AppleTalk
#7: MLIACTV Flag and the IIe Workstation Card

Written by: Mark Day & Dan Strnad November 1989

This Technical Note describes a problem using the `MLIACTV` flag with the IIe Workstation Card.

When using the Apple IIe Workstation Card, the `MLIACTV` flag does not always show that the MLI (or PFI) is active. This inconsistency can cause programs that use the `MLIACTV` flag to fail when making MLI calls from interrupt routines. Programs can correct for this problem by making all MLI calls through the `NewMLI` routine listed in this Note and checking the `NewMLIActv` flag instead of the `MLIACTV` flag. This approach solves the problem only if **all** MLI calls, including those made by any interrupt routines, are made through this routine.

The following routine is a replacement for the MLI entry point at $BF00. Programs using this routine can perform a `JSR` to `NewMLI` instead, which fixes the problem. Section 6.2.1 of the *ProDOS 8 Technical Reference Manual* details how programs can cause the MLI to return the their routine rather than the routine that originally called it. For programs using this technique that are also using the routine below, the location below labeled `NewCmdAddr` replaces `CmdAdr` ($BF9C). The steps involved in patching the MLI return location still apply, as specified in Section 6.2.1 of the ProDOS 8 Technical Reference.

```
; MLI patch for Apple II Workstation Card
; by Mark Day
;
; code shown is compatible with MPW IIGS cross-assembler
;
; Your program should use the NewMLIActv flag instead of
; MLIACTV ($BF9B), and should JSR NewMLI instead of
; JSR MLI ($BF00).
;

              machine       M6502         ; 6502 code for //e
              longa  off
              longi  off

parmptr       equ    0              ; two bytes on zero page
MLI           equ    $BF00          ; entry to the real MLI

NewMLI proc

              php                   ; save old interrupt status to
              pla                   ; temporarily disable interrupts
              sta oldp              ; so that NewCmdAddr is always valid
              sei                   ; when an interrupting routine sees
                                    ; NewMLI active.
```

```
        sec
        ror    NewMLIActv    ; NewMLI is now active!
```

```
;
; We need to get the return address from the stack so we can
; get the command number and parameter block address which
; follow the JSR NewMLI, and so we can save NewCmdAddr.
;
             clc
             pla                 ; get low byte of parm address - 1
             sta    parmptr
             adc    #4           ; get real return address
             sta    NewCmdAddr
             pla
             sta    parmptr+1    ; save high byte of parm address - 1
             adc    #0
             sta    NewCmdAddr+1 ; save real return address

             lda oldp
             pha
             plp                 ; reinstate old interrupt status
;
; Now, we copy the call number and parameter list pointer that followed
; the JSR NewMLI, and copy them after a JSR to the real MLI.
;
             tya                 ; save Y on stack
             pha
             ldy    #1           ; offset to command number
             lda    (parmptr),y  ; get command number
             sta    NewCmdNum
             iny                 ; point to parm list ptr (low)
             lda    (parmptr),y
             sta    NewParmPtr
             iny
             lda    (parmptr),y
             sta    NewParmPtr+1
             pla                 ; unstack value of y register
             tay

;
; Now, call the real MLI with the user's command and parameter list
; and jump back to our caller.
;
             jsr    MLI          ; call the real MLI
NewCmdNum    dc.b   0            ; command number
NewParmPtr   dc.w   0            ; parameter list pointer

             php                 ; save C because LSR changes it!
             lsr    NewMLIActv   ; MLI is no longer active
             plp                 ; restore C
             dc.b   $4C          ; JMP absolute instruction
NewCmdAddr   dc.w   0            ; target of jump, caller's return address

NewMLIActv   dc.b   0            ; $80 bit set if MLI active
oldp         ds.b   1            ; used to preserve processor status

             endp
             end
```

Note that this routine also works on the Apple IIGS, even though the problem with the `MLIACTV` flag only affects Apple IIe Workstation Cards.

Further Reference

- *AppleShare Programmer's Guide for the Apple IIGS*
- *ProDOS 8 Technical Reference Manual*