APPLE TALKER

Add the power of speech to your Apple without hardware. A combination of Applesoft and machine language programming makes it possible to build a disk library of sounds or words and to combine them in novel ways.

by Mike Eve
15621 S.E. 178th St.
Renton, WA 98055

Computerized speech is becoming more available for home computers. Speech boards for the Apple are now available in the $200 range. APPLE TALKER is the software alternative which allows you to add speech (and other complex sounds) to your programs without purchasing more hardware.

APPLE TALKER turns your Apple into a digital recorder. Sounds and words prerecorded on tape are introduced into the Apple via the cassette input, sampled, and stored in memory. Once in memory, the sounds may be transferred to disk, combined with other sounds, and replayed.

THEORY

Just as programs can be stored in memory as a series of 0's and 1's, so can sound. By sampling sound at a fixed rate and then playing the samples back at the same rate, the sound can be reconstructed. The sound fidelity can be made as good as desired by increasing the sample rate and the measurement accuracy. By taking very good samples, very fast, it is possible to produce a digital recording that is indistinguishable from the original.

This is the principle behind digital audio which is just now becoming available for the home. These new systems use 40 to 50 thousand samples, of 12 to 16 bits each, every second. APPLE TALKER will make 7300 1-bit samples per second. This rate and resolution are adequate to produce recognizable voice, although quite a bit of distortion is introduced. APPLE TALKER works best with short, common words.

ENTERING APPLE TALKER

APPLE TALKER consists of an Applesoft main routine which calls the assembly language routine for the time-critical tasks.

First, enter Applesoft and type in the main routine. Save it on disk under the name APPLE TALKER.

Next, enter the machine code directly into memory, or assemble the source. Use **BSAVE APPLE TALKER.OBJ,A$300,L$CE** to save your work to disk.

RUNNING APPLE TALKER

Connect your tape recorder to the cassette input port of your Apple. You need not connect the output port. Next, record some sounds on tape. To experiment with the features of APPLE TALKER, I suggest you record yourself counting from 1 to 5. Count slowly and clearly.

Now, RUN APPLE TALKER. The menu will appear along with a buffer status at the bottom of the screen. There are eight menu choices (FIGURE 1). As you step through the various options, the name of the currently selected option will appear below the menu. As you fill memory with your sounds, the pointers P1 and P2 in the status line will indicate your working buffer. The minimum and maximum values of P1 and P2 are given for reference.

Type '1' to enter the echo routine, and begin playing your cassette. As the cassette plays, your voice is sampled every 137 milliseconds. This sample is fed back to your Apple speaker. What you are hearing is exactly what your digital recording will sound like. You can control the quality somewhat by adjusting the volume control on the cassette. Adjust the volume for best response, press any key to quit echoing, and rewind your cassette.

Type '2' to enter the record routine. Play your cassette. It will be echoed until you press a key to begin recording. This gives you some additional control over the recording process. The routine will record the first five seconds of your monologue. Note that P2 has changed to point to the end of your recording. You are now done with the cassette.

Type '3' to play what you have recorded. Do not press keys 1 or 2 at this time. You will hear everything in the buffer from P1 to P2.

Type '4' to save your recording on disk. This will become the master you use for editing. The recording will be saved under the name you give with ".SOUND" appended.

BUILDING A LIBRARY

Now that you have created a master recording, you can go back and create a library of words. In option 3 (play), keys 1 and 2 can be used to adjust the buffer pointers to isolate words. Each word can be saved in its own disk file, and then combined with other words to form new phrases. After isolating and saving a word, option 6 can be used to restore P1 and P2 to the values they had before you used the 1 and 2 keys. You can then isolate another word in the master recording.

continued on page 75

If you need even finer control of P1 and P2 than provided by the play option, you can invoke option 7 to specify values for P1 and P2. You can also adjust the default recording time, and vary the sampling rate. The minimum time between samples is 87 microseconds. Additional delay can be added in 5 microsecond increments. The additional delay is defaulted to 50 microseconds for a total delay of 137 microseconds between samples. Unusual effects can be achieved by sampling at one rate and replaying at another.

After building a library, you can form new phrases using option 5 to load in the different words. Option 5 allows you to either load a word at the beginning of the buffer, or to append to the words already in place.

To exit the program, use option 8. This will restore HIMEM to its original value. If the program should crash for any reason, reboot the system to restore HIMEM.

HOW THE PROGRAM WORKS

The main routine BLOADs the assembly language routines into page 3. This page is not used by Applesoft or DOS, and keeps the routines out of the way. The main routine also resets HIMEM to make room for the sound buffer. HIMEM is set to a multiple of a thousand so that the program has at least a thousand bytes of memory available for variable and string storage. The sound buffer extends from the new HIMEM to the old HIMEM. The old HIMEM is restored when the program ends.

The assembly language routines are the most important part of APPLE TALKER. These routines have been carefully constructed so that each loop takes exactly the same amount of time, even though the loop does not always execute the same instructions.

For example, the input loop must read a bit, rotate it into the accumulator via the carry, and, if the accumulator is full (eight samples/byte), store it in the buffer. If the accumulator is not full, the address calculation is skipped. To maintain a constant loop time, a wait loop had to be added.

The output loop is similar to the input loop except that the program must compare the old sample to the new sample. If the samples are different, then the speaker is nudged; otherwise, the speaker is left alone. These two different actions required another delay loop to equalize processing time.

The echo routine is the simplest of the three sound routines, and has only one main processing path.

"If the samples are different, then the speaker is nudged . . ."

All the sound routines possess a "universal delay" loop. This loop was added so that the sample interval of all the routines can be changed by modifying only one, common parameter.

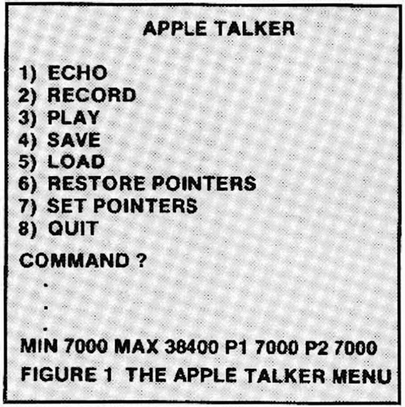
One note on program structure. As APPLE TALKER evolved, the assembly language routines moved and so did their entry points. This required editing the main routine to correct the CALL statements. I decided to place JMPs to each routine in the beginning. Now the CALLs point to these indirect entry points, and changes can be made to the assembly portion without affecting the main routine.

MODIFYING APPLE TALKER

You may wish to improve APPLE TALKER by increasing the sample rate. It is possible to decrease the sample time to 35-40 microseconds by replacing the JSRs with in-line code. This will give you a whopping 25,000 samples/second (about 3K bytes/second); however, be forewarned that APPLE TALKER's limiting factor for speech is not the sample rate, but the sample resolution. With only one bit/sample, the added samples will not be worthwhile.

The assembly language portion of APPLE TALKER may be easily incorporated into your own programs. Simply have the main routine BLOAD the assembly routines, allocate and load a sound buffer, and POKE the buffer pointers into locations 6-9. Liberal use of disk files is indicated to conserve memory.

In closing, as APPLE TALKER would say, "Have fun!"



```
          APPLE TALKER

1) ECHO
2) RECORD
3) PLAY
4) SAVE
5) LOAD
6) RESTORE POINTERS
7) SET POINTERS
8) QUIT

COMMAND ?
  .
  .
  .

MIN 7000 MAX 38400 P1 7000 P2 7000
```

FIGURE 1 THE APPLE TALKER MENU

Apple Talker (Cont.)

LISTING 1: APPLE TALKER

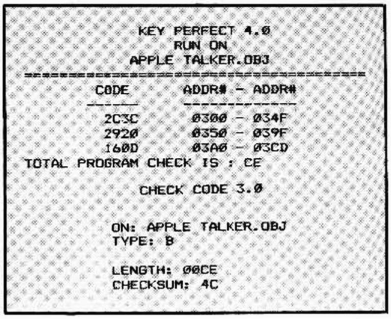
ED. RAM DISK, P.A.C. and APPLE TALKER are available on diskette for an introductory price of $19.95 plus $1.50 shipping/handling ($2.50 outside the U.S.) from NIBBLE, P.O. Box 325, Lincoln, MA 01773. Offer expires 2/29/84.

```
1   REM **********************
2   REM *   APPLE  TALKER    *
3   REM *    BY  MIKE  EVE   *
4   REM * COPYRIGHT (C) 1983 *
5   REM * BY MICROSPARC, INC *
6   REM * LINCOLN, MA. 01773 *
7   REM **********************
20  PRINT  CHR$ (4);"BLOAD APPLE TALKER.OBJ": REM     L
     OAD AT $300
30  REM  INITIALIZATION
40  DEF  FN PK(X) =  PEEK (X) + 256 *  PEEK (X + 1)
50  ONERR  GOTO 1210
60 M2 =  FN PK(115): REM  GET HIMEM
70 M1 =  FN PK(105): REM  TOP OF PROGRAM
80 M1 = (2 +  INT (M1 / 1000)) * 1000: REM  1000<VARIA
     BLE STORAGE<2000
90 P1 = M1:P2 = P1: REM    BUFFER PTRS
100 O1 = P1:O2 = P2
110 TW = 5
120 TD = 11: POKE 780, INT (TD): REM    COMMON DELAY
130  HIMEM: M1: REM    SET NEW HIMEM
140 BS = 1 / (8 * (5 * (TD - 1) + 87) * 1E - 6): REM B
     YTES/SECOND
150 S$ = ".SOUND": REM  SAVE FILE SUFFIX
160  REM    DEFINE MENU ITEMS
170 C$(1) = "ECHC"
180 C$(2) = "RECORD"
190 C$(3) = "PLAY"
200 C$(4) = "SAVE"
210 C$(5) = "LOAD"
220 C$(6) = "RESTORE POINTERS"
230 C$(7) = "SET POINTERS"
240 C$(8) = "QUIT"
250 MC = 8
260  REM    DISPLAY SCREEN
270  HOME : PRINT "** COPYRIGHT 1983 BY MICROSPARC, IN
     C. **": PRINT  TAB( 13);"APPLE  TALKER": PRINT
280  FOR I = 1 TO MC: PRINT  TAB( 10);I;") ";C$(I): NEXT
     I
290  GOSUB 1150: REM  PRINT STATUS
300  VTAB MC + 5: HTAB 10
310  INPUT "COMMAND ?";C: IF C < 1 OR C > MC THEN 260
320  HTAB 10: INVERSE : PRINT : PRINT C$(C): NORMAL : PRINT
330  ON C GOSUB 350,390,470,660,700,820,850,1050
340  GOTO 260
350  REM   ECHO
360  PRINT "HIT ANY KEY TO STOP"
370  CALL 774
380  RETURN
390  REM  RECORD
400 P1 = M1:P2 = P1 + TW * BS - 1
410  GOSUB 1090: CALL 777: REM  ZERO BUFFER
420  GOSUB 1090: REM  SETUP
430  PRINT "HIT ANY KEY TO BEGIN"
440  CALL 774: REM   ECHO TILL KEY
450  CALL 768: REM   RECORD
460  RETURN
470  REM   PLAYBACK
480  PRINT "USE KEYS 1 AND 2 TO EDIT": PRINT "ANY OTHE
     R TO EXIT"
490  GOSUB 1090: REM  SETUP
500  CALL 771: REM  PLAY
510 KY =  PEEK (25): REM  READ KEY
520  IF KY = 0 THEN  RETURN : REM  NO KEY
530 KY = KY - 128 -  ASC ("0")
540  IF KY = 1 THEN  GOTO 570
550  IF KY = 2 THEN  GOTO 620
560  RETURN
570  IF P2 <  FN PK(6) THEN 500: REM  P1 MUST BE > P2
580 O1 = P1:P1 =  FN PK(6): REM   SAVE BEGIN
590  GOSUB 1150: REM   PRINT STATUS
600  GOSUB 1130: REM   PAUSE
610  GOTO 500: REM  PLAY MORE
620 O2 = P2:P2 =  FN PK(6): REM   SAVE END
630  GOSUB 1150: REM    PRINT STATUS
640  GOSUB 1130: REM   PAUSE
650  GOTO 500
660  REM   SAVE
670  INPUT "FILENAME ? ";F$
680  PRINT  CHR$ (4)"BSAVE"F$S$",A" INT (P1);",L"; INT
     (P2 - P1)
690  RETURN
700  REM   LOAD
710  IF P2 = M1 THEN 770: REM    DO NOTHING
720  INPUT "APPEND ?(Y/N)";A$
730  IF A$ <  > "Y" AND A$ <  > "N" THEN 720
740  IF A$ = "Y" THEN 770
750 P1 = M1:P2 = M2: GOSUB 1090: CALL 777:P2 = P1: REM
      ZERO BUFFER
760  GOSUB 1150: GOTO 780: REM  PRINT STATUS
770 O1 = P1:P1 = P2:P2 = M2: GOSUB 1090: CALL 777:P2 =
     P1:P1 = O1: REM   ZERO REMAINING BUFFER
780  CALL  - 868: INPUT "FILENAME ? ";F$: PRINT  CHR$
     (4)"BLOAD"F$S$",A" INT (P2)
790 P2 = P2 +  FN PK( - 21920): REM   CALCULATE END OF
      BLOAD
800  IF P2 > M2 THEN  PRINT "ERROR:BLOAD WENT PAST HIM
     EM": PRINT "PLEASE REBOOT": END
810  RETURN
820  REM   RESET P1 AND P2
830 P1 = O1:P2 = O2
840  RETURN
850  REM   SET PARAMETERS
860  PRINT "ENTER NEW VALUE OR RETURN IF NO CHANGE"
870  PRINT : PRINT "P1="; INT (P1);
880  INPUT "  ";A$: IF  LEN (A$) = 0 THEN 910
890 T1 =  VAL (A$): IF T1 >  = M1 AND T1 <  = M2 THEN
     P1 = T1: GOTO 910: REM    ACCEPT IF IN RANGE
900  GOTC 860
910  PRINT "P2="; INT (P2);
920  INPUT "  ";A$: IF  LEN (A$) = 0 THEN 950
930 T2 =  VAL (A$): IF T2 <  = M2 AND T2 >  = P1 THEN
     P2 = T2: GOTO 950: REM    ACCEPT IF IN RANGE
940  GOTC 910
950 T1 = 5 * (TD - 1) + 87: REM COMPUTE DELAY TIME
960  PRINT "SAMPLE INTERVAL="; INT (T1);
970  INPUT "  ";A$: IF  LEN (A$) = 0 THEN  GOTO 1000
980 T2 =  VAL (A$): IF T2 < 87 OR T2 > 343 THEN  GOTO
     950: REM  343=256+87(MAX DELAY THAT FITS IN ONE B
     YTE)
990 TD =  INT ((T2 - 87) / 5) + 1: POKE 780, INT (TD):
     BS = 1 / (8 * (5 * (TD - 1) + 87) * 1E - 6)
1000  PRINT "TIME WINDCW="; INT (TW);
1010  INFUT "  ";A$: IF  LEN (A$) = 0 THEN  GOTO 1040
1020 T3 =  VAL (A$): IF T3 > 0 AND M1 + BS * T3 < M2 THEN
     TW = T3: GOTO 1040
1030  GOTO 1000
1040  RETURN
1050  REM   QUIT
1060  GOSUB 1300: REM   RESET HIMEM
1070  HOME
1080  END
1090  REM   SETUP P1 AND P2
1100  POKE 7, INT (P1 / 256): POKE 6, INT (P1 - 256 *
      INT (P1 / 256))
1110  POKE 9, INT (P2 / 256): POKE 8, INT (P2 - 256 *
      INT (P2 / 256))
1120  RETURN
1130  REM   PAUSE
1140  FOR I = 1 TO 200: NEXT I: RETURN
1150  REM   PRINT STATUS
1160 CV =  PEEK (37): REM  SAVE CURSOR ROW
1170  VTAB 22: CALL  - 868: REM  CLEAR LINE
1180  VTAB 22: INVERSE : PRINT "MIN "; INT (M1);" MAX
     "; INT (M2);" P1 "; INT (P1);" P2 "; INT (P2): NORMAL
1190  VTAB CV: REM  RESTORE CURSOR
1200  RETURN
1210  REM   ERROR
1220 ER =  PEEK (222)
1230  PRINT "ERROR # ";ER
1240  POKE 216,0
1250  PRINT "RESTARTING PROGRAM"
1260  GOSUB 1300: REM   RESTORE HIMEM
1270  CLEAR : REM  RESET SUBROUTINE STACK,ETC
1280  FOR I = 1 TO 2000: NEXT I: REM   PAUSE
1290  GOTO 30
1300  REM   RESET HIMEM
1310  POKE 116, INT (M2 / 256): POKE 115,(M2 - 256 *  PEEK
     (116))
1320  RETURN
```

```
              KEY PERFECT 4.Ø
                  RUN ON
               APPLE TALKER
==========================================
        CODE      LINE# - LINE#
        ----      -------------
        6FA9          1 -    4Ø              CHECK CODE 3.Ø
        78F3         5Ø -   14Ø
        5868        15Ø -   24Ø             ON: APPLE TALKER
        7DB9        25Ø -   34Ø             TYPE: A
        47F7        35Ø -   44Ø
        5Ø3C        45Ø -   54Ø             LENGTH: ØAAC
        5AA5        55Ø -   64Ø             CHECKSUM: 7Ø
        4CBD        65Ø -   74Ø
        9B2C        75Ø -   84Ø
        7CB7        85Ø -   94Ø
        9D19        95Ø -  1Ø4Ø
        4DF8       1Ø5Ø -  114Ø
        5BF2       115Ø -  124Ø
        5Ø57       125Ø -  132Ø
TOTAL PROGRAM CHECK IS : ØC4C
```

LISTING 2: APPLE TALKER. OBJ

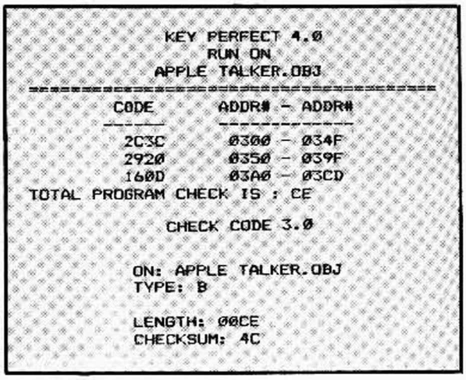
```
:ASM
                *         APPLE  TALKER
                *          ML ROUTINES
                *          BY MIKE EVE
                *      COPYRIGHT (C) 1983
                *        MICROSPARC, INC.

                P1L       EQU  $6
                P1H       EQU  P1L+1
                P2L       EQU  P1H+1
                P2H       EQU  P2L+1
                KEYSAV    EQU  $19
                KEYIN     EQU  $C000
                KEYSTR    EQU  $C010
                TAPEIN    EQU  $C060
                SPKR      EQU  $C030

                          ORG  $300

                * CONSTANT ENTRY POINTS

0300: 4C 0D 03            JMP  INPUT
0303: 4C 3B 03            JMP  OUTPUT
0306: 4C 91 03            JMP  ECHO
0309: 4C C1 03            JMP  ZERO
030C: 0B        DELAY     DFB  $0B          ;ADDITIONAL DELAY DEFAULT

                * INPUT ROUTINE

030D: A9 00     INPUT     LDA  #$00         ;ZERO SAMPLE SAVE
030F: A0 08               LDY  #$08         ;Y IS BIT COUNTER
0311: A2 00               LDX  #$00         ;FOR INDIRECT ADDR
0313: AE 0C 03  INLOOP    LDX  DELAY        ;UNIVERSAL DELAY LOOP
0316: CA        INDLY     DEX
0317: D0 FD               BNE  INDLY
0319: 48                  PHA               ;SAVE CURRENT BITS
031A: AD 60 C0            LDA  TAPEIN       ;GET NEW BIT
031D: 2A                  ROL               ;BIT TO CARRY
031E: 68                  PLA               ;RECALL CURRENT
031F: 2A                  ROL               ;APPEND NEW BIT
0320: 88                  DEY               ;ONE LESS BIT
0321: D0 10               BNE  DELAY1       ;FULL BYTE?
0323: 81 06               STA  (P1L,X)      ;YES. SAVE IT.
0325: 20 7F 03            JSR  NXTP1        ;ADDRESS NEXT BYTE
0328: B0 10               BCS  INRTS        ;RETURN IF DONE.
032A: A0 08               LDY  #08          ;RESET BIT COUNTER
032C: A2 02               LDX  #02          ;DELAY COUNTER
032E: CA        INDX      DEX               ;DELAY LOOP
032F: D0 FD               BNE  INDX
0331: F0 E0               BEQ  INLOOP       ;ALWAYS  BRANCH
0333: A2 0B     DELAY1    LDX  #$0B         ;ALTERNATE DELAY
0335: CA        INDX2     DEX
0336: D0 FD               BNE  INDX2
0338: F0 D9               BEQ  INLOOP       ;ALWAYS BRANCH
033A: 60        INRTS     RTS

                * OUTPUT ROUTINE

033B: A2 00     OUTPUT    LDX  #$00         ;ASSUME OLD=ZERO
033D: 8E CD 03            STX  OLD
0340: 86 19               STX  KEYSAV       ;NO KEY YET
0342: A1 06               LDA  (P1L,X)      ;GET FIRST BYTE
0344: A0 09               LDY  #$09         ;BIT COUNTER
0346: AE 0C 03  OUTLOOP   LDX  DELAY        ;UNIVERSAL DELAY LOOP
0349: CA        OUTDLY    DEX
034A: D0 FD               BNE  OUTDLY
034C: 88                  DEY               ;ONE LESS BIT
034D: D0 0B               BNE  WAIT1        ;BRANCH IF MORE BITS
034F: 20 7F 03            JSR  NXTP1        ;GET NEXT ADDRESS
0352: B0 2A               BCS  OUTRTS       ;EXIT IF DONE
0354: A1 06               LDA  (P1L,X)      ;GET NEXT BYTE
0356: A0 08               LDY  #$08         ;RESET BIT COUNTER
0358: 90 0C               BCC  CMPBIT       ;ALWAYS BRANCH
035A: AE 00 C0  WAIT1     LDX  KEYIN        ;KEY PRESSED?
035D: 10 02               BPL  WAIT2        ;IF NO, BRANCH
035F: 30 18               BMI  OUTKEY       ;IF SO, EXIT
0361: A2 08     WAIT2     LDX  #$08         ;DELAY
0363: CA        OUTDX     DEX
0364: D0 FD               BNE  OUTDX
0366: 48        CMPBIT    PHA               ;SAVE CURRENT BITS
0367: 4D CD 03            EOR  OLD          ;COMPARE TO OLD
036A: 10 0B               BPL  QUIET        ;SIGN BIT IMPORTANT
036C: AD 30 C0            LDA  SPKR         ;NOT SAME,HIT SPKR
036F: 68        NXTBIT    PLA               ;RECALL CURRENT
0370: 8D CD 03            STA  OLD          ;SAVE AS OLD
0373: 2A                  ROL               ;NEXT BIT TO SIGN
0374: 4C 46 03            JMP  OUTLOOP
0377: 10 F6     QUIET     BPL  NXTBIT       ;DELAY,ALWAYS BRANCH
0379: 86 19     OUTKEY    STX  KEYSAV       ;SAVE KEY
037B: AE 10 C0            LDX  KEYSTR       ;CLEAR STROBE
037E: 60        OUTRTS    RTS

                * NXTP1 ROUTINE
                *    REPLACES NXTP1 WITH EQUAL TIME
                *    BRANCHES.

037F: A5 06     NXTP1     LDA  P1L
0381: C5 08               CMP  P2L
0383: A5 07               LDA  P1H
0385: E5 09               SBC  P2H
0387: E6 06               INC  P1L
0389: D0 03               BNE  NXTRTS
038B: E6 07               INC  P1H
038D: 60                  RTS
038E: EA        NXTRTS    NOP
038F: EA                  NOP
0390: 60                  RTS

                * ECHO ROUTINE

0391: A9 00     ECHO      LDA  #$00
0393: 8D CD 03            STA  OLD
0396: AE 0C 03  ELOOP     LDX  DELAY        ;UNIVERSAL DELAY LOOP
0399: CA        ECHODLY   DEX
039A: D0 FD               BNE  ECHODLY
039C: AD 60 C0            LDA  TAPEIN
039F: 4D CD 03            EOR  OLD
03A2: 30 03               BMI  EOUTPT
03A4: EA                  NOP
03A5: 10 03               BPL  SAVE         ;ALWAYS BRANCH
03A7: AE 30 C0  EOUTPT    LDX  SPKR
03AA: 4D CD 03  SAVE      EOR  OLD
03AD: 8D CD 03            STA  OLD
03B0: AD 00 C0            LDA  KEYIN
03B3: 10 04               BPL  ECHODX
03B5: AD 10 C0            LDA  KEYSTR
03B8: 60                  RTS
03B9: A2 09     ECHODX    LDX  #$09
03BB: CA        DX        DEX
03BC: D0 FD               BNE  DX
03BE: 4C 96 03            JMP  ELOOP

                * ZERO FROM P1 TO P2

03C1: A0 00     ZERO      LDY  #00
03C3: A9 00     ZLOOP     LDA  #$00
03C5: 91 06               STA  (P1L),Y
03C7: 20 7F 03            JSR  NXTP1
03CA: 90 F7               BCC  ZLOOP
03CC: 60                  RTS

                OLD       DS   1

--End assembly--

206 bytes

Errors: 0
```



```
            KEY PERFECT 4.0
                RUN ON
           APPLE TALKER.OBJ
 ==========================================
         CODE        ADDR# - ADDR#
        ------      ---------------
          2C3C       0300 - 034F
          2920       0350 - 039F
          160D       03A0 - 03CD
 TOTAL PROGRAM CHECK IS : CE

            CHECK CODE 3.0


        ON: APPLE TALKER.OBJ
        TYPE: B

        LENGTH: 00CE
        CHECKSUM: 4C
```

Apple Talker (Cont.)

```

                    * NXTP1 ROUTINE
                    *    REPLACES NXTP1 WITH EQUAL TIME
                    *    BRANCHES.

037F: A5 06         NXTP1       LDA   P1L
0381: C5 08                     CMP   P2L
0383: A5 07                     LDA   P1H
0385: E5 09                     SBC   P2H
0387: E6 06                     INC   P1L
0389: D0 03                     BNE   NXTRTS
038B: E6 07                     INC   P1H
038D: 60                        RTS
038E: EA            NXTRTS      NOP
038F: EA                        NOP
0390: 60                        RTS

                    * ECHO ROUTINE

0391: A9 00         ECHO        LDA   #$00
0393: 8D CD 03                  STA   OLD
0396: AE 8C 03      ELOOP       LDX   DELAY         ;UNIVERSAL DELAY LOOP
0399: CA            ECHODLY     DEX
039A: D0 FD                     BNE   ECHODLY
039C: AD 60 C0                  LDA   TAPEIN
039F: 4D CD 03                  EOR   OLD
03A2: 30 03                     BMI   EOUTPT
03A4: EA                        NOP
03A5: 10 03                     BPL   SAVE          ;ALWAYS BRANCH
03A7: AE 30 C0      EOUTPT      LDX   SPKR
03AA: 4D CD 03      SAVE        EOR   OLD
03AD: 8D CD 03                  STA   OLD
03B0: AD 00 C0                  LDA   KEYIN
03B3: 10 04                     BPL   ECHODX
03B5: AD 10 C0                  LDA   KEYSTR
03B8: 60                        RTS
03B9: A2 09         ECHODX      LDX   #$09
03BB: CA            DX          DEX
03BC: D0 FD                     BNE   DX
03BE: 4C 96 03                  JMP   ELOOP

                    * ZERO FROM P1 TO P2

03C1: A0 00         ZERO        LDY   #00
03C3: A9 00         ZLOOP       LDA   #$00
03C5: 91 06                     STA   (P1L),Y
03C7: 20 7F 03                  JSR   NXTP1
03CA: 90 F7                     BCC   ZLOOP
03CC: 60                        RTS

                    OLD         DS    1


--End assembly--

206 bytes

Errors: 0
```



```
             KEY PERFECT 4.Ø
                 RUN ON
            APPLE TALKER.OBJ
=========================================
        CODE        ADDR# - ADDR#
        ----        -------------
        2C3C        Ø300  - Ø34F
        2920        Ø35Ø  - Ø39F
        160D        Ø3AØ  - Ø3CD
TOTAL PROGRAM CHECK IS : CE

             CHECK CODE 3.Ø

        ON: APPLE TALKER.OBJ
        TYPE: B

        LENGTH: ØØCE
        CHECKSUM: 4C
```