Synthetically Speaking—Part I

by David L. Engelhardt

This board gives your Model I/III speech capability. Use it in applications for education and the handicapped. It's a real chatterbox.

Talk isn't cheap, but it's getting cheaper. Once an expensive undertaking, adding speech capability to your 16K Model I or III is now affordable and fairly easy to accomplish.

In this two-part series, I'll show you how to build and use a speech board for your Model I/III. In this part, I cover the theory, construction, and testing of the board's hardware circuits. Next month I'll go into applications for the handicapped and in education, with programs that convert keystrokes and ASCII string text into synthesized speech.

Although I built this system on a Model III, it also works on a Model I since both computers use the same input/output communications signals (see Table 1). The only difference between the set-up of the two units is that the Model III uses a 50-pin connector and the Model I uses a 40-pin connector.

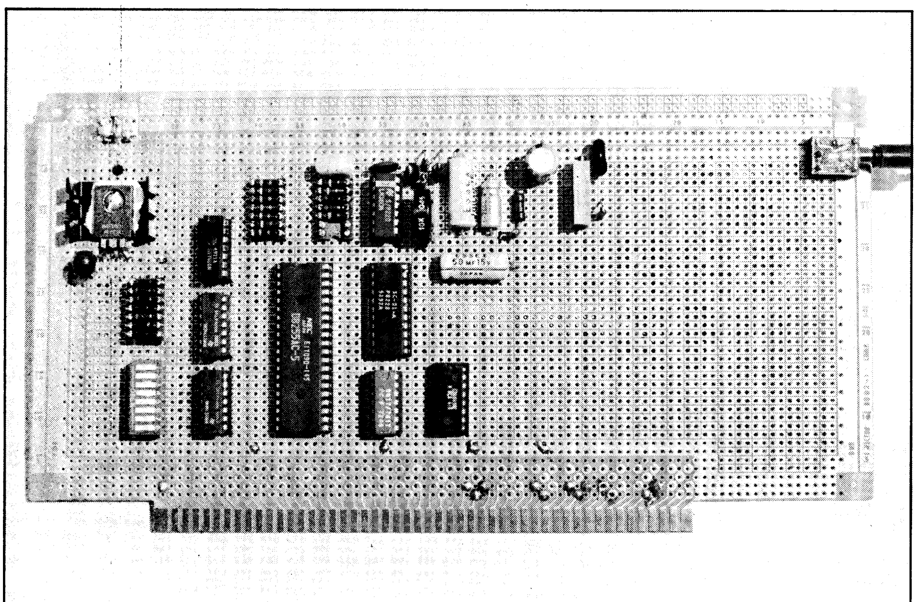

Photo. Speech board.

Parts to Buy

I based the entire speech circuit on the Votrax SC-01 speech synthesizer chip, which costs about $50. I bought the chip from Micromint Inc. (the addresses of all the manufacturers cited appear at the end of this article).

Since I have an S-100 expansion bus, I built the speech circuit on an S-100 plug-in card. The Photo shows the complete circuit, consisting of the decoding circuit, 8255 controller, SC-01 speech chip, and the amplifier that operates an external 8 ohm speaker. Notice that there's still room left on the S-100 card for future additions to the speech circuit.

The Vector Electronic Co. manufactures the S-100 plug-in card (part number 8802-1). I like to use these cards because the two-connections-per-trace design allows easy point-to-point wiring. The cards plug into an S-100 motherboard manufactured by Wameco Inc. (part

The Key Box

Models I and III
32K Disk Basic
16K Cassette Basic
Assembly Language
(If 16K—object code only)
Hardware project

number QMB-12).

The motherboard allows expansion to 12 extra cards, not including the speech board. As it stands, I have the computer interface/clock board in one slot, the port control board in another, and the speech board in a third slot. I still have plenty of room for future expansion. You must provide a 5- and 12-volt power supply to operate the speech board.

If you use a different plug-in card design, chances are it will be smaller. You can still build the speech board using smaller cards, but you may have to split up the total circuit. Using smaller cards shouldn't present any problems as you can easily link them together with ribbon cables. Adapt the required computer signals to your bus configuration according to the circuit schematics.

Port Decoding Circuit

The port decoding section shown in Fig. 1 consists of U1, U2, and two gates of U3. U1 and U2 are two-input, exclusive OR gates that make up the main port decoding section. Switches S1–S6 consist of an 8-DIP-switch (dual in-line processor) package that plugs into a 16-pin integrated circuit (IC) socket with two switches left for future use. Use these switches to set or change the speech ports to the configuration you need, but keep them within the zero to 7F hexadecimal (hex) port limits set by Radio Shack.

I use four ports to operate the speech board via an 8255 programmable peripheral interface chip that controls the Votrax chip. For simplicity, Fig. 1 shows the port decoding signals that consist of everything to the left of the 8255 chip (U4). Figure 2 details the signals that control the SC-01 chip.

I use the 8255 chip as the interface between the Model I/III and the SC-01, due to the SC-01's internal set-up requirements. I'll cover this when I describe the SC-01 synthesizer chip below.

Address lines A2–A7 in Fig. 1 tie to U1 and U2 to decode the base port value. Address lines A0 and A1 tie directly to the 8255 to decode its four internal ports. U1 and U2 turn on and stay selected for a decoding range of four, while address lines A0 and A1 actually perform the internal selection of the four ports within the 8255 chip.

Since I selected ports 16–19 decimal to control the Votrax chip, U1 and U2 set up the base, or bottom, port number (16 decimal) and stay selected while the combination of A0 and A1 make up the actual four ports (16–19) that control the 8255 chip.

U1 makes up the most significant half of the SC-01's base port number. Only one-half of U2 makes up the least significant half of the SC-01's port. The unused gates are available for future use. Select the port number by setting switches S1–S6 to either high (open) or ground. The switches you set to ground actually make up the port's decoded number.

When you don't select the port, the logic state on pin 1 of U3A is normally low. This is because the exclusive OR gate always outputs a logic low whenever either two lows or two highs are on the gate's inputs. Thus, when a decoding switch is open, one of the inputs goes to a high state. If an address line on the other input is high at the same time, it yields a logical low on the output gate that turns off the decoder.

The decoder section requires only one gate with a low output from U1 or U2 to turn off. Since a logical high is one-half the requirement on pin 1 of U3A to turn on the decoder, all decoding exclusive OR gates must have a logical high output. The exclusive OR gate supplies a logical high output only when both of the inputs on each gate are of opposite states.

For example, I decoded the speech board for ports 10–13 hex (16–19 decimal). Since the most significant port digit is 1 hex, I set switch S4 to ground and leave switches S1–S3 open. Since the least significant hex digit is a zero, switches S5 and S6 are also open. The switches tied to ground apply a logic low to one-half of a gate's inputs while

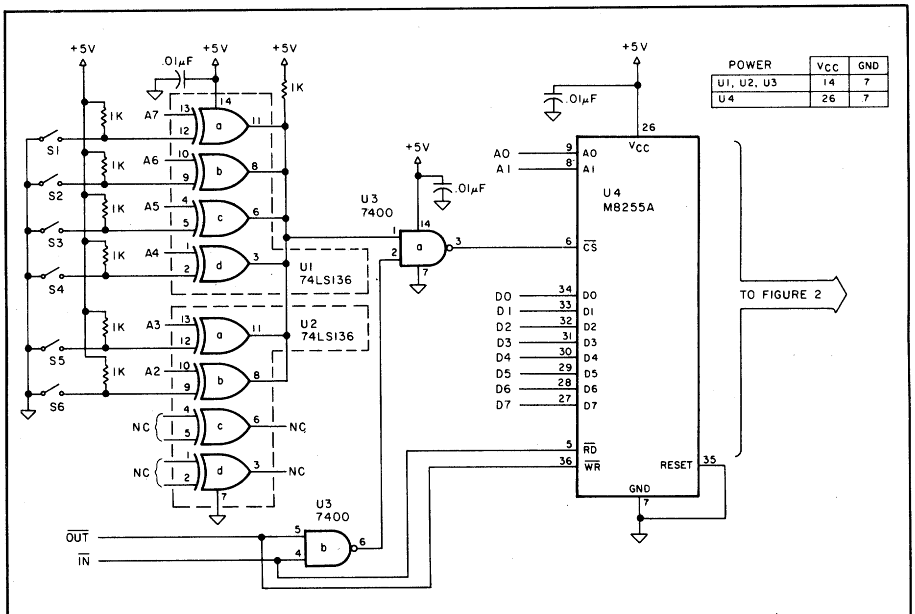

Figure 1. Decoder section.

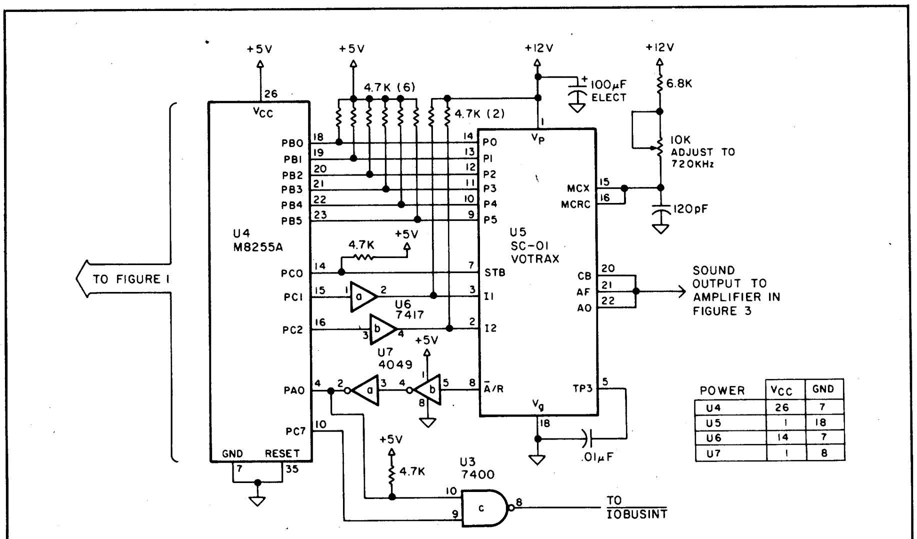

Figure 2. Main speech circuit.

the open switches apply logical highs.

At the selection of port 10 hex, address lines A5–A7 are logical low giving opposite states on the inputs to the gates. This results in logical highs on their outputs. Address line A4 is high, and since its corresponding switch is low, the inputs are of opposite states and result in a logical high as an output.

The least significant digit using address lines A2 and A3 works the same way as the most significant digit. Since address lines A2–A7 decode the base port (10 hex), the decoder enables no matter what state address lines A0 and A1 are in. While address lines A2–A7 stay selected to enable the 8255, the combination of address lines A0 and A1 changes to select the four internal ports required to operate the 8255.

Any time the Input or Output command selects a port, the computer generates a corresponding in or out signal. The computer applies these two signals to the gates of U3B and its output goes high in the presence of either one of the in or out signals. The computer applies the logical high output of U3B combined with the decoded port signal to the inputs of U3A to give a logical low

Model I pin designations

Signals	Computer Pin Designations
D0	Pin 30
D1	Pin 22
D2	Pin 32
D3	Pin 26
D4	Pin 18
D5	Pin 28
D6	Pin 24
D7	Pin 20
A0	Pin 25
A1	Pin 27
A2	Pin 40
A3	Pin 34
A4	Pin 31
A5	Pin 35
A6	Pin 38
A7	Pin 36
In*	Pin 19
Out*	Pin 12
IOBUSINT*	Pin 21

(* = negative true)

Model III pin designations

Signals	Computer Pin Designations (J2)
D0	Pin 1
D1	Pin 3
D2	Pin 5
D3	Pin 7
D4	Pin 9
D5	Pin 11
D6	Pin 13
D7	Pin 15
A0	Pin 17
A1	Pin 19
A2	Pin 21
A3	Pin 23
A4	Pin 25
A5	Pin 27
A6	Pin 29
A7	Pin 31
In*	Pin 33
Out*	Pin 35
IOBUSINT*	Pin 39

All Even
Ground >---------------< 2 TO 50

Table 1. Model I and Model III pin designations.

output.

The output of U3A enables the 8255 chip for the Read and Write commands needed to control the SC-01 speech chip. The computer also ties the in and out signals directly to the 8255 to control the direction of data flow in conjunction with the actual Input or Output command. The computer ties the in signal to the RD input and the out signal to the WR input of the 8255. The computer connects all data lines to the 8255 for reading from and writing to the SC-01 synthesizer chip.

You may wonder why U3 is a standard TTL (transistor to transistor logic) chip instead of an LS (low power Schottky) chip like the others. I discovered through part-swapping and troubleshooting that a 74LS00 doesn't have enough drive capability to enable the 8255. The system either locks up or gets lost when I used a 74LS00, but runs without trouble with a 7400 chip. Please keep this in mind when you buy the parts for the speech circuit.

"Through part-swapping and troubleshooting I discovered that the system runs without trouble with a 7400 chip."

The 8255 Controller

The programmable peripheral interface, or 8255 (U4) for short, allows configuration to any system or device with little difficulty. You can set it up to run in three different modes. I opted to run the 8255 in mode zero for the speech board application.

There are 16 possible configurations within mode zero and Table 2 shows these different configurations. Configuration 8 applies to the speech board application. (Get Intel's *Component Data Catalog* for more information on the 8255 interface chip. This book describes in detail the many combinations and configurations available with this chip in all three modes of operation.)

Writing a control word to the 8255 sets mode zero, making two 8-bit ports and two 4-bit ports available. Ports A and B are the 8-bit ports and port C is split in half to create the two 4-bit ports. All outputs are latching outputs and all inputs are non-latching inputs.

For the speech application, decoded port 19 decimal writes the control and configuration word (144 decimal) to the 8255 chip. Decoded port 18 sets up communications with the 8255's internal port C. Decoded port 17 sets up port B and port 16 communicates with the 8255's port A.

Figure 2 shows the ports that control the SC-01 speech chip. Port B, with its latching outputs, sends a code to the SC-01 and stores it there while port C tells the SC-01 to latch port B's contents. Port C also controls the SC-01's inflection inputs. Port A samples an output from the SC-01 to indicate that it's ready for a new phoneme code. I'll discuss phonemes in more detail below.

The SC-01 Speech Synthesizer

The SC-01 is a self-contained chip capable of phonetically synthesizing continuous speech by combining phonemes. This chip can reproduce the 64

Num	Value	Port A	Port B	Port C/Upper	Port C/Lower
0	128	output	output	output	output
1	129	output	output	output	input
2	130	output	input	output	output
3	131	output	input	output	input
4	136	output	output	input	output
5	137	output	output	input	input
6	138	output	input	input	output
7	139	output	input	input	input
8	144	input	output	output	output
9	145	input	output	output	input
10	146	input	input	output	output
11	147	input	input	output	input
12	152	input	output	input	output
13	153	input	output	input	input
14	154	input	input	input	output
15	155	input	input	input	input

Table 2. Mode zero configurations for the 8255.

different phonemes that emulate the English language using 6-bit access codes.

Sending out these different phonemes at a continuous rate produces synthesized speech. Table 3 lists the 64 different phonemes, their symbols, average duration time in milliseconds, and examples important in creating speech. You can create any word by combining the proper phonemes. I use a Basic program to test the hardware circuits by putting together phonemes to create actual speech (see the Program Listing).

Refer again to Fig. 2. Port B of the 8255 (U4) transfers the 6-bit phoneme code to the Votrax chip. Since the program sets the outputs of port B to latch, the phoneme code stays on the inputs (pins 9–14) of the SC-01 chip (U5) until the program sends a new one. Now that the phoneme code is on the SC-01's input pins, it must latch in through the STB input (pin 7). Latching occurs on a rising edge signal so that the PC0 of the 8255 is set high to latch in the pending phoneme code.

A time delay in the latching set-up time is necessary. The Votrax chip requires at least 100 microseconds to stabilize its internal circuits. After stabilization, the program sets the STB input high to latch in and hold the phoneme code. Due to this latching capability, the Votrax chip is capable of continually sending out the same code until it receives a new one.

Since each phoneme code has its own time duration, it's necessary to know when the SC-01 finishes creating it. This is where the acknowledge/request (A/R) output (pin 8) from the SC-01 comes into play. The A/R output indicates when the program should send the next phoneme. Upon completion of the phoneme, the A/R output goes from a low state to a high state. Since the A/R output is CMOS and the input to the 8255 is TTL, the A/R output doesn't have enough drive capability to operate the TTL gate. U7 takes care of the interface requirement.

U7 (4049) is a hex inverter/buffer capable of driving two TTL loads. Since it has CMOS input and TTL-compatible output, the SC-01 and 8255 chips are completely compatible. The 4049 inverts the signals, while two gates keep U7B's input the same as U7A's output.

A 4050 CMOS chip also works in place of the 4049. The only difference between the two is that the 4050 doesn't invert the logic signals. Should you use a 4050 chip, you need only one gate. I used a 4049 because I had one available when I built the circuit.

"You can synthesize four different voices or mix sounds together to change the pitch of a spoken sentence."

Input PA0 (pin 4) of the 8255 constantly scans the state of the A/R output. When the A/R line goes from a low to a high state, it tells the controlling program to send another phoneme to the SC-01 chip. The sequence of events using the phoneme inputs, STB line, and A/R line continues until the controlling program stops sending new phoneme codes.

I1 (pin 3) and I2 (pin 2) represent the inflection inputs to the Votrax chip. These inputs let you change inflection or pitch for a higher- or lower-sounding output. This gives you the ability to synthesize four different voices or to mix sounds together to change the pitch of a spoken sentence.

Because inflection inputs must be at a constant logical state, they require external latches. The latched outputs of PC1 and PC2 (pins 15 and 16) of U4 provide this function. The gates of U6 act as a buffer between U4 and U5 for the inflection inputs. Notice that the outputs of U6 are tied to 12 volts through 4.7K resistors. The inflection inputs require the same voltage as the SC-01's power supply and U6 provides this interfacing requirement. Remember this because the circuit doesn't function if U6's outputs are tied to 5 volts instead of 12 volts.

Gate U3C is one of two methods that uses interrupts to let the computer know it's time to send out another phoneme code via the 8255. The program sets the state of PC7 (pin 10) to high to enable the interrupt gate. When the A/R line signals the readiness for a new phoneme, its high output combines with PC7's output to give the required low signal for an interrupt.

The other method used is scanning PA0 for the low to high transition. I use this method and scan the PA0 input for a logic change. I set the interrupt usage aside for advanced applications in the future.

The master clock frequency inputs, MCRC and MCX, are connected to a resistor and a capacitor. These two inputs and components reach an oscillating frequency of approximately 720 kHz. This frequency sets up the standard phoneme timing so that, by varying the variable resistor, the voice and sound output change from a low pitch to a high pitch.

If you use the external clock source option instead of the internal one, the MCRC output (pin 16) is tied to ground while the external frequency source is applied to the MCX input (pin 15). Since I use the internal clock, both MCRC and MCX are tied together as shown in Fig. 2. The easiest way to adjust the internal clock is to send out a phoneme code or message and set the frequency for the most pleasing sound. I'll cover this in the testing section of this article.

I tied audio outputs CB, AF, and AO together so I could use them in conjunction with a class A amplifier. If you use these outputs in an amplifier other than a class A amplifier, separate them. I chose class A because it's popular and easy to build. The combined output of the three signals are tied to the amplifier's input shown in Fig. 3.

Figure 3 also shows the speech amplifier circuit, consisting of a pre-amp and a main amplifier. I decided to design an amplifier that could handle speakers larger than those in transistor radios to get better sound quality.

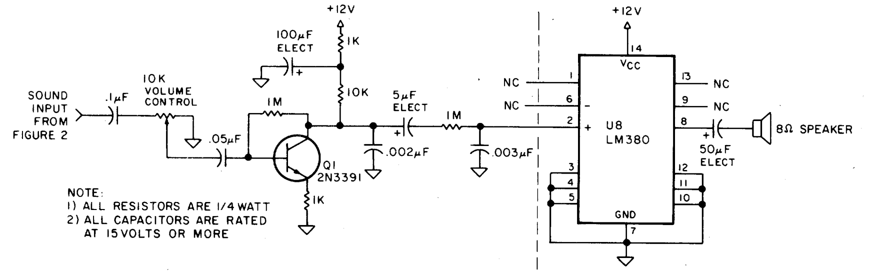

Figure 3. Sound pre-amp/amplifier circuit.

The pre-amp section consists of everything to the left of the dotted line in Fig. 3. It conditions the signal from the SC-01 and sends it to the main amplifier section, to the right of the dotted line. Note that the volume control for the speaker is located in the pre-amp section.

U8 (LM380) is an amplifier chip capable of handling speaker loads of up to 2.5 watts with proper heat sinking. A typical 6-inch car speaker shouldn't present any heat problems when used with this amplifier chip. The pre-amp/amplifier circuit is powered from the same 12-volt supply that powers the Votrax chip.

Generating a Phoneme

Here is the sequence of events that creates a phoneme sound. First, enable the external bus with the command OUT 236,16 to allow any input or output from or to the computer (Model III only). The next step is to set the 8255 interface chip to mode zero, option 8, with the OUT 19,144 command. The OUT 17,'CODE' sends a phoneme code to the Votrax chip through the 8255 (PB0 to PB5). Now consider the required set-up time delay.

If Basic controls the Votrax chip, a time delay loop isn't necessary due to the internal slow speed of the Basic language as compared to machine-language code. If you use machine-language code, the program loads register pair BC with a value and calls 60 hex for the time delay. I use this last method and will describe it under the Text-to-Speech program section next month.

After the controlling program sends the phoneme code to the SC-01 chip, the chip must latch in the code to start the sound output via the STB line input. Port 18 controls the 8255's PC0, PC1, PC2, and PC7 outputs. It always starts the SC-01 chip by making the STB line into a high state.

The value sent to port 18 controls the **STB line, inflection levels, and enables gate U3C for interrupt control. The value must always be an odd number to ensure control of the STB line. The phoneme sounds when an OUT 18,X command latches in.**

After phoneme generation, the program monitors the Votrax chip to know when to send out another code. Port 16 samples the 8255's PA0 input, which is the same as the A/R line. The program scans the A/R line in a loop until a positive value indicates the Votrax chip is ready for a new code. The Basic command that samples port 16 is A = INP (16). A positive value causes the entire sequence of events to start all over again with a new phoneme until stopped under program control.

Speech Construction

Regardless of the board system on which you build the circuit, be sure to use sockets for all the integrated circuits. Figure 4 shows the component layout of the speech board. Table 4 is the parts list for all the circuits.

Since there are a couple of CMOS **chips in the circuit, especially the Votrax chip, take extra precautions to dis**charge static electricity when handling them. A good rule to practice is never to assemble a project in an area that has a rug beneath your feet. The rug can gen-

Phoneme Code	Phoneme Symbol	Duration (ms)	Example Word
00	EH3	59	jacket
01	EH2	71	enlist
02	EH1	121	heavy
03	PA0	47	no sound
04	DT	47	butter
05	A2	71	made
06	A1	103	made
07	ZH	90	azure
08	AH2	71	honest
09	I3	55	inhibit
0A	I2	80	inhibit
0B	I1	121	inhibit
0C	M	103	mat
0D	N	80	sun
0E	B	71	bag
0F	V	71	van
10	CH*	71	chip
11	SH	121	shop
12	Z	71	zoo
13	AW1	146	lawful
14	NG	121	thing
15	AH1	146	father
16	OO1	103	looking
17	OO	185	book
18	L	103	land
19	K	80	trick
1A	J*	47	judge
1B	H	71	hello
1C	G	71	get
1D	F	103	fast
1E	D	55	paid
1F	S	90	pass
20	A	185	day
21	AY	65	day
22	Y1	80	yard
23	UH3	47	mission
24	AH	250	mop
25	P	103	past
26	O	185	cold
27	I	185	pin
28	U	185	move
29	Y	103	any
2A	T	71	tap
2B	R	90	red
2C	E	185	meet
2D	W	80	win
2E	AE	185	dad
2F	AE1	103	after
30	AW2	90	salty
31	UH2	71	about
32	UH1	103	uncle
33	UH	185	cup
34	O2	80	for
35	O1	121	aboard
36	IU	59	you
37	U1	90	you
38	THV	80	the
39	TH	71	thin
3A	ER	146	bird
3B	EH	185	get
3C	E1	121	be
3D	AW	250	call
3E	PA1	185	no sound
3F	STOP	47	no sound

/T/ must precede /CH/ to produce CH sound.
/D/ must precede /J/ to produce J sound.

Table 3. Phoneme chart.

erate static charges with a potential of thousands of volts that, in dry climates, could easily destroy CMOS chips.

I use Radio Shack wire-wrap for all of my connections. If you use wire-wrap, use a wire stripper to facilitate working with the small-sized wire. I recommend using wire-wrap because it's small, easy to work with, and comes in various colors. I standardized my color coding system so that blue wire is for address lines, yellow for data lines, green for control signals, and red for power. The color codes make trouble-shooting easy.

All of the control signals coming from the computer are labeled in the schematics. How you get them to the speech circuits is up to you. You have two options. You can run them directly from the computer or go through an interface like the one described in the March 1983 issue of *80 Micro* ("Real World, It's About Time!", p. 342).

If you don't build an interface, tie the appropriate signals from the Model III to the speech circuits directly through a 50-pin edge connector and conductor cable. Use a 40-pin edge connector and conductor cable for a Model I. Table 1 shows the signals and pin-outs of the Model I and Model III expansion buses.

You can buy the 40- and 50-pin connectors from Radio Shack, but you'll have to get the cable from an electronics dealer. Run the appropriate signals to the designated inputs shown in Figs. 1 and 2. For the Model III, be sure to connect all the even cable conductors to ground as shown in Table 1. The Model I doesn't use this configuration. (The article referred to above explains the cable/conductor assembly in greater detail.)

I mount both the 5- and 12-volt regulators on opposite sides of the S-100 card. Be sure to mount these regulators on heat sinks to keep them cool. Install a 5 μF electrolytic capacitor on each regulator's input lead to suppress any oscillations should they occur. These oscillations drop the output voltage to an undesirable value. Install the 100 μF electrolytic capacitor across the SC-01's 12-volt power supply to ground as shown in Fig. 2 to prevent noise from entering the circuit.

Although it isn't required, I highly recommend that you add .01 μF disk capacitors across each integrated circuit's power and ground pins. These ICs sometimes produce noise when they're involved in high-speed switching. The capacitors filter out most of it. I always mount mine on the back side of the S-100 card, soldering one lead directly to the chip's power input pin and the other to the chip's ground pin.

On completion of the speech board, check your circuits carefully for wiring errors, opens, and shorts before you apply power to the circuits. If everything checks out, apply power and measure the 5- and 12-volt supplies for correct operation. It's also a good idea to check each IC socket's power pins for voltage. Take time to read all of the SC-01's socket pins for voltages.

Use a miniature plug and socket to connect the speaker to the speech board as shown in the Photo. Install all of the pull-up resistors as shown in Fig. 2. These resistors ensure correct logic levels between the 8255 (U4) and SC-01 (U5) chips. I install my resistors on 14-pin component carriers to obtain a neater appearance.

With the power off, carefully install all of the ICs. Make sure you set the port address decode switches (S1 to S6) to decode the base port of 10 hex.

Decoder Section Parts List (Figure 1)

U1, U2	74LS136
U3	7400
U4	8255A
S1–S6	RS #275-1301
4-Filter Caps	.01 μF @ 25V disk
7-Resistors	1k ohm @ 1/4 watt

Main Speech Circuit Parts List (Figure 2)

U4	8255A
U5	Votrax SC-01 chip
U6	7417
U7	4049
3-Filter Caps	.01 μF @ 25V disk
1 Cap	120 pF cap @ 25V
1 Cap	100 μF @ 25V electrolytic
Resistors are 1/4 watt:	
4.7k ohm	Quantity = 10
6.8k ohm	Quantity = 1
10k ohm pot	Quantity = 1 (any power rating)

Sound Amplifier Circuit Parts List (Figure 3)

U8	LM380
Q1	2N3391
Capacitors: (rated minimum of 25V)	
100 μF elect.	Quantity = 1
50 μF elect.	Quantity = 1
5 μF elect.	Quantity = 1
.10 μF	Quantity = 1
.05 μF	Quantity = 1
.003 μF	Quantity = 1
.002 μF	Quantity = 1
Resistors are 1/4 watt:	
10k ohm pot	Quantity = 1 (any power rating)
1M ohm	Quantity = 2
10k ohm	Quantity = 1
1k ohm	Quantity = 2
Phone Jack	RS #274-249
8 ohm speaker	

Miscellaneous

Wameco QMB-12 Motherboard (if used)
Vector 8802-1 S-100 card (if used)
12V regulator
5V regulator
IC sockets
PC board
50 pin connector and cable (40 pin for Model I)
Wire-wrap, power-supply heat sinks,
solder, hardware, suitable enclosure

Table 4. Parts list.

Testing the Speech Board

The Program Listing is a Basic program that tests the speech board and sets up the master clock frequency. It is well-commented and follows my earlier discussion on the creation of phonemes. Type in and save the Listing. Turn on the speech board's power. You may hear a phoneme sounding at a constant rate due to the possible erratic power-up condition of the circuits. Type RUN to execute the test program.

The computer should now say "Hello, I am now a talking computer." After this message, the speech board is quiet since the last data value sent consists of the Votrax stop code of 63 decimal.

If the speech board doesn't talk, check the DIP switch to see if you selected the correct base port number. If the problem persists, recheck all your circuits for possible errors. If the board does talk, you can set the master clock frequency.

"When you type RUN to execute the test program, the computer should say 'Hello, I am now a talking computer.' "

Change line 270 in the Listing to RUN. This puts the program into a continuous talk mode. Now run the program and adjust the variable resistor that changes the master clock frequency as shown in Fig. 2. Adjust this resistor to obtain the most pleasing sound pitch. When the sound pitch is set, check the inflection circuits.

To do so, modify the Listing once more. Change line 170 to INPUT "ENTER INFLECTION VALUE";VO. This lets you enter different inflection values to see how they change the voice output. Line 180 ensures that the value is odd between 1 and 7. If you use the interrupt option, add a value of 128 to the 1 to 7 value.

Run the program and enter odd values from 1-7 to hear how the voice changes pitch. You should be able to detect a total of four different voices. I incorporate the ability to change the pitch or inflection inputs in the Text-To-Speech program, which I'll discuss next month. This lets you change pitch within sentences.

When you order and receive the Votrax chip, you also get the data sheets and a phonetic speech dictionary that gives you the codes to make up numerous words. The values in the data statements located in the Listing were taken from this book. You must, however, go through a conversion process to obtain the correct decimal value for the data statements.

Refer again to Table 3. Note that each phoneme code has a hex value. You must convert this value to its decimal equivalent in order to put it into data statements. To make it easier, I put

```
10 '**************************************************
20 '                L I S T I N G  1
30 '**************************************************
40 '
50 '        TEST PROGRAM FOR THE SPEECH BOARD
60 '                     BY
70 '                DAVID ENGELHARDT
80 '
90 '                   05/04/83
100 '
110 ' THIS BASIC PROGRAM IS USED TO TEST THE
120 ' HARDWARE CIRCUITS.
130 '
140 '************************************************
150 '
160 CLS
170 VO=1
180 IF VO=1 OR VO=3 OR VO=5 OR VO= 7 THEN GOTO 190  ELSE 170
190 E=43
200 DATA 27,2,35,24,35,53,55,62,62 :      ' HELLO
210 DATA 21,0,9,41,62,47,0,12,62  :       ' I AM
220 DATA 13,21,35,55,62,6,33,41,62:       ' NOW A
230 DATA 42,61,25,10,20,62          :     ' TALKING
240 DATA 25,50,12,37,34,54,55,42,58,63:   ' COMPUTER
250 '
260 GOSUB 5000  :                         ' DO IT.
270 STOP
280 '
290 '**************************************************
300 ' THIS SUBROUTINE WILL MAKE THE VOTRAX CHIP TALK
310 '**************************************************
320 '
5000 OUT236,16:
                     ' ENABLE EXTERNAL BUS
5010 FORX=1 TO E:                        ' SET UP LOOP
5020 READ C:                             ' GET PHONEME CODE
5030 OUT19,144:                          ' SET 8255 FOR MODE 0
5040 OUT17,C:                            ' SEND PHONEME CODE
5050 OUT18,VO:                           ' LATCH CODE AND TALK
5060 A=INP(16):IFA=0 GOTO 5060 :         ' SCAN A/R  HIGH = DONE
5070 NEXT X:                             ' DO AGAIN
5080 RETURN
5090 END
```

Program Listing. Test speech program.

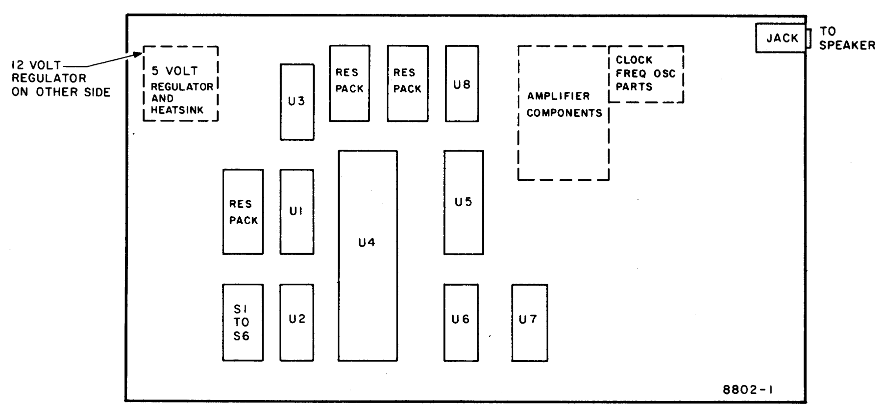

Figure 4. Parts placement for speech board.

the decimal equivalent just to the right of the hex value. Start with the phoneme code 00 and number them from zero to 63.

You can adapt the Listing to run in any Basic program. Just put the subroutine from lines 5000–5080 in your program with the appropriate data statements. Change the data statements and you can apply the talk feature to games, and so on. If the application requires only certain spoken phrases, you can save memory by putting them into data statements, when you use the Text-To-Speech program in next month's issue.

Using Table 3 might seem a little complicated at first, but it gets easier with experimentation. For practice, break the word "hello" down into its phoneme equivalent by using the example word pronunciations given in Table 3.

First find the H sound in the table, phoneme code 1B hex or 27 decimal. The next letter, E, comes closest in pronunciation to that in the word "enlist," so the phoneme code is 01 decimal. Since the next two letters are L's, add two decimal L codes, which are 24,24. The final code for the letter O is 38. The last value should always be the stop code of 63 for any spoken phrase.

Now put the decimal codes 27, 01, 24, 24, 38, and 63 into the first data statement in the Listing. Also, change the E value (line 190) to 6 for the loop count. Run the program and listen to the pronunciation of hello. You may find that some words need a little more work by adding a few extra phoneme codes.

For example, compare the phoneme codes for hello to those in the first data statement in the original Program Listing and you'll see what I mean.

When you get the speech board built and running, you're ready for the software that converts ASCII text to speech, as well as some practical applications. I'll present those next month, along with some instructions on how you can assemble the listings on your particular system. ■

Write to David Engelhardt at 10221 West 101st Place, Broomfield, CO 80020.

The addresses of the manufacturers cited in this article are: Micromint Inc., 561 Willow Ave., Cedarhurst, NY 11516 (800-645-3479; NY residents, 516-374-6793); Wameco Inc., P.O. Box 877, El Granada, CA 94018 (415-728-9121); Vector Electronic Co., 12460 Gladstone Ave., Sylmar, CA 91342 (213-365-9661).