sp synth Sym

VOTRAX INTERFACE FOR SYM

SYM

Published in MICRO - The 6502/6809 Journal No. 47 April 1982
by John Valente

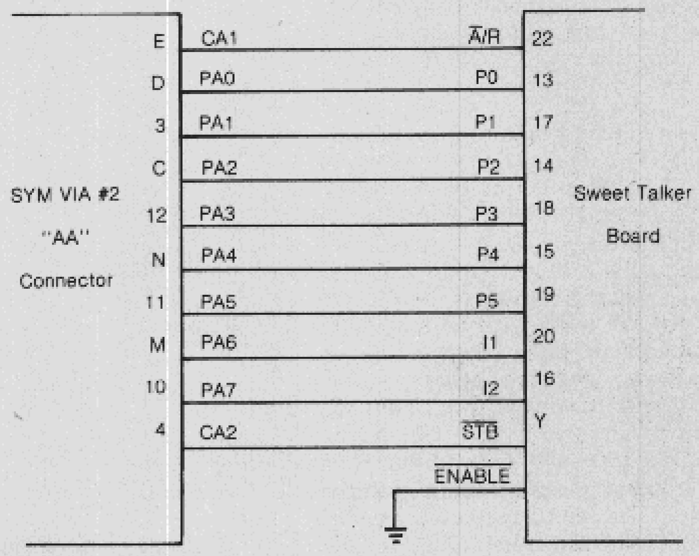
interface the Votrax Speech Synthesizer to your system's VIA. Although intended for a SYM-1, the techniques described are readily adaptable to other systems using a 6522.

VOTRAX DRIVER requires:
SYM-1
Sweet Talker Votrax
Interface Board
It is adaptable to othersystems (such as AIM) with 6522 VIA.

The Votrax SC-0l Speech Synthesizer IC lets you experiment with computer-generated speech at a reasonable cost. The Sweet Talker board, which includes the Votrax IC, allows easy interfacing to most computers. It is available from The Micro Mint, Inc., 917 Midway, Woodmere, NY 11598. While I will be describing the interface of the Votrax to my SYM-1 6522 VIA, the programs can be readily adapted to other systems using the 6522. This article provides a machine language driver, followed by a BASIC program to convert the mnemonics for each phoneme of speech into the numerical codes needed by Votrax. (A phoneme is one of the smallest units of speech that distinguishes one word from another; i.e., the m in mat and b in bat.)

Communication with the Votrax IC resembles a parallel printer interface with handshaking. Figure 1 shows the connections between the 6522 and the Sweet Talker board. This example uses VIA #2, port A, accessed through the SYM's "AA" connector. Any other VIA port can be used as long as all eight data bits and the two control lines are available. Be sure the Sweet Talker is connected to your power supply.

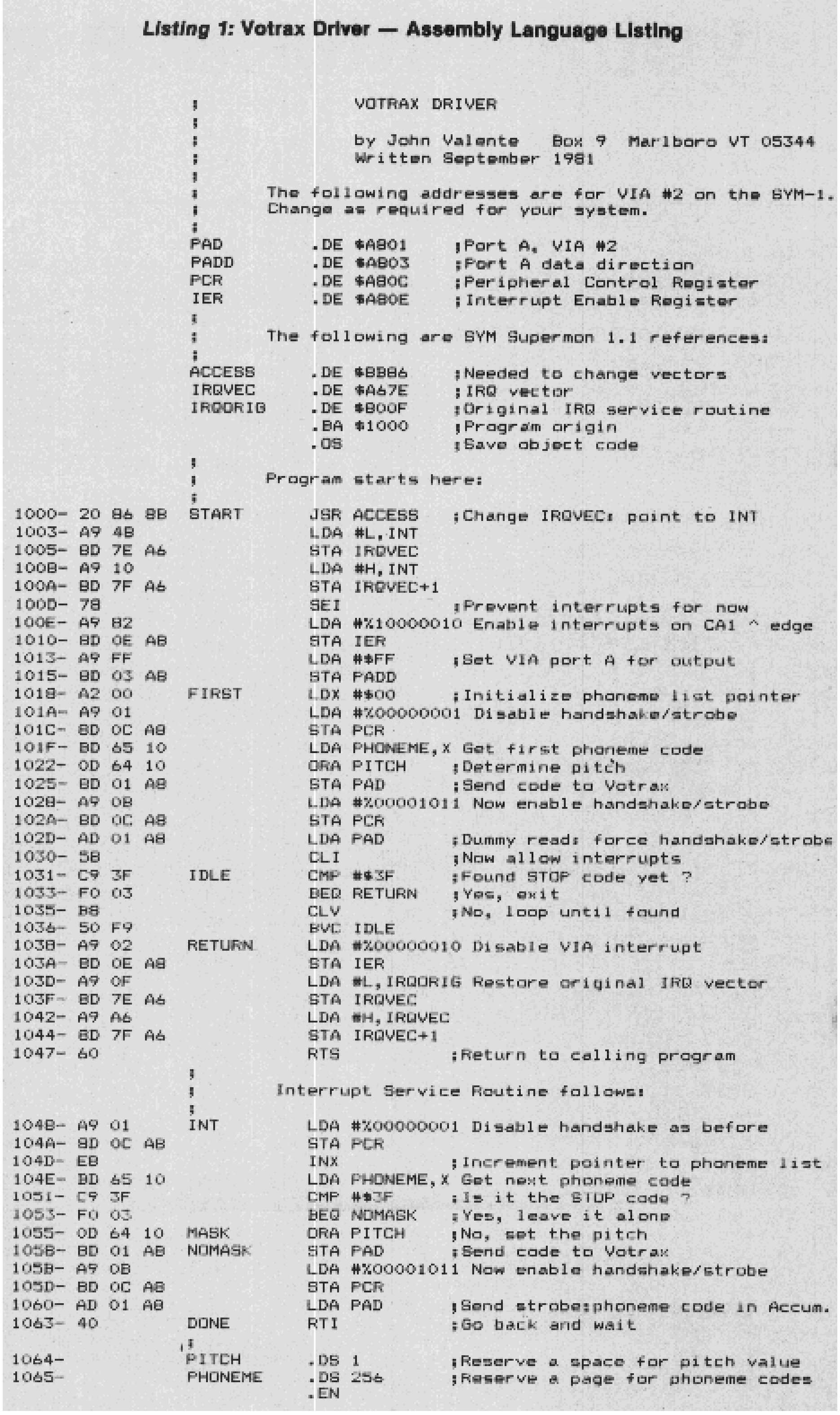
Figure 1



Bits zero through five form the code which tells Votrax which phoneme to synthesize. Bits six and seven select one of four pitch levels for the voice. Control line CA2 latches the data into the Votrax IC and starts synthesis. After the phoneme has sounded, A/R goes high. The CA1 line on the 6522 senses this transition and generates an interrupt, causing the next phoneme code to be sent out. This process repeats until a "stop" code ($3F) is sent. The ENABLE line is not used and is grounded to allow the board to operate.

Listing 1 sends a sequence of phoneme codes to the Sweet Talker with the proper timing of control signals. It is written in RAE - 1 format (SYM's Resident Assembler and Editor). The sequence of phonemes is limited to 256, due to the eight-bit length of the X index register. A block of memory to store the values for PITCH and the sequence of phonemes starting at PHONEME is reserved following the program. If you relocate this routine, you need only change the references to these locations. If your 6522 is at a different location, the register addresses will have to be modified accordingly.

Listing 1: Votrax - Assembly Language Listing



Listing 1: Votrax Driver — Assembly Language Listing

```
                    ;               VOTRAX DRIVER
                    ;
                    ;               by John Valente    Box 9  Marlboro VT 05344
                    ;               Written September 1981
                    ;
                    ;       The following addresses are for VIA #2 on the SYM-1.
                    ;       Change as required for your system.
                    ;
                    PAD         .DE $A801     ;Port A, VIA #2
                    PADD        .DE $A803     ;Port A data direction
                    PCR         .DE $A80C     ;Peripheral Control Register
                    IER         .DE $A80E     ;Interrupt Enable Register
                    ;
                    ;       The following are SYM Supermon 1.1 references:
                    ;
                    ACCESS      .DE $8B86     ;Needed to change vectors
                    IRQVEC      .DE $A67E     ;IRQ vector
                    IRQORIG     .DE $800F     ;Original IRQ service routine
                                .BA $1000     ;Program origin
                                .OS           ;Save object code
                    ;
                    ;       Program starts here:
                    ;
1000- 20 86 8B      START       JSR ACCESS    ;Change IRQVEC: point to INT
1003- A9 4B                     LDA #L,INT
1005- 8D 7E A6                  STA IRQVEC
100B- A9 10                     LDA #H,INT
100A- 8D 7F A6                  STA IRQVEC+1
100D- 78                        SEI           ;Prevent interrupts for now
100E- A9 82                     LDA #%10000010 Enable interrupts on CA1 ^ edge
1010- 8D 0E AB                  STA IER
1013- A9 FF                     LDA #$FF      ;Set VIA port A for output
1015- 8D 03 AB                  STA PADD
1018- A2 00         FIRST       LDX #$00      ;Initialize phoneme list pointer
101A- A9 01                     LDA #%00000001 Disable handshake/strobe
101C- 8D 0C A8                  STA PCR
101F- BD 65 10                  LDA PHONEME,X Get first phoneme code
1022- 0D 64 10                  ORA PITCH     ;Determine pitch
1025- 8D 01 A8                  STA PAD       ;Send code to Votrax
1028- A9 0B                     LDA #%00001011 Now enable handshake/strobe
102A- 8D 0C A8                  STA PCR
102D- AD 01 A8                  LDA PAD       ;Dummy read: force handshake/strobe
1030- 58                        CLI           ;Now allow interrupts
1031- C9 3F         IDLE        CMP #$3F      ;Found STOP code yet ?
1033- F0 03                     BEQ RETURN    ;Yes, exit
1035- B8                        CLV           ;No, loop until found
1036- 50 F9                     BVC IDLE
1038- A9 02         RETURN      LDA #%00000010 Disable VIA interrupt
103A- 8D 0E A8                  STA IER
103D- A9 0F                     LDA #L,IRQORIG Restore original IRQ vector
103F- 8D 7E A6                  STA IRQVEC
1042- A9 A6                     LDA #H,IRQVEC
1044- 8D 7F A6                  STA IRQVEC+1
1047- 60                        RTS           ;Return to calling program
                    ;
                    ;        Interrupt Service Routine follows:
                    ;
104B- A9 01         INT         LDA #%00000001 Disable handshake as before
104A- 8D 0C AB                  STA PCR
104D- E8                        INX           ;Increment pointer to phoneme list
104E- BD 65 10                  LDA PHONEME,X Get next phoneme code
1051- C9 3F                     CMP #$3F      ;Is it the STOP code ?
1053- F0 03                     BEQ NOMASK    ;Yes, leave it alone
1055- 0D 64 10      MASK        ORA PITCH     ;No, set the pitch
1058- 8D 01 AB      NOMASK      STA PAD       ;Send code to Votrax
105B- A9 0B                     LDA #%00001011 Now enable handshake/strobe
105D- 8D 0C A8                  STA PCR
1060- AD 01 A8                  LDA PAD       ;Send strobe:phoneme code in Accum.
1063- 40            DONE        RTI           ;Go back and wait
                    ;
1064-               PITCH       .DS 1         ;Reserve a space for pitch value
1065-               PHONEME     .DS 256       ;Reserve a page for phoneme codes
                                .EN
```

Listing 1. *** use browser BACK to return*

The program begins by pointing the IRQ vector to the INT routine. The SYM requires a JSR ACCESS before this operation. Then, interrupts are temporarily disabled and the 6522 Interrupt Enable Register (IER) is set up to generate an IRQ on the rising edge of CAl. The port is then configured for output.

You must send the first phoneme code manually. The X register, which is used to index into the PHONEME list, is set to zero. The next few lines of code are needed because of a timing restraint in the Votrax chip. After sending data to Votrax, wait at least 450 nsec before sending a strobe pulse to latch it. Since the 6522 normally waits only 300 nsec before pulsing, the handshake/strobe is first inhibited by sending a $01 to the Peripheral Control Register (PCR). The phoneme data is fetched from the list, OR'ed with the value of PITCH to select the voice, and sent down the line. Now the handshake is enabled by sending a $OB to the PCR. A dummy read of the port outputs the pulse, and by now much more than the required 450 nsec has elapsed.

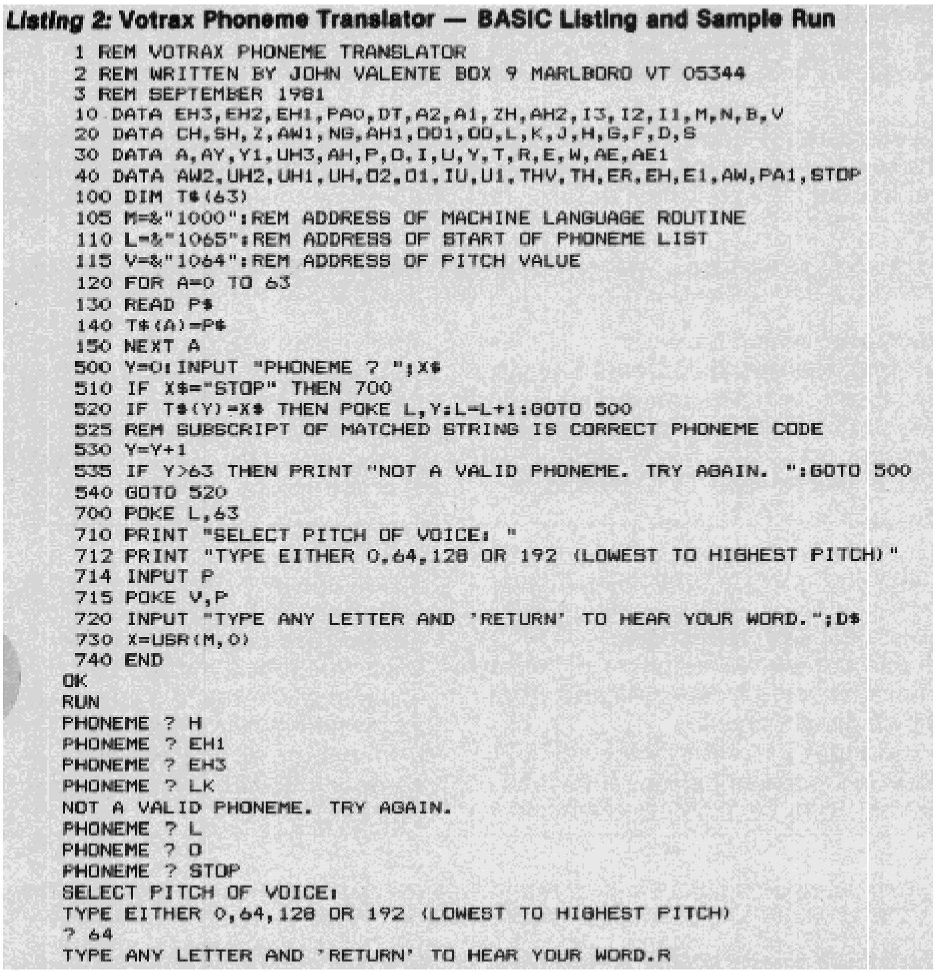
After the phoneme has sounded, the CAl line goes high and forces an interrupt. In the INT routine, the same manipulation of the handshake occurs to insure proper timing. The X register is incremented to point to the next phoneme code, and the data is fetched. But we don't want to OR the code with the value of PITCH if the "stop" code is encountered. After checking for this, the data and handshake are sent out as before. Conveniently, the dummy read at 1060 leaves the latest phoneme code in the accumulator.

As the program exits the INT routine, it loops continuously to IDLE until the "stop" code is found. Then the IER is altered to ignore further interrupts and the IRQ vector is restored to its original value. The address given is for the SYM's Supermon 1.1. Finally, control is restored to the calling program.

You can use this program directly by entering phoneme codes through the monitor, starting at address PHONEME and ending with $3F to end the speech. A value of $00, $40, $80 or $C0 must be entered at location PITCH. Try using the random values in memory as a phoneme list; you will hear some very bizarre sounds.

To compose intelligible words, however, it is more convenient to use mnemonic codes for each phoneme because these are closer to English. Listing 2, a BASIC program, prompts for the standard Votrax mnemonics, translates them into the proper numerical code, then places them sequentially into the PHONEME block. A call to the machine language driver produces the speech. Remember to reserve space for the machine language routine plus 257 bytes before entering BASIC.

Listing 2: Votrax Phoneme Translator - Basic listing and Sample Run



Listing 2: Votrax Phoneme Translator — BASIC Listing and Sample Run

```
   1 REM VOTRAX PHONEME TRANSLATOR
   2 REM WRITTEN BY JOHN VALENTE BOX 9 MARLBORO VT 05344
   3 REM SEPTEMBER 1981
   10 DATA EH3,EH2,EH1,PA0,DT,A2,A1,ZH,AH2,I3,I2,I1,M,N,B,V
   20 DATA CH,SH,Z,AW1,NG,AH1,OO1,OO,L,K,J,H,G,F,D,S
   30 DATA A,AY,Y1,UH3,AH,P,O,I,U,Y,T,R,E,W,AE,AE1
   40 DATA AW2,UH2,UH1,UH,O2,O1,IU,U1,THV,TH,ER,EH,E1,AW,PA1,STOP
   100 DIM T$(63)
   105 M=&"1000":REM ADDRESS OF MACHINE LANGUAGE ROUTINE
   110 L=&"1065":REM ADDRESS OF START OF PHONEME LIST
   115 V=&"1064":REM ADDRESS OF PITCH VALUE
   120 FOR A=0 TO 63
   130 READ P$
   140 T$(A)=P$
   150 NEXT A
   500 Y=0:INPUT "PHONEME ? ";X$
   510 IF X$="STOP" THEN 700
   520 IF T$(Y)=X$ THEN POKE L,Y:L=L+1:GOTO 500
   525 REM SUBSCRIPT OF MATCHED STRING IS CORRECT PHONEME CODE
   530 Y=Y+1
   535 IF Y>63 THEN PRINT "NOT A VALID PHONEME. TRY AGAIN. ":GOTO 500
   540 GOTO 520
   700 POKE L,63
   710 PRINT "SELECT PITCH OF VOICE: "
   712 PRINT "TYPE EITHER 0,64,128 OR 192 (LOWEST TO HIGHEST PITCH)"
   714 INPUT P
   715 POKE V,P
   720 INPUT "TYPE ANY LETTER AND 'RETURN' TO HEAR YOUR WORD.";D$
   730 X=USR(M,0)
   740 END
  OK
  RUN
  PHONEME ? H
  PHONEME ? EH1
  PHONEME ? EH3
  PHONEME ? LK
  NOT A VALID PHONEME. TRY AGAIN.
  PHONEME ? L
  PHONEME ? O
  PHONEME ? STOP
  SELECT PITCH OF VOICE:
  TYPE EITHER 0,64,128 OR 192 (LOWEST TO HIGHEST PITCH)
  ? 64
  TYPE ANY LETTER AND 'RETURN' TO HEAR YOUR WORD.R
```

Listing 2. ** *use browser BACK to return*

The program is written in Synertek Bas-1, but is easily translated into other BASICs. The &"XXXX" is Bas-1's convention for hexadecimal numbers, and X = USR (address,0) calls a machine language routine. Lines 10-40 list the standard Votrax mnemonics, and lines 130-150 READ them into an array for later comparison. Lines 105-115 assign the machine language addresses to variables; simply change these lines to conform to any relocation you have made. Lines 500-540 prompt for the desired phoneme, search for a match, and then POKE the corresponding code into the PHONEME list. When you enter STOP in answer to the prompt, the program asks you to select a pitch for the voice. After responding to line 720, the machine language driver is called and you will hear the result.

I suggest experimenting with single words hefore assembling long messages. I think you will find that the components of human speech are very complex. Two words which rhyme to our ears are often composed of different series of phonemes. what might seem to be a simple vowel sound is sometimes a series of two or even three different phonemes. Be sure to include pauses between words (two different pause mnemonics are available).

VOTRAX is a trademark of Federal Screw Works, Inc.

John Valente
: is interested in using the computer to generate sounds and musical stnictures unavailable in conventional instruments. He has been published in

Electronotes, Newsletter of die Musical Engineering Group. You can write to Valente at Box 9, Marlboro, VT 05344.