Print Error Messages From ONERR

by Thomas J. Zuchowski
304 Wood Run Ct.
Winston-Salem, NC 27103

It is generally a good practice in sophisticated **Applesoft** programming to use **ONERR ... GOTO** to turn the printer off, reset I/O hooks, etc. But it is a real inconvenience during program development to **PEEK(222)** for the **error code** and then have to look it up in the manual to find out what error caused the crash. It is a simple matter to print Applesoft error messages from the ONERR routine. Merely **reset the Onerr flag with a POKE 216,0** and then use the **RESUME** command to **re-execute the line** that caused the error. With ONERR disabled the second time around, the error will generate a normal error message and halt.

DOS ERROR MESSAGES

But what of **DOS** error messages? There are DOS errors that are not wise to repeat without correction, such as the **DISK FULL** error, which will trash the disk directory if the disk is written to without deleting the file that caused the error in the first place. It turns out to be very simple to enable the printing of DOS error messages **before** exiting to the ONERR routine, thus making the double error unnecessary. When DOS encounters an error, it checks to see if the ONERR flag is set, and if it is, DOS jumps over its own error message routine, resets some BASIC pointers, and jumps to the Applesoft error handler.

To print DOS error messages, all we need do is **remove the jump instruction from DOS**. This is easily done by inserting **two NOP instructions** in place of the jump instruction. Thereafter, the DOS error messages will always be printed.

DOS normally does a carriage return/line feed after printing an error message, and then jumps to the Applesoft error handler, which will print **BREAK IN (line #)** if ONERR is not enabled. But if ONERR is being used, the line number must be PEEKed from memory locations **218** and **219**. This can be done in a very tidy fashion by skipping the **CR/LF** routine so that the line number can be printed on the same line as the error message. **CR/LF** is disabled by POKEing NOPs to replace the jump to the **CR/LF** routine.

Now that we have tidy DOS error messages, it might be desirable to have Applesoft error messages printed in the same format. This can't be done with a simple POKE, but requires a 14 byte machine language routine. Have you ever wondered why Applesoft error codes are such off-the-wall numbers? It's because the number stored in location 222 is actually the **offset** used by the error message routine to find the correct message. These messages are stored in a **table** in ROM starting at location **$D260**. Incidentally, Prof Luebbert's 'What's Where in the APPLE?' lists these locations as containing Hi-Res graphics routines which is certainly not the case with my Apple. The error handler gets a character from the message, using the offset. After printing it, it is checked to see if its high bit is set. In machine language, the high bit is normally a sign bit signifying a negative number (when set). The Apple divides its message by setting this bit in the last character of each message. The **message routine continues to print characters from this message table until it prints one that has a negative value**, that is, has its **high bit set**. The routine saves each character on the stack to store it for sign checking after printing because the print routine alters the contents of the **A** register.

HIDING MACHINE CODE IN BASIC

The final trick used by this demo program saves the page three space for other machine code routines by POKEing directly into a **REM** statement in **Line 0**. Since BASIC ignores everything that follows a REM, it's a good place for a short machine code routine. The two requirements for doing this are that the **REM must contain as many or more characters than the routine to be imbedded in it**, and it must be the very **first line** of the BASIC program so it won't change its location in memory if the program is modified. But since Applesoft will interpret the machine code routine as BASIC tokens, listing the REM after running the program looks mighty strange!

LISTING 2

```
]LIST

0  REM +++++++++++++++THIS MUST CO
     NTAIN AT LEAST 14 CHARACTERS
     !
1  REM ************************
3  REM *ERROR MESSAGE PRINTING*
5  REM *  FROM  ONERR...GOTO  *
7  REM ************************
10  ONERR  GOTO 1000
20  FOR I = 2054 TO 2067
30  READ K
40  POKE I,K: NEXT
50  POKE  - 22816,234: POKE  - 22
     817,234: REM  ENABLE DOS ERR
     OR MESSAGES
70  POKE  - 22802,234: POKE  - 22
     803,234: POKE  - 22804,234: REM
     DISABLE CR/LF AFTER DOS ERRO
     R MESSAGE
100  REM  INSERT PROGRAM HERE
200  NEXT : REM NEXT WITHOUT FOR
     ERROR-PUT ANY ERROR YOU LIKE
      HERE
1000 E =  PEEK (222)
1005  IF E = 0 OR E > 15 THEN  PRINT
     : PRINT : CALL 2054: REM APP
     LESOFT ERROR; CALL ROUTINE I
     MBEDDED IN LINE 0
1020  IF E <  > 2 AND E <  > 3 AND
     E <  > 8 AND E <  > 11 THEN
      PRINT " ERROR";: REM THESE
     MESSAGES INCLUDE 'ERROR'
1030  PRINT  CHR$ (7);" IN LINE "
     ; PEEK (218) +  PEEK (219) *
     256
2000  DATA 166,222,189,96,210,72,
     32,92,219,232,104,16,245,96
```

LISTING 1

```
Applesoft ONERR Error Message Print Routine

0806-   A6 DE       LDX   $DE       ;Get message offset
0808-   BD 60 D2    LDA   $D260,X   ;Get character
080B-   48          PHA             ;Put character on stack
080C-   20 5C DB    JSR   $DB5C     ;Print character
080F-   E8          INX             ;Increment offset to next
                                    ;character
0810-   68          PLA             ;Pull old character off
                                    ;stack
0811-   10 F5       BPL   $0808     ;Check if high bit set
                                    ;If not get another char.
0813-   60          RTS
```