Using extended RAM in XL and XE Computer#

```
MODULE; RAMXL.ACT
;-------------------------------------
; Copyright 1985 by Daniel L. Moore.
; RAMXL may not be sold, but may be
; freely copied and distributed.
;-------------------------------------
; Last modified on 03/30/85
;-------------------------------------

;   Support routines for the "extra"
; 14K of RAM in XLs that is located
; under the OS ROM.
;   When an interrupt occurs and the OS
; is banked out, the RAMXL will bank
; the OS in, and then call the ROM OX
; interrupt handler.  When control
; returns from the ROM OS, the OS is
; banked out, and control is returned
; to the original program.

;   Only the NMI and IRQ vectors are
; supported, since the XL hardware banks
; the OS ROM in automatically when a
; chip reset occurs (the RESET button).

DEFINE  INT_VECTOR = "$FFF0"

CARD    NMI_Vector = $FFFA,
        RES_Vector = $FFFC,
        IRQ_Vector = $FFFE,
        Return_Addr

BYTE    PortB      = $D301,
        NMIEN      = $D40E,
        X_Storage

PROC OS_In=*() ; ROM OS resident
 ~[$AD PortB    ; LDA PortB
  $09 $01       ; ORA #$01  toggle OS bit to ON
  $8D PortB     ; STA PortB
  $60]          ; RTS

PROC OS_Out=*(); ROM OS not resident
 ~[$AD PortB    ; LDA PortB
  $29 $FE       ; AND #$FE  toggle OS bit to OFF
  $8D PortB     ; STA PortB
  $60]          ; RTS

PROC JMP_Vector=*()
 ~[$4C $FFFF]   ; JMP $FFFF

PROC Handle_Interrupt=*()
; Handle the interrupt that just occured.

 ~[$8E X_Storage    ; STX X_Storage
  $AA               ; TAX        A=the interrupt number
  $20 OS_In         ; JSR OS_In
```

```
; Get the address of the desired interrupt routine
  $BD INT_VECTOR   ; LDA INT_VECTOR,X
  $8D JMP_Vector+1 ; STA JMP_VECTOR
  $BD INT_VECTOR+1 ; LDA INT_VECTOR,X
  $8D JMP_Vector+2 ; STA JMP_VECTOR+1

; Setup the stack to fake an interrupt and call
; the OS ROM interrupt code.

; First the return address
  $AD Return_Addr+1; LDA Return_Addr+1
  $48              ; PHA
  $AD Return_Addr  ; LDA Return_Addr
  $48              ; PHA
; Then the proccessor status register
  $58              ; CLI      enable IRQs, for Stage 2 VBLANK
  $08              ; PHP

  $4C JMP_Vector]  ; JMP JMP_Vector

PROC Return_Here=*()
; Return here after the ROM OS interrupt code runs
; Bank the OS out, the return to the
; original program.
 ~[$20 OS_Out       ; JSR OS_Out
  $AE X_Storage    ; LDX X_Storage
  $68              ; PLA    from NMI.Handler or IRQ.Handler
  $40]             ; RTI

PROC NMI_Handler=*()
; Handle NMIs that occur while the OS is
; banked out. Save the A reg, then get
; the vector number and call Handle_Interrupt.
 ~[$48              ; PHA
  $A9 $0A          ; LDA #$0A
  $4C Handle_Interrupt]; JMP Handle_Interrrupt

PROC IRQ_Handler=*()
 ~[$48              ; PHA
  $A9 $0E          ; LDA #$0A
  $4C Handle_Interrupt]; JMP Handle_Interrrupt

;-------------------------------------
; End of actual interrupt code.
; All that is left is installing
; the vectors to the routines.
;-------------------------------------

PROC Install_CharSet()
; Copy the ROM char set at $E000 to $E3FF
; to the RAM bank, so that characters do
; not flicker when the RAM is accessed.
; If this is done, do not use the RAM
; from $E000 to $E3FF (57344 to 58367).
 BYTE POINTER where
 BYTE temp

 FOR where=$E000 TO $E3FF
```

```
 DO
  OS_In()
  temp=where^
  OS_Out()
  where^=temp
 OD
 OS_In()
RETURN

PROC Install_Interrupts()

 Return_Addr=Return_Here; Set the return address pointer

 NMIEN=0  ; Turn all NMI interrupts off.
 ~[$78]    ; SEI  Turn all IRQ interrupts off.

 OS_Out()

; Install the new interrupt routines
; vectors at $FFFA to $FFFF under the
; OS ROM.
 NMI_Vector = NMI_Handler
 IRQ_Vector = IRQ_Handler

 OS_In()

 ~[$58]    ; CLI  Turn IRQs back on.
 NMIEN=$40; Turn NMIs back on.

 Install_CharSet()
RETURN

;-------------------------------------
; Now the routine that lets you get to
; the RAM that is under the OS.
; There are actually 2 memory areas
; present:
;    4K at $C000 to $CFFF, 49152 to 53247
;   10K at $D800 to $FFFF, 55296 to 65535
;
; The last 6 bytes of the 10K area are not
; usable, since that is where the interrupt
; routines are located.  Therefore do not
; use any RAM above $FFF9 (65529) or you
; will crash the system.
;-------------------------------------

PROC MoveBlockXL(BYTE POINTER dest,source, CARD size)
; This is a version of MoveBlock that lets
; you use the extra RAM in XLs.

 OS_Out()
 FOR dest=dest TO dest+size
 DO
   dest^=source^
   source==+1
 OD
 OS_In()
RETURN
```

MODULE; For user.