Apple II
Technical Notes

$\square$

Developer Technical Support

Apple IIGS
#33:   ERRORDEATH Macro

Revised by: Jim   Mensch   &   Matt   Deatherage
            November 1988
Written by: Allan   Bell,   Apple   Australia   &   Jim   Merritt
            December 1987

This Technical Note presents a short macro which an assembly language program can invoke to handle fatal error conditions.

Early versions of Apple-approved sample assembly language code for the Apple IIGS often invoked an APW macro named `ERRORDEATH`. This macro generated code that was appropriate for handling situations where program execution simply could not proceed due to "fatal" errors, such as a failure to load one or more tools that are required to display more sophisticated error dialogs or the inability to allocate sufficient direct page space for essential tool sets. The macro libraries of prototype APW systems included `ERRORDEATH`, but the release version does not to promote the use of more sophisticated error handling techniques in commercial software packages. The MPW IIGS release never included `ERRORDEATH`.

Below are two versions of `ERRORDEATH`; one is compatible with official standard releases of APW and the other with MPW IIGS. While Apple recommends avoiding the use of `ERRORDEATH` in software intended for commercial release, we feel the code is still useful for providing minimal

error handling capability in prototype code and a brief, yet sophisticated, example of macro construction.

APW Assembler version: MPW IIGS Assembler version:

```
        MACRO                                          MACRO
&lab    ERRORDEATH &text                       ErrorDeath &text
&lab    bcc end&syscnt                          bcc @EDeathEnd
        pha                                         pha
        pea x&syscnt|-16                        pea @Message>>16
        pea x&syscnt                            pea @Message
        ldx #$1503                             ldx #$1503
        jsl $E10000                            jsl $E10000
x&syscnt dc i1'end&syscnt-x&syscnt-1'  @Message dc.B    @EDeathEnd-@Message-1
        dc c"&text"                            dc.B     &text
        dc i1'13',i1'13'                       dc.B     13
        dc c'Error was $'                      dc.B     'Error Was $'
end&syscnt anop                       @EDeathEnd
        MEND                                           MEnd
```

The "active ingredient" in the ERRORDEATH macro is the call to SysFailMgr ($1503), which is made if carry is set at the time control passes to the beginning of the expanded macro code sequence. The APW and MPW IIGS assembler macro expansion mechanisms insert the value represented by the character string argument marker, &text, into the generated code stream and provide SysFailMgr with a pointer to that string. The pseudo-argument, &syscnt, generates unique labels in the positions occupied by the expressions x&syscnt and end&syscnt, which makes it possible to invoke ERRORDEATH more than once during any particular source assembly.

In the MPW IIGS version of the macro, the MPW IIGS assembler creates a unique label for any label beginning with the at sign (@), effectively doing the equivalent of the `&syscnt` in the APW version.

To use ERRORDEATH, simply invoke it after any code sequence or subroutine call that sets the carry when it encounters an error (clears it, otherwise) and leaves an appropriate error code in the accumulator. Note that all ProDOS and Toolbox calls observe this convention. When control passes to the beginning of the ERRORDEATH code sequence, the CPU should be in full-native mode, which means the emulation bit should be clear and the accumulator and index registers should be 16-bits wide). Here is a small code segment which demonstrates invoking the macro:

```
        pushword #21            ; Dialog Manager
        pushword #0             ; Use any version
        _LoadOneTool

    ; If carry is now SET, following macro terminates program execution
    ; with the "sliding Apple" error screen.

    IfWeGoofed   ERRORDEATH 'Cannot load Dialog Manager!'

    ; *** If no error, normal execution continues here ***
```