```
OS SERIES VI
GEOFF COX
 ************************************************************************
 *                                                                      *
 *       PRINTER DRIVER                                                 *
 *                                                                      *
 ************************************************************************

 ;A=character to print

 E114    BIT     &027C   ;if bit 6 of VDU byte =1 printer is disabled
 E117    BVS     &E139   ;so E139

 E119    CMP     &0286   ;compare with printer ignore character
 E11C    BEQ     &E139   ;if the same E139

 E11E    PHP             ;else save flags
 E11F    SEI             ;bar interrupts
 E120    TAX             ;X=A
 E121    LDA     #&04    ;A=4
 E123    BIT     &027C   ;read bit 2 'disable printer driver'
 E126    BNE     &E138   ;if  set printer is disabled so exit E138
 E128    TXA             ;else A=X
 E129    LDX     #&03    ;X=3
 E12B    JSR     &E1F8   ;and put character in printer buffer
 E12E    BCS     &E138   ;if carry set on return exit, buffer empty

 E130    BIT     &02D2   ;else check buffer busy flag if 0
 E133    BPL     &E138   ;then E138 to exit
 E135    JSR     &E13A   ;else E13A to open printer cahnnel

 E138    PLP             ;get back flags
 E139    RTS             ;and exit

 E13A    LDA     &0285   ;check printer destination
 E13D    BEQ     &E1AD   ;if 0 then E1AD clear printer buffer and exit
 E13F    CMP     #&01    ;if parallel printer not selected
 E141    BNE     &E164   ;E164
 E143    JSR     &E460   ;else read a byte from the printer buffer
 E146    ROR     &02D2   ;if carry is set then 2d2 is -ve
 E149    BMI     &E190   ;so return via E190
 E14B    LDY     #&82    ;else enable interrupt 1 of the external VIA
 E14D    STY     &FE6E   ;
 E150    STA     &FE61   ;pass code to centronics port
 E153    LDA     &FE6C   ;pulse CA2 line to generate STROBE signal
 E156    AND     #&F1    ;to advise printer that
 E158    ORA     #&0C    ;valid data is
 E15A    STA     &FE6C   ;waiting
 E15D    ORA     #&0E    ;
 E15F    STA     &FE6C   ;
 E162    BNE     &E190   ;then exit

 *********:serial printer ***********************************************

 E164    CMP     #&02    ;is it Serial printer??
 E166    BNE     &E191   ;if not E191
 E168    LDY     &EA     ;else is RS423 in use by cassette??
 E16A    DEY             ;
 E16B    BPL     &E1AD   ;if so E1AD to flush buffer

 E16D    LSR     &02D2   ;else clear buffer busy flag
 E170    LSR     &024F   ;and RS423 busy flag
 E173    JSR     &E741   ;count buffer if C is clear on return
 E176    BCC     &E190   ;no room is buffer so exit
 E178    LDX     #&20    ;else
 E17A    LDY     #&9F    ;
```

```
*************************************************************************
*                                                                       *
*       OSBYTE 156 update ACIA setting and RAM copy                     *
*                                                                       *
*************************************************************************
;on entry

E17C    PHP             ;push flags
E17D    SEI             ;bar interrupts
E17E    TYA             ;A=Y
E17F    STX     &FA     ;&FA=X
E181    AND     &0250   ;A=old value AND Y EOR X
E184    EOR     &FA     ;
E186    LDX     &0250   ;get old value in X
E189    STA     &0250   ;put new value in
E18C    STA     &FE08   ;and store to ACIA control register
E18F    PLP             ;get back flags
E190    RTS             ;and exit


************ printer is neither serial or parallel so its user type *****

E191    CLC             ;clear carry
E192    LDA     #&01    ;A=1
E194    JSR     &E1A2   ;




*************************************************************************
*                                                                       *
*       OSBYTE 123 Warn printer driver going dormant                    *
*                                                                       *
*************************************************************************



E197    ROR     &02D2   ;mark printer buffer empty for osbyte
E19A    RTS             ;and exit



E19B    BIT     &02D2   ;if bit 7 is set buffer is empty
E19E    BMI     &E19A   ;so exit

E1A0    LDA     #&00    ;else A=0

E1A2    LDX     #&03    ;X=3
E1A4    LDY     &0285   ;Y=printer destination
E1A7    JSR     &E57E   ;to JMP (NETV)
E1AA    JMP     (&0222) ;jump to PRINT VECTOR for special routines



*************** Buffer handling ****************************************
                        ;X=buffer number
        ;Buffer number  Address         Flag    Out pointer     In pointer
        ;0=Keyboard     3E0-3FF         2CF     2D8             2E1
        ;1=RS423 Input  A00-AFF         2D0     2D9             2E2
```

```
        ;2=RS423 output 900-9BF         2D1     2DA             2E3
        ;3=printer      880-8BF         2D2     2DB             2E4
        ;4=sound0       840-84F         2D3     2DC             2E5
        ;5=sound1       850-85F         2D4     2DD             2E6
        ;6=sound2       860-86F         2D5     2DE             2E7
        ;7=sound3       870-87F         2D6     2DF             2E8
        ;8=speech       8C0-8FF         2D7     2E0             2E9


E1AD    CLC             ;clear carry
E1AE    PHA             ;save A
E1AF    PHP             ;save flags
E1B0    SEI             ;set interrupts
E1B1    BCS     &E1BB   ;if carry set on entry then E1BB
E1B3    LDA     &E9AD,X ;else get byte from baud rate/sound data table
E1B6    BPL     &E1BB   ;if +ve the E1BB
E1B8    JSR     &ECA2   ;else clear sound data

E1BB    SEC             ;set carry
E1BC    ROR     &02CF,X ;rotate buffer flag to show buffer empty
E1BF    CPX     #&02    ;if X>1 then its not an input buffer
E1C1    BCS     &E1CB   ;so E1CB

E1C3    LDA     #&00    ;else Input buffer so A=0
E1C5    STA     &0268   ;store as length of key string
E1C8    STA     &026A   ;and length of VDU queque
E1CB    JSR     &E73B   ;then enter via count purge vector any user routines
E1CE    PLP             ;restore flags
E1CF    PLA             ;restore A
E1D0    RTS             ;and exit


*************************************************************************
*                                                                       *
*       COUNT PURGE VECTOR      DEFAULT ENTRY                           *
*                                                                       *
*                                                                       *
*************************************************************************
;on entry if V set clear buffer
;         if C set get space left
;         else get bytes used

E1D1    BVC     &E1DA   ;if bit 6 is set then E1DA
E1D3    LDA     &02D8,X ;else start of buffer=end of buffer
E1D6    STA     &02E1,X ;
E1D9    RTS             ;and exit

E1DA    PHP             ;push flags
E1DB    SEI             ;bar interrupts
E1DC    PHP             ;push flags
E1DD    SEC             ;set carry
E1DE    LDA     &02E1,X ;get end of buffer
E1E1    SBC     &02D8,X ;subtract start of buffer
E1E4    BCS     &E1EA   ;if carry caused E1EA
E1E6    SEC             ;set carry
E1E7    SBC     &E447,X ;subtract buffer start offset (i.e. add buffer length)
E1EA    PLP             ;pull flags
E1EB    BCC     &E1F3   ;if carry clear E1F3 to exit
E1ED    CLC             ;clear carry
E1EE    ADC     &E447,X ;adc to get bytes used
E1F1    EOR     #&FF    ;and invert to get space left
E1F3    LDY     #&00    ;Y=0
E1F5    TAX             ;X=A
E1F6    PLP             ;get back flags
E1F7    RTS             ;and exit
```

```
********** enter byte in buffer, wait and flash lights if full **********

E1F8    SEI             ;prevent interrupts
E1F9    JSR     &E4B0   ;entera byte in buffer X
E1FC    BCC     &E20D   ;if successful exit
E1FE    JSR     &E9EA   ;else switch on both keyboard lights
E201    PHP             ;push p
E202    PHA             ;push A
E203    JSR     &EEEB   ;switch off unselected LEDs
E206    PLA             ;get back A
E207    PLP             ;and flags
E208    BMI     &E20D   ;if return is -ve Escape pressed so exit
E20A    CLI             ;else allow interrupts
E20B    BCS     &E1F8   ;if byte didn't enter buffer go and try it again
E20D    RTS             ;then return



*************************************************************************
*                                                                       *
*       SAVE/LOAD ENTRY                                                 *
*                                                                       *
*                                                                       *
*************************************************************************


**************: clear osfile control block workspace ********************

E20E    PHA             ;push A
E20F    LDA     #&00    ;A=0
E211    STA     &02EE,X ;clear osfile control block workspace
E214    STA     &02EF,X ;
E217    STA     &02F0,X ;
E21A    STA     &02F1,X ;
E21D    PLA             ;get back A
E21E    RTS             ;and exit


*********** shift through osfile control block *************************

E21F    STY     &E6     ;&E6=Y
E221    ROL             ;A=A*2
E222    ROL             ;*4
E223    ROL             ;*8
E224    ROL             ;*16
E225    LDY     #&04    ;Y=4
E227    ROL             ;A=A*32
E228    ROL     &02EE,X ;shift bit 7 of A into shift register
E22B    ROL     &02EF,X ;and
E22E    ROL     &02F0,X ;shift
E231    ROL     &02F1,X ;along
E234    BCS     &E267   ;if carry set on exit then register has overflowed
                        ;so bad address error
E236    DEY             ;decrement Y
E237    BNE     &E227   ;and if Y>0 then do another shift
E239    LDY     &E6     ;get back original Y
E23B    RTS             ;and exit


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

```
*                                                                      *
*       *LOAD ENTRY                                                    *
*                                                                      *
*                                                                      *
************************************************************************

E23C    LDA     #&FF    ;signal that load is being performed



************************************************************************
*                                                                      *
*       *SAVE ENTRY                                                    *
*                                                                      *
*                                                                      *
************************************************************************
;on entry A=0 for save &ff for load

E23E    STX     &F2     ;store address of rest of command line
E240    STY     &F3     ;
E242    STX     &02EE   ;x and Y are stored in OSfile control block
E245    STY     &02EF   ;
E248    PHA             ;Push A
E249    LDX     #&02    ;X=2
E24B    JSR     &E20E   ;clear the shift register
E24E    LDY     #&FF    ;Y=255
E250    STY     &02F4   ;store im 2F4
E253    INY             ;increment Y
E254    JSR     &EA1D   ;and call GSINIT to prepare for reading text line
E257    JSR     &EA2F   ;read a code from text line if OK read next
E25A    BCC     &E257   ;until end of line reached
E25C    PLA             ;get back A without stack changes
E25D    PHA             ;
E25E    BEQ     &E2C2   ;IF A=0 (SAVE)  E2C2
E260    JSR     &E2AD   ;set up file block
E263    BCS     &E2A0   ;if carry set do OSFILE
E265    BEQ     &E2A5   ;else if A=0 goto OSFILE


E267    BRK             ;
E268    DB      &FC     ;
E269    DB      'Bad Address'   ;error
E274    BRK             ;



************************************************************************
*                                                                      *
*       OSBYTE 119              ENTRY                                  *
*       CLOSE SPOOL/ EXEC FILES                                        *
*                                                                      *
************************************************************************


E275    LDX     #&10    ;X=10 issue *SPOOL/EXEC files warning
E277    JSR     &F168   ;and issue call
E27A    BEQ     &E29F   ;if a rom accepts and issues a 0 then E29F to return
E27C    JSR     &F68B   ;else close the current file
E27F    LDA     #&00    ;A=0



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

```
**                                                                      **
**      *SPOOL                                                          **
**                                                                      **
**************************************************************************
**************************************************************************

E281    PHP             ;if A=0 file is closed so
E282    STY     &E6     ;Store Y
E284    LDY     &0257   ;get file handle
E287    STA     &0257   ;store A as file handle
E28A    BEQ     &E28F   ;if Y<>0 then E28F
E28C    JSR     OSFIND  ;else close file via osfind
E28F    LDY     &E6     ;get back original Y
E291    PLP             ;pull flags
E292    BEQ     &E29F   ;if A=0 on entry then exit
E294    LDA     #&80    ;else A=&80
E296    JSR     OSFIND  ;to open file Y for output
E299    TAY             ;Y=A
E29A    BEQ     &E310   ;and if this is =0 then E310 BAD COMMAND ERROR
E29C    STA     &0257   ;store file handle
E29F    RTS             ;and exit

E2A0    BNE     &E310   ;if NE then BAD COMMAND error
E2A2    INC     &02F4   ;increment 2F4 to 00
E2A5    LDX     #&EE    ;X=&EE
E2A7    LDY     #&02    ;Y=&02
E2A9    PLA             ;get back A
E2AA    JMP     OSFILE  ;and JUMP to OSFILE

**** check for hex digit ************************************************

E2AD    JSR     &E03A   ;look for NEWline
E2B0    JSR     &E08F   ;carry is set if it finds hex digit
E2B3    BCC     &E2C1   ;so E2C1 exit
E2B5    JSR     &E20E   ;clear shift register


************** shift byte into control block ***************************

E2B8    JSR     &E21F   ;shift lower nybble of A into shift register
E2BB    JSR     &E08F   ;then check for Hex digit
E2BE    BCS     &E2B8   ;if found then do it again
E2C0    SEC             ;else set carry
E2C1    RTS             ;and exit



**************; set up OSfile control block ****************************

E2C2    LDX     #&0A    ;X=0A
E2C4    JSR     &E2AD   ;
E2C7    BCC     &E310   ;if no hex digit found EXIT via BAD Command error
E2C9    CLV             ;clear bit 6


******************READ file length from text line************************

E2CA    LDA     (&F2),Y ;read next byte from text line
E2CC    CMP     #&2B    ;is it '+'
E2CE    BNE     &E2D4   ;if not assume its a last byte address so e2d4
E2D0    BIT     &D9B7   ;else set V and M flags
E2D3    INY             ;increment Y to point to hex group

E2D4    LDX     #&0E    ;X=E
E2D6    JSR     &E2AD   ;
E2D9    BCC     &E310   ;if carry clear no hex digit so exit via error
```

```
E2DB    PHP             ;save flags
E2DC    BVC     &E2ED   ;if V set them E2ED explicit end address found
E2DE    LDX     #&FC    ;else X=&FC
E2E0    CLC             ;clear carry
E2E1    LDA     &01FC,X ;and add length data to start address
E2E4    ADC     &0200,X ;
E2E7    STA     &0200,X ;
E2EA    INX             ;
E2EB    BNE     &E2E1   ;repeat until X=0

E2ED    LDX     #&03    ;X=3
E2EF    LDA     &02F8,X ;copy start adddress to load and execution addresses
E2F2    STA     &02F4,X ;
E2F5    STA     &02F0,X ;
E2F8    DEX     ;
E2F9    BPL     &E2EF   ;
E2FB    PLP             ;get back flag
E2FC    BEQ     &E2A5   ;if end of command line reached then E2A5
                        ; to do osfile
E2FE    LDX     #&06    ;else set up execution address
E300    JSR     &E2AD   ;
E303    BCC     &E310   ;if error BAD COMMAND
E305    BEQ     &E2A5   ;and if end of line reached do OSFILE

E307    LDX     #&02    ;else set up load address
E309    JSR     &E2AD   ;
E30C    BCC     &E310   ;if error BAD command
E30E    BEQ     &E2A5   ;else on end of line do OSFILE
                        ;anything else is an error!!!!

******** Bad command error ***********************************

E310    BRK             ;
E311    DB      &FE             ;error number
E312    DB      'Bad Command'   ;
E31D    BRK
E31E    DB      &FB             ;
E31F    DB      'Bad Key'       ;
E326    BRK


*************************************************************************
*                                                                       *
*       *KEY ENTRY                                                      *
*                                                                       *
*************************************************************************

E327    JSR     &E04E   ;set up key number in A
E32A    BCC     &E31D   ;if not valid number give error
E32C    CPX     #&10    ;if key number greater than 15
E32E    BCS     &E31D   ;if greater then give error
E330    JSR     &E045   ;otherwise skip commas, and check for CR
E333    PHP             ;save flags for later
E334    LDX     &0B10   ;get pointer to top of existing key strings
E337    TYA             ;save Y
E338    PHA             ;to preserve text pointer
E339    JSR     &E3D1   ;set up soft key definition
E33C    PLA             ;get back Y
E33D    TAY             ;
E33E    PLP             ;and flags
E33F    BNE     &E377   ;if CR found return else E377 to set up new string
E341    RTS             ;else return to set null string


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

```
*       *FX   OSBYTE                                                    *
*                                                                       *
************************************************************************
        A=number

E342    JSR     &E04E   ;convert the number to binary
E345    BCC     &E310   ;if bad number call bad command
E347    TXA             ;save X


************************************************************************
*                                                                       *
*       *CODE   *MOTOR  *OPT    *ROM    *TAPE   *TV                     *
*                                                                       *
************************************************************************
        ;enter codes    *CODE   &88
                        *MOTOR  &89
                        *OPT    &8B
                        *TAPE   &8C
                        *ROM    &8D
                        *TV     &90

E348    PHA             ;save A
E349    LDA     #&00    ;clear &E4/E5
E34B    STA     &E5     ;
E34D    STA     &E4     ;
E34F    JSR     &E043   ;skip commas and check for newline (CR)
E352    BEQ     &E36C   ;if CR found E36C
E354    JSR     &E04E   ;convert character to binary
E357    BCC     &E310   ;if bad character bad command error
E359    STX     &E5     ;else save it
E35B    JSR     &E045   ;skip comma and check CR
E35E    BEQ     &E36C   ;if CR then E36C
E360    JSR     &E04E   ;get another parameter
E363    BCC     &E310   ;if bad error
E365    STX     &E4     ;else store in E4
E367    JSR     &E03A   ;now we must have a newline
E36A    BNE     &E310   ;if none then output an error

E36C    LDY     &E4     ;Y=third osbyte parameter
E36E    LDX     &E5     ;X=2nd
E370    PLA             ;A=first
E371    JSR     OSBYTE  ;call osbyte
E374    BVS     &E310   ;if V set on return then error
E376    RTS             ;else RETURN

********* *KEY CONTINUED ************************************************
                ;X points to last byte of current key definitions
E377    SEC             ;
E378    JSR     &EA1E   ;look for '"' on return bit 6 E4=1 bit 7=1 if '"'found
                        ;this is a GSINIT call without initial CLC
E37B    JSR     &EA2F   ;call GSREAD carry is set if end of line found
E37E    BCS     &E388   ;E388 to deal with end of line
E380    INX             ;point to first byte of new key definition
E381    BEQ     &E31D   ;if X=0 buffer WILL overflow so exit with BAD KEY error
E383    STA     &0B00,X ;store character
E386    BCC     &E37B   ;and loop to get next byte if end of line not found
E388    BNE     &E31D   ;if Z clear then no matching '"' found or for some
                        ;other reason line doesn't terminate properly
E38A    PHP             ;else if all OK save flags
E38B    SEI             ;bar interrupts
E38C    JSR     &E3D1   ;and move string

E38F    LDX     #&10    ;set loop counter

E391    CPX     &E6     ;if key being defined is found
```

```
E393    BEQ     &E3A3   ;then skip rest of loop
E395    LDA     &0B00,X ;else get start of string X
E398    CMP     &0B00,Y ;compare with start of string Y
E39B    BNE     &E3A3   ;if not the same then skip rest of loop
E39D    LDA     &0B10   ;else store top of string definition
E3A0    STA     &0B00,X ;in designated key pointer
E3A3    DEX             ;decrement loop pointer X
E3A4    BPL     &E391   ;and do it all again
E3A6    PLP             ;get back flags
E3A7    RTS             ;and exit

***********: set string lengths ****************************************

E3A8    PHP             ;push flags
E3A9    SEI             ;bar interrupts
E3AA    LDA     &0B10   ;get top of currently defined strings
E3AD    SEC             ;
E3AE    SBC     &0B00,Y ;subtract to get the number of bytes in strings
                        ;above end of string Y
E3B1    STA     &FB     ;store this
E3B3    TXA             ;save X
E3B4    PHA             ;
E3B5    LDX     #&10    ;and X=16

E3B7    LDA     &0B00,X ;get start offset (from B00) of key string X
E3BA    SEC             ;
E3BB    SBC     &0B00,Y ;subtract offset of string we are working on
E3BE    BCC     &E3C8   ;if carry clear (B00+Y>B00+X) or
E3C0    BEQ     &E3C8   ;result (in A)=0
E3C2    CMP     &FB     ;or greater or equal to number of bytes above
                        ;string we are working on
E3C4    BCS     &E3C8   ;then E3C8
E3C6    STA     &FB     ;else store A in &FB

E3C8    DEX             ;point to next lower key offset
E3C9    BPL     &E3B7   ;and if 0 or +ve go back and do it again
E3CB    PLA             ;else get back value of X
E3CC    TAX             ;
E3CD    LDA     &FB     ;get back latest value of A
E3CF    PLP             ;pull flags
E3D0    RTS             ;and return

***********: set up soft key definition ********************************

E3D1    PHP             ;push P
E3D2    SEI             ;bar interrupts
E3D3    TXA             ;save X
E3D4    PHA             ;push A
E3D5    LDY     &E6     ;get key number




E3D7    JSR     &E3A8   ;and set up &FB
E3DA    LDA     &0B00,Y ;get start of string
E3DD    TAY             ;put it in Y
E3DE    CLC             ;clear carry
E3DF    ADC     &FB     ;add number of bytes above string
E3E1    TAX             ;put this in X
E3E2    STA     &FA     ;and store it
E3E4    LDA     &0268   ;check number of bytes left to remove from key buffer
                        ;if not 0 key is being used (definition expanded so
```

```
                        ;error.  This stops *KEY 1 "*key1 FRED" etc.
E3E7    BEQ     &E3F6   ;if not in use continue

E3E9    BRK             ;
E3EA    DB      &FA     ;error number
E3EB    DB      'Key in use'            ;
E3F5    BRK             ;

E3F6    DEC     &0284   ;decrement consistence flag to &FF to warn that key
                        ;definitions are being changed
E3F9    PLA             ;pull A
E3FA    SEC             ;
E3FB    SBC     &FA     ;subtract &FA
E3FD    STA     &FA     ;and re store it
E3FF    BEQ     &E40D   ;if 0 then E40D

E401    LDA     &0B01,X ;else move string
E404    STA     &0B01,Y ;from X to Y
E407    INY             ;
E408    INX             ;
E409    DEC     &FA     ;for length of string
E40B    BNE     &E401   ;

E40D    TYA             ;store end of moved string(s)
E40E    PHA             ;
E40F    LDY     &E6     ;get back key number
E411    LDX     #&10    ;point at top of last string

E413    LDA     &0B00,X ;get this value
E416    CMP     &0B00,Y ;compare it with start of new or re defined key
E419    BCC     &E422   ;if less then E422
E41B    BEQ     &E422   ;if = then E422
E41D    SBC     &FB     ;shift key definitions accordingly
E41F    STA     &0B00,X ;
E422    DEX             ;point to next lowest string def
E423    BPL     &E413   ;and if =>0 then loop and do it again
E425    LDA     &0B10   ;else make top of key definitions
E428    STA     &0B00,Y ;the start of our key def
E42B    PLA             ;get new end of strings
E42C    STA     &0B10   ;and store it
E42F    TAX             ;put A in X
E430    INC     &0284   ;reset consistency flag
E433    PLP             ;restore flags
E434    RTS             ;and exit


*************** BUFFER ADDRESS HI LOOK UP TABLE ************************

E435    DB      &03
E436    DB      &0A
E437    DB      &08
E438    DB      &07
E439    DB      &07
E43A    DB      &07
E43B    DB      &07
E43C    DB      &07



*************** BUFFER ADDRESS LO LOOK UP TABLE ************************

E43D    DB      &09
E43E    DB      &00
E43F    DB      &00
```

```
E440    DB      &C0
E441    DB      &C0
E442    DB      &50
E443    DB      &60
E444    DB      &70



*************** BUFFER START ADDRESS OFFSET ***************************

E445    DB      &80
E446    DB      &00
E447    DB      &E0
E448    DB      &00
E449    DB      &40
E44A    DB      &C0
E44B    DB      &F0
E44C    DB      &F0


*******: get nominal buffer addresses in &FA/B *************************

        ; ON ENTRY X=buffer number
        ;Buffer number  Address         Flag    Out pointer     In pointer
        ;0=Keyboard     3E0-3FF         2CF     2D8             2E1
        ;1=RS423 Input  A00-AFF         2D0     2D9             2E2
        ;2=RS423 output 900-9BF         2D1     2DA             2E3
        ;3=printer      880-8BF         2D2     2DB             2E4
        ;4=sound0       840-84F         2D3     2DC             2E5
        ;5=sound1       850-85F         2D4     2DD             2E6
        ;6=sound2       860-86F         2D5     2DE             2E7
        ;7=sound3       870-87F         2D6     2DF             2E8
        ;8=speech       8C0-8FF         2D7     2E0             2E9

E450    LDA     &E43E,X ;get buffer base address lo
E453    STA     &FA     ;store it
E455    LDA     &E435,X ;get buffer base address hi
E458    STA     &FB     ;store it
E45A    RTS             ;exit


*************************************************************************
*                                                                       *
*       OSBYTE 152 Examine Buffer status                                *
*                                                                       *
*************************************************************************
;on entry X = buffer number
;on exit FA/B points to buffer start  Y is offset to next character
;if buffer is empty C=1, Y is preserved else C=0

E45B    BIT     &D9B7   ;set V and
E45E    BVS     &E461   ;jump to E461




*************************************************************************
*                                                                       *
*       OSBYTE 145 Get byte from Buffer                                 *
*                                                                       *
*************************************************************************
;on entry X = buffer number
; ON EXIT Y is character extracted
```

```
;if buffer is empty C=1, else C=0

E460    CLV             ;clear V

E461    JMP     (&022C) ;Jump via REMV


*************************************************************************
*                                                                       *
*       REMV buffer remove vector default entry point                   *
*                                                                       *
*************************************************************************
;on entry X = buffer number
;on exit if buffer is empty C=1, Y is preserved else C=0

E464    PHP             ;push flags
E465    SEI             ;bar interrupts
E466    LDA     &02D8,X ;get output pointer for buffer X
E469    CMP     &02E1,X ;compare to input pointer
E46C    BEQ     &E4E0   ;if equal buffer is empty so E4E0 to exit
E46E    TAY             ;else A=Y
E46F    JSR     &E450   ;and get buffer pointer into FA/B
E472    LDA     (&FA),Y ;read byte from buffer
E474    BVS     &E491   ;if V is set (on input) exit with CARRY clear
                        ;Osbyte 152 has been done
E476    PHA             ;else must be osbyte 145 so save byte
E477    INY             ;increment Y
E478    TYA             ;A=Y
E479    BNE     &E47E   ;if end of buffer not reached <>0 E47E

E47B    LDA     &E447,X ;get pointer start from offset table

E47E    STA     &02D8,X ;set buffer output pointer
E481    CPX     #&02    ;if buffer is input (0 or 1)
E483    BCC     &E48F   ;then E48F

E485    CMP     &02E1,X ;else for output buffers compare with buffer start
E488    BNE     &E48F   ;if not the same buffer is not empty so E48F

E48A    LDY     #&00    ;buffer is empty so Y=0
E48C    JSR     &E494   ;and enter EVENT routine to signal EVENT 0 buffer
                        ;becoming empty

E48F    PLA             ;get back byte from buffer
E490    TAY             ;put it in Y
E491    PLP             ;get back flags
E492    CLC             ;clear carry to indicate success
E493    RTS             ;and exit



*************************************************************************
*************************************************************************
**                                                                     **
**      CAUSE AN EVENT                                                 **
**                                                                     **
*************************************************************************
*************************************************************************
;on entry Y=event number
;A and X may be significant Y=A, A=event no. when event generated @E4A1
;on exit carry clear indicates action has been taken else carry set

E494    PHP             ;push flags
E495    SEI             ;bar interrupts
```

```
E496    PHA             ;push A
E497    STA     &FA     ;&FA=A
E499    LDA     &02BF,Y ;get enable event flag
E49C    BEQ     &E4DF   ;if 0 event is not enabled so exit
E49E    TYA             ;else A=Y
E49F    LDY     &FA     ;Y=A
E4A1    JSR     &F0A5   ;vector through &220
E4A4    PLA             ;get back A
E4A5    PLP             ;get back flags
E4A6    CLC             ;clear carry for success
E4A7    RTS             ;and exit

********* check event 2 character entering buffer ***********************

E4A8    TYA             ;A=Y
E4A9    LDY     #&02    ;Y=2
E4AB    JSR     &E494   ;check event
E4AE    TAY             ;Y=A



*************************************************************************
*                                                                       *
*       OSBYTE 138 Put byte into Buffer                                 *
*                                                                       *
*************************************************************************
;on entry X is buffer number, Y is character to be written


E4AF    TYA             ;A=Y

E4B0    JMP     (&022A) ;jump to INSBV



*************************************************************************
*                                                                       *
*       INSBV insert character in buffer vector default entry point     *
*                                                                       *
*************************************************************************
;on entry X is buffer number, A is character to be written


E4B3    PHP             ;save flags
E4B4    SEI             ;bar interrupts
E4B5    PHA             ;save A
E4B6    LDY     &02E1,X ;get buffer input pointer
E4B9    INY             ;increment Y
E4BA    BNE     &E4BF   ;if Y=0 then buffer is full else E4BF
E4BC    LDY     &E447,X ;get default buffer start

E4BF    TYA             ;put it in A
E4C0    CMP     &02D8,X ;compare it with input pointer
E4C3    BEQ     &E4D4   ;if equal buffer is full so E4D4
E4C5    LDY     &02E1,X ;else get buffer end in Y
E4C8    STA     &02E1,X ;and set it from A
E4CB    JSR     &E450   ;and point &FA/B at it
E4CE    PLA             ;get back byte
E4CF    STA     (&FA),Y ;store it in buffer
E4D1    PLP             ;pull flags
E4D2    CLC             ;clear carry for success
E4D3    RTS             ;and exit
```

```
E4D4    PLA             ;get back byte
E4D5    CPX     #&02    ;if we are working on input buffer
E4D7    BCS     &E4E0   ;then E4E0

E4D9    LDY     #&01    ;else Y=1
E4DB    JSR     &E494   ;to service input buffer full event
E4DE    PHA             ;push A


***** return with carry set *********************************************

E4DF    PLA             ;restore A

E4E0    PLP             ;restore flags
E4E1    SEC             ;set carry
E4E2    RTS             ;and exit



***************** CODE MODIFIER ROUTINE *********************************
*                 CHECK FOR ALPHA CHARACTER                               *
*************************************************************************
                        ;ENTRY  character in A
                        ;exit with carry set if non-Alpha character
E4E3    PHA             ;Save A
E4E4    AND     #&DF    ;convert lower to upper case
E4E6    CMP     #&41    ;is it 'A' or greater ??
E4E8    BCC     &E4EE   ;if not exit routine with carry set
E4EA    CMP     #&5B    ;is it less than 'Z'
E4EC    BCC     &E4EF   ;if so exit with carry clear
E4EE    SEC             ;else clear carry
E4EF    PLA             ;get back original value of A
E4F0    RTS             ;and Return
                        ;
                        ;

*******: INSERT byte in Keyboard buffer *********************************

E4F1    LDX     #&00    ;X=0 to indicate keyboard buffer



*************************************************************************
*                                                                       *
*       OSBYTE 153 Put byte in input Buffer checking for ESCAPE         *
*                                                                       *
*************************************************************************
;on entry X = buffer number (either 0 or 1)
;X=1 is RS423 input
;X=0 is Keyboard
;Y is character to be written


E4F3    TXA             ;A=buffer number
E4F4    AND     &0245   ;and with RS423 mode (0 treat as keyboard
                        ;1 ignore Escapes no events no soft keys)
E4F7    BNE     &E4AF   ;so if RS423 buffer AND RS423 in normal mode (1) E4AF

E4F9    TYA             ;else Y=A character to write
E4FA    EOR     &026C   ;compare with current escape ASCII code (0=match)
E4FD    ORA     &0275   ;or with current ESCAPE status (0=ESC, 1=ASCII)
E500    BNE     &E4A8   ;if ASCII or no match E4A8 to enter byte in buffer
E502    LDA     &0258   ;else get ESCAPE/BREAK action byte
E505    ROR             ;Rotate to get ESCAPE bit into carry
E506    TYA             ;get character back in A
```

```
E507    BCS     &E513   ;and if escape disabled exit with carry clear
E509    LDY     #&06    ;else signal EVENT 6 Escape pressed
E50B    JSR     &E494   ;
E50E    BCC     &E513   ;if event handles ESCAPE then exit with carry clear
E510    JSR     &E674   ;else set ESCAPE flag
E513    CLC             ;clear carry
E514    RTS             ;and exit


******** get a byte from keyboard buffer and interpret as necessary *****
;on entry A=cursor editing status 1=return &87-&8B,
;2= use cursor keys as soft keys 11-15
;this area not reached if cursor editing is normal

E515    ROR             ;get bit 1 into carry
E516    PLA             ;get back A
E517    BCS     &E592   ;if carry is set return
                        ;else cursor keys are 'soft'
E519    TYA             ;A=Y get back original key code (&80-&FF)
E51A    PHA             ;PUSH A
E51B    LSR             ;get high nybble into lo
E51C    LSR             ;
E51D    LSR             ;
E51E    LSR             ;A=8-&F
E51F    EOR     #&04    ;and invert bit 2
                        ;&8 becomes &C
                        ;&9 becomes &D
                        ;&A becomes &E
                        ;&B becomes &F
                        ;&C becomes &8
                        ;&D becomes &9
                        ;&E becomes &A
                        ;&F becomes &B

E521    TAY             ;Y=A = 8-F
E522    LDA     &0265,Y ;read 026D to 0274 code interpretation status
                        ;0=ignore key, 1=expand as 'soft' key
                        ;2-&FF add this to base for ASCII code
                        ;note that provision is made for keypad operation
                        ;as codes &C0-&FF cannot be generated from keyboard
                        ;but are recognised by OS
                        ;
E525    CMP     #&01    ;is it 01
E527    BEQ     &E594   ;if so expand as 'soft' key via E594
E529    PLA             ;else get back original byte
E52A    BCC     &E539   ;if above CMP generated Carry then code 0 must have
                        ;been returned so E539 to ignore
E52C    AND     #&0F    ;else add ASCII to BASE key number so clear hi nybble
E52E    CLC             ;clear carry
E52F    ADC     &0265,Y ;add ASCII base
E532    CLC             ;clear carry
E533    RTS             ;and exit
                        ;
********** ERROR MADE IN USING EDIT FACILITY **************************

E534    JSR     &E86F   ;produce bell
E537    PLA             ;get back A, buffer number
E538    TAX             ;X=buffer number

********get byte from buffer ******************************************

E539    JSR     &E460   ;get byte from buffer X
E53C    BCS     &E593   ;if buffer empty E593 to exit
E53E    PHA             ;else Push byte
E53F    CPX     #&01    ;and if RS423 input buffer is not the one
```

```
E541    BNE     &E549   ;then E549

E543    JSR     &E173   ;else oswrch
E546    LDX     #&01    ;X=1 (RS423 input buffer)
E548    SEC             ;set carry

E549    PLA             ;get back original byte
E54A    BCC     &E551   ;if carry clear (I.E not RS423 input) E551
E54C    LDY     &0245   ;else Y=RS423 mode (0 treat as keyboard )
E54F    BNE     &E592   ;if not 0 ignore escapes etc. goto E592

E551    TAY             ;Y=A
E552    BPL     &E592   ;if code is less that &80 its simple so E592
E554    AND     #&0F    ;else clear high nybble
E556    CMP     #&0B    ;if less than 11 then treat as special code
E558    BCC     &E519   ;or function key and goto E519
E55A    ADC     #&7B    ;else add &7C (&7B +C) to convert codes B-F to 7-B
E55C    PHA             ;Push A
E55D    LDA     &027D   ;get cursor editing status
E560    BNE     &E515   ;if not 0 (normal) E515
E562    LDA     &027C   ;else get character destination status

;Bit 0 enables  RS423 driver
;BIT 1 disables VDU driver
;Bit 2 diables  printer driver
;BIT 3 enables  printer independent of CTRL B or CTRL C
;Bit 4 disables spooled output
;BIT 5 not used
;Bit 6 disables  printer driver unless VDU 1 precedes character
;BIT 7 not used

E565    ROR             ;get bit 1 into carry
E566    ROR             ;
E567    PLA             ;
E568    BCS     &E539   ;if carry is set E539 screen disabled
E56A    CMP     #&87    ;else is it COPY key
E56C    BEQ     &E5A6   ;if so E5A6

E56E    TAY             ;else Y=A
E56F    TXA             ;A=X
E570    PHA             ;Push X
E571    TYA             ;get back Y
E572    JSR     &D8CE   ;execute edit action

E575    PLA             ;restore X
E576    TAX             ;
E577    BIT     &025F   ;check econet RDCH flag
E57A    BPL     &E581   ;if not set goto E581
E57C    LDA     #&06    ;else Econet function 6
E57E    JMP     (&0224) ;to the Econet vector


********* get byte from key string ************************************
;on entry 0268 contains key length
;and 02C9 key string pointer to next byte

E581    LDA     &0268   ;get length of keystring
E584    BEQ     &E539   ;if 0 E539 get a character from the buffer
E586    LDY     &02C9   ;get soft key expansion pointer
E589    LDA     &0B01,Y ;get character from string
E58C    INC     &02C9   ;increment pointer
E58F    DEC     &0268   ;decrement length


************** exit with carry clear ***********************************
```

```
E592    CLC             ;
E593    RTS             ;exit
                        ;
*** expand soft key strings ***********************************************
Y=pointer to sring number

E594    PLA             ;restore original code
E595    AND     #&0F    ;blank hi nybble to get key string number
E597    TAY             ;Y=A
E598    JSR     &E3A8   ;get string length in A
E59B    STA     &0268   ;and store it
E59E    LDA     &0B00,Y ;get start point
E5A1    STA     &02C9   ;and store it
E5A4    BNE     &E577   ;if not 0 then get byte via E577 and exit


********** deal with COPY key ***************************************

E5A6    TXA             ;A=X
E5A7    PHA             ;Push A
E5A8    JSR     &D905   ;read a character from the screen
E5AB    TAY             ;Y=A
E5AC    BEQ     &E534   ;if not valid A=0 so BEEP
E5AE    PLA             ;else restore X
E5AF    TAX             ;
E5B0    TYA             ;and Y
E5B1    CLC             ;clear carry
E5B2    RTS             ;and exit



************************************************************************
*                                                                      *
*        OSBYTE LOOK UP TABLE                                          *
*                                                                      *
************************************************************************


E5B3    DB      &21,&E8         ;OSBYTE   0  (&E821)
E5B5    DB      &88,&E9         ;OSBYTE   1  (&E988)
E5B7    DB      &D3,&E6         ;OSBYTE   2  (&E6D3)
E5B9    DB      &97,&E9         ;OSBYTE   3  (&E997)
E5BB    DB      &97,&E9         ;OSBYTE   4  (&E997)
E5BD    DB      &76,&E9         ;OSBYTE   5  (&E976)
E5BF    DB      &88,&E9         ;OSBYTE   6  (&E988)
E5C1    DB      &8B,&E6         ;OSBYTE   7  (&E68B)
E5C3    DB      &89,&E6         ;OSBYTE   8  (&E689)
E5C5    DB      &B0,&E6         ;OSBYTE   9  (&E6B0)
E5C7    DB      &B2,&E6         ;OSBYTE  10  (&E6B2)
E5C9    DB      &95,&E9         ;OSBYTE  11  (&E995)
E5CB    DB      &8C,&E9         ;OSBYTE  12  (&E98C)
E5CD    DB      &F9,&E6         ;OSBYTE  13  (&E6F9)
E5CF    DB      &FA,&E6         ;OSBYTE  14  (&E6FA)
E5D1    DB      &A8,&F0         ;OSBYTE  15  (&F0A8)
E5D3    DB      &06,&E7         ;OSBYTE  16  (&E706)
E5D5    DB      &8C,&DE         ;OSBYTE  17  (&DE8C)
E5D7    DB      &C8,&E9         ;OSBYTE  18  (&E9C8)
E5D9    DB      &B6,&E9         ;OSBYTE  19  (&E9B6)
E5DB    DB      &07,&CD         ;OSBYTE  20  (&CD07)
E5DD    DB      &B4,&F0         ;OSBYTE  21  (&F0B4)
E5DF    DB      &6C,&E8         ;OSBYTE 117  (&E86C)
E5E1    DB      &D9,&E9         ;OSBYTE 118  (&E9D9)
E5E3    DB      &75,&E2         ;OSBYTE 119  (&E275)
E5E5    DB      &45,&F0         ;OSBYTE 120  (&F045)
E5E7    DB      &CF,&F0         ;OSBYTE 121  (&F0CF)
```

```
E5E9    DB      &CD,&F0         ;OSBYTE 122  (&F0CD)
E5EB    DB      &97,&E1         ;OSBYTE 123  (&E197)
E5ED    DB      &73,&E6         ;OSBYTE 124  (&E673)
E5EF    DB      &74,&E6         ;OSBYTE 125  (&E674)
E5F1    DB      &5C,&E6         ;OSBYTE 126  (&E65C)
E5F3    DB      &35,&E0         ;OSBYTE 127  (&E035)
E5F5    DB      &4F,&E7         ;OSBYTE 128  (&E74F)
E5F7    DB      &13,&E7         ;OSBYTE 129  (&E713)
E5F9    DB      &29,&E7         ;OSBYTE 130  (&E729)
E5FB    DB      &85,&F0         ;OSBYTE 131  (&F085)
E5FD    DB      &23,&D9         ;OSBYTE 132  (&D923)
E5FF    DB      &26,&D9         ;OSBYTE 133  (&D926)
E601    DB      &47,&D6         ;OSBYTE 134  (&D647)
E603    DB      &C2,&D7         ;OSBYTE 135  (&D7C2)
E605    DB      &57,&E6         ;OSBYTE 136  (&E657)
E607    DB      &7F,&E6         ;OSBYTE 137  (&E67F)
E609    DB      &AF,&E4         ;OSBYTE 138  (&E4AF)
E60B    DB      &34,&E0         ;OSBYTE 139  (&E034)
E60D    DB      &35,&F1         ;OSBYTE 140  (&F135)
E60F    DB      &35,&F1         ;OSBYTE 141  (&F135)
E611    DB      &E7,&DB         ;OSBYTE 142  (&DBE7)
E613    DB      &68,&F1         ;OSBYTE 143  (&F168)
E615    DB      &E3,&EA         ;OSBYTE 144  (&EAE3)
E617    DB      &60,&E4         ;OSBYTE 145  (&E460)
E619    DB      &AA,&FF         ;OSBYTE 146  (&FFAA)
E61B    DB      &F4,&EA         ;OSBYTE 147  (&EAF4)
E61D    DB      &AE,&FF         ;OSBYTE 148  (&FFAE)
E61F    DB      &F9,&EA         ;OSBYTE 149  (&EAF9)
E621    DB      &B2,&FF         ;OSBYTE 150  (&FFB2)
E623    DB      &FE,&EA         ;OSBYTE 151  (&EAFE)
E625    DB      &5B,&E4         ;OSBYTE 152  (&E45B)
E627    DB      &F3,&E4         ;OSBYTE 153  (&E4F3)
E629    DB      &FF,&E9         ;OSBYTE 154  (&E9FF)
E62B    DB      &10,&EA         ;OSBYTE 155  (&EA10)
E62D    DB      &7C,&E1         ;OSBYTE 156  (&E17C)
E62F    DB      &A7,&FF         ;OSBYTE 157  (&FFA7)
E631    DB      &6D,&EE         ;OSBYTE 158  (&EE6D)
E633    DB      &7F,&EE         ;OSBYTE 159  (&EE7F)
E635    DB      &C0,&E9         ;OSBYTE 160  (&E9C0)
E637    DB      &9C,&E9         ;
E639    DB      &59,&E6         ;
```

```
*************************************************************************
*                                                                       *
*        OSWORD LOOK UP TABLE                                           *
*                                                                       *
*************************************************************************
```

```
E63B    DB      &02,&E9         ;OSWORD   0  (&E902)
E63D    DB      &D5,&E8         ;OSWORD   1  (&E8D5)
E63F    DB      &E8,&E8         ;OSWORD   2  (&E8E8)
E641    DB      &D1,&E8         ;OSWORD   3  (&E8D1)
E643    DB      &E4,&E8         ;OSWORD   4  (&E8E4)
E645    DB      &03,&E8         ;OSWORD   5  (&E803)
E647    DB      &0B,&E8         ;OSWORD   6  (&E80B)
E649    DB      &2D,&E8         ;OSWORD   7  (&E82D)
E64B    DB      &AE,&E8         ;OSWORD   8  (&E8AE)
E64D    DB      &35,&C7         ;OSWORD   9  (&C735)
E64F    DB      &F3,&CB         ;OSWORD  10  (&CBF3)
```

```
E651    DB      &48,&C7         ;OSWORD  11  (&C748)
E653    DB      &E0,&C8         ;OSWORD  12  (&C8E0)
E655    DB      &CE,&D5         ;OSWORD  13  (&D5CE)


*************************************************************************
*                                                                       *
*       OSBYTE 136   Execute Code via User Vector                       *
*                                                                       *
*       *CODE effectively                                               *
*                                                                       *
*************************************************************************

E658    LDA     #00     ;A=0


*************************************************************************
*                                                                       *
*       *LINE   entry                                                   *
*                                                                       *
*************************************************************************

E659    JMP     (&0200) ;Jump via USERV



*************************************************************************
*                                                                       *
*       OSBYTE  126  Acknowledge detection of ESCAPE condition          *
*                                                                       *
*************************************************************************


E65C    LDX     #&00    ;X=0
E65E    BIT     &FF     ;if bit 7 not set there is no ESCAPE condition
E660    BPL     &E673   ;so E673
E662    LDA     &0276   ;else get ESCAPE Action, if this is 0
                        ;Clear ESCAPE
                        ;close EXEC files
                        ;purge all buffers
                        ;reset VDU paging counter
E665    BNE     &E671   ;else do none of the above
E667    CLI             ;allow interrupts
E668    STA     &0269   ;number of lines printed since last halt in paged
                        ;mode = 0
E66B    JSR     &F68D   ;close any open EXEC files
E66E    JSR     &F0AA   ;clear all buffers
E671    LDX     #&FF    ;X=&FF to indicate ESCAPE acknowledged


*************************************************************************
*                                                                       *
*       OSBYTE  124  Clear ESCAPE condition                             *
*                                                                       *
*************************************************************************


E673    CLC             ;clear carry


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

```
*                                                                      *
*       OSBYTE  125  Set ESCAPE flag                                   *
*                                                                      *
************************************************************************


E674    ROR     &FF     ;clear  bit 7 of ESCAPE flag
E676    BIT     &027A   ;read bit 7 of Tube flag
E679    BMI     &E67C   ;if set TUBE exists so E67C
E67B    RTS             ;else RETURN
                        ;
E67C    JMP     &0403   ;Jump to Tube entry point


************************************************************************
*                                                                      *
*       OSBYTE  137  Turn on Tape motor                                *
*                                                                      *
************************************************************************


E67F    LDA     &0282   ;get serial ULA control setting
E682    TAY             ;Y=A
E683    ROL             ;rotate left to get bit 7 into carry
E684    CPX     #&01    ;if X=1 then user wants motor on so CARRY set else
                        ;carry is cleared
E686    ROR             ;put carry back in control RAM copy
E687    BVC     &E6A7   ;if bit 6 is clear then cassette is selected
                        ;so write to control register and RAM copy

E689    LDA     #&38    ;A=ASCII 8


************************************************************************
*                                                                      *
*       OSBYTE 08/07 set serial baud rates                             *
*                                                                      *
************************************************************************
        on entry X=baud rate
                 A=8 transmit
                 A=7 receive

E68B    EOR     #&3F    ;converts ASCII 8 to 7 binary and ASCII 7 to 8 binary
E68D    STA     &FA     ;store result
E68F    LDY     &0282   ;get serial ULA control register setting
E692    CPX     #&09    ;is it 9 or more?
E694    BCS     &E6AD   ;if so exit
E696    AND     &E9AD,X ;and with byte from look up table
E699    STA     &FB     ;store it
E69B    TYA             ;put Y in A
E69C    ORA     &FA     ;and or with Accumulator
E69E    EOR     &FA     ;zero the three bits set true
E6A0    ORA     &FB     ;set up data read from look up table + bit 6
E6A2    ORA     #&40    ;
E6A4    EOR     &025D   ;write cassette/RS423 flag

E6A7    STA     &0282   ;store serial ULA flag
E6AA    STA     &FE10   ;and write to control register
E6AD    TYA             ;put Y in A to save old contents
E6AE    TAX             ;write new setting to X
E6AF    RTS             ;and return
```