short pieces of speech, because the unfortunate fact of life is that about 95% of the information stored by this method is redundant. The redundancy problem can be overcome by doing some special coding on the information. Linear predictive coding is one such technique, and this can result in very efficient ways of storing speech.

As A Rule

Yet another method of generating speech, which certainly gives the most versatile output (and is undoubtably the most complicated solution) is SPEECH SYNTHESIS BY RULE, using a speech synthesiser model controlled by data from the computer. ▶



Block Diagram of the digital method of achieving voice storage.



The phonetic code reads almost as if it were written in English (maybe someone will write a program to convert English to phonetic code?). Before discussing the speech program or the synthesiser it is desirable to explain just how human beings generate speech.

The Vocal Tract

Speech production has been studied for centuries and there have been many historical examples of 'mechanical talkers', that is mechanical models that can be manipulated so as to produce synthetic speech. These models generally have employed bellows, reeds and moveable acoustic resonators to synthesise the speech sounds and this is not too dissimilar from the real thing, the vocal tract, Fig. 1.

Air from the lungs is expelled through the vocal cords causing them to vibrate (when you breathe in the vocal cords don't vibrate — try it!). These vibrations produce a buzz which the speaker can control in pitch and volume. This buzz is coloured by a set of acoustic resonators known as the vocal tract.

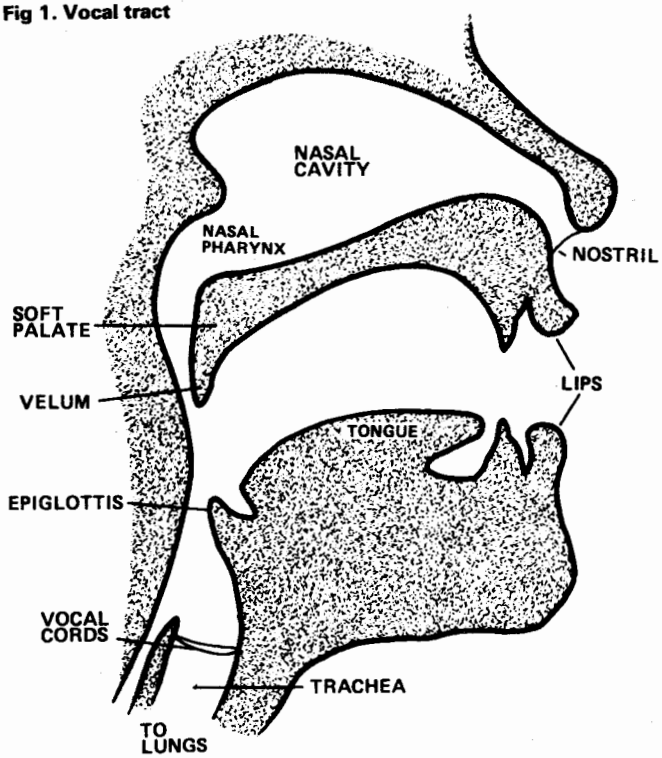
By opening and closing the mouth, by moving the tongue hump and by connecting or disconnecting the nasal cavity, the resonances of the tract can be manipulated so as to generate speech.

Take, for example some steady state vowels, AE as in HAD, EE as in HEED and OO as in WHO. Fig. 2 shows the acoustic frequency response for various vowels.

The operator types a phrase that is to be spoken. The phrase is spelled phonetically — it usually takes an operator a few hours to come to grips with the new way of spelling — and the computer converts the phrase into a series of parameters which control the speech synthesiser.

For example, the phrase 'Well, it can do with me' would be typed in as 'WEHL IHT KAAN DOO WIHTH MEE'.

Fig 1. Vocal tract



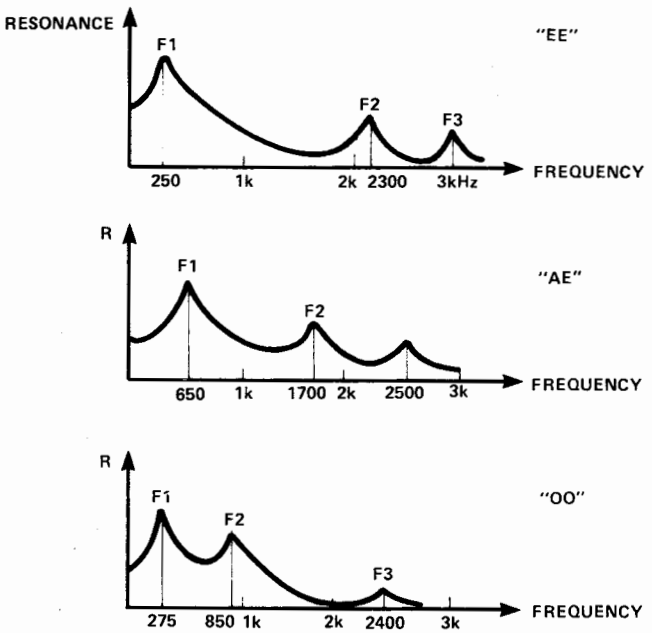


Fig 2. Acoustic response of some vowel sounds.

The first three peaks in the response, F1, 2, 3 are known as the first three formants. These are frequencies at which major resonances occur. For example, the 'OO' vowel has F1 and 2 close together at a low frequency and so the overall effect is a low frequency resonance. This is obtained by almost closing the mouth and pushing the tongue hump to the bridge of the mouth, whereas the 'AE' vowel is generated by opening the mouth and lowering the tongue hump.

Filter Vowels

It is possible to synthesise vowels by making an electronic model using active filters. If three band-pass filters (Q=5) are cascaded one after the other, set at frequencies of 660Hz, 1720Hz and 2410Hz and a saw-tooth wave form (100Hz) is injected into them, the resultant waveform will sound like the 'AE' vowel as in HAD.

A list of vowel resonances is given in Fig. 3. Note that they are for a typical MALE speaker.

A woman's voice is different in two respects. The resonances are about 10% higher because the vocal tract in women is about 10% smaller than that of a man.

Fig 3. Listing of vowel resonances.

		FORMANT (ALL IN Hz)		
		F1	F2	F3
HEED	EE	270	2290	3010
HID	I	390	1990	2550
HEAD	E	530	1840	2480
HAD	AE	660	1720	2410
HOD	AH	730	1090	2440
PAW	AW	570	840	2410
HOOD	U	440	1020	2240
WHO	OO	300	870	2240

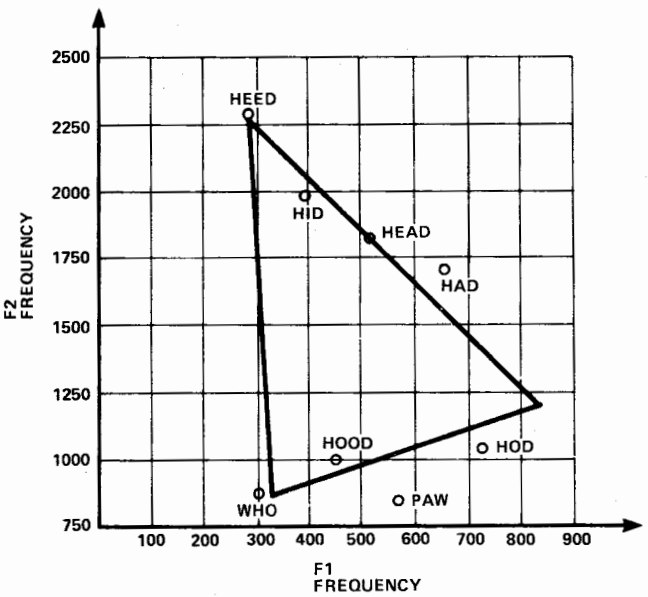


Fig 4. The vowel triangle!

Second, the pitch of the speech is perhaps an octave higher. These two effects characterise female speech as distinct from male.

Note that the formants 1 and 2 move over quite a wide range, but F3 doesn't move much at all.

However, including F3 in a model does help to improve the intelligibility. If we plot out F1 versus F2, we get what is called the 'vowel triangle', Fig. 4. Try gliding from the PAW vowel to the WHO vowel. The resulting dipthong is that which is found in HOW. Others are:— BAY, BUY, BOY and HOE.

Say Through The Nose?

When the mouth is closed, virtually no sound comes out of it(!) However there is a secondary path via the nasal cavity, which is available when the velum is open. The group of sounds generated via this route are known as NASALS. They include such sounds as 'M' as in MAN, 'n' as in NUT and 'ng' as in STING. The nasal cavity is virtually a static resonator and so all nasal sounds have an undynamic quality about them.

Vowels, dipthongs and nasals are all voiced sounds, that is they are all pitched, being generated by the vocal cords. There is a group of sounds called fricatives which are pitchless and are generated by blowing air between the teeth and lips. These sounds are the 'th', 'f', 's' and 'sh' noises and are very similar to bandpass filtered noise. 'Th' can be modelled by a bandpass filter at 8kHz whereas at the lowest frequency, 'sh' is modelled by a 2k5 Hz filter.

Constantly Talking

There are many other types of sounds but for the purposes of brevity we will consider only one more, the STOP CONSONANT. These sounds are characterised by a sudden opening of the mouth. This produces two effects.

One, there must be a period of silence (if only briefly), before the sound is generated.

Two, as the mouth opens, the formants rapidly move toward temporary target positions.

The stop consonants, 'T', 'P', 'K', 'D', 'B', 'G' are shown in Fig. 5. The vowel 'AH' has been used in this ▶

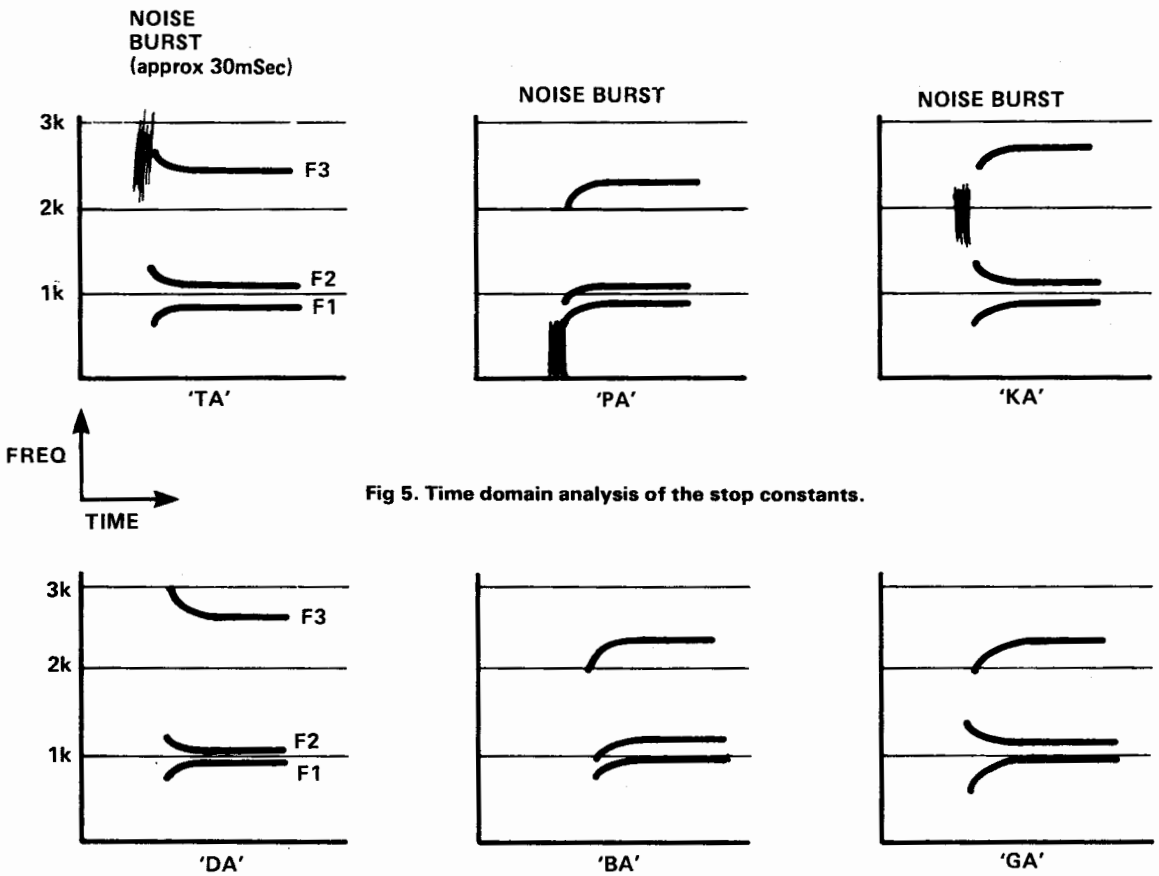


Fig 5. Time domain analysis of the stop constants.

example and so the stop consonants are 'Ta', 'Pa', 'Ka', 'Da', 'Ba', 'Ga'. The first group are characterised by having a small noise burst which preceeds the opening of the mouth.

This burst only lasts for about 30 to 50 mS and it has a different resonant frequency for each of the examples. However, it is a very important phonetic element and does much to characterise the sound.

The lower group of stop consonants has no noise burst. This is the major difference between these two sets of sounds.

Verbal Circuits

Well, that's the end of the very rapid phonetics lecture, now for the electronics. The speech synthesiser must be able to model the vocal tract. It needs a voltage controlled oscillator, a noise generator, a controlled fricative formant, a controlled set of formants F1, 2, 3 and a nasal resonator. There are 9 parameters in this model which need controlling. These are:—

AH — amplitude of aspired sounds.
AV — amplitude of vowels sounds.
AF — amplitude of fricative sounds.
AN — amplitude of nasal sounds.
F1 — frequency of formant 1.
F2 — frequency of formant 2.
F3 — frequency of formant 3.
Ff — frequency of fricative formant.
Fv — frequency of oscillator.

The model is known as a serial 3 formant synthesiser with parallel fricative and nasal formants. The computer delivers data which is converted into 9 voltages which represent the 9 parameters.

It is entirely up to the computer to generate the parameters correctly, the synthesiser merely does what it is told to do.

Speech Less Latches

The parameter generator is shown in Fig. 7. When the computer decides to deliver a frame of information it sends out an address and a data block. This address is unique to this peripheral device and is decoded by an address decoder inside the synthesiser. This decoded address generates a clock pulse which clocks a 12 bit latch.

Four of these 12 bits of data are another address which decides which of the 9 parameters is being updated. The other 8 bits are data which drive an 8 bit DAC. The analogue output from this DAC is fed to a demultiplexer which drives 9 sample and hold units.

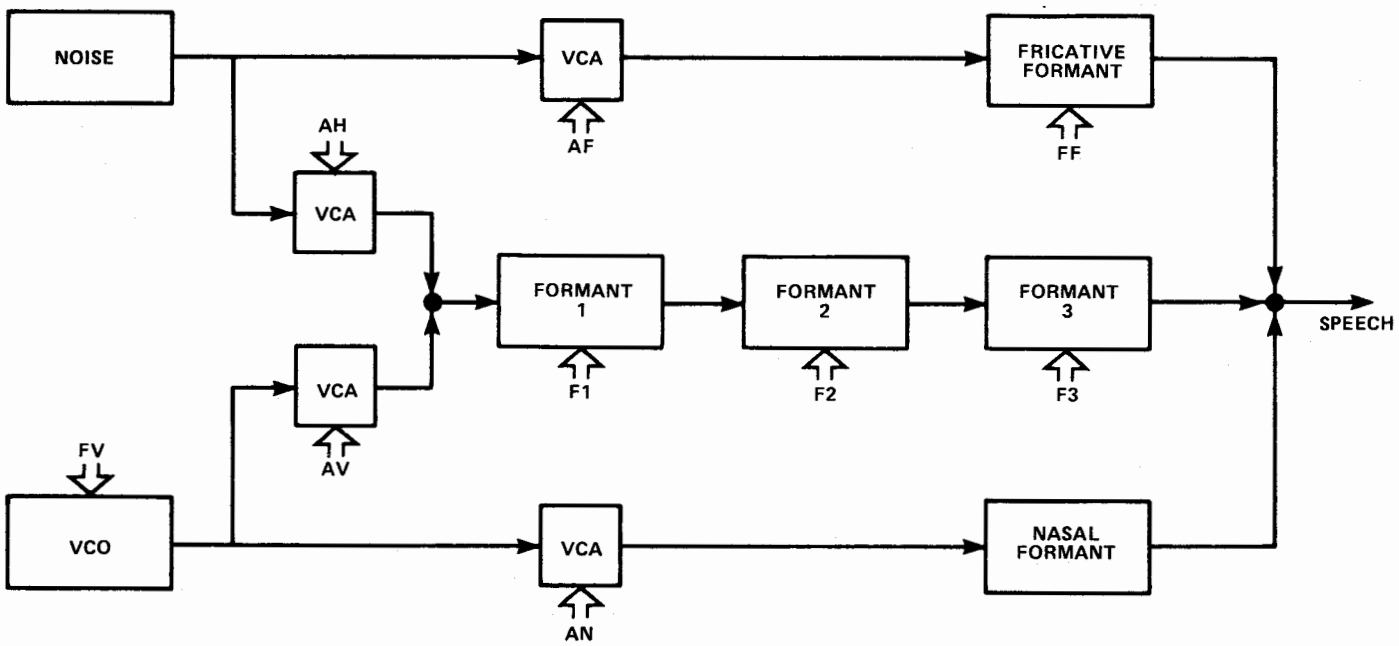
Thus the 8 bit data word is converted into a control voltage and is then steered by the 4 bit address into the correct sample and hold. The whole frame of 9 parameters is updated 50 times a second. This consumes only a small percentage of the computer time, and yet it allows the speech program to be run on a slower time scale without the steps between frames becoming noticeable.

Pitch In

The program was written so as to make the operator's job as easy as possible. There is a listing of about 50 phonemes which can be used to generate speech. Gaps can be typed in and changes to existing sentences can easily be implemented.

The pitch of the speech is controllable so that the correct pitch inflections can be used to stress various words. Also, an external sound source can be used in place of the VCO so that effects such as 'talking music' can be produced. ETI

Fig 6. Block diagram of a three formant speech synthesiser.



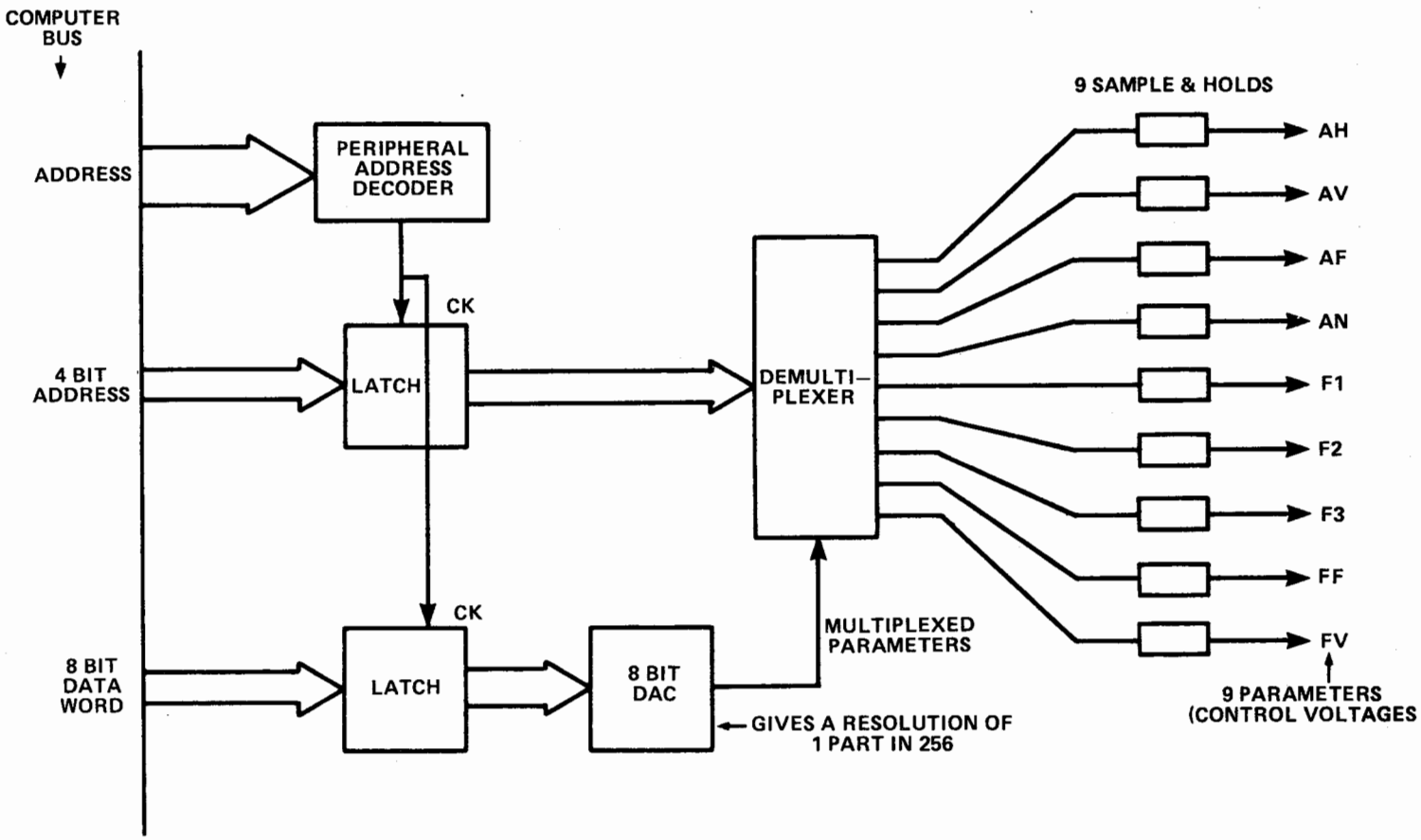


Fig 7. Block diagram of a parameter generator for a speech synthesis machine.

Resumé of Speech Products

The number of speech products that are being produced is rapidly increasing. Here is a list of some of them.

Texas Instruments have brought out a teaching aid called 'speak and spell'. This unit has an alphabetical keyboard plus display. The word that is typed in is spoken by a ROM that uses a linear predecive coding technique, enabling more than 200 words to be stored.

Federal Screw works make a speech synthesiser called Votrax. It generates speech by rule and it can be used as a computer peripheral or as a stand alone unit.

They also make a speech synthesiser which is a bit like a large pocket calculator, except that words are printed next to the buttons. This is intended as a limited talker for people with speech loss.

Telesensory systems make a 'talking' pocket calculator, a 'speaking chip set' and they are also working on a reading tool for the blind. This uses a little hand-held camera which converts the printed text into letters which are then converted into speech.

OVE III made by Fonema is a speech synthesiser similar to that described in this article. However, it uses lots of parameters and the speech output can be better than the real thing!

Speech Lab made by Heuristics is a microprocessor peripheral. This device recognises the spoken word (after you have trained it to do so). The manufacturers claim real time operation and a 95% correct recognition rate.

Computalker made by Computalker Consultants is a microprocessor peripheral speech synthesiser using the vocal tract analogue as described in this article.

Microspeech made by Richard Monkhouse and Tim Orr. A microprocessor peripheral speech synthesiser designed to run from 6800 orientated systems.

Vocoder and Vocoder 2000. The first commercially available channel vocoders for the music market manufactured by EMS. Enables normally inarticulate sounds to speak. (For example, talking pianos.)

Vocaliser pedal made by Coloursound. A music product, not a Wah-wah pedal but a vowel pedal. Vowels available EE to AH to OO.

Dipthoniser made by Coloursound. Produces dipthong filter sweeps primarily for bass guitar. Sounds such as BOW, YEH, WAH and YAE are available.

SPEECH SYNTHESIZER

How to add a voice to your Commodore 64.

Ricardo Jiminez and Adrian Valle

■If you want to increase the potential of your Commodore 64 with a speech synthesizer this project is for you. When your computer talks, you can have many advantages; for example, if you are processing information with numbers or letters, it will be a lot easier, because you will hear the number pressed on the keyboard without looking at the screen. And your children can learn to speak and spell by looking at the letters and numbers while they are listening to the correct pronunciation. Another advantage is that all the components for this project are available at any Radio Shack store.

Construction

The construction of the circuit is not critical, since only two chips are involved. You can assemble a working version on a solderless breadboard in less than two hours. We assembled the project in a small plastic experimenter's box. The 3.12 MHz crystal XTAL-1 can be replaced by a 3.14 MHz crystal, part number X004 for $1.35, distributed by Digi-Key Corp. (P.O. Box 677, Thief River Falls, MN, 56701-9988).

Crystal XTAL-1 and capacitors C1 and C2 should be placed as close to IC1 as possible. A 12/24 pin card-edge connector was used to interface the electronic circuit with the C64 computer.

About the audio output; if you own the Commodore video monitor model 1701 you don't have to use an external speaker, just plug the audio output cables into the audio input of the monitor (located at the rear of the screen) via a phono plug. In this case the volume is controlled by the volume control of the monitor.

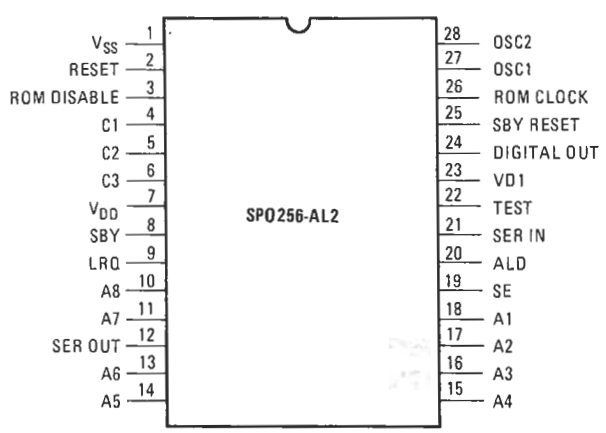
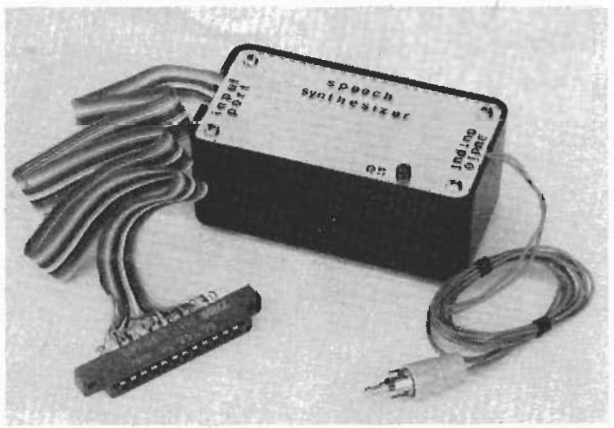


FIG. 1—PINOUT DIAGRAM for the SPO256-AL2. Refer to text for a detailed explanation.



THE COMPLETED SYNTHESIZER packaged and ready for use. As you can see, it's a compact unit that doesn't occupy much space considering the big job that it does.

Finally, the 5-volt DC power supply is provided by the Commodore 64 (pin 2 of the user port), which delivers a maximum current of 100 mA. Since the circuit has a current consumption of 90 mA, such power is suitable for this project.

About the SPO256-AL2

The speech processor (SPO256-AL2) is a single chip N-Channel MOS LSI device that is able, using its stored program, to synthesize speech or complex sounds. Figure 1 shows the pin configuration. We will look at the most important functions of the SPO256-AL2 which permit us to interface with the C64 computer.

A1-A8: 8-bit address which defines any one of 256 speech entry points. ALD: ADDRESS LOAD. A negative pulse on this input loads the 8 address bits into the input port and starts a speech command. This input is controlled by PC2 from the C64 computer.

SBY: STANDBY. Stays high until an addrress is loaded, then it goes low until the chip stops talking. This output indicates to the C64 computer when the chip is ready.

DIGITAL OUT: Pulse width modulated digital speech output which, when filtered by a low-pass filter and amplified, will drive a loudspeaker.

Let's start with some basic linguistic terms in order to understand how this project works.

Phoneme. This Is the basic unit of distinctive sound. It represents different sounds depending on its position within a word. Each of these positional variants is an allophone of the same phoneme.

How to use allophones

When you use allophones you have to think in terms of sounds, not letters. And because each sound is acoustically different depending upon its position within a word, this method is called "Allophone Speech Synthesis." With this technique you can synthesize an unlimited vocabulary by addressing the 59 allophones plus 5 pauses in the appropriate sequence. Figure 2 shows the 64 allophone address table which are contained in the 16K ROM of the SPO256-AL2. The first column (decimal address) is the respective binary address for each particular sound. Since there are 64 addresses only, you will need six

DECIMAL ADDRESS		ALLOPHONE	SAMPLE WORD	DURATION
000		PA1	PAUSE	10MS
001		PA2	PAUSE	30MS
002		PA3	PAUSE	50MS
003		PA4	PAUSE	100MS
004		PA5	PAUSE	200MS
005		/OY/	Boy	420MS
006		/AY/	Sky	260MS
007	*	/EH/	End	70MS
008		/KK3/	Comb	120MS
009		/PP/	Pow	210MS
010		/JH/	Dodge	140MS
011		/NN1/	Thin	140MS
012	*	/IH/	Sit	70MS
013		/TT2/	to	140MS
014		/RR1/	Rural	170MS
015	*	/AX/	Succeed	70MS
016		/MM/	Milk	180MS
017		/TT1/	Part	100MS
018		/DH1/	They	290MS
019		/IY/	See	250MS
020		/EY/	Beige	280MS
021		/DD1/	Could	70MS
022		/UW1/	To	100MS
023	*	/AO/	Aught	100MS
024	*	/AA/	Hot	100MS
025		/YY2/	Year	180MS
026	*	/AE/	Hat	120MS
027		/HH1/	He	130MS
028		/BB1/	Business	80MS
029	*	/TH/	Thin	180MS
030	*	/UH/	Book	100MS
031		/UW2/	Food	260MS
032		/AW/	Out	370MS
033		/DD2/	Do	160MS
034		/GG3/	Wig	140MS
035		/VV/	Vest	190MS
036		/GG1/	Got	80MS
037		/SH/	Ship	160MS
038		/ZH/	Azure	190MS
039		/RR2/	Brain	120MS
040	*	/FF/	Food	150MS
041		/KK2/	Sky	190MS
042		/KK1/	Can't	160MS
043		/ZZ/	Zoo	210MS
044		/NG/	Anchor	220MS
045		/LL/	Lake	110MS
046		/WW/	Wool	180MS
047		/XR/	Repair	360MS
048		/WH/	Whig	200MS
049		/YY1/	Yes	130MS
050		/CH/	Church	190MS
051		/ER1/	Starter	160MS
052		/ER2/	Bird	300MS
053		/OW/	Close	240MS
054		/DH2/	They	240MS
055	*	/SS/	Vest	90MS
056		/NN2/	No	190MS
057		/HH2/	Hoe	180MS
058		/OR/	Store	330MS
059		/AR/	Alarm	290MS
060		/YR/	Clear	350MS
061		/GG2/	Guest	40MS
062		/EL/	Saddle	190MS
063		/BB2/	Business	50MS

FIG. 2—ALLOPHONE ADDRESS TABLE.This will prove extremely handy when setting up your synthesizer.

input address pins to make all these combinations. The second column represents all the allophone sounds (except pauses PA1 to PA5). The third column shows how each allophone sounds in a sample word. The fourth column is the duration in milliseconds for each.

Allophones marked with an asterisk can be doubled; in the word "four" the "f" sound is long, therefore, this word must be formed as follows: "FF, FF, OR, PA5." Note that PA5 is not a sound, is a pause necessary to make the chip stop talking the last allophone.

Each allophone has its own address and sound. To demonstrate this let's see how it happens. The initial K sound (KK3, address 8), used in words like "comb" sounds different from the K's in words like "can't." These small variations are due to the vowel which follow them, in this case, "o" and "a."

The computer controls the SPO256-AL2 with simple software. Let's look at the important instructions used in the programs. First we need to access the user port as follows. Pin C, D, E, F, H, and J (PBO to PB5) in the output mode. Pins K and L (PB6 and PB7) in the input mode. This is made with the statement.

POKE 56579,63

Where PBO to PB6 send the desired address to the speech processor IC1 (see Fig. 3). And PB6 receives the logic status of the STANDBY output (pin 8 of IC1). Remember that the decimal number 63 is equal to 00111111 in binary code. Therefore, when we send an address the ALD input is activated and the chip (IC1) starts talking. Then the computer reads the user port to know the STANDBY status (PB6) which tells the computer when the chip is ready to be triggered again. The instruction PEEK(56577) reads the user port, but we need to read PB6 only which represents the decimal number 64. This is made by using the AND function. If PB6 is a logic "1" then the computer sends a new address.

Testing the speech synthesizer

As was mentioned before, the poke statement (line 20) is used to access the user port. Let's begin with the program shown in Fig. 4. With this program you can listen the particular sounds of the 59 allophones. The FOR-NEXT loop (line 30) is used to increment the value

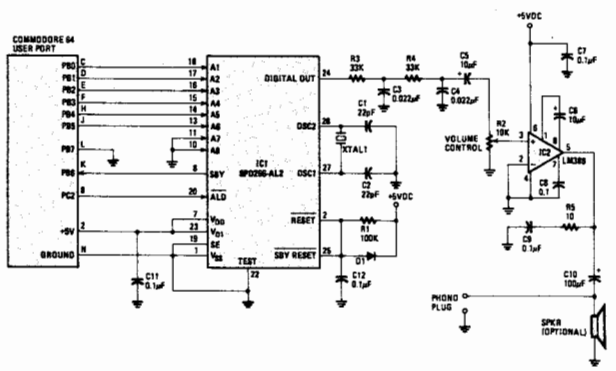


FIG. 3—SCHEMATIC DIAGRAM is relatively simple, uses only few components. Refer to the text for a full and complete explanation.

```
10 REM ALLOPHONES EVALUATION
12 REM BY R. JIMENEZ AND A. VALLE
20 POKE 56579,63
30 FORA=0 TO 63:PRINTA
40 POKE 56577,A
50 POKE 56577,0
60 PB=PEEK(56577)
70 F=PBAND64
80 IF F<>64 THEN 60
90 NEXT A
100 POKE 56577,1
120 END
```

FIG. 4—THIS SIMPLE, BRIEF PROGRAM allows you to evaluate the 59 available allophones.

of A from D to 63. The purpose is to use the variable A in the POKE statement (line 40) in order to address each allophone. The poke 56577,0 clears and prepares the user port for the next data. The PEEK statement (line 60) reads the user port, storing its value in the variable PB. Then PB is compared with the number 64 on the IF statement (line 80). If F is not equal to 64, that means that the speech synthesizer is not ready to receive new data. And the computer goes back to line 60 automatically. Otherwise, new data is sent.

The POKE Statement 56577,1 makes the chip (IC1) stop saying the last allophone. If you wish to listen to the allophones slowly, just press the (CTRL) control key.

Using the speech synthesizer

Making words with the SPO256-AL2 is easy, all you have to do is to look for the sounds you need from the Allophone Address Table. Figure 5 shows a program which makes the speech synthesizer say the sentence "I am a talking computer." This program works like the first one (fig. 4), using the same instructions to write and read by means of the user port. The only difference is that the data (lines 103-110) are going to be sent with the READ statement (line 70). Note that the data are written in decimal numbers. The words used in this sentence were taken from the dictionary included with the SPO256-AL2 package.

Figure 6 shows a routine that can be used for data processing. This routine makes the speech synthesizer say numbers from 0 to 9 when you press the respective key number. Lines 65 to 100 are used exclusively to assign the data found in lines 100 to 128 to the dimensioned variables A (I) and B (I,J). Here you can

```
10 REM I'M A TALKING COMPUTER
60 POKE 56579,63
65 FORJ=1TO 27
70 READ A
80 POKE 56577,A
85 POKE 56577,0
90 PB=PEEK(56577)
94 F=PBAND64
96 IF F<>64 THEN 90
100 NEXT J
102 DATA 24,6,0:REM    I
104 DATA 7,7,16,2:REM  AM
106 DATA 24,2:REM      A
108 DATA 13,23,23,2,42,12,44,0:REM TALKING
110 DATA 42,15,16,9,49,22,13,51,1,4:REM COMPUTER
160 RESTORE
170 FORT=1 TO 500:NEXTT:GOTO 65
200 END
```

FIG. 5—USE THIS PROGRAM, and you'll hear your computer actually say the words "I am a talking computer."

PARTS LIST

Resistors
All resistors ¼watt, 5% unless otherwise specified
R1—100,000 ohm
R2—10,000 ohm trimmer potentiometer
R3, R4—33,000 ohm
R5—10 ohm,

Capacitors
C1, C2—22pF., ceramic disc
C3, C4—0.022μF., ceramic disc
C5, C6—10μF., 10 volts, electrolytic
C7, C8, C9, C11, C12—0.1μF., ceramic disc
C10—100μF., 10 volts, electrolytic

Semiconductors
IC1—SP0256-AL2 speech processor (Radio Shack 276-1784)
IC2—LM386 audio amplifier (Radio Shack 276-1731)
D1—1N914 switching diode

Other components
XTAL-1—3.12 MHz crystal (see text)
SPKR—8-ohms (optional, see text)
12/24 pin card-edge socket (Jameco Electronics)
1.5 ft. - 14-conductor ribbon cable
Insulated phone plug (Radio Shack 274-321)
Experimenter box (Radio Shack 270-231)
Experimenter's IC perfboard (Radio Shack 270-150)
28 pin IC socket
8 pin IC socket

```
12 REM DATA PROCESSING
15 REM BY R. JIMENEZ AND A. VALLE
50 PRINT"(SC)"
60 POKE 56579,63
65 FOR I=0 TO 9
70 READ A(I)
75 FOR J=1 TO A(I)
78 READ B(I,J)
100 NEXT J,I
110 DATA 4,43,60,53,4 :REM            ZERO
112 DATA 5,57,15,15,11,4:REM           ONE
114 DATA 3,13,31,4 :REM                TWO
116 DATA 4,29,14,19,4    :REM        THREE
118 DATA 4,40,40,58,4  :REM           FOUR
120 DATA 5,40,40,6,35,4:REM           FIVE
122 DATA 8,55,55,12,12,2,41,55,4:REM SIX
124 DATA 8,55,55,7,7,35,12,11,4:REMSEVEN
126 DATA 4,20,2,13,4   : REM         EIGHT
128 DATA 5,56,24,6,11,4  :REM         NINE
150 V=V+1
160 IF V=10 THEN V=0 : PRINT"(SC)"
170 PRINT"  VALUE OF X(";V;")";
200 GET C$:IF C$="" THEN 200
205 C=ASC(C$)
210 IF C<48 OR C>57 THEN 200
215 C=C-48 : PRINTC
217 POKE 56577,0
220 FOR I=1 TO A(C)
230 POKE 56577,B(C,I)
240 POKE 56577,0
250 PB=PEEK(56577)
260 F=PB AND 64
270 IF F<>64 THEN 250
280 NEXT I : POKE 56577,1
290 GOTO 150
300 END
```

FIG. 6—THIS PROGRAM will cause your computer to clearly speak each number when the appropriate number key is pressed.

see how A (I) contains the data that the speech synthesizer will use to pronounce a particular word. In this case ten numbers can be pronounced. For example, A (I) holds the number 4 when the variable "I" has a value of 0. And the statement READ (line 70) is executed by the computer. That means that the word "zero" will be spoken by the speech synthesizer, by just sending the four data contained in the vector B (I,J) for values of "I" equal to zero with "J" varying 1 to 4.

The FOR-NEXT loops are used as an auxilliary to the statement READ. Observe how A (I) holds the first data to be used as a variable of the statement FOR (line 75), in order to read to exact quantity of data for each spoken number. Data from lines 110 to 128 are the decimal number values which we need to make the speech synthesizer talk. Vector A (I) will store the first data contained in such lines (4, 5, 3, 4, 4, 5, 8, 8, 4, 5), which as we said before indicates the data contained in the vector B (I, J) respectively. We recommend that you compare these numbers with the allophone address table (fig. 2). For example, the data of line 110 forms the word "ZZ YR OW PA5" where the respective data are 43, 60, 53, and 4.

Lines 150 to 215 give an example of how to make the speech synthesizer more versatile. The screen will display "VALUE OF X (V)?" where V varies from 0 to 9.

The A (C) works as a variable in line 220 of the statement FOR, which means that "I" varies from 1 to the A (C) value, where C indicates the number that will be spoken. Line 217 POKEs a zero so that the new data can be accepted without problems. Line 240 serves the same purpose. And lines 250 to 270 are used to read the STANDBY condition as we explained before.

Finally, the capacity of the speech synthesizer depends upon the imagination of the user. And we hope you enjoy this project as much as we did.