APPLE-LIS'NER™

by Bob Bishop

- *Software speech recognition for your Apple II*
- *Make your own "lis'ner" programs*

"Your Apple's Ears"

APPLE-LIS'NER™ CASSETTE ALB-978 ™

10756 Vanowen, North Hollywood, Ca. 91605

APPLE-LIS'NER DESCRIPTION
APPLE-LIS'NER IS A SOFTWARE PACKAGE THAT ALLOWS YOU TO PERFORM VOICE RECOGNITION EXPERIMENTS WITH YOUR APPLE II COMPUTER. IT HAS THE ABILITY TO RECOGNIZE UP TO 31 DIFFERENT "WORDS". EACH "WORD" CAN BE A SOUND OR A SERIES OF CONTINUOUS SOUNDS OF UP TO ABOUT 5 SECONDS DURATION. THE AUDIO INFORMATION IS ENTERED INTO YOUR COMPUTER THROUGH THE CASSETTE INPUT PORT. YOUR STANDARD CASSETTE RECORDER IS USED AS AN AMPLIFIER WHEN PLACED IN THE RECORD MODE. THE INFORMATION IS DIGITIZED AND STORED IN NUMBERED TABLES IN YOUR COMPUTER'S MEMORY (RAM). IT IS LATER USED DURING THE "RECOGNITION MODE" WHEN NEW AUDIO INFORMATION IS COMPARED WITH THAT STORED IN MEMORY DURING THE "LEARNING MODE". THE SOFTWARE ROUTINES RESIDE ON MEMORY PAGE NINE (HEX LOCATIONS $0900 TO $09FF),AND THE RECOGNITION TABLES ARE LOCATED FROM $0A00 TO $0AFF.THUS, APPLE-LIS'NER USES VERY LITTLE MEMORY TO MAKE YOUR COMPUTER "LISTEN"! AS AN ADDED CONVENIENCE, WE HAVE INCLUDED AN APPLE-LIS'NER PREFIX. THIS PREFIX MAY BE APPENDED TO YOUR INTEGER BASIC PROGRAMS. (NOTE. APPLE-LIS'NER CANNOT BE USED WITH APPLESOFT BASIC DUE TO MEMORY ALLOCATION CONFLICTS). THIS PREFIX AUTOMATICALLY SETS LOMEM AND MOVES THE MACHINE LANGUAGE SUBROUTINES INTO THEIR PROPER LOCATIONS FOR EXECUTION. USING THIS PACKAGE AND A FEW LINES OF BASIC YOU CAN GIVE ANY APPLE PROGRAM THE POWER OF VOICE RECOGNITION WITHOUT HARDWARE COSTS.

***APPLE II AND APPLESOFT ARE REGISTERED TRADEMARKS OF APPLE COMPUTER INC.**

USING APPLE-LIS'NER SUBROUTINES
1. ROUTINES MUST BE RESIDENT IN MEMORY AT HEX LOCATIONS $0900 TO $09FF.
2. LOMEM MUST BE SET AT 2816 DECIMAL ($B00 HEX).
3. CONDITIONS 1 AND 2 ABOVE ARE MET WHEN USING THE PREFIX PROGRAM. IT ALLOWS YOU TO MAKE THE ENTIRE PROGRAM LOAD IN BASIC.
4. THE FIRST LINE OF THE PROGRAM SHOULD READ.

```
0  LEARN = 2304 : RECOG = 2307 : BUILD = 2425 : SAVE = 2313
```

LEARNING MODE
WHEN YOU TEACH THE APPLE-LIS'NER A WORD, IT IS IDENTIFIED BY AN INTEGER NUMBER FROM 1 TO 31. POKE DECIMAL 16 WITH THE I.D. NUMBER FOR THE WORD AND THEN CALL LEARN (CALL 2304) IN YOUR PROGRAM. THE COMPUTER WILL WAIT FOR YOU TO SPEAK. AFTER IT DETECTS THE END OF THE "WORD" (INDICATED BY THE ABSENCE OF SOUND), IT WILL STORE A REPRESENTATION OF THE "WORD" IN THE I.D. PORTION OF THE RECOGNITION TABLES AND THEN RETURN TO YOUR PROGRAM

example. 10 POKE 16,n : CALL LEARN

RECOGNITION MODE

AFTER ALL "n" WORDS (1 ≤ n ≤ 31) HAVE BEEN TAUGHT, THE COMPUTER CAN NOW BE PUT INTO THE RECOGNITION MODE. SIMPLY POKE DECIMAL 16 IN MEMORY WITH THE TOTAL NUMBER OF WORDS, "n", AND THEN CALL RECOG (CALL 2307). THE COMPUTER WILL AGAIN WAIT FOR YOU TO SPEAK. THIS TIME IT WILL COMPARE THE INPUT "WORD" WITH THE "WORDS" IN THE RECOGNITION TABLES TO FIND WHAT IT THINKS IS THE CLOSEST MATCH. THE I.D. NUMBER OF THIS MATCHED "WORD" WILL BE STORED IN DECIMAL LOCATION 16 OF MEMORY ($10 HEX) AND A 16 BIT MEASURE OF THE ERROR WILL APPEAR IN LOCATION 17 ($11 HEX) AND 18 ($12 HEX).** THIS PERMITS USER SELECTABLE TOLERANCE RANGE FOR RECOGNITION. (NOTE. THE RECOGNITION PROCESS INVOLVES SEARCHING THE RECOGNITION TABLES FROM I.D = 1 TO I.D. = n. THUS YOU MUST NOT LEAVE ANY "HOLES" IN THE TABLES WHEN INITIALLY TEACHING THE WORDS TO THE COMPUTER; ie., ALL I.D." BETWEEN 1 AND n MUST BE SPECIFIED AT LEARN TIME).

example.

```
20    POKE 16, n : CALL LEARN
30    ID = PEEK (16) : ERR = PEEK (17)
40    IF PEEK (18) > 0 THEN 20
```

BUILD MODE (OPTIONAL)

THIS WILL ACCEPT VOICE RESPONSE AND BUILD A RECOGNITION TABLE BUT NOT PERMANENTLY STORE IT.

example:

```
50   CALL BUILD
```

SAVE MODE (OPTIONAL)

THIS WILL TAKE THE TEMPORARY DATA LIST BUILT AND STORE IT IN LOCATION n (n = 1 TO 31)

example.

```
60   POKE 16, n : CALL SAVE
```

USING THE APPLE-LIS'NER PREFIX

THIS PROGRAM LOADS FROM INTEGER BASIC AND CAN BE APPENDED TO YOUR BASIC PROGRAM TO MAKE IT LISTEN. THE FIRST LINE OF THE USER PROGRAM SH0ULD ALWAYS BE 0 OR A **BAD BRANCH ERR ** WILL OCCUR WHEN YOU TYPE RUN. IF YOU HAVE AN APPEND PROGRAM THEN.

1. RUN RENUM/APPEND PROGRAM
2. SELECT APPEND FUNCTION
3. LOAD IN USER PROGRAM
4. APPEND APPLE-LIS'NER PREFIX (LOAD)
5. SAVE FINISHED PROGRAM TO TAPE OR DISKETTE

IF YOU DO NOT HAVE A RENUMBER/APPEND PROGRAM, THEN BY HAND.

1. LOAD USER PROGRAM
2. CALL -35 (IGNORE HEX DUMP AND BELL)
3. TYPE FROM MONITOR *CA.CB (RETURN). THE APPLE WILL PRINT:
   00CA - XX YY (THESE ARE VARIABLES FOR STEP 5)
4. TYPE FROM MONITOR *4C.4D (RETURN), THE APPLE WILL PRINT:
   004C - AA BB (THESE ARE VARIABLES FOR STEP 9)
5. TYPE FROM MONITOR *4C:XX YY (RETURN). THIS SETS HIMEM.
6. TYPE A CONTROL C AND RETURN TO GO TO BASIC
7. LOAD APPLE-LIS'NER PREFIX PROGRAM (LOAD)
8. CALL -35 (IGNORE HEX DUMP AND BELL)
9. TYPE FROM MONITOR *4C:AA BB (RETURN). THIS RESETS HIMEM.
10. TYPE A CONTROL C AND RETURN TO GO TO BASIC
11. SAVE FINISHED PROGRAM TO TAPE OR DISKETTE

**** STRICTLY SPEAKING,**

THE ERROR IS ACTUALLY: ERROR = PEEK (17) + PEEK (18) , 256
BUT LOCATION 18 SHOULD ALMOST ALWAYS HAVE A ZERO IN IT.