APPLEKEYS

Turn the Open-Apple and Closed-Apple keys into versatile function keys

The Apple keys on the IIe, IIc, and IIGS (the Option key on the IIGS is equivalent to the Closed-Apple key) have made commercial programs for those computers easier to use than programs for earlier Apples.

The Apple keys have other uses, however. For instance, you can have one type out any string of characters when it is pressed. The Open-Apple key might become the CATALOG command, and the Closed-Apple key might be LIST. Later in this article, we'll take a look at a program that does exactly that.

First, let's look at how character keys on the Apple IIe keyboard work. When a character key is pressed on the Apple keyboard, its ASCII value is stored at −16384, and the high bit of that memory location is turned on. Therefore, if PEEK(−16384) is greater than or equal to 128, a key has been pressed and the input character is CHR$(PEEK(−16384)−128). Before you read another keypress, you must reset the high bit at −16384. You do this with the command POKE −16368,0.

The Open-Apple and Closed-Apple keys work somewhat differently. You read the Open-Apple key with PEEK (−16287). The Closed-Apple key is read with PEEK (−16286). When either Apple key is down, the value at its memory location will be 128 or greater. When the keys are up, the values are 127 or less. Only the high bit is significant; the rest of the bits have no meaning. The Open-Apple key is equivalent to the button on paddle 0; the Closed-Apple key is equivalent to the button on paddle 1.

How can we incorporate the Apple keys into the keyboard input routine? The most obvious way is to modify the standard keyboard input routine, and include code to detect these keys and output our own characters or strings when one is pressed. This, in effect, is what the accompanying program does.

USING THE PROGRAM

Install AppleKeys by typing BRUN APPLEKEYS. When AppleKeys is operational, pressing the Open-Apple key is equivalent to typing CATALOG (or CAT, under ProDOS) and then pressing Return. Pressing the Closed-Apple key types out the LIST command. After pressing the key, you may press Return or type a line number or range of line numbers to be listed and then press Return.

To disconnect AppleKeys type Control-Reset or a PR#3 command. AppleKeys is not operational when the cursor is beyond column 1.

ENTERING THE PROGRAM

If you have an assembler, enter the source code for AppleKeys as shown in **Listing 1**. Assemble the source program and save the object code as APPLEKEYS. If you do not have an assembler, enter the hex code in **Listing 2** and save it to disk with the command

```
BSAVE APPLEKEYS,A$300,L$79
```

For help entering *Nibble* programs, see the Typing Tips section in this issue.

HOW THE PROGRAM WORKS

The code that installs the program first checks to see which operating system, DOS 3.3 or ProDOS, is active, and branches to the appropriate point (**lines 22-39**). After installation, the program exits.

The main loop (**lines 50-55**) continually checks for a keypress. If a (non-Apple key) keypress is detected, the program jumps to the normal KEYIN routine, $FD1B. If no keypress is detected, the program loops back. If an Apple key is pressed, the program sends the CATALOG or LIST command to Applesoft.

MODIFICATIONS

You can change the values of the function keys by changing the strings in **lines 84-91** of **Listing 1**. Be sure to end the command strings with a character whose MSB (most significant bit) is off.

You can put this routine anywhere in memory by changing the ORG location, but be careful not to overwrite any areas of memory used by DOS 3.3 or ProDOS.

LISTING 1: APPLEKEYS Source Code

```
 1 *
 2 * APPLEKEYS Source Code
 3 * by Eli M. Ben-Natan
 4 * Copyright(c) 1988
 5 * MicroSPARC, Inc.
 6 * Concord, MA 01742
 7 *
 8 CR       EQU   $0D        RETURN key code
 9 KEYIN    EQU   $FD1B      Apple keyboard input routine
10 OPENAP   EQU   $C061      Open Apple key (switch 0)
11 KSWL     EQU   $38        Input hook
12 DOS      EQU   $3EA       Reconnect DOS
13 CLOSAP   EQU   $C062      Closed Apple key (switch 1)
14 DELAY    EQU   $FCA8      Delay loop
15 PROCHK   EQU   48896       ProDOS or DOS 3.3?
16 COUT     EQU   $FDED      Character output
17 VECTIN   EQU   $BE32      ProDOS input vector
18 KSW      EQU   $38        system input vector
19
```

```
20            ORG   $300
21
22 INSTALL    LDA   PRODOS
23            CMP   #76
24            BEQ   PRODOS
25 *
26 DOS33      LDA   #<START
27            STA   KSWL          Set input hooks
28            LDA   #>START
29            STA   KSWL+1
30            JSR   DOS           Reconnect DOS
31            RTS
32
33 PRODOS     LDA   #<START
34            STA   KSWL          Set input hooks on zero-
35            STA   VECTIN         and global page
36            LDA   #>START
37            STA   KSWL+1
38            STA   VECTIN+1
39            RTS
40
41 *      Check for Applekeys press      *
42 * and substitute function key codes *
43
44 START      PHA                 Save A reg on stack
45 XSAVE      LDA   #00           Get saved X register
46            BNE   GETCH         If nonzero, function in progress
47            TXA
48            BNE   EXIT
49
50 WAIT       LDA   $C000         Check for key press
51            BMI   EXIT
52            LDA   OPENAP        Open Apple key pressed ?
53            BMI   OPENON        Yes, handle Open Apple key
54            LDA   CLOSAP        Closed Apple key pressed ?
55            BPL   WAIT          No, wait for some key press
56 CLOSON     LDA   #<CLFUN       Get adrs of Closed Apple function
57            BNE   PUTADR
58 OPENON     LDA   PROCHK
59            CMP   #76           ProDOS?
60            BNE   OPEN3
61            LDA   #<OPFUN4      Use 40 column CAT
62            BNE   PUTADR        branch always
63 OPEN3      LDA   #<OPFUN       DOS 3.3, CATALOG command
64 PUTADR     STA   ADR+1         Save function address
65 GETCH      LDX   XSAVE+1       Set X reg to next function char
66            PLA                 Pull garbage off stack!
67 ADR        LDA   FUNC,X        Get next function character
68            BPL   DONE          If MSB not set, then done!
69            INC   XSAVE+1       Else, increment next character
70            RTS                 And return with char in A reg
71
72 DONE       PHA                 Save A reg on stack
73            LDA   #0            Clear next char pointer
74            STA   XSAVE+1
75            PLA                 Restore A reg
76            EOR   #$80          Turn on sign bit
77            RTS
78
79 EXIT       PLA
80            JMP   KEYIN         Go to regular key input
81
82 FUNC       EQU   *
83
84 OPFUN      ASC   "CATALOG"
85            HEX   0D            RETURN code
86
87 OPFUN4     ASC   "CAT"         CAT for ProDOS
88            HEX   0D
89
90 CLFUN      ASC   "LIST"
91            HEX   20            SPACE code
92
93            LST   OFF
```

END OF LISTING 1

LISTING 2: APPLEKEYS

Start: 300 Length: 79

```
86 0300:AD 00 BF C9 4C F0 0C A9
C4 0308:22 85 38 A9 03 85 39 20
B5 0310:EA 03 60 A9 22 85 38 8D
CF 0318:32 BE A9 03 85 39 8D 33
F0 0320:BE 60 48 A9 00 D0 26 8A
35 0328:D0 3A AD 00 C0 30 35 AD
FA 0330:61 C0 30 09 AD 62 C0 10
61 0338:F1 A9 74 D0 0D AD 00 BF
D7 0340:C9 4C D0 04 A9 70 D0 02
86 0348:A9 68 8D 52 03 AE 24 03
A7 0350:68 BD 68 03 10 04 EE 24
7B 0358:03 60 48 A9 00 8D 24 03
F0 0360:68 49 80 60 68 4C 1B FD
84 0368:C3 C1 D4 C1 CC CF C7 0D
8D 0370:C3 C1 D4 0D CC C9 D3 D4
BD 0378:20
```

TOTAL: AEEC

END OF LISTING 2

READING APPLE KEYS

In "Ask Nibble" (*Nibble* Vol. 9/No. 2), Mr. Wagner answers the question "How do I read commands such as Open-Apple-A?" by giving some examples such as

```
IF PEEK(49249) > 127 AND
 PEEK(49152) = ASC("A")+128 THEN
 PRINT "OPEN-APPLE-A"
```

This method reads this Open-Apple location first, and then checks to see if "A" was pressed. Well, there is a practical and more nearly perfect way:

```
10 POKE 49168,0: REM CLEAR STROBE
20 GET A$
30 IF A$ < > "A" THEN 10
40 IF PEEK (49249) > 127 THEN
   PRINT "OA-A!": GOTO 70
50 IF PEEK (49250) > 127 THEN
   PRINT "CA-A!": GOTO 70
60 PRINT "YOU PRESSED THE A KEY!"
70 END
```

This routine checks the keyboard first, and then the Open-Apple key. If you think about it long enough, you'll see that it works (it gives you time to hold down the Open-Apple key before it checks for "A").