OS SERIES 10
LAST PART
GEOFF COX

```
 ****************************** LOAD *************************************

 F9B4    TYA             ;A=Y
 F9B5    BEQ     &F9C4   ;
 F9B7    JSR     &FA46   ; print message following call

 F9BA    DB      &0D     ;
 F9BB    DB      'Loading';
 F9C2    DB      &0D     ;
 F9C3    BRK             ;

 F9C5    STA     &BA     ;current block flag
 F9C6    LDX     #&FF    ;X=&FF
 F9C8    LDA     &C1     ;Checksum result
 F9CA    BNE     &F9D9   ;if not 0 F9D9
 F9CC    JSR     &FA72   ;else check filename header block matches searched
                         ;filename if this returns NE then no match
 F9CF    PHP             ;save flags on stack
 F9D0    LDX     #&FF    ;X=&FF
 F9D2    LDY     #&99    ;Y=&99
 F9D4    LDA     #&FA    ;A=&FA this set Y/A to point to 'File?' FA99
 F9D6    PLP             ;get back flags
 F9D7    BNE     &F9F5   ;report a query unexpected file name

 F9D9    LDY     #&8E    ;making Y/A point to 'Data' FA8E for CRC error
 F9DB    LDA     &C1     ;Checksum result
 F9DD    BEQ     &F9E3   ;if 0 F9E3
 F9DF    LDA     #&FA    ;A=&FA
 F9E1    BNE     &F9F5   ;jump to F9F5

 F9E3    LDA     &03C6   ;block number
 F9E6    CMP     &B4     ;current block no. lo
 F9E8    BNE     &F9F1   ;if not eual F9F1
 F9EA    LDA     &03C7   ;block number hi
 F9ED    CMP     &B5     ;current block no. hi
 F9EF    BEQ     &FA04   ;if equal FA04

 F9F1    LDY     #&A4    ;Y=&A4
 F9F3    LDA     #&FA    ;A=&FA  point to 'Block?' error unexpected block no.

                         ;at this point an error HAS occurred

 F9F5    PHA             ;save A on stack
 F9F6    TYA             ;A=Y
 F9F7    PHA             ;save Y on stack
 F9F8    TXA             ;A=X
 F9F9    PHA             ;save X on stack
 F9FA    JSR     &F8B6   ;print CR if indicated by current block flag
 F9FD    PLA             ;get back A
 F9FE    TAX             ;X=A
 F9FF    PLA             ;get back A
 FA00    TAY             ;Y=A
 FA01    PLA             ;get back A
 FA02    BNE     &FA18   ;jump to FA18

 FA04    TXA             ;A=X
 FA05    PHA             ;save A on stack
 FA06    JSR     &F8A9   ;report
 FA09    JSR     &FAD6   ;check loading progress, read another byte
 FA0C    PLA             ;get back A
 FA0D    TAX             ;X=A
 FA0E    LDA     &BE     ;CRC workspace
```

```
FA10    ORA     &BF     ;CRC workspace
FA12    BEQ     &FA8D   ;
FA14    LDY     #&8E    ;Y=&8E
FA16    LDA     #&FA    ;A=&FA  FA8E points to 'Data?'
FA18    DEC     &BA     ;current block flag
FA1A    PHA             ;save A on stack
FA1B    BIT     &EB     ;CFS Active flag
FA1D    BMI     &FA2C   ;if active FA2C
FA1F    TXA             ;A=X
FA20    AND     &0247   ;filing system flag 0=CFS 2=RFS
FA23    BNE     &FA2C   ;
FA25    TXA             ;A=X
FA26    AND     #&11    ;
FA28    AND     &BB     ;current OPTions
FA2A    BEQ     &FA3C   ;ignore errors
FA2C    PLA             ;get back A
FA2D    STA     &B9     ;store A on &B9
FA2F    STY     &B8     ;store Y on &B8
FA31    JSR     &F68B   ;do *EXEC 0 to tidy up
FA34    LSR     &EB     ;halve CFS Active flag to clear bit 7




FA36    JSR     &FAE8   ;bell, reset ACIA & motor
FA39    JMP     (&00B8) ;display selected error report

FA3C    PLA             ;get back A
FA3D    INY             ;Y=Y+1
FA3E    BNE     &FA43   ;
FA40    CLC             ;clear carry flag
FA41    ADC     #&01    ;Add 1
FA43    PHA             ;save A on stack
FA44    TYA             ;A=Y
FA45    PHA             ;save Y on stack
FA46    JSR     &E7DC   ;check if free to print message
FA49    TAY             ;Y=A
FA4A    PLA             ;get back A
FA4B    STA     &B8     ;&B8=8
FA4D    PLA             ;get back A
FA4E    STA     &B9     ;&B9=A
FA50    TYA             ;A=Y
FA51    PHP             ;save flags on stack
FA52    INC     &B8     ;
FA54    BNE     &FA58   ;
FA56    INC     &B9     ;
FA58    LDY     #&00    ;Y=0
FA5A    LDA     (&B8),Y ;get byte
FA5C    BEQ     &FA68   ;if 0 Fa68
FA5E    PLP             ;get back flags
FA5F    PHP             ;save flags on stack
FA60    BEQ     &FA52   ;if 0 FA52 to get next character
FA62    JSR     OSASCI  ;else print
FA65    JMP     &FA52   ;and do it again


FA68    PLP             ;get back flags
FA69    INC     &B8     ;increment pointers
FA6B    BNE     &FA6F   ;
FA6D    INC     &B9     ;
FA6F    JMP     (&00B8) ;and print error message so no error condition
                        ;occcurs
```

```
************ compare filenames ******************************************

FA72    LDX     #&FF    ;X=&FF inx will mean X=0

FA74    INX             ;X=X+1
FA75    LDA     &03D2,X ;sought filename byte
FA78    BNE     &FA81   ;if not 0 FA81
FA7A    TXA             ;else A=X
FA7B    BEQ     &FA80   ;if X=0 A=0 exit
FA7D    LDA     &03B2,X ;else A=filename byte
FA80    RTS             ;return
        ;
FA81    JSR     &E4E3   ;set carry if byte in A is not upper case Alpha
FA84    EOR     &03B2,X ;compare with filename
FA87    BCS     &FA8B   ;if carry set FA8B
FA89    AND     #&DF    ;else convert to upper case
FA8B    BEQ     &FA74   ;and if A=0 filename characters match so do it again
FA8D    RTS             ;return
        ;
FA8E    BRK             ;
FA8F    DB      &D8     ;error number
FA90    DB      'Data'  ;
FA96    BRK             ;

FA97    BNE     &FAAE   ;

FA99    BRK             ;
FA9A    DB      &DB     ;error number
FA9B    DB      'File?' ;
FAA1    BRK             ;

FAA2    BNE     &FAAE   ;

FAA4    BRK             ;
FAA5    DB      &DA     ;error number
FAA6    DB      'Block?'
FAAD    BRK             ;

FAAE    LDA     &BA     ;current block flag
FAB0    BEQ     &FAD3   ;if 0 FAD3 else
FAB2    TXA             ;A=X
FAB3    BEQ     &FAD3   ;If X=0 FAD3
FAB5    LDA     #&22    ;A=&22
FAB7    BIT     &BB     ;current OPTions checking bits 1 and 5
FAB9    BEQ     &FAD3   ;if neither set no  retry so FAD3 else
FABB    JSR     &FB46   ;reset ACIA
FABE    TAY             ;Y=A
FABF    JSR     &FA4A   ;print following message

FAC2    DB      &0D     ;Carriage RETURN
FAC3    DB      &07     ;BEEP
FAC4    DB      'Rewind Tape'   ;
FACF    DW      &0D0D   ;two more newlines
FAD1    BRK             ;

FAD2    RTS             ;return
        ;

FAD3    JSR     &F24D   ;print CR if CFS not operational
FAD6    LDA     &C2     ;filename length/progress flag
FAD8    BEQ     &FAD2   ;if 0 return else
FADA    JSR     &F995   ;confirm ESC not set and CFS not executing
FADD    LDA     &0247   ;filing system flag 0=CFS 2=RFS
FAE0    BEQ     &FAD6   ;if CFS FAD6
FAE2    JSR     &F588   ;else set up ACIA etc
```

```
FAE5    JMP     &FAD6   ;and loop back again


********** sound bell, reset ACIA, motor off ****************************

FAE8    JSR     &E7DC   ;check if free to print message
FAEB    BEQ     &FAF2   ;enable second processor and reset serial system
FAED    LDA     #&07    ;beep
FAEF    JSR     OSWRCH  ;
FAF2    LDA     #&80    ;
FAF4    JSR     &FBBD   ;enable 2nd proc. if present and set up osfile block
FAF7    LDX     #&00    ;
FAF9    JSR     &FB95   ;switch on motor
FAFC    PHP             ;save flags on stack
FAFD    SEI             ;prevent IRQ interrupts
FAFE    LDA     &0282   ;get serial ULA control register setting
FB01    STA     &FE10   ;write to serial ULA control register setting
FB04    LDA     #&00    ;A=0
FB06    STA     &EA     ;store A RS423 timeout counter
FB08    BEQ     &FB0B   ;jump FB0B


FB0A    PHP             ;save flags on stacksave flags
FB0B    JSR     &FB46   ;release ACIA (by &FE08=3)
FB0E    LDA     &0250   ;get last setting of ACIA
FB11    JMP     &E189   ;set ACIA and &250 from A before exit

FB14    PLP             ;get back flags
FB15    BIT     &FF     ;if bit 7of ESCAPE flag not set
FB17    BPL     &FB31   ;then FB31
FB19    RTS             ;else return as unserviced ESCAPE is pending



*************************************************************************
*                                                                       *
*       Claim serial system for sequential Access                       *
*                                                                       *
*************************************************************************


FB1A    LDA     &E3     ;get cassette filing system options byte
                        ;high nybble used for LOAD & SAVE operations
                        ;low nybble used for sequential access

                        ;0000   Ignore errors,          no messages
                        ;0001   Abort if error,         no messages
                        ;0010   Retry after error,      no messages
                        ;1000   Ignore error            short messages
                        ;1001   Abort if error          short messages
                        ;1010   Retry after error       short messages
                        ;1100   Ignore error            long messages
                        ;1101   Abort if error          long messages
                        ;1110   Retry after error       long messages

FB1C    ASL             ;move low nybble into high nybble
FB1D    ASL             ;
FB1E    ASL             ;
FB1F    ASL             ;
FB20    STA     &BB     ;current OPTions save into &BB
FB22    LDA     &03D1   ;get sequential block gap
FB25    BNE     &FB2F   ;goto to &FB2F

*************************************************************************
```

```
*                                                                        *
*       claim serial system for cassette etc.                            *
*                                                                        *
**************************************************************************

FB27    LDA     &E3     ;get cassette filing system options byte
                        ;high nybble used for LOAD & SAVE operations
                        ;low nybble used for sequential access

                        ;0000   Ignore errors,          no messages
                        ;0001   Abort if error,         no messages
                        ;0010   Retry after error,      no messages
                        ;1000   Ignore error            short messages
                        ;1001   Abort if error          short messages
                        ;1010   Retry after error       short messages
                        ;1100   Ignore error            long messages
                        ;1101   Abort if error          long messages
                        ;1110   Retry after error       long messages

FB29    AND     #&F0    ;clear low nybble
FB2B    STA     &BB     ;as current OPTions
FB2D    LDA     #&06    ;set current interblock gap
FB2F    STA     &C7     ;to 6


FB31    CLI             ;allow interrupts
FB32    PHP             ;save flags on stack
FB33    SEI             ;prevent interrupts
FB34    BIT     &024F   ;check if RS423 is busy
FB37    BPL     &FB14   ;if not FB14
FB39    LDA     &EA     ;see if RS423 has timed out
FB3B    BMI     &FB14   ;if not FB14

FB3D    LDA     #&01    ;else load RS423 timeout counter with
FB3F    STA     &EA     ;1 to indicate that cassette has 6850
FB41    JSR     &FB46   ;reset ACIA with &FE80=3
FB44    PLP             ;get back flags
FB45    RTS             ;return
        ;

FB46    LDA     #&03    ;A=3
FB48    BNE     &FB65   ;and exit after resetting ACIA



**********************  set ACIA control register  **********************

FB4A    LDA     #&30    ;set current ACIA control register
FB4C    STA     &CA     ;to &30
FB4E    BNE     &FB63   ;and goto FB63

                        ;if bit 7=0 motor off 1=motor on


***************** control cassette system *******************************

FB50    LDA     #&05    ;set &FE10 to 5
FB52    STA     &FE10   ;setting a transmit baud rate of 300,motor off

FB55    LDX     #&FF    ;
FB57    DEX             ;delay loop
FB58    BNE     &FB57   ;

FB5A    STX     &CA     ;&CA=0
FB5C    LDA     #&85    ;Turn motor on and keep baud rate at 300 recieve
FB5E    STA     &FE10   ;19200 transmit
```

```
FB61    LDA     #&D0    ;A=&D0

FB63    ORA     &C6     ;
FB65    STA     &FE08   ;set up ACIA control register
FB68    RTS             ;returnand return

        ;
FB69    LDX     &03C6   ;block number
FB6C    LDY     &03C7   ;block number hi
FB6F    INX             ;X=X+1
FB70    STX     &B4     ;current block no. lo
FB72    BNE     &FB75   ;
FB74    INY             ;Y=Y+1
FB75    STY     &B5     ;current block no. hi
FB77    RTS             ;return
        ;
FB78    LDY     #&00    ;
FB7A    STY     &C0     ;filing system buffer flag


*****************set (zero) checksum bytes ******************************

FB7C    STY     &BE     ;CRC workspace
FB7E    STY     &BF     ;CRC workspace
FB80    RTS             ;return
        ;


*********** copy sought filename routine ********************************

FB81    LDY     #&FF    ;Y=&FF
FB83    INY             ;Y=Y+1
FB84    INX             ;X=X+1
FB85    LDA     &0300,X ;
FB88    STA     &03D2,Y ;sought filename
FB8B    BNE     &FB83   ;until end of filename (0)
FB8D    RTS             ;return
        ;
FB8E    LDY     #&00    ;Y=0


********************** switch Motor on **********************************

FB90    CLI             ;allow   IRQ interrupts
FB91    LDX     #&01    ;X=1
FB93    STY     &C3     ;store Y as current file handle


********************: control motor ************************************

FB95    LDA     #&89    ;do osbyte 137
FB97    LDY     &C3     ;get back file handle (preserved thru osbyte)
FB99    JMP     OSBYTE  ;turn on motor



****************** confirm file is open  ********************************

FB9C    STA     &BC     ;file status or temporary store
FB9E    TYA             ;A=Y
FB9F    EOR     &0247   ;filing system flag 0=CFS 2=RFS
FBA2    TAY             ;Y=A
FBA3    LDA     &E2     ;CFS status byte
FBA5    AND     &BC     ;file status or temporary store
FBA7    LSR             ;A=A/2
FBA8    DEY             ;Y=Y-1
```

```
FBA9    BEQ     &FBAF   ;
FBAB    LSR             ;A=A/2
FBAC    DEY             ;Y=Y-1
FBAD    BNE     &FBB1   ;
FBAF    BCS     &FBFE   ;


FBB1    BRK             ;
FBB2    DB      &DE     ;error number
FBB3    DB      'Channel' ;
FBBA    BRK             ;


************* read from second processor ********************************

FBBB    LDA     #&01    ;A=1
FBBD    JSR     &FBD3   ;check if second processor file test tube prescence
FBC0    BEQ     &FBFE   ;if not exit
FBC2    TXA             ;A=X
FBC3    LDX     #&B0    ;current load address
FBC5    LDY     #&00    ;Y=00
FBC7    PHA             ;save A on stack
FBC8    LDA     #&C0    ;filing system buffer flag
FBCA    JSR     &0406   ;and out to TUBE
FBCD    BCC     &FBCA   ;
FBCF    PLA             ;get back A
FBD0    JMP     &0406   ;


*************** check if second processor file test tube prescence ******

FBD3    TAX             ;X=A
FBD4    LDA     &B2     ;current load address high word
FBD6    AND     &B3     ;current load address high word
FBD8    CMP     #&FF    ;
FBDA    BEQ     &FBE1   ;if &FF then its for base processor
FBDC    LDA     &027A   ;&FF if tube present
FBDF    AND     #&80    ;to set bit 7 alone
FBE1    RTS             ;return
        ;

******** control ACIA and Motor *****************************************

FBE2    LDA     #&85    ;A=&85
FBE4    STA     &FE10   ;write to serial ULA control register setting
FBE7    JSR     &FB46   ;reset ACIA
FBEA    LDA     #&10    ;A=16
FBEC    JSR     &FB63   ;set ACIA to CFS baud rate
FBEF    JSR     &F995   ;confirm ESC not set and CFS not executing
FBF2    LDA     &FE08   ;read ACIA status register
FBF5    AND     #&02    ;clear all but bit 1
FBF7    BEQ     &FBEF   ;if clear FBEF
FBF9    LDA     #&AA    ;else A=&AA
FBFB    STA     &FE09   ;transmit data register
FBFE    RTS             ;return
        ;
FBFF    BRK             ;


************** FRED 1MHz Bus memory-mapped I/O *************************

FC00    ;test hardware
FC10-13 ;teletext
FC14-1F ;Prestel
FC20-27 ;IEEE interface
FC30    ;
```

```
FC40-47 ;winchester disc interface
FC50    ;
FC60    ;
FC70    ;
FC80    ;
FC90    ;
FCA0    ;
FCB0    ;
FCC0    ;
FCD0    ;
FCE0    ;
FCF0    ;
FCFF    ;paging register for JIM expansion memory


************** JIM 1MHz Bus memory-expansion page ***********************

FD00-FF ;


FDFE    ;Ecosoak Vector


************** SHEILA MOS memory-mapped I/O ***************************


        ;DEVICE         WRITE                   READ
FE00    ;6845 CRTC      address register
FE01    ;6845 CRTC      register file
FE02    ;
FE03    ;
FE04    ;
FE05    ;
FE06    ;
FE07    ;
FE08    ;6850 ACIA      control register        status register
FE09    ;6850 ACIA      transmit data           recieve data
FE0A    ;
FE0B    ;
FE0C    ;
FE0D    ;
FE0E    ;
FE0F    ;
FE10    ;SERIAL ULA     control register
FE11    ;
FE12    ;
FE13    ;
FE14    ;
FE15    ;
FE16    ;
FE17    ;
FE18    ;68B54 ADLC     Disable interrupts      Econet station ID
FE19    ;
FE1A    ;
FE1B    ;
FE1C    ;
FE1D    ;
FE1E    ;
FE1F    ;
FE20    ;Video ULA      control register
FE21    ;Video ULA      palette register        palette register
FE22    ;
FE23    ;
FE24    ;
FE25    ;
FE26    ;
```

```
FE27    ;
FE28    ;
FE29    ;
FE2A    ;
FE2B    ;
FE2C    ;
FE2D    ;
FE2E    ;
FE2F    ;
FE30    ;ROM latch      paged ROM ID            write only
FE31    ;ALTAIR         RAM protect
FE32    ;
FE33    ;
FE34    ;Shadow RAM     B+ only         note different OS
FE35    ;
FE36    ;
FE37    ;
FE38    ;
FE39    ;
FE3A    ;
FE3B    ;
FE3C    ;
FE3D    ;
FE3E    ;
FE3F    ;
FE40    ;MOS 6522 VIA Output Register B                 Input Register B
FE41    ;MOS 6522 VIA Output Register A                 Input Register A
FE42    ;MOS 6522 VIA data direction register B
FE43    ;MOS 6522 VIA data direction register A
FE44    ;MOS 6522 VIA T1C-L  latches                    T1 low Order counter
FE45    ;MOS 6522 VIA T1C-H  counter
FE46    ;MOS 6522 VIA T1L-L low order latches
FE47    ;MOS 6522 VIA T1L-H high order latches
FE48    ;MOS 6522 VIA T2C-L latches                     T2C-L lo order counter
FE49    ;MOS 6522 VIA T2C-H T2 high order counter
FE4A    ;MOS 6522 VIA shift register
FE4B    ;MOS 6522 VIA auxilliary control register ACR
FE4C    ;MOS 6522 VIA Peripheral control register PCR
FE4D    ;MOS 6522 VIA Interrupt  flag    register IFR
FE4E    ;MOS 6522 VIA Interrupt enable   register IER
FE4F    ;MOS 6522 VIA ORB/IRB but no handshake
FE50    ;
FE51    ;
FE52    ;
FE53    ;
FE54    ;
FE55    ;
FE56    ;
FE57    ;
FE58    ;
FE59    ;
FE5A    ;
FE5B    ;
FE5C    ;
FE5D    ;
FE5E    ;
FE5F    ;
FE60    ;USER 6522 VIA Output Register B                Input Register B
FE61    ;USER 6522 VIA Output Register A                Input Register A
FE62    ;USER 6522 VIA data direction register B
FE63    ;USER 6522 VIA data direction register A
FE64    ;USER 6522 VIA T1C-L  latches                   T1 low Order counter
FE65    ;USER 6522 VIA T1C-H  counter
FE66    ;USER 6522 VIA T1L-L low order latches
FE67    ;USER 6522 VIA T1L-H high order latches
FE68    ;USER 6522 VIA T2C-L latches                    T2C-L lo order counter
```

```
FE69    ;USER 6522 VIA T2C-H T2 high order counter
FE6A    ;USER 6522 VIA shift register
FE6B    ;USER 6522 VIA auxilliary control register ACR
FE6C    ;USER 6522 VIA Peripheral control register PCR
FE6D    ;USER 6522 VIA Interrupt  flag    register IFR
FE6E    ;USER 6522 VIA Interrupt enable   register IER
FE6F    ;USER 6522 VIA ORB/IRB but no handshake
FE70    ;
FE71    ;
FE72    ;
FE73    ;
FE74    ;
FE75    ;
FE76    ;
FE77    ;
FE78    ;
FE79    ;
FE7A    ;
FE7B    ;
FE7C    ;
FE7D    ;
FE7E    ;
FE7F    ;
FE80    ;8271 FDC       command register                status register
FE81    ;8271 FDC       parameter register              result register
FE82    ;8271 FDC       reset register
FE83    ;8271 FDC       illegal                         illegal
FE84    ;8271 FDC       data                            data
FE85    ;
FE86    ;
FE87    ;
FE88    ;
FE89    ;
FE8A    ;
FE8B    ;
FE8C    ;
FE8D    ;
FE8E    ;
FE8F    ;
FE90    ;
FE91    ;
FE92    ;
FE93    ;
FE94    ;
FE95    ;
FE96    ;
FE97    ;
FE98    ;
FE99    ;
FE9A    ;
FE9B    ;
FE9C    ;
FE9D    ;
FE9E    ;
FE9F    ;
FEA0    ;68B54 ADLC     control register 1              status register 1
FEA1    ;68B54 ADLC     control register 2/3            status register 2/3
FEA2    ;68B54 ADLC     Tx FIFO (frame continue)        Rx      FIFO
FEA3    ;68B54 ADLC     Tx FIFO (frame terminate)       Rx      FIFO
FEA4    ;
FEA5    ;
FEA6    ;
FEA7    ;
FEA8    ;
FEA9    ;
FEAA    ;
```

```
FEAB    ;
FEAC    ;
FEAD    ;
FEAE    ;
FEAF    ;
FEB0    ;
FEB1    ;
FEB2    ;
FEB3    ;
FEB4    ;
FEB5    ;
FEB6    ;
FEB7    ;
FEB8    ;
FEB9    ;
FEBA    ;
FEBB    ;
FEBC    ;
FEBD    ;
FEBE    ;
FEBF    ;
FEC0    ;7002 ADC       data latch A/D start            status
FEC1    ;7002 ADC       hi data byte
FEC2    ;7002 ADC       lo data byte
FEC3    ;
FEC4    ;
FEC5    ;
FEC6    ;
FEC7    ;
FEC8    ;
FEC9    ;
FECA    ;
FECB    ;
FECC    ;
FECD    ;
FECE    ;
FECF    ;
FED0    ;
FED1    ;
FED2    ;
FED3    ;
FED4    ;
FED5    ;
FED6    ;
FED7    ;
FED8    ;
FED9    ;
FEDA    ;
FEDB    ;
FEDC    ;
FEDD    ;
FEDE    ;
FEDF    ;
FEE0    ;TUBE FIFO1     status register
FEE1    ;TUBE FIFO1
FEE2    ;TUBE FIFO2     status register
FEE3    ;TUBE FIFO2
FEE4    ;TUBE FIFO3     status register
FEE5    ;TUBE FIFO3
FEE6    ;TUBE FIFO4     status register
FEE7    ;TUBE FIFO4
FEE8    ;
FEE9    ;
FEEA    ;
FEEB    ;
FEEC    ;
```

```
FEED    ;
FEEE    ;
FEEF    ;
FEF0    ;
FEF1    ;
FEF2    ;
FEF3    ;
FEF4    ;
FEF5    ;
FEF6    ;
FEF7    ;
FEF8    ;
FEF9    ;
FEFA    ;
FEFB    ;
FEFC    ;
FEFD    ;
FEFE    ;
FEFF    ;



********** EXTENDED VECTOR ENTRY POINTS**********************************
;vectors are pointed to &F000 +vector No. vectors may then be directed thru
;a three byte vector table whose XY address is given by osbyte A8, X=0, Y=&FF
;this is set up as lo-hi byte in ROM and ROM number

FF00    JSR     &FF51   ;E USERV
FF03    JSR     &FF51   ;E BRKV
FF06    JSR     &FF51   ;E IRQ1V
FF09    JSR     &FF51   ;E IRQ2V
FF0C    JSR     &FF51   ;E CLIV
FF0F    JSR     &FF51   ;E BYTEV
FF12    JSR     &FF51   ;E WORDV
FF15    JSR     &FF51   ;E WRCHV
FF18    JSR     &FF51   ;E RDCHV
FF1B    JSR     &FF51   ;E FILEV
FF1E    JSR     &FF51   ;E ARGSV
FF21    JSR     &FF51   ;E BGETV
FF24    JSR     &FF51   ;E BPUTV
FF27    JSR     &FF51   ;E GBPBV
FF2A    JSR     &FF51   ;E FINDV
FF2D    JSR     &FF51   ;E FSCV
FF30    JSR     &FF51   ;E EVENTV
FF33    JSR     &FF51   ;E UPTV
FF36    JSR     &FF51   ;E NETV
FF39    JSR     &FF51   ;E VDUV
FF3C    JSR     &FF51   ;E KEYV
FF3F    JSR     &FF51   ;E INSV
FF42    JSR     &FF51   ;E REMV
FF45    JSR     &FF51   ;E CNPV
FF48    JSR     &FF51   ;E IND1V
FF4B    JSR     &FF51   ;E IND2V
FF4E    JSR     &FF51   ;E IND3V

;at this point the stack will hold 4 bytes (at least)
;S 0,1 extended vector address
;S 2,3 address of calling routine
;A,X,Y,P will be as at entry

FF51    PHA             ;save A on stack
FF52    PHA             ;save A on stack
FF53    PHA             ;save A on stack
FF54    PHA             ;save A on stack
FF55    PHA             ;save A on stack
FF56    PHP             ;save flags on stack
```

```
FF57    PHA             ;save A on stack
FF58    TXA             ;A=X
FF59    PHA             ;save X on stack
FF5A    TYA             ;A=Y
FF5B    PHA             ;save Y on stack
FF5C    TSX             ;get stack pointer into X (&F2 or less)
FF5D    LDA     #&FF    ;A=&FF
FF5F    STA     &0108,X ;A
FF62    LDA     #&88    ;
FF64    STA     &0107,X ;
FF67    LDY     &010A,X ;this is VECTOR number*3+2!!
FF6A    LDA     &0D9D,Y ;lo byte of action address
FF6D    STA     &0105,X ;store it on stack
FF70    LDA     &0D9E,Y ;get hi byte
FF73    STA     &0106,X ;store it on stack
                        ;at this point stack has YXAP and action address
                        ;followed by return address and 5 more bytes
FF76    LDA     &F4     ;
FF78    STA     &0109,X ;store original ROM number below this
FF7B    LDA     &0D9F,Y ;get new rom number
FF7E    STA     &F4     ;store it as ram copy
FF80    STA     &FE30   ;and switch ti that ROM
FF83    PLA             ;get back A
FF84    TAY             ;Y=A
FF85    PLA             ;get back A
FF86    TAX             ;X=A
FF87    PLA             ;get back A
FF88    RTI             ;get back flags and jump to ROM vectored entry
                        ;leaving return address and 5 more bytes on stack


************ return address from ROM indirection ***********************

;at this point stack comprises original ROM number,return from JSR &FF51,
;return from original call the return from FF51 is garbage so;

FF89    PHP             ;save flags on stack
FF8A    PHA             ;save A on stack
FF8B    TXA             ;A=X
FF8C    PHA             ;save X on stack
FF8D    TSX             ; (&F7 or less)
FF8E    LDA     &0102,X ;STORE A AND P OVER
FF91    STA     &0105,X ;return address from (JSR &FF51)
FF94    LDA     &0103,X ;hiding garbage by duplicating A and X just saved
FF97    STA     &0106,X ;
                        ;now we have
                        ;flags,
                        ;A,
                        ;X,
                        ;Rom no.,
                        ;A,
                        ;flags,
                        ;and original return address on stack
                        ;so
FF9A    PLA             ;get back X
FF9B    TAX             ;X=A
FF9C    PLA             ;get back A lose next two bytes
FF9D    PLA             ;get back A lose
FF9E    PLA             ;get back A rom number
FF9F    STA     &F4     ;store it
FFA1    STA     &FE30   ;and set it
FFA4    PLA             ;get back A
FFA5    PLP             ;get back flags
FFA6    RTS             ;return and exit pulling original return address
                        ;from stack
;FFA6 is also default input for CFS OSBPGB, VDUV, IND1V,IND2V,IND3V
```

```
;as these functions are not implemented by the OS but may be used
;by software or other filing systems or ROMs



*************************************************************************
*                                                                       *
*       OSBYTE &9D    FAST BPUT                                         *
*                                                                       *
*************************************************************************
FFA7    TXA             ;A=X
FFA8    BCS     &FFD4   ;if carry set BPUT



*************************************************************************
*                                                                       *
*       OSBYTE &92      READ A BYTE FROM FRED                           *
*                                                                       *
*************************************************************************
;

FFAA    LDY     &FC00,X ;read a byte from FRED area
FFAD    RTS             ;return


*************************************************************************
*                                                                       *
*       OSBYTE &94      READ A BYTE FROM JIM                            *
*                                                                       *
*************************************************************************
;
        ;
FFAE    LDY     &FD00,X ;read a byte from JIM area
FFB1    RTS             ;return


*************************************************************************
*                                                                       *
*       OSBYTE &96      READ A BYTE FROM SHEILA                         *
*                                                                       *
*************************************************************************
;
        ;
FFB2    LDY     &FE00,X ;read a byte from SHEILA memory mapped I/O area
FFB5    RTS             ;return


*********** DEFAULT VECTOR TABLE ****************************************

FFB6    DB      36      ;length of look up table in bytes
FFB7    DB      40      ;low byte of address of this table
FFB8    DB      D9      ;high byte of address of this table



**************************************************************************
**************************************************************************
**                                                                      **
**      OPERATING SYSTEM FUNCTION CALLS                                 **
```

```
 **                                                                      **
 **************************************************************************
 **************************************************************************

 FFB9    JMP     DC0B    ;OSRDRM get a byte from sideways ROM
 FFBC    JMP    &C4C0    ;VDUCHR VDU character output
 FFBF    JMP    &E494    ;OSEVEN generate an EVENT
 FFC2    JMP    &EA1E    ;GSINIT initialise OS string
 FFC5    JMP    &EA2F    ;GSREAD read character from input stream
 FFC8    JMP    &DEC5    ;NVRDCH non vectored OSRDCH
 FFCB    JMP    &E0A4    ;NVWRCH non vectored OSWRCH
 FFCE    JMP     (&021C) ;OSFIND open or close a file
 FFD1    JMP     (&021A) ;OSGBPB transfer block to or from a file
 FFD4    JMP     (&0218) ;OSBPUT save a byte to file
 FFD7    JMP     (&0216) ;OSBGET get a byte from file
 FFDA    JMP     (&0214) ;OSARGS read or write file arguments
 FFDD    JMP     (&0212) ;OSFILE read or write a file
 FFE0    JMP     (&0210) ;OSRDCH get a byte from current input stream
 FFE3    CMP    #&0D     ;OSASCI output a byte to VDU stream expanding
 FFE5    BNE     &FFEE   ;Carriage returns (&0D) to CR/LF (&0A,&0D)
 FFE7    LDA    #&0A     ;OSNEWL output a CR/LF to VDU stream
 FFE9    JSR    OSWRCH   ;
 FFEC    LDA    #&0D     ;
 FFEE    JMP     (&020E) ;OSWRCH output a character to the VDU stream
 FFF1    JMP     (&020C) ;OSWORD perform operation using parameter table
 FFF4    JMP     (&020A) ;OSBYTE perform operation on single byte !
 FFF7    JMP     (&0208) ;OSCLI  pass string to command line interpreter


 *************************************************************************
 *                                                                       *
 *       6502 Vectors                                                    *
 *                                                                       *
 *************************************************************************

 FFFA    DW      &0D00   ;NMI   address
 FFFC    DW      &D9CD   ;RESET address
 FFFE    DW      &DC1C   ;IRQ   address
```

That's it the end of the series and the end of Micronet.

See you on the new system or in the paper mags.

Geoff