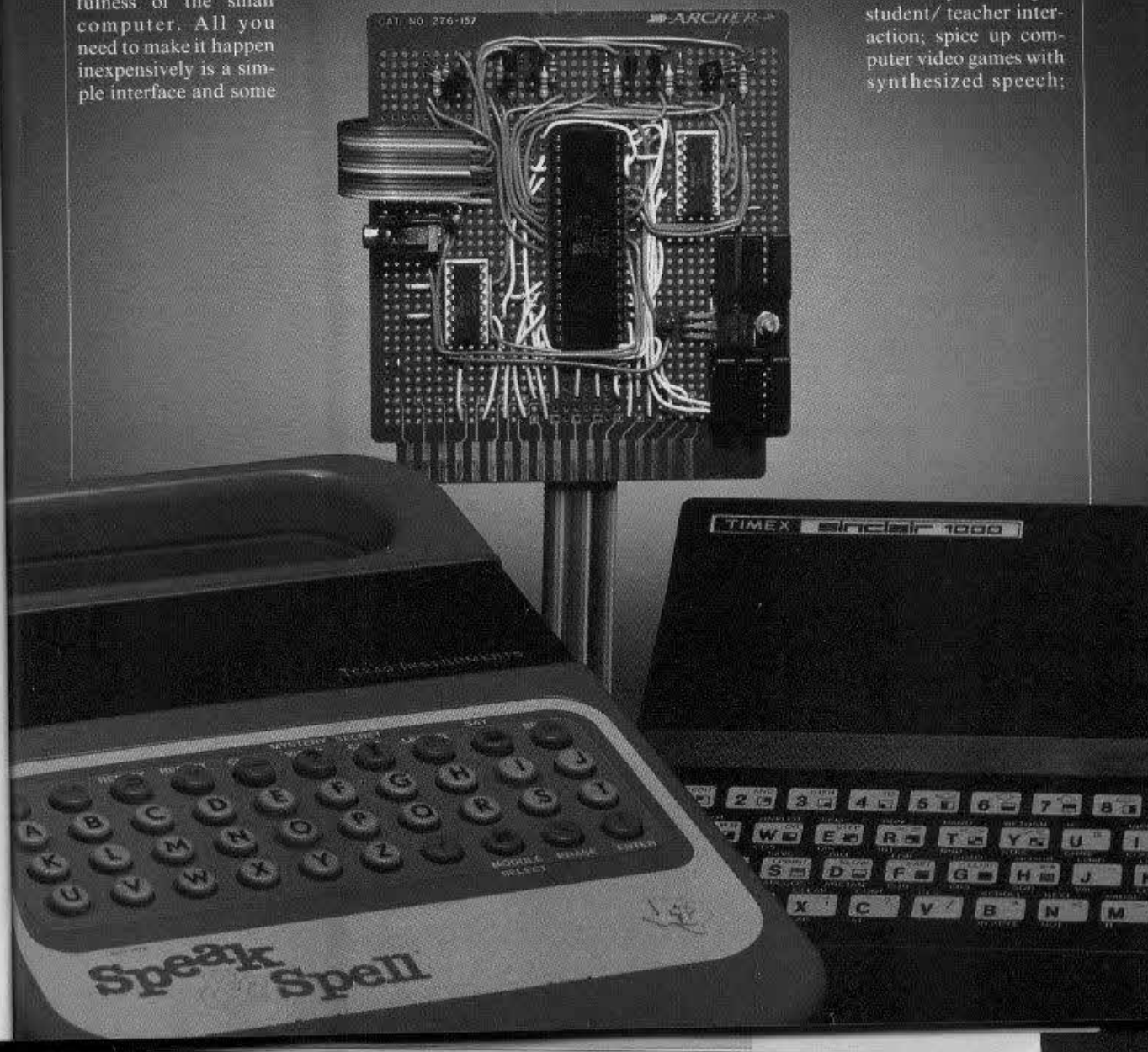
Computers & Electronics
FEBRUARY 1983

TALK CAN BE CHEAP

Hundreds of words can be "spoken" by a Sinclair or Timex computer when adding an interface and a "Speak & Spell"

By Larry Dighera

A "talking" computer is not necessarily expensive—not if you mate one of the low-cost computers (Sinclair ZX-80, ZX-81, or Timex 1000) with Texas Instruments' popularly priced "Speak & Spell" learning device. The combination gives you several hundred clearly articulated words that expand the usefulness of the small computer. All you need to make it happen inexpensively is a simple interface and some software, all described here. Using these ideas, you might design a low-cost security/fire-alarm that vocalizes the nature of a problem ("FIRE," "THEFT," etc.). You could also enhance your computer's portability by making its output audible instead of displaying it on a video screen; write educational programs with truly meaningful student/ teacher interaction; spice up computer video games with synthesized speech;



create useful programs for the visually impaired; etc. Here's how it can be done.

System Overview. The Speak & Spell consists of a pushbutton keyboard, microprocessor, display, ROM (contains speech data), voice synthesizer, and loudspeaker. A block diagram of the system is shown in Fig. 1. The microprocessor communicates with the speech units through a 6-line bus with CNTL 1, 2, 4, and 8 forming a 4-bit data bus and PDC (processor data clock) and CS (chip select) forming a control bus. The control commands used in the Speak & Spell are listed in Table I.

The ROM contains the binary-coded speech data for synthesis of the spoken word. Each word has a specific starting address. When it is desired to output a particular word, the ROM address of the beginning of the word is sent to the voice synthesizer in five 4-bit nybbles, preceded by the LOAD ADDRESS (code 2) command. The data is then clocked into the voice synthesizer by the PDC signal. Once the 5-nybble word address is loaded, READ ROM (code 8) and SPEAK (code 10) commands are sent to cause speech to be generated. If the BUSY SPEAKING? (code 14) command is now sent, the voice synthesizer will raise the CNTL 1 line high until it finishes vocalizing.

A schematic of the interface circuit between the computer and Speak & Spell is shown in Fig. 2. The microprocessor in the Speak & Spell uses PMOS devices that operate at −21 V. (A typical I/O line is shown within the processor.) Because PMOS uses passive pulldown resistors, output current is limited. If ground potential is impressed on these lines, no harm will result, regardless of their state.

The Z80A Parallel Input/Output (PIO) chip in the interface used for *IC1* provides two bidirectional I/O ports: port A uses CMOS inverters (*IC2*) and open-collector pnp driver transistors (*Q1* through *Q6*) as the outputs. The emitters of these drivers are connected to the +5-volt line, which is also connected to the positive (COM) of the Speak & Spell. Thus, when a transistor is conducting, the S&S MPU "sees" a logic 1 (0 V); when the transistor is off, the PMOS pulldown resistors bring the line to logic 0.

Port B of the PIO is used for input and receives its signal from *R7* through *R12*, which limit the incoming signal from the Speak & Spell. In addition to interfacing with the Speak & Spell, with appropriate software, the PIO can provide a much-needed parallel I/O capability for the computer, allowing use of joysticks and such functions as music, control, and process monitoring.

Construction. The circuit can be built on a dual 22-contact card similar to the Radio Shack No. 276-154. If you use the same edge-contact arrangement as in the computer, except for the clock line, the card is compatible with ZX bus expansion cards currently available. Use of sockets for the ICs and a miniature phone jack to interconnect the power supply are recommended.

The Speak & Spell draws about 200 mA and the interface about 70 mA at 5 V. If you are using a 16K RAM extension, the larger power supply can handle the extra load. Arrange switching so that both computer and interface power up at the same time. If you elect to use batteries in the Speak & Spell, disconnect the ground line by opening the jumper (see Fig. 2). Recheck the interface circuit before connecting it to the computer.

To make the connections to the Speak & Spell, carefully remove the back plate and locate the 40-pin microprocessor immediately below the display. Pin 1 is in the lower right-hand corner. Connections are made to pins 10 through 14, 36, 38, and the negative terminal of the bat-

(Continued on page 47)

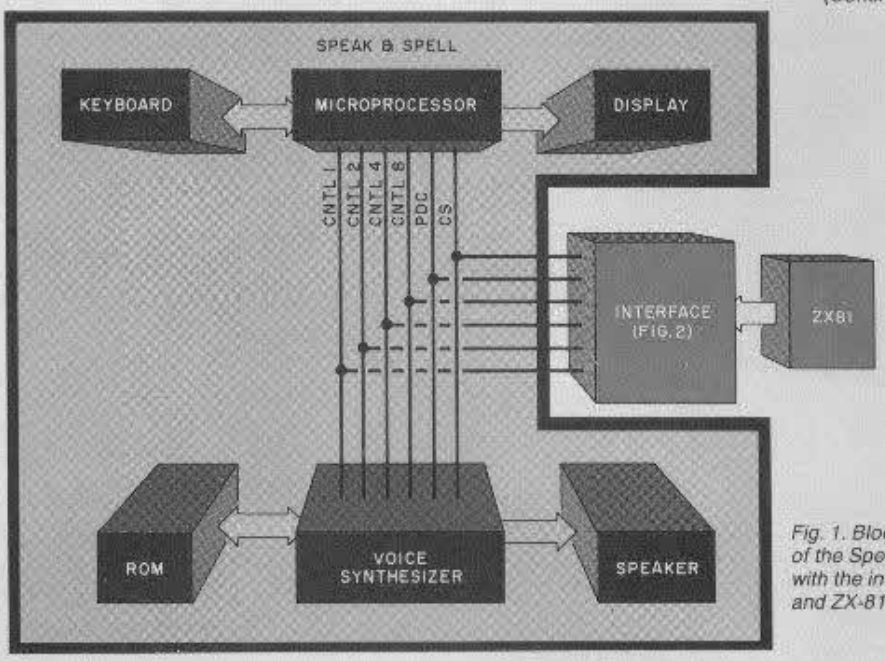


Fig. 1. Block diagram of the Speak & Spell with the interface and ZX-81 computer.

(Continued from page 40)

tery. Use fine insulated wire to make these connections and work very carefully to avoid creating short circuits.

The nine leads can be terminated in a 14-pin DIP socket, with the cable brought out through the battery compartment. Slip a short length of heat-shrinkable tubing over each wire before soldering it to the DIP socket. Then shrink it over the soldered connection.

The interface is wired to the Z80 microprocessor in the computer as shown in Fig. 2.

Software. Because it is necessary to supply the voice synthesizer with data at a rate beyond the capability of the BASIC interpreter built into the computer, machine language must be used during programming. The programming code given in Table II covers six program modules. The first, labelled PIO, is the initialization routine for the Z80 PIO chip (*IC1*). When power is first applied to the system, the PIO is in an inactive state and must be initialized (told what to do) before it can be used.

The listing in Table II can be entered into the computer using the BASIC program shown in Table III. The three lines without numbers at the beginning of the program are keyed directly into the machine to reserve the top 6K of RAM for the remainder of the program. After entering Table III, run it and enter each number shown in the decimal-code column of Table II, referring to the check sum as you go. If an error is detected, use B to move back. Moving forward without altering the data that has already been entered may be accomplished by entering S (for skip).

At this point, it is possible to test operation of the PIO by entering the following:

```
POKE 26624,62      ;LD A, data
POKE 26625,0
POKE 26626,211     ;OUT port A,A
POKE 26627,1
POKE 26628,201     ;return
```

Now enter:

```
1000 PRINT USR 16514
1010 INPUT A
1020 POKE 26625,A
1030 PRINT USR 26624
1040 GOTO 1000
```

By entering a number between 0 and 63, the six low-order bits of port A are controlled. These can be metered at the outputs of *IC2*. After testing, delete the BASIC programs. At this point, the listing in Table II should still be in the machine; it can be SAVED on cassette for future use.

Not all Speak & Spell ROMs are programmed with the same word addresses. Hence, it is necessary to construct a "word map" for your particular device. One way to find the starting address of a word would be to send the voice synthesizer successive addresses and note which produce usable speech output. However, this tedious method is not necessary since the interface can be used as a form of logic analyzer that allows you to monitor the bus and capture the addresses sent by the microprocessor. This can be accomplished by entering the BASIC program shown in Table IV. The REM at line 1 reminds you that the machine code of Table II should be in the machine.

To locate the address of a particular word, use the DETERMINE WORD ADDRESS routine and press the GO key until you hear that word. Hit any key to start recording (which calls STODATA of Table II) and then operate the REPEAT key to put the address on the bus. The STODATA routine allows taking "snapshots" of the voice synthesizer bus at approximately 12-microsecond intervals and storing this data in the top 5K of RAM. (See RAM Memory Map in Table V.) This data can be graphically displayed via the LOGIC DISPLAY routine in Table IV. Successive frames can be viewed by hitting ENTER, or specific frames can be selected by hitting the appropriate key. You can also choose numerical readout by using the PRINT WORD ADDRESS portion of Table IV.

The machine-code module SPEAK (Table II) is the software driver for speech production. Because Speak & Spell uses CNTL 1, 2, 4, and 8 to turn on the display segments and convey data, there are always extraneous signals on the bus. This "noise" can be minimized by pressing ON, ON, GO. This leaves only the cursor on, which causes CNTL 8 to periodically go high. Because of this, SPEAK contains a trap that waits for the bus to clear before sending data. Once clear, five nybbles, stored in the 1K above RAMTOP, are clocked into the voice synthesizer. Then the READ ROM, SPEAK, and BUSY SPEAKING? control words are clocked in. When finished speaking, the voice synthesizer causes the CNTL 1 line to go low to allow the RESET control code to be clocked in. Finally, PIO port A is configured with all lines low via the OFF program module.

PDC is the machine-code module that clocks in the data presented by SPEAK. When called, it waits approximately 124 microseconds, brings the processor data clock high for 124 μs, then holds the data for 124 μs before returning. The 124-μs timing is effected by the delay loop at 16645 of Table II.

If sentences rather than individual words are required, RAM address 16542 can be POKEd with the location of the next word to be spoken, then SPEAK called again. This is repeated until all words are vocalized. It is possible to store more than 200, 5-nybble word addresses in the 1K space above RAMTOP.

TABLE I—VOICE SYNTHESIZER CONTROL COMMANDS

Code	Use	Input/Output
0	RESET	Input
2	LOAD ADDRESS	Input
4	PLACE VOICE DATA ON BUS	Output
6	SPEAK SLOWLY	Output
8	READ VOICE DATA FROM ROM	Input
10	SPEAK	Output
12	BRANCH	Input
14	BUSY SPEAKING?	Output

TABLE II—MACHINE CODE LIST

Address	Label	Mnemonic	Code Decimal	CK Sum	Comment
16514	PIO	LD A,207	62,207	62	Mode control word (Mode 3)
16516		OUT (3),A	211,3	480	Mode control word to PORT A control register
16518		LD A,192	62,192	545	Data direction word:bits 0-5 = out,6&7 = in
16520		OUT (3),A	211,3	948	Data direction word to PORT A control register
16522		LD A,207	62,207	1013	Mode control word (Mode 3)
16524		OUT (7),A	211,7	1431	Mode control word to PORT B control register
16526		LD A,255	62,255	1500	Data direction word:all bits input
16528		OUT (7),A	211,7	1966	Data direction word to PORT B control register
16530		LD A,7	62,7	2035	Interrupt control word: disable interrupts
16532		OUT (3),A	211,3	2253	Interrupt control word to PORT A control reg.
16534		OUT (7),A	211,7	2467	Interrupt control word to PORT B control reg.
16536		RET	201	2675	Return
16537		NOP	0,0,0,0	2675	No Operation
16541	SPEAK	LD HL,0,104	33,0,104	2708	Set NYBL pointer to RAMTOP
16544	AAA	LD BC,0,5	1,0,5	2813	Set NYBL counter = 5
16547		IN A,PORT B	219,5	3037	Get current state of synthesizer bus
16549		ADD A,0	198,0	3240	Set zero flag if no data present
16551		JR Z,AAA	40,250	3280	Wait until data present
16553	BBB	IN A,PORT B	219,5	3749	Get current state of synthesizer bus
16555		ADD A,−8	198,248	3952	Wait until bus clear (CNTL 81)
16557		JR Z,BBB	40,250	4240	Loop until bus clear
16559		LD A,CS	62,32	4552	Get Chip Select/reset word
16561		CALL,PDC	205,236,64	4789	Clock in reset
16564		ADD A,2	198,2	5287	2 = "LOAD ADDRESS"
16566		CALL,PDC	205,236,64	5494	Clock in "LOAD ADDRESS" command
16569		SUB,2	214,2	6008	Remove command
16571	CCC	ADD A,(HL)	134	6144	Get NYBL
16572		CALL,PDC	205,236,64	6349	Clock in NYBL
16575		SUB,(HL)	150	6799	Remove NYBL
16576		INC,HL	35	6834	Increment NYBL pointer
16577		DJNZ,CCC	16,241	6850	Loop if less than 5 NYBLs
16579		ADD A,8	198,8	7289	8 = "READ ROM"
16581		CALL,PDC	205,236,64	7502	Clock in "READ ROM" command
16584		ADD A,2	198,2	8000	10 = "SPEAK"
16586		CALL,PDC	205,236,64	8207	Clock in "SPEAK" command
16589		ADD A,4	198,4	8705	14 = "BUSY?"
16591		CALL,PDC	205,236,64	8914	Clock in "BUSY?" command
16594		LD A,CS	62,32	9276	0 = "RESET" command
16596		CALL, PDC	205,236,64	9513	Clock in "RESET" command + CS
16599	DDD	IN A,PORT B	219,5	10032	Get synthesizer bus status
16601		BIT 0,A	203,71	10240	Check bit 0 = 0
16603		JRNZ,DDD	32,250	10343	If bit 0 ≠ 0,then still speaking, so loop
16605		LD A,CS	62,32	10655	0 = "RESET"
16607		CALL,PDC	205,236,64	10892	Clock in "RESET" + chip select
16610	OFF	LD A,0	62,0	11254	0 = off
16612		OUT PORT A,A	211,1	11465	Turn off PORT A
16614		RET	201	11667	Return
16618		NOP	0,0,0,0,0	11667	No operation
16620	PDC	OUT PORT A,A	211,1	11878	Send data to synthesizer
16622		CALL,DELAY	205,5,65	12084	Set up time
16625		ADD A,PDC	198,16	12352	16 = Processor Data Clock
16627		OUT PORT A,A	211,1	12579	Set clock high
16629		CALL,DELAY	205,5,65	12785	Clock pulse width
16632		SUB 16	214,16	13069	Remove Processor Data Clock
16634		OUT PORT A,A	211,1	13296	Let clock fall
16636		CALL,DELAY	205,5,65	13502	Hold time
16639		RET	201	13773	Return
16640		NOP	0,0,0,0,0	13773	No operation
16645	DELAY	PUSH,BC	197	13970	Save NYBL counter
16646		LD C,21	14,21	13984	Initialize delay-loop counter
16648	EEE	DEC C	13	14018	Reduce delay-loop counter by 1
16649		JRNZ EEE	32,253	14050	Loop until time-out
16651		POP,BC	193	14496	Retrieve NYBL counter
16652		RET	201	14697	Return
16653		NOP	0,0,0,0,0	14697	No operation
16658	STODATA	LD, HL,108	33,0,108	14730	Set data pointer to storage address
16661		LD C,5	14,5	14852	PORT B data register address
16663	FFF	IN A,PORT B	219,5	15076	Get current state of synthesizer bus
16665		CP 50	254,50	15335	50 = CS + PDC + "LOAD ADDRESS" command
16667		JRNZ,FFF	32,250	15417	Loop until 50 present
16669	GGG	INI	237,162	15904	Send bus data to storage & increment pointer
16671		LD A,129	62,129	16128	129 = 2's complement of 32512
16673		ADD A,H	132	16389	Test if H byte = 32512
16674		JRNZ,GGG	32,249	16421	Loop until H byte = 32512
16676		RET	201	16687	Return
16677		NOP	0 x 14	16687	No operation
16690					End of REM statement

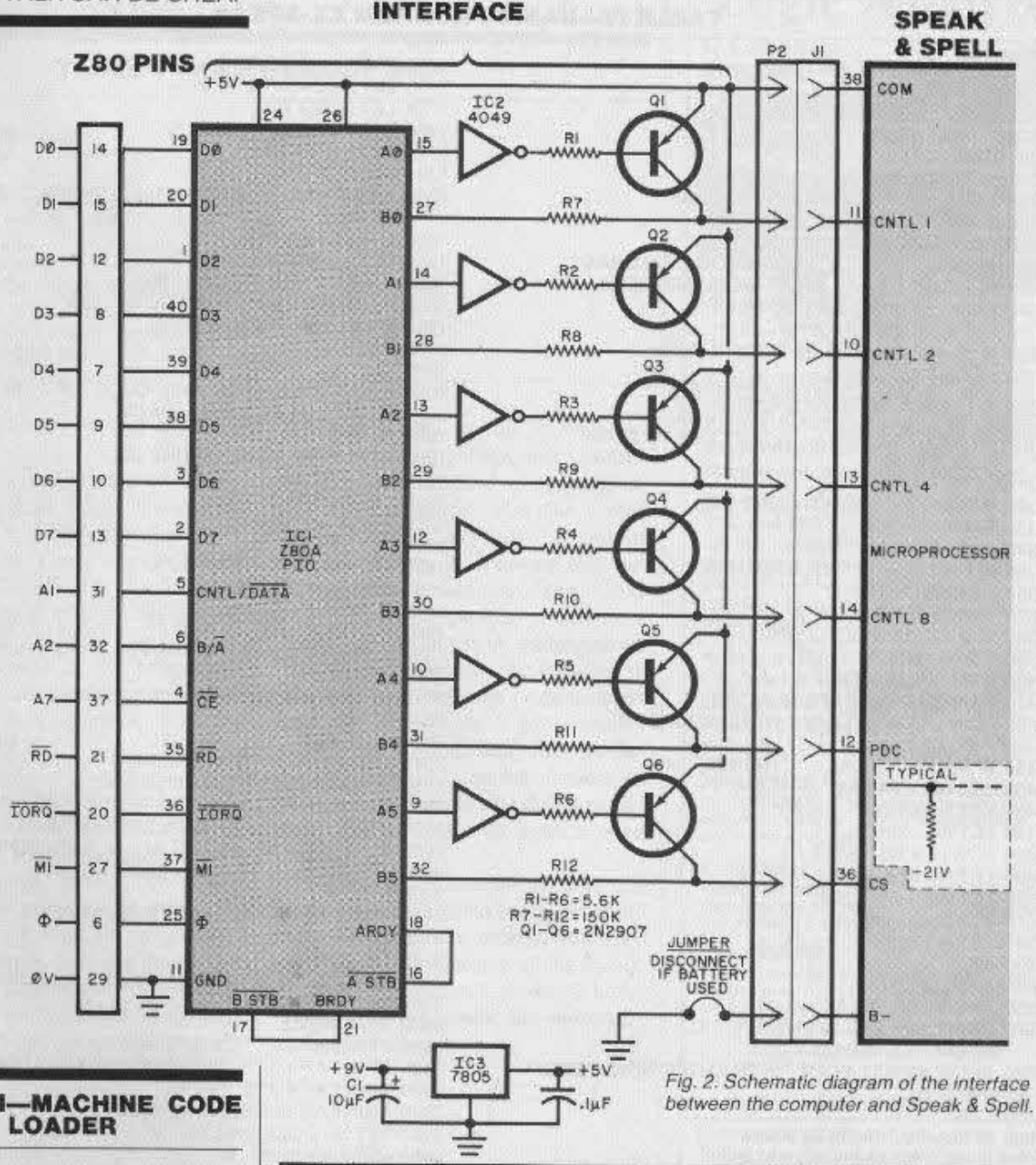


Fig. 2. Schematic diagram of the interface between the computer and Speak & Spell.

TABLE III—MACHINE CODE LOADER

```
    POKE 16389, 104
    NEW
    FAST
  1 REM (178 CHARACTERS)
 10 LET S = -1
 20 LET B = 256
 30 LET A1 = 0
 40 FOR I = 16514 to 16690
 50 PRINT AT 0,0; "ENTER"; I
 60 INPUT A
 70 IF A < 0 THEN GOTO 110
 80 IF A > 255 THEN GOTO 150
 90 SCROLL
100 POKE I, A
110 LET A1 = A1 + PEEK I
120 PRINT I; "="; PEEK I, A1
130 NEXT I
140 STOP
150 LET A1 = A1 - PEEK (I-1)
160 LET I = I - 2
170 NEXT I
```

PARTS LIST

C1—10-μF, 16-V electrolytic
C2—0.1-μF ceramic capacitor
IC1—Z80A PIO
IC2—4049 hex inverter
IC3—7805 5-V regulator
J1—14-pin DIP socket
P2—14-pin DIP connector
Q1 through Q6—2N2907 transistor
R1 through R6—5.6-kilohm, ¼-W resistor
R7 through R13—150-kilohm, ¼-W resistor
Misc.—Sockets, prototype board (Radio Shack 276-154 or similar), ribbon cable

Note: A cassette of software for the project is available from the author at PO Box 12100, Santa Ana, CA 92712, for $6.00. California residents, add sales tax.

Operation of the BASIC program of Table IV is straight-forward. You will be prompted whenever an input is required. To return to the menu, enter M. When your word map is complete, you can delete all but the REM statement containing the machine code and write your own programs for speech production. To output speech, POKE the addresses of the words you want spoken into the area above RAMTOP, by adding a loop to the SPEAK WORD routine (line 1400) and let the loop incre-

TABLE IV—BASIC PROGRAM ZX-SPEAK

```
   1 REM (Machine Code Here)
1000 REM ZX-SPEAK REV. 3.1
1010 REM (c) L. DIGHERA 1982
1020 IF PEEK 16389 < > 104 THEN STOP
1030 FAST
1040 GOTO 3000
1190 REM DETERMINE WORD ADDRESS
1200 SLOW
1210 PRINT AT 1,9;"WORD LOCATER"
1220 PRINT AT 4,3;"1. . .DISPLAY LOGIC DIAGRAM"
1230 PRINT AT 6,3;"2. . .PRINT WORD ADDRESS"
1240 PRINT AT 10,9;S$(A + 1);"MODE"
1250 LET S = (CODE INKEY$) - 28
1260 IF S = (CODE"M") - 28 THEN RETURN
1270 LET A = ABS(A - 1)
1280 IF S < 1 OR S > 2 THEN GOTO 1240
1290 IF INKEY$ < > " " THEN GOTO 1290
1300 PRINT AT 10,0;H$(A + 1);"ANY KEY TO START
     RECORDING."
1310 LET A = ABS(A - 1)
1320 IF INKEY$ = " "THEN GOTO 1300
1330 FAST
1340 CLS
1350 IF INKEY$ = "M" THEN RETURN
1360 RAND USR PIO
1370  RAND USR STODATA
1380 IF S - 1 THEN GOTO 1650
1390 GOTO 1900
1400 REM SPEAK WORD
1410 PRINT AT 1,10; "SPEAK WORD"
1420 PRINT AT 4,5; "ENTER WORD ADDRESS"
1430 INPUT A$
1440 IF A$ = "M" OR A$ = " "THEN RETURN
1450 LET WA = VAL A$
1460 LET H = 65536
1470 LET AS = 26624
1480 FOR I = 4 TO 0 STEP - 1
1490 LET N = INT(WA/H)
1500 LET WA = WA - N*H
1510 POKE AS + I,N
1520 LET H = H/16
1530 NEXT I
1540 RAND USR PIO
1550 RAND USR SPEAK
1560 PRINT AT 8,3;" ' ' ' 'ENTER' ' ' ' TO
     SPEAK";A$;"AGAIN."
1570 PRINT AT 4,5;" ' ' ' 'N' ' ' ' TO ENTER NEW WORD."
1580 INPUT B$
1590 CLS
1600 IF B$ = "M" THEN RETURN
1610 IF B$ = "N" THEN GOTO 1400
1620 GOTO 1540
1640 REM LOGIC DISPLAY
1650 FOR F = DATA TO 32488 STEP 60
1660 PRINT TAB 3; "** LOGIC SIGNAL DISPLAY**"
1670 PRINT "FRAME:";(F -
     DATA + 60)/60;TAB12;"12US/■";TAB22;"60■ /FRAME"
1680 PRINT AT 3,0;"CS",,,,,,"CK",,,,,,,,
     "C8",,,,,,"C4",,,,,,"C2",,,,,,"C1"
1690 PRINT AT 8,0"# #  +▬+▬+▬+▬+▬+▬+▬+▬+"
1700 FOR X - 0 TO 59
1710 LET L = PEEK(F + X)
1720 LET I$ = "72 + ?■■"
1730 PLOT X + 4,CODE I$ + INT(L/32)*2
1732 LET L = L - INT(L/32)*32
1734 LET I$ = I$(2 TO )
1736 PLOT X + 4,CODE I$ + INT(L/16)*2
1738 LET L = L - INT(L/16)*16
1740 LET I$ = I$(2 TO )
1742 PLOT X + 4,CODE I$ + INT(L/8)*2
1744 LET L = L - INT(L/8)*8
1746 LET I$ = I$(2 TO )
1748 PLOT X + 4,CODE I$ + INT(L/4)*2
1750 LET L = L - INT(L/4)*4
1752 LET I$ = I$(2 TO )
1754 PLOT X + 4,CODE I$ + INT(L/2)*2
1756 LET L = L - INT(L/2)*2
1758 LET I$-I$(2 TO )
1760 PLOT X + 4,CODE I$ + L*2
1770 NEXT X
1780 SLOW
1790 PRINT AT 21,4;H$(A + 1);" ""ENTER"" FOR NEXT
     FRAME"
1800 LET L$ = INKEY$
1810 LET A = ABS(A - 1)
1820 IF L$ = " " THEN GOTO 1800
1830 IF INKEY$ < > " " THEN GOTO 1830
1840 IF L$ = "M" THEN RETURN
1850 FAST
1860 CLS
1880 IF CODE L$ > 28 AND CODE L$ < 64 THEN LET
     F = (CODE L$ - 30)*60 + DATA
1890 NEXT F
1900 REM PRINT WORD ADDRESS
1910 LET F - DATA
1920 LET ADDR = 0
1930 FOR P = 0 TO 4
1940 LET F1 = 0
1950 LET F2 = 0
1960 FOR F = F TO 32512
1970 IF PEEK F = 32 + 16 + 2 THEN LET F1 = 1
1980 IF PEEK F < 32 + 16 AND F1 = 1 THEN LET F2 = 1
1990 IF PEEK F > = 32 + 16 AND F2 = 1 THEN GOTO 2010
2000 NEXT F
2010 LET ADDR = ADDR + 16**P*((PEEK F) - 32 - 16)
2020 LET F = F + 5
2030 NEXT F
2040 REM KEYBOARD ADDRESSES
2050 IF ADDR > 803 THEN GOTO 2140
2060 FOR F = F TO 32512
2070 IF PEEK F = 32 + 15 THEN LET F1 = F1 + 1
2080 IF PEEK F = 32 + 15 AND F1 > 14 THEN GOTO 1920
2090 IF PEEK F < > 32 + 15 THEN LET F1 = 0
2100 NEXT F
2110 PRINT AT 5,7;"ADDRESS NOT FOUND"
2120 SLOW
2130 GOTO1300
2140 PRINT AT 15,7;"WORD ADDRESS: ";ADDR
2150 GOTO 1190
2900 REM MENU
2910 PRINT AT 1,7;"**ZX-SPEAK**"
2920 PRINT AT 4,3;"1...DETERMINE WORD ADDRESS"
2930 PRINT AT 6,3;"2...SPEAK"
2940 PRINT AT 10,9;S$(A + 1);"MODE"
2950 LET S = CODE INKEY$ - 28
2960 LET A = ABS(A - 1)
2970 IF S < 1 OR S > 2 THEN GOTO 2940
2980 IF INKEY$ < > " "THEN GOTO 2980
2990 FAST
2998 RETURN
3000 REM****EXEC****
3010 LET MENU = 2900
3020 LET PI0 = 16514
3030 LET SPEAK = 16541
3040 LET OFF = 16610
3050 LET STODATA = 16658
3060 LET DATA = 27648
3070 DIM S$(2,8)
3080 LET S$(1) = " SELECT "
3090 LET S$(2) = "▮SELECT▮"
3100 DIM H$(2,5)
3110 LET H$(1) = " HIT "
3120 LET H$(2) = "▮HIT▮"
3130 LET A = 1
3140 SLOW
3150 GOSUB MENU
3160 CLS
3170 GOSUB S*200 + 1000
3180 CLS
3190 GOTO 3140
```

TABLE V—RAM MEMORY MAP

16514	PIO Initialization
16541	Speak
16610	Off
16620	PDC (H=64, L=236)
16648	Delay (H=65, L=5)
16658	Sto-Data
16690	End of REM
26624	RAMTOP (POKE 16389,104)
27647	End of NYBL Storage
27648	Bus Data Storage Begins (H=108)
32512	End of Data Storage (A-129)

ment variable AS by five for each word.

When all words are stored, delete all but the first REM statement. Then write a subroutine that calls PIO at 16514. Then, after POKEing 16542 and 16543 with the location of the word to be spoken, call SPEAK at 16541. Or, if desired, you can arrange the words in the proper sequence; and, after calling SPEAK initially, call 16341 (SPEAK +3) for the next word.

Conclusion. Only the basics of using the Speak & Spell vocalizer with Sinclair and Timex computers have been discussed here. There are many things you can do with the system beyond what we've presented. For example, you can locate the addresses of individual word sounds (phonemes) contained in ROM and string them together to make words that don't exist in the ROM's vocabulary, making it possible to build an almost unlimited dictionary of words. You might trim the prefix from the word "anything" to obtain "thing" simply by locating and using the starting address of the suffix.

Another approach to obtaining a larger vocabulary is by adding more ROMs to the system. Currently, as many as 16 ROMs can be connected into the system, each individually accesssed through the address-decoded ROM chip select. Access to data output from the ROMs is available at the Speak & Spell's edge connector.

The more you use the system, the more you're likely to learn about it. As you experiment with it, you may discover many features of the Speak & Spell we haven't covered here. You may even crack the word-encoding scheme. ◇